MP/M II™

Operating System

SYSTEM IMPLEMENTOR'S GUIDE

Digital Research
P. O. Box 579
801 Lighthouse Avenue
Pacific Grove, CA 93950
(408) 649-3896
TWX 910 360 5001

COPYRIGHT

DISCLAIMER

Digital Research makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, Digital Research reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Digital Research to notify any person of such revision or changes.

TRADEMARKS

FOREWORD

MP/M II™ is a multi-user operating system for any microcomputer based on an 8-bit Zilog Z80® or Intel 8080 or 8085 microprocessor. Typically, an MP/M II system resides in approximately 27k. 16k of the operating system must reside in common memory.

The version of MP/M II that Digital Research ships cannot be directly booted on any specific hardware configuration. However, all the hardware-dependent code is isolated in specific subroutines that can be modified by the user.

This document describes the procedures required to implement MP/M II for a custom hardware environment. At minimum, the custom hardware environment must include an 8080, 8085, or Z-80 processor, 32K bytes of random access memory (RAM), a system console, and a real-time clock. This manual assumes the reader is familiar with the following Digital Research publications:

- MP/M II User's Guide

- MP/M II Programmer's Guide

It is also assumed that the reader has already implemented a CP/M 2 Basic Input Output System (BIOS), preferrably on the target MP/M II machine.

TABLE OF CONTENTS

TABLE OF CONTENTS

(continued)

APPENDIXES

SECTION 1

MP/M II ALTERATION PROCEDURE

The MP/M II operating system is designed so that the user can alter a specific set of subroutines that define the hardware operating environment. By modifying these subroutines, the user can produce a diskette that operates with any IBM-3740 format compatible diskette subsystem and other peripheral devices.

Although the standard MP/M II is shipped on single-density floppy disks, field-alteration features allow the user to adapt MP/M II to a wide variety of disk subsystems, including single drive minidisks and high-capacity "hard disk" systems.

To achieve device independence, MP/M II has isolated all hardware-dependent code into an XIOS module. The user can rewrite the distributed version of the XIOS to customize the interface between the remaining MP/M II modules and the user's own hardware system. The user can also rewrite the distributed version of the LDRBIOS, which loads the MP/M II system from the disk.

There are actually two versions of the XIOS: the RESXIOS for non-banked systems, and the BNKXIOS for banked memory systems. To avoid repeating both names for each reference, the term XIOS refers to both versions.

1.1 Preparation for MP/M II Alteration

To simplify the alteration process, this document assumes that a CP/M 2 BIOS has already been implemented on the target MP/M II machine. You must implement both the BIOS as well as the XIOS because the MP/M II loader uses a CP/M 2 BIOS to load the MP/M II system. Once loaded, MP/M II uses the XIOS and not the BIOS. The CP/M 2 BIOS used by the MP/M II loader is called the LDRBIOS.

Another good reason for implementing CP/M 2 on the target MP/M machine is that debugging your XIOS is simpler when you can run SID or DDT under a CP/M 2 system.

1.2 Customizing the MPMLDR

To customize the MPMLDR, you must integrate a LDRBIOS for your hardware configuration into the MPMLDR.COM file supplied on the distribution disk. The required LDRBIOS can be simply a version of your CP/M 2 BIOS, altered as described below and renamed to LDRBIOS.

The customized LDRBIOS must have an ORG of 1700H, perform console output functions, and be able to read data from a single disk drive. The first call MPMLDR makes to LDRBIOS is SELDSK: select disk. If your system has devices that require initialization, place initialization code or perhaps a call to the LDRBIOS cold start at the beginning of the SELDSK handler.

The LDRBIOS need only perform the operations described above. Other functions can be deleted to conserve space. There is only one restriction on memory space for LDRBIOS: it cannot extend above the base of the MPM.SYS which it is loading. (GENSYS Lists MP/M II's base address in its load map.) However, if you plan to boot MP/M II from floppy disks, you will encounter a LDRBIOS upper address limit of 1A00H in order to place the MPMLDR.COM file on two system tracks.

Test LDRBIOS completely to ensure that it properly performs console character output and disk reads. Be especially careful that no disk write operations occur accidently during read operations, and check that the proper track and sectors are addressed on all reads.

Use the following steps to integrate a custom LDRBIOS into the MPMLDR.COM:

1. Obtain access to a CP/M system and prepare a LDRBIOS.HEX file.

2. Read the MPMLDR.COM file into memory using either DDT or SID.

```
A>DDT MPMLDR.COM
DDT VERS 2.0
NEXT  PC
1780 0100
```

3. Using the input command (I), specify that the LDRBIOS.HEX file is to be read in and then read (R) in the file. This operation overlays the LDRBIOS portion of the MP/M loader.

```
-ILDRBIOS.HEX
-R
NEXT PC
1A00 0000
```

4. Exit the debugger, returning to the CCP by executing a jump to location zero.

   ```
   -G0
   ```

5. Write the updated memory image onto a disk file. Use the CP/M SAVE command to write the updated memory image onto a disk file. In the example below, the X in front of the filename simply designates an experimental version, and preserves the original.

   ```
   A>SAVE 26 XMPMLDR.COM
   ```

6. Test XMPMLDR.COM and then rename it to MPMLDR.COM.

1.3 Customizing the XIOS

As you are tailoring MP/M II for your computer system, your new XIOS will require software development and testing. Two sample XIOS's are listed in the Appendixes, and can be used as models for the customized package.

The XIOS entry points, including both basic and extended, are described in Sections 2 and 3. These sections, along with the appendixes, give you the information you need to write your XIOS. Your initial implementation of an XIOS should use polled I/O without any interrupts. This initial system can run without a clock interrupt. Implement interrupts only after your XIOS is fully developed and tested.

Follow the procedure below to prepare a BNKXIOS.SPR or RESXIOS.SPR file from your customized XIOS:

1. Assemble your BNKXIOS.ASM or RESXIOS.ASM with RMAC or any other assembler that can generate a file of type REL in Microsoft's relocatable object file format.

   ```
   A>RMAC BNKXIOS
   ```

2. Link the BNKXIOS.REL or RESXIOS.REL file using the Digital Research LINK-80 to produce the BNKXIOS.SPR or RESXIOS.SPR file.

   ```
   A>LINK BNKXIOS [OS]
   ```

1.4 Debugging an XIOS

You can debug an XIOS or a resident system process with DDT or SID running under CP/M. The debugging technique is outlined in the following steps:

1. Determine the amount of memory available to MP/M II when the debugger and CP/M are resident. Do this by loading the debugger and then listing the jump instruction at location 0005H. This jump is to the base of the debugger.

```
A>DDT
DDT VERS 2.0

-L5

0005  JMP C800
```

2. Using GENSYS running under CP/M, generate and MPM.SYS file that specifies the top of memory determined by the previous step, allowing at least 256 bytes for a patch area.

```
...
Top page of operating system (xx) ? C6
...
```

Also while executing GENSYS, specify a breakpoint restart number different from the one used by the CP/M debugger you plan to use. The suggested MP/M II restart is #6; however, any restart from #1 to #6 can usually be used. The CP/M debuggers normally use restart #7.

```
...
Breakpoint RST (xx) ? 6
...
```

Note: If you are also debugging a resident system process, be sure to select it for inclusion in MPM.SYS during GENSYS execution.

3. Using CP/M, load the MPMLDR.COM file into memory.

```
A>DDT MPMLDR.COM
DDT VERS 2.0
NEXT  PC
1A00 0100
```

4. Place the characters "$B" into locations 005DH and 005EH of the default FCB based at 005CH. This operation can be done with the I command:

   ```
   -I$B
   ```

   The "$B" causes the MPMLDR to break after loading the MPM.SYS file. You can specify the breakpoint restart to be executed by the MPMLDR by adding one additional character to the string in the fourth position of the default FCB.

   ```
   -I$B6
   ```

   In the example above, a restart #6 is to be executed by the MPMLDR when loading of the MPM.SYS file is completed. If no restart number is supplied, the default restart is #7. Remember, the restart number at the location 5FH is the CP/M debugger restart number, not the MP/M debugger restart.

5. Execute the MPMLDR.COM program by entering a G command:

   ```
   -G
   ```

6. After the G command, the MP/M II loader loads the MP/M II operating system into memory and displays a memory map. You may obtain a hard copy of your load map during the GENSYS operation by entering a ^P before executing GENSYS.

7. If you are debugging an XIOS, note the address of the BNKXIOS.SPR or RESXIOS.SPR memory segment. You must also note the address of SYSTEM.DAT. If you are debugging a resident system process, note its address as well. The debugger lists actual addresses at the console. If your hard copy listing of the XIOS or RSP starts at zero, you must add the base address listed in the GENSYS load map to each address on the listing to make the listing reflect actual addresses. Or you can assemble the code again with an additional ORG statement specifying the base listed in the load map, although the object code generated by this assembly is unusable.

8. Using the X command, determine the MP/M II beginning execution address. The address is the first location past the current program counter.

   ```
   -X
   ......................... P = 09F2 .....
   ```

   In the example shown above, MP/M II execution starts at address 09F3H, which is the first instruction after the restart at 09F2H.

9. Begin execution of MP/M II using the G command, specifying the start address and any breakpoints you need in your code. The actual memory address can be determined by entering an H command to add the code segment base address given in the memory map to the relative displacement address in your XIOS or resident system process listing.

   The following example shows how to set a breakpoint in an XIOS at the list subroutine entry point given in the memory map:

   ```
   ...
   XIOSJMP TBL   C300H  0100H

   -G9F3,C30F
   ```

   09F3H is the beginning MP/M II execution address and C30FH is the XIOS jump vector address of the list subroutine.

10. At this point, you have MP/M II running with CP/M and the CP/M debugger also in memory. Because interrupts are left enabled during operation of the CP/M debugger, ensure that interrupt-driven code does not execute through a breakpoint.

    Because the CP/M debugger operates with interrupts left enabled, it is a somewhat difficult task to debug an interrupt-driven console handler. Approach this problem by leaving console #0 in a polled mode while debugging the other consoles in an interrupt-driven mode. Once this is done, very little, if any, debugging is required to adapt the interrupt-driven code from another console to console #0. It is further recommended that you maintain a debug version of your XIOS that has polled I/O for console #0. Otherwise, it is not possible to run the CP/M debugger underneath the MP/M II system because the CP/M debugger cannot get any console input, as all of it is sent to the MP/M interrupt-driven console #0 handler.

1.5 Directly Booting MP/M II

In systems where MP/M II is to be booted directly at cold start rather than loaded and run as a transient program under CP/M, the customized MPMLDR.COM file and cold start loader cn be placed on the first two tracks of an eight-inch floppy disk. If a CP/M SYSGEN.COM program is available, use it to write the MPMLDR.COM file on the first two tracks. If a SYSGEN.COM program is not available, or if SYSGEN.COM does not work because a different media such as a five-inch floppy disk or hard disk is to be used, the user must write two programs: a simple memory loader, called GETSYS, which brings the MP/M loader into memory, and a program called PUTSYS, which places the MPMLDR on the first two tracks of a disk. If you have implemented a CP/M 2 BIOS, you have probably already prepared GETSYS and PUTSYS.

You can use either the SID or DDT debugger instead of writing a GETSYS program. This method is shown in the following example, which also uses SYSGEN in place of PUTSYS. Sample skeletal GETSYS and PUTSYS programs are given in Section 1.5.3.

To load and run the MP/M system automatically, you must also supply a cold start loader that loads the MP/M loader into memory from the first two tracks of the diskette. Modify the CP/M 2 cold start loader in the following manner: change the load address to 0100H and the execution address to 0100H.

The following bootstrap techniques are specific to the Intel MDS-800, which has a boot ROM that loads the first track into location 3000H. However, the steps shown can be applied in a general sense to any custom hardware environment.

1.5.1 Preparing an MP/M II Boot Using SYSGEN

If a SYSGEN program is available, use the following steps to prepare a diskette that cold starts in MP/M II:

1. Prepare the MPMLDR.COM file by integrating your custom LDRBIOS as described in Section 1.2. Test the MPMLDR.COM and verify that it operates properly.

2. Execute either DDT or SID.

```
A>DDT
DDT VERS 2.0
```

3. Using the input command (I), specify that the MPMLDR.HEX file is to be read in then read (R) in the file with an offset of 880H bytes.

```
-IMPMLDR.HEX
-R880
NEXT  PC
2480 0100
```

4. Using the I command, specify that the BOOT.HEX file is to be read in and then read in the file with an offset that loads the boot into memory at 900H. You can use the H command to calculate the offset.

```
-H900 3000
3900 D900

-IBOOT.HEX
-RD900
NEXT  PC
2480 0000
```

5. Return to the CP/M console command processor (CCP) by jumping to location zero.

```
-G0
```

6. Use the SYSGEN program to write the new cold start loader onto the first two tracks of the diskette.

```
A>SYSGEN
SYSGEN VER 2.0
SOURCE DRIVE NAME (OR RETURN TO SKIP)<cr>
DESTINATION DRIVE NAME (OR RETURN TO REBOOT)B
DESTINATION ON B, THEN TYPE RETURN<cr>
FUNCTION COMPLETE
```

1.5.2 Custom Generation of an MP/M II Boot

If a SYSGEN program is not available, then use the following steps to prepare a diskette that cold starts MP/M II:

1. Write a GETSYS program that reads the custom MPMLDR.COM file into location 3380H and the cold start loader (or boot program) into location 3300H. Code GETSYS so that it starts at location 100H (base of the TPA).

   Or, as in the previous example, you can use either SID or DDT to perform this function instead of writing a GETSYS program.

2. Run the GETSYS program using an initialized MP/M II diskette to see if GETSYS loads the MP/M loader starting at 3380H (the operating system actually starts 128 bytes later at 3400H).

3. Write a PUTSYS program that writes memory starting at 3380H back onto the first two tracks of the diskette. The PUTSYS program should be located at 200H.

4. Test the PUTSYS program using a blank, uninitialized diskette by writing a portion of memory to the first two tracks; clear memory and read it back. Test PUTSYS completely, because you will use this program to alter the MP/M II system diskette.

5. Use PUTSYS to place the MP/M II loader and cold start loader onto the first two tracks of a blank diskette.

1.5.3 Sample GETSYS and PUTSYS Programs

The following programs provide a framework for the GETSYS and PUTSYS program. You must insert WRITESEC subroutines to write the specific sectors.

```
;    GETSYS PROGRAM - READ TRACKS 0 AND 1 TO MEMORY AT 3380H
;    REGISTER                USE
;       A                (SCRATCH REGISTER)
;       B                TRACK COUNT (0, 1)
;       C                SECTOR COUNT (1,2,...,26)
;       DE               (SCRATCH REGISTER PAIR)
;       HL               LOAD ADDRESS
;       SP               SET TO STACK ADDRESS
;
START:  LXI    SP,3380H      ;SET STACK POINTER TO SCRATCH AREA
        LXI    H, 3380H      ;SET BASE LOAD ADDRESS
        MVI    B, 0          ;START WITH TRACK 0
RDTRK:                       ;READ NEXT TRACK (INITIALLY 0)
        MVI    C,1           ;READ STARTING WITH SECTOR 1
RDSEC:                       ;READ NEXT SECTOR
        CALL   READSEC       ;USER-SUPPLIED SUBROUTINE
        LXI    D,128         ;MOVE LOAD ADDRESS TO NEXT 1/2 PAGE
        DAD    D             ;HL = HL + 128
        INR    C             ;SECTOR = SECTOR + 1
        MOV    A,C           ;CHECK FOR END OF TRACK
        CPI    27
        JC    RDSEC          ;CARRY GENERATED IF SECTOR < 27
;
;    ARRIVE HERE AT END OF TRACK, MOVE TO NEXT TRACK
        INR    B
        MOV    A,B           ;TEST FOR LAST TRACK
        CPI    2
        JC    RDTRK          ;CARRY GENERATED IF TRACK < 2
;
;    ARRIVE HERE AT END OF LOAD, HALT FOR NOW
        HLT
;
;    USER-SUPPLIED SUBROUTINE TO READ THE DISK
READSEC:
;    ENTER WITH TRACK NUMBER IN REGISTER B,
;          SECTOR NUMBER IN REGISTER C, AND
;          ADDRESS TO FILL IN HL
;
        PUSH    B            ;SAVE B AND C REGISTERS
        PUSH    H            ;SAVE HL REGISTERS
        ........................................
        perform disk read at this point, branch to
         label START if an error occurs
        ........................................
        POP     H            ;RECOVER HL
        POP     B            ;RECOVER B AND C REGISTERS
        RET                  ;BACK TO MAIN PROGRAM
        END     START
```

```
;   PUTSYS PROGRAM - WRITE TRACKS 0 AND 1 FROM MEMORY AT 3380H
;   REGISTER                 USE
;      A             (SCRATCH REGISTER)
;      B             TRACK COUNT (0, 1)
;      C             SECTOR COUNT (1,2,...,26)
;      DE            (SCRATCH REGISTER PAIR)
;      HL            LOAD ADDRESS
;      SP            SET TO STACK ADDRESS
;
START: LXI   SP,3380H    ;SET STACK POINTER TO SCRATCH AREA
       LXI   H, 3380H    ;SET BASE LOAD ADDRESS
       MVI   B, 0        ;START WITH TRACK 0
WRTRK:                   ;WRITE NEXT TRACK (INITIALLY 0)
       MVI   C,1         ;WRITE STARTING WITH SECTOR 1
WRSEC:                   ;WRITE NEXT SECTOR
       CALL  WRITESEC    ;USER-SUPPLIED SUBROUTINE
       LXI   D,128       ;MOVE LOAD ADDRESS TO NEXT 1/2 PAGE
       DAD   D           ;HL = HL + 128
       INR   C           ;SECTOR = SECTOR + 1
       MOV   A,C         ;CHECK FOR END OF TRACK
       CPI   27
       JC   WRSEC        ;CARRY GENERATED IF SECTOR < 27
;
;   ARRIVE HERE AT END OF TRACK, MOVE TO NEXT TRACK
       INR   B
       MOV   A,B         ;TEST FOR LAST TRACK
       CPI   2
       JC   WRTRK        ;CARRY GENERATED IF TRACK < 2
;
;   ARRIVE HERE AT END OF LOAD, HALT FOR NOW
       HLT
;
;   USER-SUPPLIED SUBROUTINE TO WRITE THE DISK
WRITESEC:
;   ENTER WITH TRACK NUMBER IN REGISTER B,
;         SECTOR NUMBER IN REGISTER C, AND
;         ADDRESS TO FILL IN HL
;
       PUSH   B          ;SAVE B AND C REGISTERS
       PUSH   H          ;SAVE HL REGISTERS
       ..........................................
       perform disk read at this point, branch to
        label START if an error occurs
       ..........................................
       POP    H          ;RECOVER HL
       POP    B          ;RECOVER B AND C REGISTERS
       RET               ;BACK TO MAIN PROGRAM

       END    START
```

1.6 Loading MPM.SYS Without the MPMLDR

The MPM.SYS file is a fully-relocated absolute file that can be moved directly into memory and then executed without the use of the MPMLDR. The format of the MPM.SYS file is in Table 1-1, below.

Table 1-1. MPM.SYS File Format

Record	Contents
1	First 128 bytes of the SYSDAT page
2	Second 128 bytes of the SYSDAT page
3-n	MP/M operating system in reverse order, top down.

The actual base of the SYSDAT page in memory is specified in byte 000 of the SYSDAT page. The rest of MP/M II operating system is to be located directly below the SYSDAT page. In Table 1-1, n represents the number of records. Bytes 120-121 of the SYSDAT page contain the value of n. The execution address of MP/M is specified by the page address given in byte 011 of the SYSDAT page.

MPMLDR could load the MPM.SYS file into memory and then move it to its destination specified in the SYSDAT page (byte 000). Or the user could write a separate custom program to produce a directly loadable memory image from the MPM.SYS file.

1.7 Digital Research Copyright and Trademark

Read your MP/M II Licensing Agreement; it specifies your legal responsibilities when copying the MP/M II system. Place the copyright notice:

Copyright © 1981 Digital Research

on the label of each copy you make of your customized MP/M II diskette. Digital Research also requests that you place your MP/M II serial number on the label of any copies you make. Remember also that MP/M II is a trademark of Digital Research, and the first time it appears on a disk label or in a document, it should be followed by a trademark symbol, as shown below:

MP/M II™

1.8 Disk Organization

This section describes MP/M II sector allocation for a system in which the MPMLDR resides on the first two tracks of a single density diskette. The first sector (see Table 1-2) contains an optional software boot section. Disk controllers are often set up to bring track 0, sector 1 into memory at a specific location, often location 0000H. The program in this sector, called BOOT, is responsible for bringing the remaining sectors into memory starting at location 0100H. If your controller does not have a built-in sector load, you can ignore the program in track 0, sector 1, and begin the load from track 0 sector 2 to location 0100H.

As an example, the Intel MDS-800 hardware cold start loader brings track 0, sector 1 into absolute address 3000H. When this sector is loaded, control transfers to location 3000H, where the bootstrap operation commences by loading the remainder of track 0, and all of track 1 into memory, starting at 0100H. Remember that this bootstrap loader is of little use in a non-MDS environment, but it is useful to examine it because you will have to duplicate some of its actions in your own cold start loader.

Table 1-2. MP/M II Sample Disk Organization

Track#	Sector#	Page#	Memory Address	MP/M Module name
00	01		(boot address)	Cold Start Loader
00	02	00	0100H	MPMLDR
"	03	"	0180H	"
"	04	01	0200H	"
"	05	"	0280H	"
"	06	02	0300H	"
"	07	"	0380H	"
"	08	03	0400H	"
"	09	"	0480H	"
"	10	04	0500H	"
"	11	"	0580H	"
"	12	05	0600H	"
"	13	"	0680H	"
"	14	06	0700H	"
"	15	"	0780H	"
"	16	07	0800H	"
"	17	"	0880H	"
"	18	08	0900H	"
"	19	"	0980H	"
"	20	09	0A00H	"
"	21	"	0A80H	"
"	22	10	0B00H	"
"	23	"	0B80H	"
"	24	11	0C00H	"
00	25	"	0C80H	MPMLDR
00	26	12	0D00H	LDRBDOS
01	01	"	0D80H	"
"	02	13	0E00H	"
"	03	"	0E80H	"
"	04	14	0F00H	"
"	05	"	0F80H	"
"	06	15	1000H	"
"	07	"	1080H	"
"	08	16	1100H	"
"	09	"	1180H	"
"	10	17	1200H	"
"	11	"	1280H	"
"	12	18	1300H	"
"	13	"	1380H	"
"	14	19	1400H	"
"	15	"	1480H	"
"	16	20	1500H	"
"	17	"	1580H	"
"	18	21	1600H	"
01	19	"	1680H	LDRBDOS
01	20	22	1700H	LDRBIOS
"	21	"	1780H	"
"	22	23	1800H	"
"	23	"	1880H	"
"	24	24	1900H	"
"	25	"	1980H	"
01	26	25	1A00H	LDRBIOS

SECTION 2

MP/M II BIOS

2.1 MP/M II BIOS Overview

The MP/M II BDOS and XDOS access peripheral devices as "logical" devices within the BIOS and XIOS. To customize MP/M II for a specific hardware environment, the system implementor must prepare the BIOS and XIOS subroutines upon which the BDOS and XDOS depend. This section describes how the logical portions of MP/M II expect to interact with the BIOS; Section 3 describes the same for the XIOS.

The BDOS and XDOS call BIOS subroutines through a "jump vector" located at the base of the BIOS as shown below and in Appendixes D and E. The jump vector is a sequence of 26 jump instructions that send program control to the individual BIOS subroutines. All subroutines must be represented in the jump vector during MP/M II system regeneration. However, certain subroutines may be "empty", that is, they may contain only a single RET instruction.

The BIOS jump vector must take the form shown below. The individual jump addresses for each entry point are listed to the left. Note that the XIOS entry points immediately follow the last BIOS entry point.

```
BIOS+00H    JMP COMMONBASE   ; COMMONBASE, TERMINATE PROCESS
BIOS+03H    JMP WBOOT        ; WARM BOOT, TERMINATE PROCESS
BIOS+06H    JMP CONST        ; CHECK FOR CONSOLE CHAR READY
BIOS+09H    JMP CONIN        ; READ CONSOLE CHARACTER IN
BIOS+0CH    JMP CONOUT       ; WRITE CONSOLE CHARACTER OUT
BIOS+0FH    JMP LIST         ; WRITE LIST CHARACTER OUT
BIOS+12H    JMP PUNCH        ; not used by MP/M II
BIOS+15H    JMP READER       ; not used by MP/M II
BIOS+18H    JMP HOME         ; MOVE TO TRACK 00
BIOS+1BH    JMP SELDSK       ; SELECT DISK DRIVE
BIOS+1EH    JMP SETTRK       ; SET TRACK NUMBER
BIOS+21H    JMP SETSEC       ; SET SECTOR NUMBER
BIOS+24H    JMP SETDMA       ; SET DMA ADDRESS
BIOS+27H    JMP READ         ; READ SELECTED SECTOR
BIOS+2AH    JMP WRITE        ; WRITE SELECTED SECTOR
BIOS+2DH    JMP LISTST       ; not used by MP/M II
BIOS+30H    JMP SECTRAN      ; SECTOR TRANSLATE SUBROUTINE
```

Each jump address corresponds to a particular subroutine that performs a specific function, as outlined in Section 2.3. Three major functions are performed by calls to the jump table: process termination from COMMONBASE and WBOOT; simple character I/O from CONST, CONIN, CONOUT, and LIST; and disk I/O from HOME, SELDSK, SETTRK, SETSEC, SETDMA, READ, WRITE, and SECTRAN.

All simple character I/O operations are assumed to be performed in ASCII, upper and lower case, with high-order (parity) bit set to zero. The BDOS depends on only the CONST, CONIN, and CONOUT subroutines for simple character I/O. An ASCII ^Z (1AH) is interpreted as an end-of-file condition for an input device.

2.2 BIOS Device Characteristics and Entry Points

The BIOS generally supports three types of devices: consoles, list devices and disks. The characteristics of each device are described below:

Consoles are the principal interactive devices that communicate with operators, and are accessed through CONST, CONIN, and CONOUT. Typically, consoles are devices such as CRTs of teletypes. MP/M II supports up to 16 consoles or character I/O devices.

List Devices, if they exist on your system, are usually hard-copy devices, such as printers or teletypes. MP/M II supports up to 16 list devices.

Disks are accessed through a sequence of calls on the various disk I/O subroutines. These subroutines set up the disk number to access, the track and sector on a particular disk, and the direct memory access (DMA) address involved in the I/O operation. After all these parameters have been set up, a call is made to the READ or WRITE function to perform the actual I/O operation. Note that there is often a single call to SELDSK to select a disk drive, followed by a number of read or write operations to the selected disk before selecting another drive for subsequent operations. Similarly, there may be a single call to set the DMA address, followed by several calls which read or write from the selected DMA address before the DMA address is changed. The track and sector subroutines are always called before the READ or WRITE operations are performed.

Note that the READ and WRITE routines should perform several retries (10 is standard) before reporting an error condition to the BDOS. If the error condition is returned to the BDOS, it reports the error to the user. The HOME subroutine may or may not actually perform the track 00 seek, depending upon your controller characteristics; the important point is that track 00 has been selected for the next operation, and is often treated in exactly the same manner as SETTRK with a parameter of 00.

Table 2-1 outlines the exact responsibilities of each subroutine entered through the BIOS jump table.

Table 2-1. BIOS Subroutine Summary

Subroutine	Description
COMMONBASE	The COMMONBASE entry point establishes the base address of the portion of the XIOS that must reside in common memory. The COMMONBASE entry point also contains a jump vector that enables the XIOS to access user and system memory bank switching subroutines, the MP/M II dispatcher, the XDOS and BDOS, the SYSDAT page, and COLDSTART. The effect of a call to COMMONBASE is to terminate the calling program. Other external procedures accessed by COMMONBASE are described in Section 2.4.
WBOOT	The WBOOT subroutine performs an XDOS terminate process call, terminating the calling process. The subroutine must be re-entrant and this entry point must be above the COMMONBASE label.
CONST	The CONST subroutine obtains the status of the console device specified by register D and returns 0FFH in register A if a character is ready to read, or 00H in register A if no console characters are ready. This subroutine must be re-entrant and this entry point must be above the COMMONBASE label.
CONIN	The CONIN subroutine reads the next character from the console device specified by register D into register A, and sets the parity bit (high-order bit) to zero. If no console character is ready, CONIN waits until a character is typed before returning. This subroutine must be re-entrant and this entry point must be above the COMMONBASE label.

Table 2-1. (continued)

Subroutine	Description
CONOUT	The CONOUT subroutine sends the character from register C to the console output device specified by register D. The character is in ASCII, with high-order parity bit set to zero. You may want to include a delay on a line feed or carriage return if your console device requires some time interval at the end of the line (such as a TI Silent 700 terminal). You can, if you wish, filter out control characters that cause your console device to react in a strange way. For example, a ^Z causes the Lear-Seigler terminal to clear the screen, and could be filtered out by CONOUT. This subroutine must be re-entrant and this entry point must be above the COMMONBASE label.
LIST	The LIST subroutine sends the character from register C to the list output device specified by register D. The character is in ASCII with zero parity. This subroutine must be re-entrant and this entry point must be above the COMMONBASE label.
PUNCH	The punch device is not implemented under MP/M II. The transfer vector position is preserved to maintain CP/M compatibility. Note that MP/M II supports up to 16 character I/O devices, any of which can be a reader/punch.
READER	The reader device is not implemented under MP/M II. See the note above for PUNCH.
HOME	The HOME subroutine returns the disk head of the currently-selected disk to the track 00 position. If your controller allows access to the track 0 flag from the drive, step the head until the track 0 flag is detected. If your controller does not support this feature, you can translate the HOME call into a call on SETTRK with a parameter of 0.

Table 2-1. (continued)

Subroutine Description

SELDSK

The SELDSK subroutine selects the disk drive given by register C for further operations, where register C contains 0 for drive A, 1 for drive B, and so up to 15 for drive P. On each disk select, SELDSK must return in HL the base address of a 16-byte area, called the Disk Parameter Header, described in Section 2.3. For standard floppy disk drives, the contents of the header and associated tables does not change, and thus the program segment included in the sample XIOS performs this operation automatically. If there is an attempt to select a non-existent drive, SELDSK returns HL=0000H as an error indicator.

On entry to SELDSK, it is possible to determine whether it is the first time the specified disk has been selected. Register E, bit 0 (least significant bit) is a zero if the drive has not been previously selected. This information is of interest in systems that read configuration information from the disk to set up a dynamic disk definition table.

Although SELDSK must return the header address on each call, it is advisable to postpone the actual physical disk select operation until an I/O function (read or write) is actually performed. This is because disk selects often occur without ultimately performing any disk I/O, and many controllers unload the head of the current disk before selecting the new drive. This unloading can cause an excessive amount of noise and disk wear.

The first SELDSK subroutine call that MP/M II makes is only for getting the DIRBUF address and need not perform any actual I/O.

Table 2-1. (continued)

Subroutine	Description
SETTRK	For the SETTRK subroutine, register BC contains the track number for subsequent disk accesses on the currently selected drive. You can choose to seek the selected track at this time, or delay the seek until the next read or write actually occurs. Register BC can take on values in the range 0-76 corresponding to valid track numbers for standard floppy disk drives, and 0-65535 for non-standard disk subsystems.
SETSEC	For the SETSEC subroutine, register BC contains the translated sector number for subsequent disk accesses on the currently selected drive (see SECTRAN, below). You can choose to send this information to the controller at this point, or instead delay sector selection until a read or write operation occurs. Register BC can take on values in the range 1-26 corresponding to valid sector numbers for standard floppy disk drives, and 0-65535 for non-standard disk subsystems.
SETDMA	For the SETDMA subroutine, register BC contains the DMA (disk memory access) address for subsequent read or write operations. For example, if B = 00H and C = 80H when SETDMA is called, then all subsequent read operations read their data into 80H through 0FFH, and all subsequent write operations get their data from 80H through 0FFH, until the next call to SETDMA occurs. The initial DMA address is assumed to be 80H (relative to the base of the memory segment from which the call was made). Note that the controller need not actually support direct memory access. If, for example, all data is received and sent through I/O ports, the XIOS you construct can use the 128 byte area starting at the selected DMA address for the memory buffer during subsequent read or write operations. A special case of the SETDMA subroutine occurs when the passed parameter in register BC contains a 0FFFFH. This parameter indicates that the blocking buffer, if it exists, must be flushed.

Table 2-1. (continued)

Subroutine Description

Thus, a call to the SETDMA subroutine is interpreted as a flush buffer call when a parameter of 0FFFFH is passed. The BDOS function to flush buffers is translated to this form of a SETDMA subroutine call. If the flush buffer operation performed as a result of the 0FFFFH parameter is successful a simple return should be executed. However, if a disk error occurs, the current return address should be popped from the stack and one of the following error codes should be returned in the register A:

1 non-recoverable error condition occurred
2 disk read/only

READ Assuming the drive has been selected, the track has been set, the sector has been set, and the DMA address has been specified, the READ subroutine attempts to read one sector based upon these parameters, and returns the following error codes in register A:

0 no error occurred
1 non-recoverable error condition occurred

If the value in register A is 0, then MP/M II assumes that the disk operation was completed properly. If an error occurs, however, the XIOS should attempt at least 10 retries to see if the error is recoverable. When an error is reported, the BDOS prints the message "BDOS ERR ON x: BAD SECTOR". Then, depending on the error mode of the calling process, the calling process is terminated or returned an error code.

An additional parameter containing the absolute record number for the disk read is now passed by MP/M II on entry to the READ subroutine. The parameter is three bytes in length, with the high-order byte in register B and the low-order two bytes in register DE. This parameter may be useful in blocking/deblocking algorithms.

Table 2-1. (continued)

Subroutine Description

The BNKXIOS of MP/M II allows portions of the XIOS to reside in bank-switched memory (non-common). This reduces the common memory requirements. The XIOS code for all the disk operations including READ and WRITE can reside in non-common memory with one exception: the code that actually performs the transfer of data into the DMA address must reside in common memory. Two additional entry points within the XIOS, name SWTUSER and SWTSYS, enable switching between the user's memory bank and the system bank containing the BNKXIOS. SWTUSER and SWTSYS are described in Section 2.4.

If you perform deblocking in your READ and WRITE code, you must choose whether to place your deblocking buffer in common memory and then perform a single move into the user's DMA, or to place your deblocking buffer in non-common memory. If you choose the latter, you must then perform an extra move to first move the sector into common memory and then another move into the user's DMA. Blocking and deblocking are discussed in Section 2.5.

WRITE

The WRITE subroutine writes the data from the currently selected DMA address to the currently selected drive, track, and sector. The data should be marked as "non deleted data" to maintain compatibility with other CP/M nd MP/M systems. WRITE returns the following error codes in register A, as shown below:

0 no error occurred
1 non-recoverable error condition occurred
2 disk read/only

If the value in register A is 0, then MP/M II assumes that the disk operation completed properly. If an error occurs, however, the XIOS should attempt at least 10 retries to see if the error is recoverable. When an error is reported, the BDOS prints the message "BDOS ERR ON x: BAD SECTOR". Then, depending on the error mode of the calling process, the calling

Table 2-1. (continued)

Subroutine	Description
	process is terminated or returned an error code. On entry to the WRITE subroutine a parameter is passed in the C register which is intended for use by blocking/deblocking algorithms. This parameter is described in Section 2.5 on blocking/deblocking. An additional parameter containing the absolute record number for the disk write is now passed by MP/M II on entry to the WRITE subroutine. The parameter is three bytes in length, with the high-order byte in register B and the low-order two bytes in register DE. This parameter can be useful in blocking/deblocking algorithms. See the previous section on disk READ for a discussion of placing disk WRITE code in bank-switched memory and deblocking in your WRITE code.
LISTST	The LISTST subroutine returns the ready status of the list device specified by register D. The value 00 is returned in A if the list device is not ready to accept a character, and 0FFH if a character can be sent to the printer. Note that a 00 value always suffices. LISTST must be re-entrant. This entry point is maintained solely for compatibility with CP/M and can generally be omitted from the MP/M II XIOS as none of the standard utilities use this entry point.
SECTRAN	The SECTRAN subroutine performs logical sector to physical sector translation and can improve the overall response of MP/M II. Standard MP/M II systems are shipped with a "skew factor" of 6, where six physical sectors are skipped between each logical read operation. This skew factor allows enough time between sectors for most programs to load their buffers without missing the next sector.

Table 2-1. (continued)

Subroutine	Description
	For computer systems that use fast processors, memory and disk subsystems, you can change the skew factor to improve overall response. Note, however, that you should maintain a single-density IBM-compatible version of MP/M II for information transfer into and out of your computer system, using a skew factor of 6. In general, SECTRAN receives a logical sector number in BC and a translate table address in DE. SECTRAN uses the sector number as an index into the translate table, and returns the resulting physical sector number in HL. For standard systems, the tables and indexing code are provided in the XIOS and need not be changed.

2.3 BIOS Disk Definition Tables

This section presents the organization and construction of tables within the BIOS that define the characteristics of a particular disk system used with MP/M II. These tables can be either hand-coded or automatically generated using the DISKDEF utility provided with MP/M II. The elements of these tables are presented below.

2.3.1 Disk Parameter Table Format

In general, each disk drive has an associated (16-byte) Disk Parameter Header which both contains information about the disk drive and provides a scratchpad area for certain BDOS operations. The format of the Disk Parameter Header for each drive is shown below.

Disk Parameter Header

XLT	0000	0000	0000	DIRBUF	DPB	CSV	ALV
16b	16b	16b	16b	16b	16b	16b	16b

Each element is a word (16-bit) value. The meaning of eah Disk Parameter Header (DPH) element is given in Table 2-2.

Table 2-2. Disk Parameter Header Elements

Element	Description
XLT	Offset of the logical to physical translation vector, if used for this particular drive, or the value 0000H if no sector translation takes place (i.e., the physical and logical sector numbers are the same). Disk drives with identical sector skew factors share the same translate tables.
0000	Scratchpad values for use within the BDOS (initial value is unimportant).
DIRBUF	Offset of a 128 byte scratchpad area for directory operations within BDOS. All DPHs address the same scratchpad area. The same DIRBUF is used by all drives.
DPB	Offset of a disk parameter block for this drive. Drives with identical disk characteristics address the same disk parameter block.
CSV	Offset of a scratchpad area used for software check for changed disks. This offset is different for each DPH.
ALV	Offset of a scratchpad area used by the BDOS to keep disk storage allocation information. This offset is different for each DPH.

Given n disk drives, the DPHs are arranged in a table whose first row of 16 bytes corresponds to drive 0, with the last row corresponding to drive n-1. The table thus appears as:

```
DPBASE

00   XLT 00   0000   0000   0000  DIRBUF DBP 00 CSV 00 ALV 00
01   XLT 01   0000   0000   0000  DIRBUF DBP 01 CSV 01 ALV 01
                      .                         .
                      .                         .
                      .                         .
n-1  XLTn-1   0000   0000   0000  DIRBUF DBPn-1 CSVn-1 ALVn-1
```

where the label DPBASE defines the offset of the DPH table relative to the beginning of the operating system.

A responsibility of the SELDSK subroutine, defined in the previous section, is to return the offset of the DPH from the beginning of the operating system for the selected drive. The following sequence of operations returns the table offset, with a 0000H returned if the selected drive does not exist.

```
NDISKS   EQU    4   ;NUMBER OF DISK DRIVES
......
SELDSK:
         ;SELECT DISK N GIVEN BY C
         LXI    H,0000H     ;READY FOR ERR
         MOV    A,C
         CPI    NDISKS      ;N BEYOND MAX DISKS?
         RNC                ;RETURN IF SO
                            ;0 <= N < NDISKS
         MOV    L,C
         DAD    H           ;READY FOR * 16
         DAD    H
         DAD    H
         DAD    H
         LXI    D,DPBASE
         DAD    D           ;DPBASE + N * 16
         RET
```

The translation vectors (XLT 00 through XLTn-1) are located elsewhere in the BIOS, and simply correspond one-for-one with the logical sector numbers zero through the sector count-1. The Disk Parameter Block (DPB) for each drive is more complex. A particular DPB, which is addressed by one or more DPHs, takes the general form:

SPT	BSH	BLM	EXM	DSM	DRM	AL0	AL1	CKS	OFF
16b	8b	8b	8b	16b	16b	8b	8b	16b	16b

where each is a byte or word value, as shown by the "8b" or "16b" indicator below the field. The fields are defined in Table 2-3.

Table 2-3. Disk Parameter Block Fields

Field	Definition
SPT	is the total number of sectors per track.
BSH	is the data allocation block shift factor, determined by the data block allocation size.
BLM	is the block mask which is also determined by the data block allocation size.
EXM	is the extent mask, determined by the data block allocation size and the number of disk blocks.
DSM	determines the total storage capacity of the disk drive.
DRM	determines the total number of directory entries which can be stored on this drive.
AL0,AL1	determine reserved directory blocks.
CKS	is the size of the directory check vector, a CKS of 8000H marks the drive as permanent with no directory records checked.
OFF	is the number of reserved tracks at the beginning of the (logical) disk.

Although these table values are produced automatically by DISKDEF, it is worthwhile reviewing the derivation of each field so that the values may be cross-checked when necessary. The values of BSH and BLM determine (implicitly) the data allocation size BLS, which is not an entry in the disk parameter block. Given that you have selected a value for BLS, the values of BSH and BLM are shown in Table 2-4 below, where all values are in decimal.

Table 2-4. BSH and BLM Values for Selected BLS

BLS	BSH	BLM
1,024	3	7
2,048	4	15
4,096	5	31
8,192	6	63
16,384	7	127

The value of EXM depends upon both the BLS and whether the DSM value is less than 256 or greater than 255, as shown in the following table.

Table 2-5. Maximum EXM Values

BLS	DSM < 256	DSM > 255
1,024	0	N/A
2,048	1	0
4,096	3	1
8,192	7	3
16,384	15	7

The value of DSM is the maximum data block number supported by this particular drive, measured in BLS units. The product BLS times (DSM+1) is the total number of bytes held by the drive and, of course, must be within the capacity of the physical disk, not counting the reserved operating system tracks.

The DRM entry is one lss than the total number of directory entries, which can take on a 16-bit value. The values of AL0 and AL1, however, are determined by DRM. The two values AL0 and AL1 cn together be considered a string of 16-bits, as shown below.

```
              AL0                            AL1

        00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
```

where position 00 corresponds to the high-order bit of the byte labeled AL0, and 15 corresponds to the low-order bit of the byte labeled AL1. Each bit position reserves a data block for a number of directory entries, thus allowing a total of 16 data blocks to be assigned for directory entries (bits are assigned starting at 00 and filled to the right until position 15). Each directory entry occupies 32 bytes, as shown in Table 2-6.

Table 2-6. BLS and Number of Directory Entries

BLS	Directory Entries
1,024	32 times # bits
2,048	64 times # bits
4,096	128 times # bits
8,192	256 times # bits
16,384	512 times # bits

Thus, if DRM = 127 (128 directory entries), and BLS = 1024, then there are 32 directory entries per block, requiring 4 reserved blocks. In this case, the 4 high-order bits of AL0 are set, resulting in the values AL0 = 0F0H and AL1 = 00H.

The CKS value is determined as follows: if the disk drive media is removable, then CKS = (DRM+1)/4, where DRM is the last directory entry number. If the media is fixed, then set CKS = 8000H (no directory records are checked in this case and drive marked as permanent).

Finally, the OFF field determines the number of tracks which are skipped at the beginning of the physical disk. This value is automatically added whenever SETTRK is called, and can be used as a mechanism for skipping reserved operating system tracks, or for partitioning a large disk into smaller segmented sections.

To complete the discussion of the DPB, recall that several DPHs can address the same DPB if their drive characteristics are identical. Further, the DPB can be dynamically changed when a new drive is addressed by simply changing the pointer in the DPH since the BDOS copies the DPB values to a local area whenever the SELDSK function is invoked.

Returning back to the DPH for a particular drive, note that the two address values CSV and ALV remain. Both addresses reference an area of uninitialized memory following the BIOS. The areas must be unique for each drive, and the size of each area is determined by the values in the DPB.

The size of the area addressed by CSV is CKS bytes, which is sufficient to hold the directory check information for this particular drive. If CKS = (DRM+1)/4, then you must reserve (DRM+1)/4 bytes for directory check use. If CKS = 0, indicating no checked directory entries, or CKS = 8000H, marking the drive as permanent with no checked directory entries, then no storage is reserved.

The size of the area addressed by ALV is determined by the maximum number of data blocks allowed for this particular disk, and is computed as (DSM/8)+1.

2.3.2 The DISKDEF Macro Library

A macro library called DISKDEF greatly simplifies the table construction process. You must have access to the MAC macro assembler or the RMAC relocatable macro assembler distributed with MP/M II to use the DISKDEF facility. The macro library is included with all MP/M II distribution disks.

A BIOS disk definition consists of the following sequence of macro statements:

```
MACLIB   DISKDEF
 .....
DISKS    n
DISKDEF  0,...
DISKDEF  1,...
 .....
DISKDEF  n-1
 .....
ENDEF
```

where the MACLIB statement loads the DISKDEF.LIB file (on the same disk as you BIOS) into MAC's internal tables. The DISKS macro call follows, which specifies the number of drives to be configured with your system, where n is an integer in the range 1 to 16. A series of DISKDEF macro calls then follow, which define the characteristics of each logical disk, 0 through n-1 (corresponding to logical drives A through P). Note that the DISKS and DISKDEF macros generate the in-line fixed data tables described in the previous section, and thus must be placed in a non-executable portion of your BIOS, typically directly following the BIOS jump vector.

The remaining portion of your BIOS is defined following the DISKDEF macros, with the ENDEF macro call immediately preceding the END statement. The ENDEF (End of Diskdef) macro generates the necessary uninitialized RAM areas that are located in memory above your BIOS.

The form of the DISKDEF macro call is

```
DISKDEF  dn,fsc,lsc,[skf],bls,dks,dir,cks,ofs,[k16],[prm]
```

where

dn is the logical disk number, 0 to n-1
fsc is the first physical sector number (0 or 1)
lsc is the last sector number
skf is the option sector skew factor
bls is the data allocation block size
dks is the total number of blocks on the drive
dir is the number of directory entries
cks is the number of "checked" directory entries
ofs is the tract offset to logical track 00
k16 is an optional 1.4 compatibility flag which forces 16K/directory entry
prm is an optional flag which indicates that the drive is permanent (cannot be removed)

The value **dn** is the drive number being defined with this DISKDEF macro invocation. The **fsc** parameter accounts for differing sector numbering systems, and is usually 0 or 1. The **lsc** is the last numbered sector on a track. When present, the **skf** parameter defines the sector skew factor which is used to create a sector translation table according to the skew. If the number of sectors is less than 256, a single-byte table is created, otherwise each translation table element occupies two bytes. No translation table is created if the skf parameter is omitted (or equal to 0).

The **bls** parameter specifies the number of bytes allocated to each data block, and takes on the values 1024, 2048, 4096, 8192 or 16384. Generally, performance increases with larger data block sizes since there are fewer directory references and logically connected data records are physically close on the disk. Also, each directory entry addresses more data, and the BIOS-resident RAM space is reduced. The **dks** specifies the total disk size in **bls** units. That is, if the bls = 2048 and dks = 1000, then the total disk capacity is 2,048,000 bytes. If dks is greater than 255, then the block size parameter bls must be greater than 1024. The value of dir is the total number of directory entries which may exceed 255, is desired.

The **cks** parameter determines the number of directory items to check on each directory scan and is used internally to detect changed disks during system operation. When this situation is detected, MP/M II automatically marks the disk read/only, so that data is not subsequently destroyed. As stated in the previous section, the value of cks equals dir when the media is easily changed, as is the case with a floppy disk subsystem. If the disk is permanently mounted, then the value of cks is typically 0 and thus the **prm** parameter should be included to indicate that the drive is permanent.

The **ofs** value determines the number of tracks to skip when this particular drive is addressed, which can be used to reserve additional operating system space or to simulate several logical drives on a single large-capacity physical drive.

The **k16** parameter is included when file compatibility is required with versions of CP/M 1.4 that have been modified for higher density disks. This parameter ensures that only 16K is allocated for each directory record, as was the case for previous versions. Normally, this parameter is left null. Finally, the prm parameter can be used to indicate that the drive is permanent. This parameter should only be included if the disk media cannot be removed from the drive.

For convenience and economy of table space, the special form

```
        DISKDEF     i,j
```

gives disk i the same characteristics as the previously defined drive j. A standard four-drive single density system, which is compatible with CP/M 1.4, is defined using the following macro invocations:

```
DISKS       4
DISKDEF     0,1,26,6,1024,243,64,64,2
DISKDEF     1,0
DISKDEF     2,0
DISKDEF     3,0
 ...
ENDEF
```

with all disks having the same parameter values of 26 sectors per track (numbered 1 through 26), with 6 sectors skipped between each access, 1024 bytes per data block, 243 data blocks for a total of 243k byte disk capacity, 64 checked directory entries, and two operating system tracks.

The DISKS macro generates n Disk Parameter Headers (DPHs), starting at the DPH table address DPBASE generated by the macro. Each disk header block contains sixteen bytes, as described above, and corresponds one-for-one to each of the defined drives. In the four drive standard system, for example, the DISKS macro generates a table of the form:

```
DPBASE   EQU   $
DPE0:    DW    XLTO,0000H,0000H,0000H,DIRBUF,DPB0,CSV0,ALV0
DPE1:    DW    XLTO,0000H,0000H,0000H,DIRBUF,DPB0,CSV1,ALV1
DPE2:    DW    XLTO,0000H,0000H,0000H,DIRBUF,DPB0,CSV2,ALV2
DPE3:    DW    XLTO,0000H,0000H,0000H,DIRBUF,DPB0,CSV3,ALV3
```

where the DPH labels are included for reference purposes to show the beginning table addresses for each drive, 0 through 3. The values contained within the disk parameter header are described in detail in the previous section. The check and allocation vector addresses are generated by the ENDEF macro in the RAM area following the BIOS code and tables.

Note that if the skf (skew factor) parameter is omitted (or equal to 0), the translation table is omitted, and a 0000H value is inserted in the XLT position of the disk parameter header for the disk. In a subsequent call to perform the logical to physical translation, SECTRAN receives a translation table address of DE = 0000H, and simply returns the original logical sector from BC in the HL register pair. A translate table is constructed when the skf parameter is present, and the (non-zero) table address is placed into the corresponding DPHs. The table shown below, for example, is constructed when the standard skew factor skf = 6 is specified in the DISKDEF macro call:

```
XLT0:  DB   1,7,13,19,25,5,11,17,23,3,9,15,21
       DB   2,8,14,20,26,6,12,18,24,4,10,16,22
```

Following the ENDEF macro call, a number of uninitialized data area are defined. These data areas need not be a part of the BIOS that is loaded upon cold start, but must be available between the BIOS and the end of memory. The size of the uninitialized RAM area is determined by EQU statements generated by the ENDEF macro. For a standard four-drive system, the ENDEF macro might produce:

```
4C72 =       BEGDAT EQU $
             (data areas)
4DB0 =       ENDDAT EQU $
013C =       DATSIZ EQU $-BEGDAT
```

which indicates that uninitialized RAM begins at location 4C72H, ends at 4DB0H-1, and occupies 013CH bytes. You must ensure that these addresses are free for use after the system is loaded.

After modification, you can use the STAT program to check your drive characteristics, because STAT uses the disk parameter block to decode the drive information. The STAT command form

```
STAT d:DSK:
```

decodes the disk parameter block for drive d (d=A,...,P) and displays the values shown below.

```
r: 128 Byte Record Capacity
k: Kilobyte Drive  Capacity
d: 32  Byte Directory Entries
c: Checked  Directory Entries
e: Records/ Extent
b: Records/ Block
s: Sectors/ Track
t: Reserved Tracks
```

Three examples of DISKDEF macro invocations are shown below with corresponding STAT parameter values. The last example produces an 8-megabyte system.

```
DISKDEF 0,1,58,,2048,256,128,128,2
r=4096, k=512, d=128, c=128, e=256, b=16, s=58, t=2

DISKDEF 0,1,58,,2048,1024,300,0,2
r=16384, k=2048, d=300, c=0, e=128, b=16, s=58, t=2

DISKDEF 0,1,58,,16384,512,128,128,2
r=65536, k=8192, d=128, c=128, e=1024, b=128, s=58, t=2
```

2.4 External Procedure Access

To help the XIOS access other MP/M entry points, a jump vector is dynamically built by the MP/M II GENSYS program and placed at the COMMONBASE subroutine entry point. The dynamic portion of the jump vector contains five entry points that provide access to user and system memory bank switching, the MP/M II dispatcher, the XDOS, and the SYSDAT page. Table 2-7 describes external procedure entry points.

The following example illustrates the code used to access external procedures:

```
COMMONBASE
          JMP     COLDSTART
SWTUSER:  JMP     $-$
SWTSYS:   JMP     $-$
PDISP:    JMP     $-$
XDOS:     JMP     $-$
SYSDAT:   DW      $-$

COLDSTART:
WBOOT:
          MVI     C,0
          JMP     XDOS      ;terminate process
```

Table 2-7. External Procedure Summary

Subroutine	Description
SWTUSER	The SWTUSER entry point restores the bank of the user's calling program. There are no parameters passed or returned. The purpose of SWTUSER is to enable BIOS disk read and write code to transfer data from a disk controller or buffer in common memory to/from the DMA buffer in the user's calling program. This procedure must be called only from common memory, that is above the COMMONBASE label, and it must be used only from BIOS disk functions. Internally the SWTUSER procedure disables and then re-enables interrupts. Thus, if you disable interrupts before calling SWTUSER, they will be enabled on returning from SWTUSER.
SWTSYS	The SWTSYS entry point restores the bank of the BNKBDOS. There are no parameters passed or returned. The purpose of SWTSYS is to restore the bank containing the banked portion of the BDOS following the transfer of data from a disk controller or buffer in common memory to/from the DMA buffer in the user's calling program. This procedure must be called only from common memory. Internally the SWTSYS procedure disables and then re-enables interrupts. Thus, if you disable interrupts before calling SWTSYS, they will be enabled on returning from SWTSYS.
PDISP	The PDISP entry point forces a dispatch call. It is intended to be used at the conclusion of interrupt handling when a process is to be dispatched. It is effectively a null procedure call from the point of view of the calling program.
XDOS	The XDOS entry point provides access to XDOS functions. XDOS functions are required for flag operations, queue operations and polling devices.
SYSDAT	The SYSDAT entry is not a true entry point, but the address of the system data page. Section 4 provides a definition of the system data page.

2.5 Blocking and Deblocking Algorithms

Upon each call to the BIOS WRITE entry point, the BDOS includes information that allows effective sector blocking and deblocking where the host disk subsystem has a sector size which is a multiple of the basic 128-byte unit. This section presents a general-purpose algorithm that can be included within your BIOS that uses the BDOS information to perform the operations automatically.

Upon each call to WRITE, the BDOS provides the following information in register C:

```
0  =  deferred write sector
1  =  non-deferred write sector
2  =  deferred write to the first sector
        of a new data block
3  =  non-deferred write to the first sector
        of a new data block
```

Conditions 0 and 2 occur only for permanent drives and allow deferred writes. Conditions 1 and 3 occur for non-permanent (removable) drives and force immediate (non-deferred) writes. Condition 1 also occurs on permanent drives for writes to the directory.

Condition 2 and 3 occur when a write operation is made to the first sector of a new data block. The blocking/deblocking algorithm does not perform physical record pre-reads if sequential writes ae made to a new data block. In most cases, application programs read or write multiple 128-byte sectors in sequence, and thus there is little overhead involved in either operation when blocking and deblocking records because pre-read operations can be avoided when writing records.

The blocking and deblocking algorithm is listed in Appendix B in skeletal form. The file is included on your MP/M II disk. Generally, the algorithms map all MP/M II sector read operations onto the host disk through an intermediate buffer which is the size of the host disk sector. Throughout the program, values and variables which relate to the sector involved in a seek operation are prefixed by "sek", while those related to the host disk system are prefixed by "hst". The equate statements beginning on line 24 define the mapping between MP/M II and the host system, and must be changed if other than the sample host system is involved.

The SELDSK entry point clears the host buffer flag whenever a new disk is logged-in. Note that although the SELDSK entry point computes and returns the Disk Parameter Header address, it does not physically select the host disk at this point (it is selected later at READHST or WRITEHST). Further, SETTRK, SETSEC, and SETDMA simply store the values, but do not take any other action at this point. SECTRAN performs a trivial function of returning the physical sector number.

The principal entry points are READ and WRITE. These subroutines take the place of your previous READ and WRITE operations.

The actual physical read or write takes place at either WRITEHST or READHST, where all values have been prepared: hstdsk is the host disk number, hsttrk is the host track number, and hstsec is the host sector number (which may require translation to a physical sector number). You must insert code at this point which performs the full host sector read or write into, or out of, the buffer at hstbuf of length hstsiz. All other mapping functions are performed by the algorithms.

2.6 Common Memory Portion of the BNKXIOS

Take care when selecting which XIOS code is to be placed in common memory. This section should give you some helpful guidelines.

In general, all XIOS and BIOS entries (with the exception of the disk I/O entries) must be above the COMMONBASE subroutine entry point. Thus, the BNKXIOS enables you to place your disk drivers in a portion of code that is not in common memory. There are, however, some exceptions that affect both the code and data areas of the disk handlers.

The Disk Parameter Headers and Disk Parameter Blocks must be in common memory.

The DIRBUF data structure, which is referenced by the disk parameter blocks, must reside in common memory.

All disk device polling code and interrupt handlers must reside in common memory.

While it is possible to place a deblocking buffer in non-common memory, it requires a sector buffer in common memory and an extra move of 128 bytes to move the data first into common memory and then into the users DMA buffer. Also, bank switching cannot be permitted while a physical DMA from a disk controller to a deblocking buffer in non-common memory is in operation.

SECTION 3

MP/M II XIOS

3.1 MP/M II XIOS Overview

The Extended Input/Output System (XIOS) must include the hardware dependent code that polls devices, handles interrupts and performs memory management functions.

The MP/M II system implementor must prepare subroutines that perform the functions described in Table 3-1, then place a jump vector containing the XIOS entry points immediately following the BIOS jump vector. Most of the XIOS subroutines need to be re-entrant. The XIOS jump vector must take the following form:

```
BIOS+33H    JMP SELMEMORY     ; SELECT MEMORY
BIOS+36H    JMP POLLDEVICE    ; POLL DEVICE
BIOS+39H    JMP STARTCLOCK    ; START CLOCK
BIOS+3CH    JMP STOPCLOCK     ; STOP CLOCK
BIOS+3FH    JMP EXITREGION    ; EXIT CRITICAL REGION
BIOS+42H    JMP MAXCONSOLE    ; MAXIMUM CONSOLE NUMBER
BIOS+45H    JMP SYSTEMINIT    ; SYSTEM INITIALIZATION
BIOS+48H    JMP IDLE          ; IDL PROCEDURE (Optional)
```

3.2 MP/M XIOS Entry Points

Each jump address corresponds to a particular subroutine that performs the specific function. Table 3-1 outlines the exact responsibilities of each XIOS entry point subroutine.

Table 3-1. XIOS Subroutine Summary

Subroutine	Function
SELMEMORY	The SELMEMORY subroutine identifies the segment of memory where a process is to execute. Each time a process is dispatched for execution, the operating system makes a call to this XIOS select memory procedure. If the hardware environment has memory bank selection/protection, SELMEMORY can use the passed parameter to select/protect areas of memory. The passed parameter (in registers BC) is a pointer to a memory descriptor from which the memory base, size, attributes and bank of the executing process can be determined. Thus, all other regions of memory can be write-protected.

Table 3-1. (continued)

Subroutine	Function
	MP/M II calls SELMEMORY with interrupts disabled from within the dispatcher. The SELMEMORY subroutine must not enable interrupts. This subroutine must reside above the COMMONBASE entry point.
POLLDEVICE	A polled environment can be created by coding XIOS device poll handlers. The purpose of implementing a polled environment is to avoid typical busy-wait code for device operation completion. There are also peripheral devices that may not operate efficiently under interrupts. XDOS calls the device poll handlr (POLLDEVICE) with the device to be polled in the C register as a single parameter. The user-written POLLDEVICE procedure can be coded to access the device polling routines via a table that contains the addresses of the device polling procedures. An association is made between a device number to be polled and the polling procedure itself. The polling procedures must return a value of 0FFH in the accumulator if the device is ready, or 00H if the device is not ready. POLLDEVICE is called from a critical region within the dispatcher; therefore, the POLLDEVICE subroutine must not enable interrupts. This subroutine must reside above the COMMONBASE entry point.
STARTCLOCK	The STARTCLOCK and STOPCLOCK procedures eliminate unnecessary overhead for the system clock interrupt handler. The system clock provides a time base for both the real time flag and the system tick procedure. However, the system tick procedure is needed only when there is a process on the delay list. MP/M II calls STARTCLOCK when a process enters the delay list to initiate the system tick time base (see Section 3.4).

Table 3-1. (continued)

Subroutine	Function
	In some hardware environments, it is not possible to shut off the system time unit clock while maintaining the one-second flag used for keeping time of day. In this situation, the STARTCLOCK procedure simply sets a boolean variable to true, indicating that there is a delayed process. The clock interrupt handler can then determine if system time unit flag is to be set by testing the boolean. This subroutine must reside above the COMMONBASE entry point.
STOPCLOCK	When the system delay list is emptied, MP/M II calls the STOPCLOCK procedure to stop the system tick time base. This eliminates unnecessary overhead for the system clock interrupt handler. In some hardware environments, it is not possible to shut off the system time unit clock while maintaining the one second flag used for keeping time of day; that is, a single clock/timer interrupt source is used. In this situation, the STOPCLOCK procedure simply sets a boolean variable to false, indicating that there are no delayed processes. The clock interrupt handler can then determine if the system time unit flag is to be set by testing the boolean. This subroutine must reside above the COMMONBASE entry point.
EXITREGION	MP/M II calls the EXITREGION procedure to test a local parameter called the PREEMPT flag. If PREEMPT is true, EXITREGION leaves interrupts disabled. If PREEMPT is false, EXITREGION enables interrupts. Interrupt service routines must set the PREEMPT flag true at beginning of the interrupt handling. This procedure allows an interrupt service routine to make a flag set MP/M II system call, leaving interrupts disabled until completion of the interrupt handling. This subroutine must reside above the COMMONBASE entry point.

Table 3-1. (continued)

Subroutine	Function
MAXCONSOLE	The maximum console procedure enables the calling program to determine the number of physical consoles the BIOS is capable of supporting. The number of physical consoles is returned in the A register. This subroutine must reside above the COMMONBASE entry point.
SYSTEMINIT	The system initialization procedure performs the required MP/M cold start initialization. The following is a typical initialization for a banked system: first, MP/M II initializes bank 0, disables interrupts and calls SYSTEMINIT. Then, SYSTEMINIT sets up interrupt jump vectors, interrupt masks, and the base page of each bank before returning to MP/M II. Finally, MP/M II enables interrupts. A typical initialization for a non-banked system would perform the same steps, but only one bank would be initialized. MP/M II disables interrupts and calls the SYSTEMINIT entry point prior to any other XIOS call. As stated above, MP/M II enables interrupts immediately upon return from SYSTEMINIT. This subroutine must reside above the COMMONBASE entry point. In systems with bank switched memory, it is necessary to set up the base page (0000H - 00FFH) within each bank of memory. Both the MPMLDR and MP/M itself assume that the base bank (bank #0) is switched in when the MPMLDR is executed. The base bank is properly initialized by MP/M prior to entering SYSTEMINIT. The information required for the initialization of other banks is provided on entry to SYSTEMINIT in the registers defined below:
C	MP/M debugger restart #
DE	MP/M entry point address for the debugger. Place a jump at the proper debugger restart location to the address contained in DE.

Table 3-1. (continued)

Subroutine	Function
HL	BIOS direct jump table address. Place a jump instruction at location 0000H in each bank's base page to the address contained in HL.
IDLE	An IDLE process is the anchor of the process ready list. The MP/M II nucleus calls the IDLE procedure when there are no other processes ready to run. The normal IDLE procedure is a call to the dispatcher. This most efficiently serves polled devices. If your system is entirely interrupt-driven (i.e. no polled devices), you can supply your own IDLE procedure, which should be as follows: IDLE: HALT RET If you do not supply an IDLE procedure, place three bytes of zero at the BIOS +48H location.

3.3 Interrupt Service Routines

The MP/M II operating system is designed to work with virtually any interrupt architecture, be it flat or vectored. The code operating at the interrupt level saves the required registers, determines the cause of the interrupt, removes the interrupting condition, sets an appropriate flag, and then forces a dispatch to take place.

Be sure to use a minimum number of stack levels when saving the state of the interrupted process. This is because the interrupted application program, especially if it has been written for a CP/M environment, is not likely to provide extra stack area as a contingency for interrupts. The example Extended Input/Output Systems shown in the Appendixes illustrates a technique whereby no additional levels of stack are required beyond that of the interrupt restart itself. This technique is highly recommended.

Operation of the flags is described in Section 3 of the MP/M II Programmer's Guide, under the discussion of the Flag Set and Flag Wait XDOS Functions. Briefly, flags synchronize a process to an asynchronous event. In general, an interrupt service routine sets a particular flag while another process waits for the flag to be set.

At a logical level above the physical interrupts, the flags can be regarded as providing 256 levels of virtual interrupts (32 flags are supported under MP/M II). Thus, logical interrupt handlers wait on flags set by the physical interrupt handlers. This mechanism allows a common XDOS to operate on potentially all 8080, 8085 and Z80® microcomputers, regardless of the hardware environment.

As an example, consider a hardware environment with a flat interrupt structure. That is, a single interrupt level is provided and devices must be polled to determine the cause of the interrupt. Once the interrupt cause is determined, a specific flag is set indicating that that particular interrupt has occurred.

At the conclusion of the interrupt processing, a jump should be made to the MP/M II dispatcher. This is done by jumping to the PDISP entry point. This jump gives the processor resource to the highest priority ready process, usually the process readied by setting the flag in the interrupt handler, and then enables interrupts before jumping to resume execution of that process.

The only XDOS or BDOS call that should be made from an interrupt handler is 133: Flag Set. Any other XDOS or BDOS call results in a dispatch which would then enable interrupts before the execution of the interrupt handler is completed.

It is recommended that interrupts be used only for asynchronous operations such as console input or disk operation complete. In general, operations such as console output should not be interrupt-driven, because the system has more elasticity when performing polled console outputs while idling, rather than incurring the dispatch overhead for each character transmitted. This is particularly true at higher baud rates.

If a system requires the execution of a return from interrupt (RETI) instruction, the interrupt handler must execute the RETI before branching to the dispatcher via the PDISP entry point.

3.4 Time Base Management

The XIOS must provide two time bases: a one second flag for real time and a system tick for managing the delay list. The one second flag operation is logically separate from the system tick operation even though it may physically share the same clock/timer interrupt source. The one second flag procedure sets flag #2 at each one second of real time. MP/M II uses flage #2 to maintain a time of day clock.

The system tick procedure, when enabled by STARTCLOCK, sets flag #1 at system time unit intervals. The recommended time unit is a period of 16.67 milliseconds, corresponding to a tick frequency of 60 Hz. When operating with 50 Hz, use a 20 millisecond period. MP/M II uses the system tick to manage the delay list until the delay list is empty, at which time the system tick procedure is disabled by STOPCLOCK.

The system tick frequency is critical because it determines the dispatch frequency for compute-bound processes. If the frequency is too high, a significant amount of system overhead is incurred by excessive dispatches. If the frequency is too low, compute-bound processes keep the CPU resource for accordingly longer periods.

SECTION 4

MP/M II SYSTEM FILE COMPONENTS

The MP/M II system file, MPM.SYS, consists of a number of components: the system data page, the customized XIOS, the RESBDOS and BNKBDOS, the XDOS and BNKXDOS, the TMP, and the resident system processes. MPM.SYS resides in the directory with a user code of 0 and usually has the Read Only attribute. The MP/M II loader reads the MPM.SYS file into memory to bring up the MP/M II system.

4.1 System Data

The system data page contains 256 bytes used by GENSYS to dynamically configure the MP/M II system. The system data page can be prepared using the GENSYS program or it can be manually prepared using DDT or SID. The Table 4-1 describes the byte assignments.

Table 4-1. System Data Byte Assignments

Byte	Contents
000-000	Mem$top, top page of memory
001-001	Nmb$cns, number of system consoles (TMPs)
002-002	Brkpt$RST, breakpoint RST #
003-003	Add system call user stacks, boolean
004-004	Bank switched, boolean
005-005	Z80 version, boolean
006-006	banked bdos, boolean
007-007	XIOS jump table page
008-008	RESBDOS base page
009-010	CP/NET master configuration table address
011-011	XDOS base page
012-012	RSPs (BNKXIOS top+1) base page
013-013	BNKXIOS base page
014-014	BNKBDOS base page
015-015	Max$mem$seg, max memory segment number
016-047	Initial memory segment table
048-063	Breakpoint vector table, filled in by debuggers
064-079	Reserved for MP/M II
080-095	System call user stack pointer table
096-119	Reserved for MP/M II
120-121	Nmb records in MPM.SYS file
122-122	# ticks/sec
123-123	System Drive
124-124	Common Memory Base Page
125-125	Number of RSPs
126-127	Listcp array Address
128-143	Subflg, submit flag array

Table 4-1. (continued)

Byte	Contents
144-186	Reserved for MP/M II
187-187	Max locked records/process
188-188	Max open files/process
189-190	# list items
191-192	Pointer to base of lock table free space
193-193	Total system locked records
194-194	Total system open files
195-195	Dayfile logging, boolean
196-196	Temporary file drive
197-197	Number of printers
197-241	Reserved for MP/M II
242-242	Banked XDOS base page
243-243	TMP process descriptor base
244-244	Console.dat base
245-246	BDOS/XDOS entry point
247-247	TMP.spr base
248-248	Nmbrsps, number of banked RSPs
249-249	Brsp base address
250-251	Brspl, non-resident rsp process link
252-253	Sysdatadr, XDOS internal data segment address
254-255	Rspl, resident system process link

4.2 Customized XIOS

The customized XIOS is obtained either from a file named RESXIOS.SPR, or a file named BNKXIOS.SPR. The XIOS file of type SPR contains the page relocatable version of the user-customized XIOS. The standard method for the generation of the XIOS is to use the Digital Research LINK program. An alternative method is described in Section 1.

4.3 BDOS

The Basic Disk Operating System (BDOS) resides in two page-relocatable files name the RESBDOS and the BNKBDOS. These two files contain the console, list and disk file management code.

4.3.1 RESBDOS

The file named RESBDOS.SPR is a page relocatable file containing the logical console and list handling, as well as the resident portion of the disk file system that provides an interface to the BNKBDOS.

4.3.2 BNKBDOS

The file named BNKBDOS.SPR is a page relocatable file containing the non-resident portion of the banked BDOS.

4.4 XDOS

The XDOS file named XDOS.SPR is a page-relocatable file containing the priority-driven MP/M II nucleus. The nucleus contains the following code pieces: root module, dispatcher, queue management, flag management, memory management, terminal handler, terminal message process, command line interpreter, file name parser, and time base management.

4.5 Resident System Processes

A file type of RSP identifies a resident system process. The RSP files distributed with MP/M II include: run-time system status display (MPMSTAT), printer spooler (SPOOL), abort named process (ABORT), and a scheduler (SCHED). At system generation time, GENSYS prompts you to select which RSPs to include in the MPM.SYS file.

It is possible for the user to prepare custom resident system processes. The resident system processes must follow these rules:

- The file must be page-relocatable. Page relocatable files can be generated by LINK, or by the submit files MACSPR.SUB or ASMSPR.SUB. The output file must be renamed to type RSP.
- The first two bytes of the resident system process are reserved for the address of the BBOS/XDOS. Thus, a resident system process can access the BDOS/XDOS by loading the two bytes at relative 0000-0001H and then performing a PCHL.
- The process descriptor for the resident system process must begin at the third byte position.

4.6 Banked Resident System Processes

A banked resident system process consists of two parts: a resident portion and the code for the process. The resident portion contains the process descriptor, and queues or other data structures that must be in common memory. This portion follows the rules given above for resident system processes. The presence of a banked portion is specified by setting the process descriptor memory segment index to zero rather than 0FFH. The name provided in the process descriptor is used to obtain the banked portion which has a fil type of BRS.

The second part of a banked system process is the actual cod piece for the process. The rules for the BRS portion are as follows:

- The file must be page relocatable. Page relocatable files can b generated by LINK, or the procedure outlined in Section 1. Th output file must be renamed to type BRS.

- Bytes 0000-0001H of the banked RSP are reserved for the addres of the resident portion of the RSP. Thus, a banked RSP mus access the BDOS/XDOS functions by indirectly loading from the tw bytes at relative 0000-0001H, which point to the base of th resident portion of the RSP, which in turn contain the BDOS/XDO entry point address.

- Bytes 0002-0003H of the banked RSP must contain the initial stac pointer value for the process. Thus, the stack for the banke RSP is in the banked portion of the RSP, and should b initialized such that the return address on top of the stack i the banked RSP entry point address.

- Bytes 0004-000BH of the banked RSP must contain an ASCII name fo the process. This is used for display purposes during GENSYS an MPMLDR execution.

SECTION 5

SYSTEM GENERATION

5.1 GENSYS Operation

MP/M II system generation consists of preparing a system data file and concatenating both required and optional code files to produce a file name MPM.SYS. A GENSYS program reforms these tasks and can be run under either MP/M II or CP/M. The GENSYS automates the system generation process by prompting the user for optional parameters and then prepares the MPM.SYS file. The following sample execution illustrates GENSYS operation.

```
0A>gensys

MP/M-80 V2.0 System Generation
Copyright (C) 1981, Digital Research

Default entries are shown in (parens).
Default base in Hex, precede entry with # for decimal

Use SYSTEM.DAT for defaults (Y) ?
Top page of operating system (FF) ?
Number of TMPs (system consoles) (#2) ?
Number of Printers (#1) ?
Breakpoint RST (06) ?
Add system call user stacks (Y) ?
Z80 CPU (Y) ?
Number of ticks/second (#60) ?
System Disk (E:) ?
Temporary file drive (E:) ?
Maximum locked records/process (#16) ?
Total locked records/system (#32) ?
Maximum open files/process (#16) ?
Total open files/system (#32) ?
Bank switched memory (Y) ?
Number of user memory segments (#3) ?
Common memory base page (C0) ?
Dayfile logging at console (Y) ?

 SYSTEM  DAT  FF00H  0100H
 TMPD    DAT  FE00H  0100H
 USERSYS STK  FD00H  0100H
 XIOSJMP TBL  FC00H  0100H

Accept new system data page entries (Y) ?

 RESBDOS SPR  F000H  0C00H
 XDOS    SPR  CE00H  2200H

Select Resident System Processes:
```

```
SCHED   RSP (N) ?
ABORT   RSP (N) ? Y
SPOOL   RSP (N) ? Y
MPMSTAT RSP (N) ? Y

ABORT   RSP  CD00H  0100H
SPOOL   RSP  CC00H  0100H
MPMSTAT RSP  CB00H  0100H

BNKXIOS SPR  B800H  1300H
BNKBDOS SPR  9500H  2300H
BNKXDOS SPR  9200H  0300H
TMP     SPR  8F00H  0300H

SPOOL   BRS  8700H  0800H
MPMSTAT BRS  7900H  0E00H

LCKLSTS DAT  7700H  0200H
CONSOLE DAT  7500H  0200H

Enter memory segment table:

Base,size,attrib,bank (75,8b,80,00) ?
Base,size,attrib,bank (00,C0,00,01) ?
Base,size,attrib,bank (00,C0,00,02) ?
Base,size,attrib,bank (00,C0,00,03) ? 00,ff,0,0
*** Memory conflict - segment trimmed ***
Base,size,attrib,bank (00,75,00,00) ?

MP/M II Sys  7500H  8B00h  Bank 00
Memseg  Usr  0000H  C000H  Bank 01
Memseg  Usr  0000H  C000H  Bank 02
Memseg  Usr  0000H  7500H  Bank 00

Accept new memory segment table entries (Y) ?

** GENSYS DONE **
```

5.2 System Generation Parameters

This section discusses the issue involved in answering each o the GENSYS queries shown in the example above.

5.2.1 Defaults

The GENSYS program displays default entry values withi parentheses. The base is hex unless a # character precedes the valu to indicate a decimal base. The initial prompt determines if th internal GENSYS defaults are to be used, or those of the most recentl generated SYSTEM.DAT file.

5.2.2 Top Page of Operating System

Enter two hex ASCII digits to give the top page of the operating system. The highest address used by MP/M II is XXFFH, where XX is the entry.

5.2.3 Number of System Consoles

This entry determines the number of system consoles for which Terminal Message Processes (TMP's) are created to generate user prompts and send command lines to the Command Line Interpreter (CLI). A region of common memory called TMPD.DAT is reserved for the TMP process descriptors. Four TMP process descriptors can be placed in each page of the TMPD.DAT. Each system console also requires 256 bytes of memory for stack and buffer areas in a non-resident region of memory called CONSOLE.DAT. MP/M II supports up to a maximum of 16 character I/O console devices, of which 8 can be system consoles and have associated TMPs. During MP/M II initialization, an XIOS call obtains the actual maximum number of physical consoles supported by the XIOS. This number is used if it is less than the number specified during the GENSYS.

5.2.4 Number of Printers

This entry determines the number of physical printers which the XIOS is capable of supporting. This number is used by the MPMSTAT program when it displays the status of the system printers.

5.2.5 Breakpoint RST

Enter the breakpoint restart number to be used by the MP/M debuggers. Recommended restarts are RST #1 to RST #6.

5.2.6 System Call User Stacks

If you want to execute CP/M *.COM files, enter yes. An affirmative response forces a stack switch to occur when system calls are made rom a user program. BDOS calls require more stack space under MP/M II than under CP/M. An affirmative response causes GENSYS to allocate a region of common memory called USERSYS.STK. The size of this region is determined by the number of user memory segments, where 0-3 segments require 100h bytes and 4-7 segments require 200h bytes.

Note that this affects BDOS calls only, not XDOS calls. The XDOS is re-entrant and performs no stack switching. Therefore, if your program makes any XDOS calls, you need to make certain that you have allocated sufficient stack.

5.2.7 Z80 CPU

An affirmative response should only be made if you do have a Z80 CPU. If specified, the MP/M II dispatcher saves and restores the Z80 alternate register set.

5.2.8 Number of Ticks / Second

This entry value can be used by applications programs to determine the number of ticks per second. This value may vary among MP/M II systems.

5.2.9 System Disk

The drive entered here is used for a second search if the file requested to the CLI is not found on the default drive.

5.2.10 Temporary File Drive

The drive entered here is used as the drive for temporary disk files. This entry is used by SUBMIT when it generates the n.SUB temporary file. This entry can also be accessed in the system data page by application programs as the drive on which to create temporary files.

5.2.11 Maximum Locked Records / Process

This entry specifies the maximum number of records that a single process (usually one program) can lock at any given time. This number can range from 0 to 255 and must be less than or equal to the total locked records for the system.

5.2.12 Total Locked Records / System

This entry specifies the total number of locked records for all the processes executing under MP/M II at any given time. This number can range from 0 to 255 and should be greater than or equal to the maximum locked records per process.

It is possible to allow each process to either use up the total system lock record space, or to allow each process to lock only a fraction of the system total. The first technique implies a dynamic storage region in which one process can force other processes to block because it has consumed all available resources.

5.2.13 Maximum Open Files / Process

This entry specifies the maximum number of files that a single process (usually one program) can open at any given time. This number can range from 0 to 255 and must be less than or equal to the total open files for the system.

5.2.14 Total Open Files / System

This entry specifies the total number of open files for all the processes executing under MP/M II at any given time. This number can range from 0 to 255 and should be greater than or equal to the maximum open files per process.

It is possible either to allow each process to use up the total system open file space, or to allow each process to only open a fraction of the system total. The first technique implies a dynamic storage region in which one process can force other processes to block because it has consumed all available resources.

5.2.15 Bank Switched Memory

If your system does not have bank-switched memory, then you should respond with an "N". Otherwise, respond with a "Y" and additional questions and responses (as shown in Section 5.2.2) are required.

5.2.16 Number of User Memory Segments

The number of user memory segments must be in the range 1 to 7 and should be greater than or equal to the number of system consoles.

5.2.17 Common Memory Base Page

In response to this prompt, enter the address of the lowest page of memory common to all banks. GENSYS checks that all modules requiring residence in common memory are located above this address.

5.2.18 Dayfile Logging at Console

An affirmative response causes the generated MP/M II system to display the current time, file name and type, and user number of each executed command file.

5.2.19 Accept System Data Page Entries

If the entries made for the first 16 queries are acceptable, then enter yes. Otherwise, any or all of the entries made can be changed by re-cycling through the GENSYS queries, entering a carriage return where values are not to be changed.

5.2.20 Select Resident System Processes

GENSYS searches the directory for all files of type RSP. Each file found is listed and included in the generated system file if you respond with a "Y". Tests are performed to make certain that the specified RSPs reside at or above the common base address.

5.2.21 Memory Segment Table

Memory segmentation is defined by the entries which are made. You are prompted for the base, size, attributes, and bank for each memory segment. The GENSYS program only allows you to enter the number of segments specified in the response to the query regarding the number of user memory segments.

The first default entry made is for the operating system. This becomes the segment zero entry in the memory segment table. It is switched in during the baked MP/M II execution of the BNKXIOS, BRS's, and the BNKBDOS. The first entry is not counted in your number of user memory segments.

A significant amount of error checking is performed using a memory bit map to ensure that no memory segments overlap each other. It will be possible to customize the GENSYS program such that non-existent memory for a particular hardware configuration is pre-allocated in the bit map.

The order of entries in the memory segment table is also critical. The first entry is reserved for the operating system. The remaining entries ca be specified by user. In specifying the user memory segments, the absolute TPA regions (segments based at 0000H) should be specified in order of size, from the largest to the smallest. Entering the segments in this order causes the MP/M II memory manager to allocate the largest available TPA region for execution by a COM program because it linearly searches through the memory segment table for the first available segment based at zero. The ordering of relocatable segments (those not based at 0000H) is not critical because the MP/M II memory manager does a best fit for those segments.

The attribute byte is normally defined as 00. However, if you wish to pre-allocate a memory segment, specify a value of FFH.

The bank byte value is an index which can be used by the XIOS to obtain a value to be sent to the bank switching hardware to select the specified bank. Values of 0,1,2,... are used to identify the memory ɔanks. A bank byte value of 0 is used for the non-resident portion of MP/M II.

5.2.22 Accept Memory Segment Table

A negative response to this query allows memory segment entries to be re-edited prior to acceptance.

5.3 GENSYS Execution

The GENSYS program has an automatic mode which simplifies repetitive generation of MPM.SYS files. This is useful in a debug mode of testing, XIOS editing, and a subsequent GENSYS execution to produce a new MPM.SYS file. The automatic mode is specified as follows:

```
0A>GENSYS $A
```

The effect of the automatic mode is to simulate the entry of a <cr> for each GENSYS query.

SECTION 6

MP/M LOADER

6.1 MP/M Loader Operation and Display

The MPMLDR program loads the MPM.SYS file and branches to the execution address of the MP/M II operating system. MPMLDR can be run under CP/M or loaded from the first two tracks of a disk by the cold start loader.

The MPMLDR displays system loading and configuration. It does not require any operator interaction. In the following example, the MPM.SYS file prepared by the first GENSYS example shown in Section 5 is loaded into memory and executed.

```
MP/M-II V2.0 Loader
Copyright (C) 1981, Digital Research

Nmb of consoles      = 2
Breakpoint RST #     = 6
Z80 Alternate register set saved/restored by dispatcher

Memory Segment Table:
SYSTEM   DAT  FF00H  0100H
TMPD     DAT  FE00H  0100H
USERSYS  STK  FD00H  0100H
XIOSJMP  TBL  FC00H  0100H
RESBDOS  SPR  F000H  0C00H
XDOS     SPR  CE00H  2200H
ABORT    RSP  CD00H  0100H
Spool    RSP  CC00H  0100H
MPMSTAT  RSP  CB00H  0100H
BNKXIOS  SPR  B800H  1300H
BNKBDOS  SPR  9500H  2300H
BNKXDOS  SPR  9200H  2300H
TMP      SPR  8F00H  0300H
Spool    BRS  8700H  0800H
Mpmstat  BRS  7900H  0E00H
LCKLSTS  DAT  7700H  0200H
CONSOLE  DAT  7500H  0200H
-------------------------
MP/M II Sys  7500H  8BH0H  Bank  0
Memseg  Usr  0000H  C000H  Bank  1
Memseg  Usr  0000H  C000H  Bank  2
Memseg  Usr  0000H  7500H  Bank  0

MP/M II V2.0
Copyright (C) 1981, Digital Research
0A>
```

6.2 MPMLDR Execution

Two parameters may be specified to the MPMLDR. The first parameter is used to cause a break to a CP/M debugger after the loading is completed. The parameter is a $Bn character string placed in the default FCB filename field beginning at 005DH. The character n is the CP/M debugger restart number. If n is not entered, a default of 7 is used. An example of this parameter is shown in Section 1.4.

The second parameter can specify an alternate filename of loading other than the standard MPM.SYS file. This parameter is specified by placing a filename with a filetype of SYS in the default FCB beginning at 005CH, or, if the $Bn parameter is also being specified, in the second default FCB beginning at 006CH. A good application of this second parameter would be to incorporate a menu-driven SYS file selection in the LDRBIOS at the SELDSK entry point. Thus, the operator would be prompted to select the appropriate SYS file for his MP/M environment. Custom code at the SELDSK entry point would prompt the operator for a file name and then place the selected SYS file name into the default FCB beginning at 005CH.

APPENDIX A

DISK DEFINITION MACRO

```
;       MP/M II V2.0 disk re-definition library
;
;       Copyright (c) 1979, 1980, 1981
;       Digital Research
;       Box 579
;       Pacific Grove, CA
;       93950
;
;       MP/M II logical disk drives are defined using the
;       macros given below, where the sequence of calls
;       is:
;
;       disks   n
;       diskdef parameter-list-0
;       diskdef parameter-list-1
;       ...
;       diskdef parameter-list-n
;       endef
;
;       where n is the number of logical disk drives attached
;       to the MP/M II system, and parameter-list-i defines the
;       characteristics of the ith drive (i=0,1,...,n-1)
;
;       each parameter-list-i takes the form
;               dn,fsc,lsc,[skf],bls,dks,dir,cks,ofs,[k16],[prm]
;       where
;       dn      is the disk number, 0,1,...,n-1
;       fsc     is the first sector number (usually 0 or 1)
;       lsc     is the last sector number on a track
;       skf     is the optional "skew factor" for sector translate
;       bls     is the data block size (1024,2048,...16384)
;       dks     is the disk size in bls increments (word)
;       dir     is the number of directory elements (word)
;       cks     is the number of directory elements to checksum
;       ofs     is the number of tracks to skip (word)
;       k16     is an optional 0 which forces 16K/directory entry
;       prm     is an optional 0 which marks drive as permanent
;
;       for convenience, the form
;               dn,dm
;       defines disk dn as having the same characteristics as
;       a previously defined disk dm.
;
;       a standard four drive MP/M II system is defined by
;               disks   4
;               diskdef 0,1,26,6,1024,243,64,64,2
;       dsk     set     0
;               rept    3
;       dsk     set     dsk+1
;               diskdef %dsk,0
;               endm
;               endef
;
;       the value of "begdat" at the end of assembly defines the
;       beginning of the uninitialize ram area above the bios,
;       while the value of "enddat" defines the next location
;       following the end of the data area.  the size of this
;       area is given by the value of "datsiz" at the end of the
;       assembly.  note that the allocation vector will be quite
;       large if a large disk size is defined with a small block
;       size.
;
dskhdr  macro   dn
;;      define a single disk header list
dpe&dn: dw      xlt&dn;,0000h      ;translate table
        dw      0000h,0000h        ;scratch area
        dw      dirbuf,dpb&dn      ;dir buff,parm block
        dw      csv&dn,alv&dn      ;check, alloc vectors
        endm
;
disks   macro   nd
;;      define nd disks
ndisks  set     nd          ;;for later reference
dpbase  equ     $           ;base of disk parameter blocks
;;      generate the nd elements
dsknxt  set     0
        rept    nd
        dskhdr %dsknxt
dsknxt  set     dsknxt+1
        endm
        endm
;
dpbhdr  macro   dn
dpb&dn  equ     $                  ;disk parm block
        endm
;
ddb     macro   data,comment
;;      define a db statement
        db      data               comment
        endm
;
ddw     macro   data,comment
;;      define a dw statement
        dw      data               comment
        endm
;
gcd     macro   m,n
;;      greatest common divisor of m,n
;;      produces value gcdn as result
;;      (used in sector translate table generation)
gcdm    set     m       ;;variable for m
gcdn    set     n       ;;variable for n
gcdr    set     0       ;;variable for r
```

```
        rept    65535
gcdx    set     gcdm/gcdn
gcdr    set     gcdm - gcdx*gcdn
        if      gcdr = 0
        exitm
        endif
gcdm    set     gcdn
gcdn    set     gcdr
        endm
        endm
;
diskdef macro   dn,fsc,lsc,skf,bls,dks,dir,cks,ofs,k16
;;      generate the set statements for later tables
cksz    set     (cks)/4
        if      nul lsc
;;      current disk dn same as previous fsc
dpb&dn  equ     dpb&fsc ;equivalent parameters
als&dn  equ     als&fsc ;same allocation vector size
css&dn  equ     css&fsc ;same checksum vector size
xlt&dn  equ     xlt&fsc ;same translate table
        else
secmax  set     lsc-(fsc)           ;;sectors 0...secmax
sectors set     secmax+1;;number of sectors
als&dn  set     (dks)/8 ;;size of allocation vector
        if      ((dks) mod 8) ne 0
als&dn  set     als&dn+1
        endif
css&dn  set     cksz    ;;number of checksum elements
;;      generate the block shift value
blkval  set     bls/128 ;;number of sectors/block
blkshf  set     0       ;;counts right 0's in blkval
blkmsk  set     j0      ;;fills with 1's from right
        rept    16      ;;once for each bit position
        if      blkval=1
        exitm
        endif
;;      otherwise, high order 1 not found yet
blkshf  set     blksf+1
blkmsk  set     (blkmsk shl 1) or 1
blkval  set     blkval/2
        endm
;;      generate the extent mask byte
blkval  set     bls/1024            ;;number of kilobytes/block
extmsk  set     0       ;;fill from right with 1's
        rept    16
        if      blkval=1
        exitm
        endif
;;      otherwise more to shift
extmsk  set     (extmsk shl 1) or 1
blkval  set     blkval/2
        endm
;;      may be double byte allocation
        if      (dks) > 256
extmsk  set     (extmsk shr 1)
```

```
        endif
;;      may be optional [0] in last position
        if      not nul k16
extmsk  set     k16
        endif
;;      now generate directory reservation bit vector
dirrem  set     dir      ;;# remaining to process
dirbks  set     bls/32   ;;number of entries per block
dirblk  set     0        ;;fill with 1's on each loop
        rept    16
        if      dirrem=0
        exitm
        endif
;;      not complete, iterate once again
;;      shift right and add 1 high order bit
dirblk  set     (dirblk shr 1) or 8000h
        if      dirrem > dirbks
dirrem  set     dirrem-dirbks
        else
dirrem  set     0
        endif
        endm
        dpbhdr dn        ;;generate equ $
        ddw     %sectors,<;sec per track>
        ddb     %blkshf,<;block shift>
        ddb     %blkmsk,<;block mask>
        ddb     %extmsk,<;extnt mask>
        ddw     %(dks)-1,<;disk size-1)
        ddw     %(dir)-1,<;directory max>
        ddb     %dirblk shr 8,<;alloc0>
        ddb     %dirblk and 0ffh,<;alloc1>
        if      nul prm
        ddw     %(cks)/4,<;check size>
        else
        ddw     8000h+cksz,<;permanent disk with check size>
        endif
        ddw     %ofs,<;offset>
;;      generate the translate table, if requested
        if      nul skf
xlt&dn  equu    0                ;no xlate table
        else
        if      skf = 0
xlt&dn  equ     0                ;no xlate table
        else
;;      generate the translate table
nxtsec  set     0        ;;next sector to fill
nxtbas  set     0        ;;moves by one on overflow
        gcd     %sectors,skf
;;      gcdn = gcd(sectors,skew)
neltst  set     sectors/gcdn
;;      neltst is number of elements to generate
;;      before we overlap previous elements
nelts   set     neltst  ;;counter
xlt&dn  equ     $                ;translate table
        rept    sectors ;;once for each sector
        if      sectors < 256
        ddb     %nxtsec+(fsc)
        else
        ddw     %nxtsec+(fsc)
        endif
nxtsec  set     nxtsec+(skf)
        if      nxtsec >= sectors
nxtsec  set     nxtsec-sectors
        endif
nelts   set     nelts-1
        if      nelts = 0
nxtbas  set     nxtbas+1
nxtsec  set     nxtbas
nelts   set     neltst
        endif
        endm
        endif   ;;end of nul fac test
        endif   ;;end of nul bls test
        endm
;
defds   macro   lab,space
lab:    ds      space
        endm
;
lds     macro   lb,dn,val
        defds   lb&dn,%val&dn
        endm
;
endef   macro
;;      generate the necessary ram data areas
begdat  equ     $
dirbuf: ds      128     ;directory access buffer
dsknxt  set     0
        rept    ndisks  ;;once for each disk
        lds     alv,%dsknxt,als
        lds     csv,%dsknxt,css
dsknxt  set     dsknxt+1
        endm
enddat  equ     $
datsiz  equ     $-begdat
force:  db      0       ;force out last byte in hex file
        endm
;
```

APPENDIX B

SECTOR DEBLOCKING ALGORITHMS FOR MP/M II

```
                        page    0
                ;*****************************************************
                ;*                                                   *
                ;*      Sector Deblocking Algorithms for MP/M II V2.0 *
                ;*                                                   *
                ;*****************************************************
                ;
                ;       utility macro to compute sector mask
                smask   macro   hblk
                ;;      compute log2(hblk), return @x as result
                ;;      (2 ** @x = hblk on return)
                @y      set     hblk
                @x      set     0
                ;;      count right shifts of @y until = 1
                        rept    8
                        if      @y = 1
                        exitm
                        endif
                ;;      @y is not 1, shift right one position
                @y      set     @y shr 1
                @x      set     @x + 1
                        endm
                        endm
                ;
                ;*****************************************************
                ;*                                                   *
                ;*          MP/M to host disk constants              *
                ;*                                                   *
                ;*****************************************************
0800 =          blksiz  equ     2048            ;MP/M allocation size
0200 =          hstsiz  equ     512             ;host disk sector size
0014 =          hstspt  equ     20              ;host disk sectors/trk
0004 =          hstblk  equ     hstsiz/128      ;MP/M sects/host buff
0050 =          cpmspt  equ     hstblk * hstspt ;MP/M sectors/track
0003 =          secmsk  equ     hstblk-1        ;sector mask
                        smask   hstblk          ;compute sector mask
0002 =          secshf  equ     @x              ;log2(hstblk)
                ;
                ;*****************************************************
                ;*                                                   *
                ;*         BDOS constants on entry to write          *
                ;*                                                   *
                ;*****************************************************
0000 =          wrall   equ     0               ;write to allocated
0001 =          wrdir   equ     1               ;write to directory
0002 =          wrual   equ     2               ;write to unallocated
                ;
                ;*****************************************************
                ;*      The BDOS entry points given below show the      *
                ;*      code which is relevant to deblocking only.      *
                ;*                                                      *
                ;********************************************************
                ;
                ;
                ;       DISKDEF macro, or hand coded tables go here
0000 =          dpbase  equ     $               ;disk param block base
                ;
                boot:
                wboot:
                        ;enter here on system boot to initialize
0000 AF                 xra     a               ;0 to accumulator
0001 326901             sta     hstact          ;host buffer inactive
0004 326B01             sta     unacnt          ;clear unalloc count
0007 C9                 ret
                ;
                home:
                        ;home the selected disk
0008 3A6A01             lda     hstwrt          ;check for pending write
000B B7                 ora     a
000C C21200             jnz     homed
000F 326901             sta     hstact          ;clear host active flag
                homed:
0012 C9                 ret
                ;
                seldsk:
                        ;select disk
0013 79                 mov     a,c             ;selected disk number
0014 326001             sta     sekdsk          ;seek disk number
0017 6F                 mov     l,a             ;disk number to HL
0018 2600               mvi     h,0
                        rept    4               ;multiply by 16
                        dad     h
                        endm
001A+29                 DAD     H
001B+29                 DAD     H
001C+29                 DAD     H
001D+29                 DAD     H
001E 110000             lxi     d,dpbase        ;base of parm block
0021 19                 dad     d               ;hl=.dpb(curdsk)
0022 C9                 ret
                ;
                settrk:
                        ;set track given by registers BC
0023 60                 mov     h,b
0024 69                 mov     l,c
0025 226101             shld    sektrk          ;track to seek
0028 C9                 ret
                ;
                setsec:
                        ;set sector given by register c
0029 79                 mov     a,c
002A 326301             sta     seksec          ;sector to seek
002D C9         ret
```

```
                ;
                setdma:
                       ;set dma address given by BC
002E 60                mov     h,b
002F 69                mov     l,c
0030 227401            shld    dmaadr
0033 C9                ret
                ;
                sectran:
                       ;translate sector number BC
0034 60                mov     h,b
0035 69                mov     l,c
0036 C9                ret
                ;
                ;*****************************************************
                ;*                                                   *
                ;*     The READ entry point takes the place of      *
                ;*     the previous BIOS definition for READ.       *
                ;*                                                   *
                ;*****************************************************
                read:
                       ;read the selected MP/M sector
0037 AF                xra     a
0038 326B01            sta     unacnt          ;unacnt = 0
003B 3C                inr     a
003C 327201            sta     readop          ;read operation
003F 327101            sta     rsflag          ;must read data
0042 3E02              mvi     a,wrual
0044 327301            sta     wrtype          ;treat as unalloc
0047 C3B500            jmp     rwoper          ;to perform the read
                ;
                ;*****************************************************
                ;*                                                   *
                ;*     The WRITE entry point takes the place of     *
                ;*     the previous BIOS definition for WRITE       *
                ;*                                                   *
                ;*****************************************************
                write:
                       ;write the selected MP/M sector
004A AF                xra     a               ;0 to accumulator
004B 327201            sta     readop          ;not a read operation
004E 79                mov     a,c             ;write type in c
004F 327301            sta     wrtype
0052 E602              ani     wrual           ;write unallocated?
0054 CA6E00            jz      chkuna          ;check for unalloc
                ;
                ;      write to unallocated, set parameters
0057 3E10              mvi     a,blksiz/128    ;next unalloc recs
0059 326B01            sta     unacnt
005C 3A6001            lda     sekdsk          ;disk to seek
005F 326C01            sta     unadsk          ;unadsk = setdsk
0062 2A6101            lhld    sektrk
0065 226D01            shld    unatrk          ;unatrk = sectrk
0068 3A6301            lda     seksec
006B 326F01            sta     unasec          ;unasec = seksec
```

```
                    ;
                    chkuna:
                            ;check for write to unallocated sector
006E 3A6B01                 lda     unacnt          ;any unalloc remain?
0071 B7                     ora     a
0072 CAAD00                 jz      alloc           ;skip if not
                    ;
                    ;       more unallocated records remain
0075 3D                     dcr     a               ;unacnt = unacnt-1
0076 326B01                 sta     unacnt
0079 3A6001                 lda     sekdsk          ;same disk?
007C 216C01                 lxi     h,unadsk
007F BE                     cmp     m               ;sekdsk = unadsk?
0080 C2AD00                 jnz     alloc           ;skip if not
                    ;
                    ;       disks are the same
0083 216D01                 lxi     h,unatrk
0086 CD5201                 call    sektrkcmp       ;sektrk = unatrk?
0089 C2AD00                 jnz     alloc           ;skip if not
                    ;
                    ;       tracks are the same
008C 3A6301                 lda     seksec          ;same sector?
008F 216F01                 lxi     h,unasec
0092 BE                     cmp     m               ;seksec = unasec?
0093 C2AD00                 jnz     alloc           ;skip if not
                    ;
                    ;       match,move to next sector for future ref
0096 34                     inr     m               ;unasec = unasec+1
0097 7E                     mov     a,m             ;end of track?
0098 FE50                   cpi     cpmspt          ;count MP/M sectors
009A DAA600                 jc      noovf           ;skip if no overflow
                    ;
                    ;       overflow to next track
009D 3600                   mvi     m,0             ;unasec = 0
009F 2A6D01                 lhld    unatrk
00A2 23                     inx     h
00A3 226D01                 shld    unatrk          ;unatrk = unatrk+1
                    ;
                    noovf:
                            ;match found, mark as unnecessary read
00A6 AF                     xra     a               ;0 to accumulator
00A7 327101                 sta     rsflag          ;rsflag = 0
00AA C3B500                 jmp     rwoper          ;to perform the write
                    ;
                    alloc:
                            ;not an unallocated record, requires pre-read
00AD AF                     xra     a               ;0 to accum
00AE 326B01                 sta     unacnt          ;unacnt = 0
00B1 3C                     inr     a               ;1 to accum
00B2 327101                 sta     rsflag          ;rsflag = 1
                    ;
                    ;*****************************************************
                    ;*                                                   *
                    ;*      Common code for READ and WRITE follows       *
                    ;*                                                   *
```

```
                        ;*****************************************************
                        rwoper:
                                ;enter here to perform the read/write
00B5 AF                         xra     a               ;zero to accum
00B6 327001                     sta     erflag          ;no errors (yet)
00B9 3A6301                     lda     seksec          ;compute host sector
                                rept    secshf
                                ora     a               ;carry = 0
                                rar
                                endm
00BC+B7                         ORA     A               ;CARRY = 0
00BD+1F                         RAR                     ;SHIFT RIGHT
00BE+B7                         ORA     A               ;CARRY = 0
00BF+1F                         RAR                     ;SHIFT RIGHT
00C0 326801                     sta     sekhst          ;host sector to seek
                        ;
                        ;       active host sector?
00C3 216901                     lxi     h,hstact        ;host active flag
00C6 7E                         mov     a,m
00C7 3601                       mvi     m,1             ;always becomes 1
00C9 B7                         ora     a               ;was it already?
00CA CAF100                     jz      filhst          ;fill host if not
                        ;
                        ;       host buffer active, same as seek buffer?
00CD 3A6001                     lda     sekdsk
00D0 216401                     lxi     h,hstdsk        ;same disk?
00D3 BE                         cmp     m               ;sekdsk = hstdsk?
00D4 C2EA00                     jnz     nomatch
                        ;
                        ;       same disk, same track?
00D7 216501                     lxi     h,hsttrk
00DA CD5201                     call    sektrkcmp       ;sektrk = hsttrk?
00DD C2EA00                     jnz     nomatch
                        ;
                        ;       same disk, same track, same buffer?
00E0 3A6801                     lda     sekhst
00E3 216701                     lxi     h,hstsec        ;sekhst = hstsec?
00E6 BE                         cmp     m
00E7 CA0E01                     jz      match           ;skip if match
                        ;
                        nomatch:
                                ;proper disk, but not correct sector
00EA 3A6A01                     lda     hstwrt          ;host written?
00ED B7                         ora     a
00EE C45E01                     cnz     writehst        ;clear host buff
                        ;
                        filhst:
                                ;may have to fill the host buffer
00F1 3A6001                     lda     sekdsk
00F4 326401                     sta     hstdsk
00F7 2A6101             lhld    sektrk
00FA 226501                     shld    hsttrk
00FD 3A6801                     lda     sekhst
0100 326701                     sta     hstsec
0103 3A7101                     lda     rsflag          ;need to read?
```

```
0106 B7                 ora     a
0107 C45F01             cnz     readhst         ;yes, if 1
010A AF                 xra     a               ;0 to accum
010B 326A01             sta     hstwrt          ;no pending write
                ;
                match:
                        ;copy data to or from buffer
010E 3A6301             lda     seksec          ;mask buffer number
0111 E603               ani     secmsk          ;least signif bits
0113 6F                 mov     l,a             ;ready to shift
0114 2600               mvi     h,0             ;double count
                        rept    7               ;shift to left 7
                        dad     h
                        endm
0116+29                 DAD     H
0117+29                 DAD     H
0118+29                 DAD     H
0119+29                 DAD     H
011A+29                 DAD     H
011B+29                 DAD     H
011C+29                 DAD     H
                ;       hl has relative host buffer address
011D 117601             lxi     d,hstbuf
0120 19                 dad     d               ;hl = host address
0121 EB                 xchg                    ;now in DE
0122 2A7401             lhld    dmaadr          ;get/put MP/M data
0125 0E80               mvi     c,128           ;length of move
0127 3A7201             lda     readop          ;which way?
012A B7                 ora     a
012B C23401             jnz     rwmove          ;skip if read
                ;
                ;       write operation, mark and switch direction
012E 3E01               mvi     a,1
0130 326A01             sta     hstwrt          ;hstwrt = 1
0133 EB                 xchg                    ;source/dest swap
                ;
                rwmove:
                        ;C initially 128, DE is source, HL is dest
0134 1A                 ldax    d               ;source character
0135 13                 inx     d
0136 77                 mov     m,a             ;to dest
0137 23                 inx     h
0138 0D                 dcr     c               ;loop 128 times
0139 C23401             jnz     rwmove
                ;
                ;       data has been moved to/from host buffer
013C 3A7301             lda     wrtype          ;write type
013F E601               ani     wrdir           ;to directory?
0141 3A7001             lda     erflag          ;in case of errors
0144 C8                 rz                      ;no further processing
                ;
                ;       clear host buffer for directory write
0145 B7                 ora     a               ;errors?
0146 C0                 rnz                     ;skip if so
0147 AF                 xra     a               ;0 to accum
```

```
0148 326A01           sta     hstwrt           ;buffer written
014B CD5E01           call    writehst
014E 3A7001           lda     erflag
0151 C9               ret
                ;
                ;*****************************************************
                ;*                                                   *
                ;*      Utility subroutine for 16-bit compare        *
                ;*                                                   *
                ;*****************************************************
                sektrkcmp:
                      ;HL = .unatrk or .hsttrk, compare with sektrk
0152 EB               xchg
0153 216101           lxi     h,sektrk
0156 1A               ldax    d                ;low byte compare
0157 BE               cmp     m                ;same?
0158 C0               rnz                      ;return if not
                ;     low bytes equal, test high 1s
0159 13               inx     d
015A 23               inx     h
015B 1A               ldax    d
0115C BE              cmp     m       ;sets flags
015D C9               ret
                ;
                ;*****************************************************
                ;*                                                   *
                ;*      WRITEHST performs the physical write to      *
                ;*      the host disk, READHST reads the physical    *
                ;*      disk.                                        *
                ;*                                                   *
                ;*****************************************************
                writehst:
                      ;hstdsk = host disk #, hsttrk = host track #,
                      ;hstsec = host sect #. write "hstsiz" bytes
                      ;from hstbuf and return error flag in erflag.
                      ;return erflag non-zero if error
015E C9               ret
                ;
                readhst:
                      ;hstdsk = host disk #, hsttrk = host track #,
                      ;hstsec = host sect #. read "hstsiz" bytes
                      ;into hstbuf and return error flag in erflag.
015F C9               ret
                ;
                ;*****************************************************
                ;*                                                   *
                ;*      Unitialized RAM data areas                   *
                ;*                                                   *
                ;*****************************************************
                ;
0160            sekdsk: ds    1                ;seek disk number
0161            sektrk: ds    2                ;seek track number
0163            seksec: ds    1                ;seek sector number
                ;
0164            hstdsk: ds    1                ;host disk number
```

```
0165            hsttrk: ds     2                  ;host track number
0167            hstsec: ds     1                  ;host sector number
                ;
0168            sekhst: ds     1                  ;seek shr secshf
0169            hstact: ds     1                  ;host active flag
016A            hstwrt: ds     1                  ;host written flag
                ;
016B            unacnt: ds     1                  ;unalloc rec cnt
016C            unadsk: ds     1                  ;last unalloc disk
016D            unatrk: ds     2                  ;last unalloc track
016F            unasec: ds     1                  ;last unalloc sector
                ;
0170            erflag: ds     1                  ;error reporting
0171            rsflag: ds     1                  ;read sector flag
0172            readop: ds     1                  ;1 if read operation
0173            wrtype: ds     1                  ;write operation type
0174            dmaadr: ds     2                  ;last dma address
0176            hstbuf: ds     hstsiz             ;host buffer
                ;
                ;*****************************************************
                ;*                                                   *
                ;*     The ENDEF macro invocation goes here          *
                ;*                                                   *
                ;*****************************************************
0376                   end


00AD ALLOC      0800 BLKSIZ     0000 BOOT       006E CHKUNA
0050 CPMSPT     0174 DMAADR     0000 DPBASE     0170 ERFLAG
00F1 FILHST     0008 HOME       0012 HOMED      0169 HSTACT
0004 HSTBLK     0176 HSTBUF     0164 HSTDSK     0167 HSTSEC
0200 HSTSIZ     0014 HSTSPT     0165 HSTTRK     016A HSTWRT
010E MATCH      00EA NOMATCH    00A6 NOOVF      0037 READ
015F READHST    0172 READOP     0171 RSFLAG     0134 RWMOVE
00B5 RWOPER     0003 SECMSK     0002 SECSHF     0034 SECTRAN
0160 SEKDSK     0168 SEKHST     0163 SEKSEC     0161 SEKTRK
0152 SEKTRKCMP  0013 SELDSK     002E SETDMA     0029 SETSEC
0023 SETTRK     016B UNACNT     016C UNADSK     016F UNASEC
016D UNATRK     0000 WBOOT      0000 WRALL      0001 WRDIR
004A WRITE      015E WRITEHST   0173 WRTYPE     0002 WRUAL
```

APPENDIX C

SAMPLE MP/M II LOADER BIOS

```
                       page    0
                       title   'Skeleton MP/M-80 V2.0 Ldrbios'


                ;      Copyright (C) 1978, 1979, 1980, 1981
                ;      Digital Research
                ;      Box 579, Pacific Grove
                ;      California, 93950

0000 =          false  equ     0
FFFF =          true   equ     not false


1700                   org     1700h


0080 =          buff   equ     0080h   ;default buffer address

                ;      jump vector for individual routines

1700 C33317            jmp     boot
1703 C33317     wboote:jmp     wboot
1706 C33617            jmp     const
1709 C33417            jmp     conin
170C C33517            jmp     conout
170F C33917            jmp     list
1712 C33817            jmp     punch
1715 C33717            jmp     reader
1718 C33C17            jmp     home
171B C33B17            jmp     seldsk
171E C33D17            jmp     settrk
1721 C33E17            jmp     setsec
1724 C33F17            jmp     setdma
1727 C34117            jmp     read
172A C34217            jmp     write
172D C33A17            jmp     list$st         ; list status poll
1730 C34017            jmp     sect$tran       ; sector translation


                boot:
                wboot:
                gocpm:
1733 C9                ret

                crtin:                ; crt: input
1734 C9                ret
                crtout                ; crt: output
1735 C9                ret
```

```
               crtst:                ; crt: status
1736 C9                ret
               ttyin:                ; tty: input
1737 C9                ret
               ttyout:               ; tty: output
1738 C9                ret
               lptout:               ; lpt: output
1739 C9                ret
               lpt$st:
173A C9                ret

1734 =         conin   equ     crtin
1736 =         const   equ     crtst
1735 =         conout  equ     crtout
1737 =         reader  equ     ttyin
1738 =         punch   equ     ttyout
1739 =         list    equ     lptout
173A =         listst  equ     lptst

               seldsk:         ;select disk given by register c
173B C9                ret
               ;
               home:   ;move to home position
173C C9                ret
               ;
               settrk: ;set track number given by c
173D C9                ret
               ;
               setsec: ;set sector number given by c
173E C9                ret
               ;
               setdma: ;set dma address given by regs b,c
173F C9                ret
               ;
               sect$tran:            ; translate the sector # in <c reg>
1740 C9                ret
               ;
               read:   ;read next disk record (assuming disk/trk/sec/ selected
1741 C9                ret
               ;
               write:  ;disk write function
1742 C9                ret
               ;
1743                   end
```

APPENDIX D

SAMPLE XIOS SOURCE LISTING

```
                        page    0
                        title   'MP/M II V2.0 DSC-2 Basic & Extended I/O
                        cseg
                        maclib  diskdef
                ;
                ; bios for micro-2 computer
                ;
                ;
0000 =          false   equ     0
FFFF =          true    equ     not false
                ;
FFFF =          debug   equ     true
FFFF =          ldcmd   equ     true
                ;
FFFF =          MHz4    equ     true

                        if      MHz4
0086 =          dlycnst equ     086h
                        else
                dlycnst equ     054h
                        endif
                ;
                ;       org     0000h

                ;pdisp  equ     $-3
                ;xdos   equ     pdisp-3
                ;
                ;       jump vector for individual subroutines
                ;       jmp     coldstart       ;coldstart
0000 C34900             jmp     commonbase
                wboot:
0003 C35A00             jmp     warmstart       ;warm start
0006 C35F00             jmp     const           ;console status
0009 C36800             jmp     conin           ;console character in
000C C37100             jmp     conout          ;console character out
000F C3DF00             jmp     list            ;list character out
0012 C38100             jmp     rtnempty        ;punch not implemented
0015 C38100             jmp     rtnempty        ;reader not implemented
0018 C3CA02             jmp     home            ;move head to home
001B C3DB02             jmp     seldsk          ;select disk
001E C30503             jmp     settrk          ;set track number
0021 C32203             jmp     setsec          ;set sector number
0024 C33A03             jmp     setdma          ;set dma address
0027 C34003             jmp     read            ;read disk
002A C34503             jmp     write           ;write disk
002D C30101             jmp     pollpt          ;list status
0030 C32803             jmp     sectran         ;sector translate
```

```
0033 C30C02          jmp     selmemory        ; select memory
0036 C3F301          jmp     polldevice       ; poll device
0039 C30D02          jmp     startclock       ; start clock
003C C31302          jmp     stopclop         ; stop clock
003F C31802          jmp     exitregion       ; exit region
0042 C31F02          jmp     maxconsole       ; maximum console number
0045 C32202          jmp     systeminit       ; system initialization
0048 00              db      0                ; force use of internal
             ;       jmp     idle             ; idle procedure
             ;
             commonbase:
0049 C35A00          jmp     coldstart
004C C30000  swtuser:jmp     $-$
004F C30000  swtsys: jump    $-$
0052 C30000  pdisp:  jmp     $-$
0055 C30000  xdos:   jmp     $-$
0058 0000    sysdat: dw      $-$

             coldstart:
             warmstart:
005A 0E00            mvi     c,0
005C C35500          jmp     xdos             ; system reset, terminal
             ;
             ;
             ;I/O handlers
             ;
             ;
             ; MP/M II V2.0   Console Bios
             ;
             ;
0003 =       nmbcns  equ     3      ; number of consoles

0083 =       poll    equ     131    ; XDOS poll function
0086 =       makeque equ     134    ; XDOS make queue function
0089 =       readque equ     137    ; XDOS read queue function
008B =       writeque equ    139    ; XDOS write queue function
008D =       xdelay  equ     141    ; XDOS delay function
0090 =       create  equ     144    ; XDOS create process function

0000 =       pllpt   equ     0      ; poll printer
0001 =       plco0   equ     1      ; poll console out #0
0002 =       plco2   equ     2      ; poll console out #1
0003 =       plco3   equ     3      ; poll console out #2 (Port 3)
0004 =       plcoi3  equ     4      ; poll console in #2 (Port 3)
                     if      debug
0005 =       plci0   equ     5      ; poll console in #0
                     endif

             ;
             const:                 ; Console Status
005F CD7A00          call   ptbljmp; compute and jump to hndlr
0062 8E00            dw      pt0st ; console #0 status routine
0064 0901            dw      pt2st ; console #1 (Port 2) status reg
0066 C301            dw      pt3st ; Console #2 (Port 3) status reg
```

```
                conin:                  ; Console Input
0068 CD7A00             call    ptbljmp; compute and jump to hndlr
006B 9D00               dw      pt0in  ; console #0 input
006D 9901               dw      pt2in  ; console #1 (Port 2) input
006F CB01               dw      pt3in  ; console #2 (Port 3) input

                conout:                 ; Console Output
007A CD7A00             call    ptbljmp; compute and jump to hndlr
0074 C200               dw      pt0out ; console #0 output
0076 A701               dw      pt2out ; console #1 (Port 2) output
0078 D701               dw      pt3out ; console #2 (Port 3) output

                ;
                ptbljmp:                ; compute and jump to handlr
                                        ; d = console #
                                        ; do not destroy d !
007A 7A                 mov     a,d
007B FE03               cpi     nmbcns
007D DA8300             jc      tbljmp
0080 F1                 pop     psw     ; throw away table address
                rtnempty:
0081 AF                 xra     a
0082 C9                 ret
                tbljmp:                 ; compute and jump to handler
                                        ; a = table index
0083 87                 add     a       ; double table index for adr offset
0084 E1                 pop     h       ; return adr points to jump tbl
0085 5F                 mov     e,a
0086 1600               mvi     d,0
0088 19                 dad     d       ; add table index * 2 to tbl base
0089 5E                 mov     e,m     ; get handler address
008A 23                 inx     h
008B 56                 mov     d,m
008C EB                 xchg
008D E9                 pchl            ; jump to computed cns handler

                ;
                ; ASCII Character Equates
                ;
005F =          uline   equ     5fh
007F =          rubout  equ     7fh
0020 =          space   equ     20h
0008 =          backsp  equ     8h
005F =          altrub  equ     uline
                ;
                ; Input / Output Port Address Equates
                ;
0040 =          data0   equ     40h
0041 =          sts0    equ     data0+1
0041 =          cd0     equ     sts0
0048 =          data1   equ     48h
0049 =          sts1    equ     data1+1
0049 =          cd1     equ     sts1
0050 =          data2   equ     50h
0051 =          sts2    equ     data2+1
```

```
0051 =          cd2     equ     sts2
0058 =          data3   equ     58h
0059 =          sts3    equ     data3+1
0059 =          cd3     equ     sts3
                ;
                ; Poll Console #0 Input
                ;
                        if      debug
                polci0:
                pt0st:
                        if      ldcmd
008E 3AAF00             lda     pt0cntr
0091 B7                 ora     a
0092 3E00               mvi     a,0
0094 C0                 rnz
                        endif

0095 BD41               in      sts0
0097 E602               ani     2
0099 C8                 rz
009A 3EFF               mvi     a,0ffh
009C C9                 ret
                ;
                pt0in:
                        if      ldcmd
009D 21AF00             lxi     h,pt0cntr
00A0 73                 mov     a,m
00A1 B7                 ora     a
00A2 CAB600             jz      ldcmd0empty
00A5 35                 dcr     m
00A6 2AB000             lhld    pt0ptr
00A9 7E                 mov     a,m
00AA 23                 inx     h
00AB 22B000             shld    pt0ptr
00AE C9                 ret
                pt0cntr:
00AF 04                 db      ldcm0empty-pt0ldcmd
                pt0ptr:
00B0 B200               dw      pt0ldcmd
                pt0ldcmd:
00B2 746F6420           db      'tod '
                ldcm0empty:
                        endif

00B6 0E83               mvi     c,poll
00B8 1E05               mvi     e,plci0
00BA CD5500             call    xdos
00BD DB40               in      data0
00BF E67F               ani     7fh
00C1 C9                 ret
                ;
                        else
                pt0st   ;
                                        ; return 0ffh if ready,
                                        ;           000h if not
```

```
        lda     c0inmsgcnt
        ora     a
        rz
        mvi     a,0ffh
        ret
;
; Console #0 Input
;
c0inpd:
        dw      c2inpt ; pl
        db      0      ; status
        db      32     ; priority
        dw      c0instk+18 ; stkptr
        db      'c0in     ' ; name
        db      0      ; console
        db      0ffh   ; memseg
        ds      36

c0instk:
        dw      0c7c7h,0c7c7h,0c7c7h
        dw      0c7c7h,0c7c7h,0c7c7h
        dw      0c7c7h,0c7c7h,0c7c7h
        dw      c0inp  ; starting address

c0inq:
        dw      0      ; ql
        db      'c0inque ' ; name
        dw      1      ; msglen
        dw      4      ; nmbmsgs
        ds      8
c0inmsgcnt:
        ds      2      ; msgcnt
        ds      4      ; buffer

c0inqcb:
        dw      c0inq  ; pointer
        dw      ch0in ; msgadr
ch0in:
        db      0

c0inuqcb:
        dw      c0inq  ; pointer
        dw      char0in; msgadr
char0in:
        db      0

c0inp:
        mvi     c,makeque
        lxi     d,c0inq
        call    xdos   ; make the c0inq

c0inloop:
        mvi     c,flagwait
        mvi     e,6
        call    xdos   ; wait for c0 in intr flag
```

```
                       mvi     c,writeque
                       lxi     d,c0inqcb
                       call    xdos   ; write c0in queue
                       jmp     c0inloop


               pt0in:
                                      ; return character in reg A
                       mvi     c,readque
                       lxi     d,c0inuqcb
                       call    xdos           ; read from c0 in queue
                       lda     char0in        ; get character
                       ani     7fh            ; strip parity bit
                       ret
               ;
                       endif
               ;
               ; Console #0 Output
               ;
               pt0out:
                                      ; Reg C = character to output
00C2 DB41              in      sts0
00C4 E601              ani     01h
00C6 C2D200            jnz     tx0rdy
00C9 C5                push    b
00CA 0E83              mvi     c,poll
00CC 1E01              mvi     e,plco0
00CE CD5500            call    xdos   ; poll console #0 output
00D1 C1                pop     b
               tx0rdy:
00D2 79                mov     a,c
00D3 D340              out     data0
00D5 C9                ret
               ;
               ; poll console #0 output
               ;
               polco0:
00D6 DB41              in      sts0
00D8 E601              ani     01h
00DA C8                rz
00DB 3EFF              mvi     a,0ffh
00DD C9                ret
               ;
               ;
               ; Line Printer Driver: TI 810 Serial Printer
               ;                      TTY Model 40
               ;
               initflag:
00DE 00                db      0      ; printer initialization flag

               list:                  ; List Output
               ptlout:
                                      ; Reg c = Character to print
00DF 3ADE00            lda     initflag
00E2 B7                ora     a
```

```
00E3 C2ED00              jnz     ptlxx
00E6 3E27                mvi     a,27h
00E8 D349                out     49h               ; TTY Model 40 init
00EA 32DE00              sta     initflag
                ptlxx:
00ED DB49                in      stsl
00EF E601                ani     01h
00F1 C2FD00              jnz     txlrdy
00F4 C5                  push    b
00F5 0E83                mvi     c,poll
00F7 1E00                mvi     e,pllpt
00F9 CD5500              call    xdos              ; poll printer output
00FC C1                  pop     b
                txlrdy:
00FD 79                  mov     a,c               ; char to register a
00FE D348                out     datal
0100 C9                  ret
                ;
                ; Poll Printer Output
                ;
                pollpt
                                           ; return 0ffh if ready
                                           ;        000h if not
0101 DB49                in      stsl
0103 E601                ani     01h
0105 C8                  rz
0106 3EFF                mvi     a,0ffh
0108 C9                  ret
                ;
                ; Poll Console #1 (Port 2) Input
                ;
                pt2st:
                                           ; return 0ffh if ready,
                                           ;        000h if not
0109 3A6F01              lda     c2inmsgcnt
010C B7                  ora     a
010D C8                  rz
010E 3EFF                mvi     a,0ffh
0110 C9                  ret
                ;
                ; Console #1 (Port 2) Input
                ;
                c2inpd:
0111 0000                dw      0         ; pl
0113 00                  db      0         ; status
0114 22                  db      34        ; priority
0115 5701                dw      c2instk+18 ; stkptr
0117 6332696E20          db      'c2in    '  ; name
011F 02                  db      2         ; console
0120 FF                  db      0ffh      ; memseg
0121                     ds      36

                c2instk:
0145 C7C7C7C7C7          dw      0c7c7h,0c7c7h,0c7c7h
014B C7C7C7C7C7          dw      0c7c7h,0c7c7h,0c7c7h
```

```
0151 C7C7C7C7C7         dw      0c7c7h,0c7c7h,0c7c7h
0157 7F01               dw      c2inp  ; starting address

               c2inq:
0159 0000               dw      0       ; ql
015B 6332696E71         db      'c2inque ' ; name
0163 0100               dw      1       ; msglen
0165 0400               dw      4       ; nmbmsgs
0167                    ds      8
               c2inmsgcnt:
016F                    ds      2       ; msgcnt
0171                    ds      4       ; buffer

               c2inqcb:
0175 5901               dw      c2inq  ; pointer
0177 7901               dw      ch2in ; msgadr
               ch2in:
0179 00                 db      0

               c2inuqcb:
017A 5901               dw      c2inq  ; pointer
017C 7E01               dw      char2in; msgadr
               char2in:
017E 00                 db      0

               c2inp:
017F 0E86               mvi     c,makeque
0181 115901             lxi     d,c2inq
0184 CD5500             call    xdos    ; make the c2inq

               c2inloop:
0187 0E84               mvi     c,flagwait
0189 1E08               mvi     e,8
018B CD5500             call    xdos    ; wait for c2 in intr flag
018E 0E8B               mvi     c,writeque
0190 117501             lxi     d,c2inqcb
0193 CD5500             call    xdos    ; write c2in queue
0196 C38701             jmp     c2inloop


               pt2in:
                                        ; return character in reg A
0199 0E89               mvi     c,readque
019B 117A01             lxi     d,c2inuqcb
019E CD5500             call    xdos            ; read from c2 in queue
01A1 3A7E01             lda     char2in         ; get character
01A4 E67F               ani     7fh             ; strip parity bit
01A6 C9                 ret
               ;
               ; Console #1 (Port 2) Output
               ;
               pt2out:
                                        ; Reg C = character to output
01A7 DB51               in      sts2
01A9 E601               ani     01h
```

```
01AB C2B701            jnz     tx2rdy
01AE C5                push b
01AF 0E83              mvi     c,poll
01B1 1E02              mvi     e,plco2
01B3 CD5500            call    xdos   ; poll console #1 output
01B6 C1                pop     b
               tx2rdy:
01B7 79                mov     a,c
01B8 D350              out     data2
01BA C9                ret
               ;
               ; poll console #1 output
               ;
               polco2:
01BB DB51              in      sts2
01BD E601              ani     01h
01BF C8                rz
01C0 3EFF              mvi     a,0ffh
01C2 C9                ret
               ;
               ; Poll Console #2 (Port 3) Input
               ;
               polci3:
               pt3st:                 ; return 0ffh if ready,
                                      ;        000h if not
01C3 DB59              in      sts3
01C5 E602              ani     2
01C7 C8                rz
01C8 3EFF              mvi     a,0ffh
01CA C9                ret
               ;
               ; Console #2 (Port 3) Input
               ;
               pt3in:                 ; return character in reg A
01CB 0E83              mvi     c,poll
01CD 1E04              mvi     e,plci3
01CF CD5500            call    xdos            ; poll console #0 input
01D2 DB58              in      data3           ; read character
01D4 E67F              ani     7fh             ; strip parity bit
01D6 C9                ret
               ;
               ; Console #2 (Port 3) Output
               ;
               pt3out:                ; Reg C = character to output
01D7 DB59              in      sts3
01D9 E601              ani     01h
01DB C2E701            jnz     tx3rdy
01DE C5                push    b
01DF 0E83              mvi     c,poll
01E1 1E03              mvi     e,plco3
01E3 CD5500            call    xdos            ; poll console #2 (Port)
01E6 C1                pop     b
               tx3rdy:
01E7 79                mov     a,c
01E8 D358              out     data3           ; transmit character
```

```
01EA C9                ret
               ;
               ; Poll Console #2 (Port 3) Output
               ;
               polco3:
                                      ; return 0ffh if ready,
                                      ;       000h if not
01EB DB59              in      sts3
01ED E601              ani     01h
01EF C8                rz
01F0 3EFF              mvi     a,0ffh
01F2 C9                ret
               ;
               ;
               ;  MP/M II V2.0   Xios
               ;
               ;
               polldevice:
                                      ; Reg C = device # to be polled
                                      ; return 0ffh if ready,
                                      ;       000h if not
01F3 79                mov     a,c
01F4 FE06              cpi     nmbdev
01F6 DAFB01            jc      devok
01F9 3E06              mvi     a,nmbdev; if dev # >= nmbdev,
                                       ; set to nmbdev
               devok:
01FB CD8300            call    tbljmp ;jump to dev poll code

               devtbl:
01FE 0101              dw      pollpt ; poll printer output
0200 D600              dw      polco0 ; poll console #0 output
0202 BB01              dw      polco2 ; poll console #1 output
0204 EB01              dw      polco3 ; poll console #2 output
0206 C301              dw      polci3 ; poll console #2 input
                       if      debug
0208 8E00              dw      polci0 ; poll console #0 input
                       endif
0006 =         nmbdev  equ     ($-devtbl)/2    ; number of devices to
020A 8100              dw      rtnempty; bad device handler
               ;

               ; Select / Protect Memory
               ;
               selmemory:
                                      ; Reg BC = adr of mem descript
                                      ; BC ->  base   1 byte,
                                      ;        size   1 byte,
                                      ;        attrib 1 byte,
                                      ;        bank   1 byte.
               ; this hardware does not have memory protection or
               ; bank switching
020C C9                ret
               ;
               ; Start Clock
               ;
               startclock:
                                      ; will cause flag #1 to be set
                                      ;  at each system time unit tick
020D 3EFF              mvi     a,0ffh
020F 322F04            sta     tickn
0212 C9                ret
               ;
               ; Stop Clock
               ;
               stopclock:
                                      ; will stop flag #1 setting at
                                      ;  system time unit tick
0213 AF                xra     a
0214 322F04            sta     tickn
0217 C9                ret
               ;
               ; Exit Region
               ;
               exitregion:
                                      ; EI if not preempted or in disable
                                      ;  interrupt if preempted
0218 3A3104            lda     preemp
021B B7                ora     a
021C C0                rnz
021D FB                ei
021E C9                ret
               ; Maximum Console Number
               ;
               maxconsole:
021F 3E03              mvi     a,nmbcns
0221 C9                ret
               ;
               ; System Initialization
               ;
               systeminit:
               ;
               ; This is the place to insert code to initialize
               ; the time of day clock, if it is desired on each
               ; booting of the system.
               ;
0222 3EC3              mvi     a,0c3h
0224 323800            sta     0038h
0227 214702            lxi     h,inthnd
022A 223900            shld    0039h           ; JMP INTHND at 0038H

022D 0E90              mvi     c,create
                       if      debug
022F 111101            lxi     d,c2inpd
                       else
                       lxi     d,c0inpd
                       endif
0232 CD5500            call    xdos

0235 3A3004    lda             intmsk
```

```
0238 D360                       out 60h                         ; init interrupt mask

023A ED56                       db      0edh,056h               ; Interrupt Mode 1
                                                                ; ** Z80 Instruction *
023C FB                         ei
023D CDCA02                     call    home
0240 0E84                       mvi     c,flagwait
0242 1E05                       mvi     e,5
0244 C35500                     jmp     xdos                    ; clear first disk int
                        ;                                       ;   & return


                        ;
                        ; Idle procedure
                        ;
                        ;idle:
                        ;       ret

                        ;       -or-

                        ;       ei
                        ;       hlt
                        ;       ret                             ; for full interrupt system


                        ;
                        ; MP/M II V2.0   Interrupt Handlers
                        ;

0084 =                  flagwait equ    132
0085 =                  flagset equ     133
008E =                  dsptchq equ     142

                        inthnd:
                                                        ; Interrupt handler entry point
                                                        ;  All interrupts gen a RST 7
                                                        ;  Location 0038H contains a jmp
                                                        ;  to INTHND.
0247 222904                     shld    svdhl
024A E1                         pop     h
024B 222D04                     shld    svdret
024E F5                         push    psw
024F 210000                     lxi     h,0
0252 39                         dad     sp
0253 222B04                     shld    svdsp                   ; save users stk ptr
0256 312904                     lxi     sp,lstintstk            ; lcl stk for intr hnd
0259 D5                         push    d
025A C5                         push    b

025B 3EFF                       mvi     a,0ffh
025D 323104                     sta     preemp      ; set preempted flag

0260 DB60                       in      60h                     ; read interrupt mask
0262 E640                       ani     01000000b               ; test & jump if clk idle
0264 C28F02                     jnz     clk60hz
                        ;
0267 DB80                       in      stat                    ; read disk status port
```

```
0269 E608                   ani     08h
026B C27802                 jnz     diskintr

                            if      not debug
                            in      sts0
                            ani     2
                            jnz     con0in
                            endif

026E DB51                   in      sts2
0270 E602                   ani     2
0272 C28002                 jnz     con2in

                    ;       ...                     ; test/handle other interrupt
                    ;
0275 C3B502                 jmp     intdone

                    diskintr:
0278 AF                     xra     a
0279 D380                   out     cmdl            ; reset disk interrupt
027B 1E05                   mvi     e,5
027D C38702                 jmp     concmn          ; set flag #5

                            if      not debug
                    con0in:
                            in      data0
                            sta     ch0in
                            mvi     e,6
                            jmp     concmn          ; set flag #6
                            endif

                    con2in:
0280 DB50                   in      data2
0282 327901                 sta     ch2in
0285 1E08                   mvi     e,8
                    ;       jmp     concmn          ; set flag #8

                    concmn:
0287 0E85                   mvi     c,flagset
0289 CD5500                 call    xdos
028C C3B502                 jmp     intdone

                    clk60hz:
                                                    ; 60 Hz clock interrupt
028F 3A2F04                 lda     tickn
0292 B7                     ora     a               ; test tickn, indicate
                                                    ;  delayed process(es)
0293 CA9D02                 jz      notickn
0296 0E85                   mvi     c,flagset
0298 1E01                   mvi     e,1
029A CD5500                 call    xdos            ; set flag #1 each tick
                    notickn:
029D 210004                 lxi     h,cnt60
02A0 35                     dcr     m               ; dec 60 tick cntr
02A1 C2AD02                 jnz     notlsec
```

```
02A4 363C               mvi     m,60
02A6 0E85               mvi     c,flagset
02A8 1E02               mvi     e,2
02AA CD5500             call    xdos            ; set flag #2 @ 1 sec
                notlsec:
02AD AF                 xra     a
02AE D360               out     60h
02B0 3A3004             lda     intmsk
02B3 D360               out     60h             ; ack clock interrupt
                ;       jmp     intdone
                ;
                ;       ...
                ; Other Interrupt handlers
                ;       ...
                ;
                intdone:
02B5 AF                 xra     a
02B6 323104             sta     preempt  ; clear preempted flag
02B9 C1                 pop     b
02BA D1                 pop     d
02BB 2A2B04             lhld    svdsp
02BE F9                 sphl                    ; restore stk ptr
02BF F1                 pop     psw
02C0 2A2D04             lhldd   svdret
02C3 E5                 push    h
02C4 SA2904             lhld    svdhl
                ; The following dispatch call will force round robin
                ;  scheduling of processes executing at the same prior
                ;  each 1/60th of a second.
                ; Note:  Interrupts are not enabled until the dispatcher
                ;  resumes the next process.  This prevents interrupt
                ;  over-run of the stacks when stuck or high frequency
                ;  interrupts are encountered.
02C7 C35200             jmp     pdisp           ; MP/M dispatch
                ;
                ;
                ;       Disk I/O Drivers
                ;
                ; Disk Port Equates
                ;
0080 =          cmdl    equ     80h
0080 =          stat    equ     80h
0081 =          haddr   equ     81h
0082 =          laddr   equ     82h
0083 =          cmd2    equ     83h
                ;
                ;
                home:   ;move to the track o0 position of current drive
02CA CDA03              call    headload
                ; h,l point to word with track for selected disk
                homel:
02CD 3600               mvi     m,00    ;set current track ptr back to
02CF DB80               in      stat    ;read fdc status
02D1 E604               ani     4       ;test track 0 bit
02D3 C8                 rz              ;return if at 0
```

```
02D4 37                stc            ;direction=out
02D5 CDC203            call    step   ;step one track
02D8 C3CD02            jmp     homel  ;loop
               ;
               seldsk:
                       ;drive number in c
02DB 210000            lxi     h,0    ;0000 in hl produces select error
02DE 79                mov     a,c    ;a is disk number 0 ... ndisks
02DF FE02              cpi     ndisks ;less than ndisks?
02E1 D0                rnc            ;return with HL = 0000 if not
               ;make sure dummy is 0 (for use in double add to h,l)
02E2 AF                xra     a
02E3 323A04            sta     dummy
02E6 79                mov     a,c
02E7 E607              ani     07h    ;get only disk select bits
02E9 323904            sta     diskno
02EC 4F                mov     c,a
               ;set up the second command port
02ED 3A3C04            lda     port
02F0 E6F0              ani     0f0h   ;clear out old disk select bit
02F2 B1                ora     c      ;put in new disk select bits
02F3 F608              ori     08h    ; force double density
02F5 323C04            sta     port
               ;       proper disk number, return dpb element address
02F8 69                mov     l,c
02F9 29                dad     h      ;*2
02FA 29                dad     h      ;*4
02FB 29                dad     h      ;*8
02FC 29                dad     h      ;*16
02FD 113F04            lxi     d,dpbase
0300 19                dad     d      ;HL=.dpb
0301 226E04            shld    tran   ;translate table base
0304 C9                ret
               ;
               ;
               ;
               settrk: ;set track given by register c
0305 CDDA03            call    headload
               ;h,l reference correct track indicator according to
               ;selected disk
0308 79                mov     a,c    ;desired track
0309 BE                cmp     m
030A C8                rz             ;we are already on the track
               settkx:
030B CDC203            call    step   ;step track-carry has direction
                                      ;step will update trk indicator
030E 79                mov     a,c
030F BE                cmp     m      ;are we where we want to be
0310 C20B03            jnz     settkx ;not yet
               ;have stepped enough
               seekrt:
               ;need 10 msec delay for final step time and head settle
0313 3E14              mvi     a,20d
               ;       call    delay
               ;       ret            ;end of settrk routine
```

```
                    ;
                    delay:  ;delay for c[A] X .5 milliseconds
0315 C5                     push    b
                    delay1:
0316 0E86                   mvi     c,dlycnst ;constant adjusted to .5 ms
                    delay2:
0318 0D                     dcr     c
0319 C21803                 jnz     delay2
031C 3D                     dcr     a
031D C21603                 jnz     delay1
0320 C1                     pop     b
0321 C9                     ret             ;end of delay routine

                    ;
                    setsec: ;set sector given by register c
0322 0C                     inr     c
0323 79                     mov     a,c
0324 323604                 sta     sector
0327 C9                     ret
                    ;
                    sectran:
                            ;sector number in c
                            ;translate logical to physical sector
0328 2A6E04                 lhld    tran    ;hl=..translate
032B 5E                     mov     e,m     ;E=low(.translate)
032C 23                     inx     h
032D 56                     mov     d,m     ;DE=.translate
032E 7B                     mov     a,e     ;zero?
032F B2                     ora     d       ;00 or 00 = 00
0330 2600                   mvi     h,0
0332 69                     mov     l,c     ;HL = untranslated sector
0333 C8                     rz              ;skip if so
0334 EB                     xchg
0335 42                     mov     b,d     ;BC=00ss
0336 09                     dad     b       ;HL=.translate(sector)
0337 6E                     mov     l,m
0338 62                     mov     h,d     ;HL=translate(sector)
0339 C9                     ret
                    ;
                    setdma: ;set dma address given by registers b and c
033A 69                     mov     l,c     ;low order address
033B 60                     mov     h,b     ;high order address
033C 223704                 shld    dmaad   ;save the address
033F C9                     ret
                    ;
                    ;
                    read:   ;perform read operation.
                            ;this is similar to write, so set up read
                            ; command and use common code in write
0340 0640                   mvi     b,040h ;set read flag
0342 C34703                 jmp     waitio ;to perform the actual I/O
                    ;
                    write:  ;perform a write operation
0345 0680                   mvi     b,080h ;set write command
```

```
               ;
               waitio:
               ;enter here from read and write to perform the actual
               ; I/O operation.  return a 00h in register a if the
               ; operation completes properly, and 01h if an error
               ; occurs during the read or write
               ;
               ;in this case, the disk number save in 'diskno'
               ;                       the track number in 'track'
               ;                       the sector number in 'sector'
               ;                       the dma address in 'dmaad'
               ;                       ;b still has r/w flag
0347 3E0A              mvi     a,10d   ;set error count
0349 323B04            sta     errors  ;retry some failures 10 times
                                       ;before giving up
               tryagn:
034C C5                push    b
034D CDDA03            call    headload
               ;h,l point to track byte for selected disk
0350 C1                pop     b
0351 4E                mov     c,m
               ;decide whether to allow disk write precompensation
0352 3E27              mvi     a,39d   ;inhibit precomp on trks 0-39
0354 B9                cmp     c
0355 DA5C03            jc      allowit
               ;inhibit precomp
0358 3E10              mvi     a,10h
035A B0                ora     b
035B 47                mov     b,a     ;goes out on the same port
                                       ; as read/write
               allowit:
035C 2A3704            lhld    dmaad   ;get buffer address
035F C5                push    b       ;b has r/w code   c has track
0360 2B                dcx     h       ;save and replace 3 bytes belo
                                       ;buf with trk,sectr,adr mark
0361 5E                mov     e,m
               ;figure correct address mark

0362 3A3C04            lda     port
0365 E608              ani     08h
0367 3EFB              mvi     a,0fbh
0369 CA6E03            jz      sin
036C E60F              ani     0fh     ;was double
                                       ;0bh is double density
                                       ;0fbh is single density
               sin:
036E 77                mov     m,a
               ;fill in sector
036F 2B                dcx     h
0370 56                mov     d,m
0371 3A3604            lda     sector  ;note that invalid sector number
                                       ;will result in head unloaded
                                       ;error, so dont check
0374 77                mov     m,a
               ;fill in track
```

```
0375 2B                 dcx     h
0376 C1                 pop     b
0377 79                 mov     a,c
0378 4E                 mov     c,m
0379 77                 mov     m,a
037A 7C                 mov     a,h     ;set up fdc dma address
037B D381               out     haddr   ;high byte
037D 7D                 mov     a,l
037E D382               out     laddr   ;low byte
0380 78                 mov     a,b     ;get r/w flag
0381 D380               out     cmdl    ;start disk read/write

                rwwait:
0383 C5                 push    b
0384 D5                 push    d
0385 E5                 push    h

0386 0E84               mvi     c,flagwait
0388 1E05               mvi     e,5
038A CD5500             call    xdos            ; wait for disk intrpt

038D E1                 pop     h
038E D1                 pop     d
038F C1                 pop     b
0390 71                 mov     m,c     ;restore 3 bytes below buf
0391 23                 inx     h
0392 72                 mov     m,d
0393 23                 inx     h
0394 73                 mov     m,e
0395 DB80               in      stat    ;test for errors
0397 E6F0               ani     0f0h
0399 C8                 rz              ;a will be 0 if no errors

                ; error from disk
039A F5                 push    psw     ;save error condition
                ;check for 10 errors
039B 213B04             lxi     h,errors
039E 35                 dcr     m
039F C2A603             jnz     redo    ;not ten yet.  do a retry
                ;we have too many errors. print out hex number for last
                ;received error type. cpm will print perm error message
03A2 F1                 pop     psw     ;get code
                ;set error return for operating system
03A3 3E01               mvi     a,1
03A5 C9                 ret
                redo:
                ;b still has read/write flag
03A6 F1                 pop     psw     ;get error code
03A7 E6E0               ani     0e0h    ;retry if not track error
03A9 C24C03             jnz     tryagn  ;
                ;was a track error so need to reseek
03AC C5                 push    b       ;save   read/write indicator
                ;figure out the desired track
03AD 113204             lxi     d,track
03B0 2A3904             lhld    diskno  ;selected disk
```

```
03B3 19                   dad     d         ;point to correct trk indicator
03B4 7E                   mov     a,m       ;desired track
03B5 F5                   push    psw       ;save it
03B6 CDCA02               call    home
03B9 F1                   pop     psw
03BA 4F                   mov     c,a
03BB CD0503               call    settrk
03BE C1                   pop     b         ;get read/write indicator
03BF C34C03               jmp     tryagn
                  ;
                  ;
                  ;
                  step:                     ;step head out towards zero
                                            ;if carry is set; else
                                            ;step in
                  ; h,l point to correct track indicator word
03C2 DAD503               jc      outx
03C5 34                   inr     m         ;increment current track byte
03C6 3E04                 mvi     a,04h     ;set direction = in
                  dostep:
03C8 F602                 ori     2
03CA D380                 out     cmdl      ;pulse step bit
03CC E6FD                 ani     0fdh
03CE D380                 out     cmdl      ;turn off pulse
                  ;the fdc-2 had a stepp ready line.  the fdc-3 relies on
                  ;software time out
03D0 3E10                 mvi     a,16d     ;delay 8 ms
03DS C31503               jmp     delay
                  ;       ret
                  ;
                  outx:
03D5 35                   dcr     m         ;update track byte
03D6 AF                   xra     a
03D7 C3C803               jmp     dostep
                  ;
                  headload:
                  ;select and load the head on the correct drive
03DA 213D04               lxi     h,prtout          ;old slect info
03DD 46                   mov     b,m
03DE 2B                   dcx     h         ;new select info
03DF 7E                   mov     a,m
03E0 23                   inx     h
03E1 77                   mov     m,a

03E2 F610                 ori     10h       ; enable interrupt

03E4 D383                 out     cmd2      ;select the drive
03E6 E6EF                 ani     0efh
                  ;set up h.l to point to track byte for selected disk
03E8 113204               lxi     d,track
03EB 2A3904               lhld    diskno
03EE 19                   dad     d
                  ;now check for needing a 35 ms delay
                  ;if we have changed drives or if the head is unloaded
                  ;we need to wait 35 ms for head settle
```

```
03EF B8                cmp     b       ;are we on the same drive
03F0 C2F803            jnz     needdly
               ;we are on the same drive
               ;is the head loaded?
03F3 DB80              in      stat
03F5 E680              ani     80h
0EF7 C8                rz              ;already loaded
               needdly:
03F8 AF                xra     a
03F9 D380              out     cmdl    ;load the head
03FB 3E46              mvi     a,70d
03FD C31503            jmp     delay
               ;       ret

               ; BIOS Data Segment
               ;
0400 3C        cnt60:  db      60      ; 60 tick cntr = 1 sec
               intstk:                 ; local intrpt stk
0401 C7C7C7C7C7        dw      0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h
040B C7C7C7C7C7        dw      0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h
0415 C7C7C7C7C7        dw      0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h
041F C7C7C7C7C7        dw      0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h
               lstintstk:
0429 0000      svdhl:  dw      0       ; saved Regs HL during int hnd
042B 0000      svdsp:  dw      0       ; saved SP during int hndl
042D 0000      svdret: dw      0       ; saved return during int hndl
042F 00        tickn:  db      0       ; ticking boolean,true = delay
                       if      debug
0430 44        intmsk: db      44h     ; intrpt msk, enables clk intrpt
                       else
               intmsk: db      54h     ; intrpt msk, enables clk intrpt
                       endif
0431 00        preemp:db       0       ; preempted boolean
               ;
               scrat:                  ; start of scratch area
0432 00        track:  db      0       ; current trk on drive 0
0433 00        trak1:  db      0       ; current trk on drive 1
0434 00        trak2:  db      0
0435 00        trak3:  db      0
0436 00        sector: db      0       ; currently selected sctr
0437 0000      dmaad:  dw      0       ; current dma address
0439 00        diskno: db      0       ; current disk number
043A 00        dummy:  db      0       ; must be 0 for dbl add
043B 00        errors: db      0
043C 00        port:   db      0
043D 00        prtout; db      0
043E 00        dnsty:  db      0
               ;
                        disks   2
043F+=          DPBASE  EQU     $       ;BASE OF DISK PARAMETER BLOCKS
043F+00000000   DPE0:   DW      XLT0,0000H      ;TRANSLATE TABLE
0443+00000000           DW      0000H,0000H     ;SCRATCH AREA
0447+70045F04           DW      DIRBUF,DPB0     ;DIR BUFF,PARM BLOCK
044B+1005F004           DW      CSV0,ALV0       ;CHECK, ALLOC VECTORS
```

```
044F+00000000  DPE1:   DW      XLT1,0000H      ;TRANSLATE TABLE
0453+00000000          DW      0000H,0000H     ;SCRATCH AREA
0457+70045F04          DW      DIRBUF,DPB1     ;DIR BUFF,PARM BLOCK
045B+50053005          DW      CSV1,ALV1       ;CHECK, ALLOC VECTORS
0800 =         bpb     equ     2*1024  ;bytes per block
0010 =         rpb     equ     bpb/128 ;records per block
00FF =         maxb    equ     255     ;max block number
                       diskdef 0,1,58,,bpb,maxb+1,128,128,2,0
045F+=         DPB0    EQU     $               ;DISK PARM BLOCK
045F+3A00              DW      58              ;SEC PER TRACK
0461+04                DB      4               ;BLOCK SHIFT
0462+0F                DB      15              ;BLOCK MASK
0463+00                DB      0               ;EXTNT MASK
0464+FF00              DW      255             ;DISK SIZE-1
0466+7F00              DW      127             ;DIRECTORY MAX
0468+C0                DB      192             ;ALLOC0
0469+00                DB      0               ;ALLOC1
046A+2000              DW      32              ;CHECK SIZE
046C+0200              DW      2               ;OFFSET
0000+=         XLT0    EQU     0               ;NO XLATE TABLE
                       diskdef 1,0
045F+=         DPB1    EQU     DPB0    ;EQUIVALENT PARAMETERS
0020+=         ALS1    EQU     ALS0    ;SAME ALLOCATION VECTOR SIZE
0020+=         CSS1    EQU     CSS0    ;SAME CHECKSUM VECTOR SIZE
0000+=         XLT1    EQU     XLT0    ;SAME TRANSLATE TABLE
               ;
046E           tran:   ds      2
               ;
                       endef
0470+=         BEGDAT  EQU     $
0470+          DIRBUF: DS      128     ;DIRECTORY ACCESS BUFFER
ALV0:          ALV0:   DS      32
0510+          CSV0:   DS      32
0530+          ALV1:   DS      32
0550+          CSV1:   DS      32
0570+=         ENDDAT  EQU     $
0100+=         DATSIZ  EQU     $-BEGDAT
0570+00        FORCE:  DB      0       ;FORCE OUT LAST BYTE IN HEX FI

0571 00                db      0       ;force out last byte in hex fi

0572                   end
```

APPENDIX E

SAMPLE M/M II BANKED XIOS

```
        page    0
        TITLE   'XIOS200, Copyright 1980, DIGITAL RESEARCH
;--------------------------------------------------------------
;
;       DIGITAL RESEARCH
;       P.O. BOX 579, 801 LIGHTHOUSE AVENUE
;       PACIFIC GROVE, CALIFORNIA 93950
;
;       Copyright 1980, DIGITAL RESEARCH
;
;       This program is a copyright program product of
;       DIGITAL RESEARCH and is distributed to the
;       owners of DYNABYTE computers for use as an
;       example only.  Any other use of this software
;       constitutes a breach of the copyright license
;       to the purchaser.  However, permission is
;       granted to use this listing as a sample for the
;       construction of the reader's own XIOS.
;
;       VERSION NUMBER: 1.12*
;       VERSION DATE:   June 28, 1980
;       .       Add support for CP/M version 1.0
;       .       Add support for Hard disk drives
;       .       Add support for disk MODE selection
;       .       Provide compatability MODE for 1.4 operation
;       .       Remove CTC/1791 counter reset
;       .       CORRECT HARD DISK SEEK PROBLEM
;       .       Add code to recover from WD1791 going to sleep
;       .       Initialize parallel port for Centronics printer
;       VERSION DATE:   March 17, 1981
;       .       Virtual disk in banks 1,2,3: M DISK !;
;       VERSION DATE:   April 11, 1981
;       .       Conditional assembly for virtual disks
;       .       Conditional assembly for MP/M 2.0
;       VERSION DATE:   April 14, 1981
;       .       Equates added for LDRBIOS hooks  !
;       VERSION DATE:   April 16, 1981
;       .       Testing for bank setup added
;
;--------------------------------------------------------------
;
;
;--------------------------------------------------------------
;
;       Mode    0       single density
;               1       double density Version 2.0
```

```
                ;               2       double density Version 1.4
                ;               3       hard disk Version 2.0 (8 MEG
                ;               4       HARD DISK VERSION 2.0 (8 MEG
                ;               5       HARD DISK VERSION 2.0 (8 MEG
                ;               6       HARD DISK VERSION 2.0 (4 MEG
                ;
                ;----------------------------------------------------------

                ;----------------------------------------------------------
                ;
                ;       ASSEMBLER CONTROL STATEMENTS
                ;
                ;----------------------------------------------------------

                        MACLIB DISKDEG
                        MACLIB Z80S

FFFF =          TRUE    EQU     0FFFFH          ;VALUE FOR TRUE
0000 =          FALSE   EQU     NOT TRUE        ;VALUE FOR FALSE

0000 =          mdisk   equ     false           ;Virtual Disk cond asm boolean
FFFF =          mpm20   equ     true            ;MP/M 2.0 cond asm boolean

                ;----------------------------------------------------------
1700 =          ldrbiosbase equ 1700h   ;                       for M

0037 =          density$mask$offset equ 37h ;density mask offset from LDRBIOS
00BB =          misc$params$offset equ 0bbh ;misc. parameters offset from LDRBI

                ;----------------------------------------------------------
                ;
                ;
                ;       THE FOLLOWING EQUATES ARE USER MODIFIABLE BASED ON
                ;       PARTICULAR USER SYSTEM AND OPTIONS SELECTED.
                ;
                ;----------------------------------------------------------

FFFF =          DMA     EQU     TRUE            ;DMA HARDWARE SUPPORT ??
FFFF =          HARDSK  EQU     TRUE            ;HARD DISK SUPPORT

                ;----------------------------------------------------------
                ;
                ;       THE FOLLOWING CONSTANTS APPLY TO THE DEBLOCKING
                ;       OF SECTORS LARGER THAN 128 FOR DOUBLE DENSITY
                ;       AND HARD DISK.
                ;
                ;----------------------------------------------------------

4000 =          BLKSIZ  EQU     16384           ;CP/M ALLOCATION SIZE
0400 =          HSTSIZ  EQU     1024            ;HOST DISK SECTOR SIZE
0010 =          HSTSPT  EQU     16              ;HOST DISK SECTORS PER TRACK
0008 =          HSTBLK  EQU     HSTSIZ/128      ;CP/M SECTORS PER HOST BUFF
0080 =          CPMSPT  EQU     HSTBLK * HSTSPT ;CP/M SECTORS PER TRACK
0007 =          SECMSK  EQU     HSTBLK - 1      ;SECTOR MASK
0003 =          SECSHF  EQU     3               ;LOG2(HHSTBLK)
```

```
                        PAGE

                ;-------------------------------------------------------
                ;
                ;       THE FOLLOWING EQUATES APPLY TO THE RELOCATABLITY
                ;       OF THE CBIOS AND SHOULD NOT BE USER ALTERED.
                ;
                ;-------------------------------------------------------

FFFF =          RELOC   EQU     TRUE            ;RELOCATABLE VERSION ??

                ;-------------------------------------------------------

                        if      mdisk
                maxdsk  equ     13
                        else
                        IF      HARDSK
000C =          MAXDSK  EQU     12              ;MAXIMUM NUMBER OF LOGICAL DRIVE
                        ELSE
                MAXDSK  EQU     4               ;MAXIMUM NUMBER OF LOGICAL DRIVE
                        ENDIF
                        endif

                        IF      RELOC
0000                    ORG     0000H
                        ELSE
                        ORG     0C000H
                        ENDIF

0000 =          BASE    EQU     $

                ;-------------------------------------------------------

0000 =          WRALL   EQU     0               ;WRITE TO ALLOCATED
0001 =          WRDIR   EQU     1               ;WRITE TO DIRECTORY
0002 =          WRUAL   EQU     2               ;WRITE TO UNALLOCATED


0004 =          NMBCNS  EQU     4       ; NUMBER OF CONSOLES

0083 =          POLL    EQU     131     ; XDOS POLL FUNCTION
0084 =          FLAGWT  EQU     132     ; XDOS FLAG WAIT FUNCTION
0085 =          FLAGST  EQU     133     ; XDOS FLAG SET FUNCTION

0005 =          HDFLAG  EQU     5       ;HARD DISK FLAG FOR WAIT & SET
0006 =          FPYFLAG EQU     6       ;FLOPPY DISK FLAG FOR WAIT & SET

0000 =          PLLPT   EQU     0       ; POLL PRINTER
0001 =          PLCO0   EQU     PLLPT+1 ; POLL CONSOLE OUT #0 (CRT:)
0002 =          PLCO1   EQU     PLCO0+1 ; POLL CONSOLE OUT #1 (CRT:)
0003 =          PLCO2   EQU     PLCO1+1 ; POLL CONSOLE OUT #2 (CRT:)
0004 =          PLCO3   EQU     PLCO2+1 ; POLL CONSOLE OUT #3 (CRT:)
0005 =          PLCI0   EQU     PLCO3+1 ; POLL CONSOLE IN #0 (CRT:)
0006 =          PLCI1   EQU     PLCI0+1 ; POLL CONSOLE IN #1 (CRT:)
```

```
0007 =          PLCI2   EQU     PLCI1+1 ; POLL CONSOLE IN #2 (CRT:)
0008 =          PLCI3   EQU     PLCI2+1 ; POLL CONSOLE IN #3 (CRT:)

0009 =          MEMPORT EQU     009H    ; MEMORY SELECT PORT
0002 =          MEMSK   EQU     002H    ; MEMORY SELECT MASK


                        PAGE

                ;---------------------------------------------------------
                ;
                ;       JUMP VECTORS FOR ENTRIES TO CBIOS ROUTINES
                ;
                ;---------------------------------------------------------

                ;       EXTERNAL JUMP TABLE (BELOW XIOS BASE)

                ;PDISP  EQU     $-3
                ;XDOS   EQU     PDISP-3

                        if      mpm20
0000 C3040B             jmp     commonbase
                        else
                        JMP     COLDSTART       ;COLD START
                        endif
                WBOTE:
0003 C3150B             JMP     WARMSTART       ;WARM START
0006 C3790B             JMP     CONST           ;CONSOLE STATUS
0009 C3840B             JMP     CONIN           ;CONSOLE CHARACTER IN
000C C38F0B             JMP     CONOUT          ;CONSOLE CHARACTER OUT
000F C3A90C             JMP     LIST            ;LIST CHARACTER OUT - THIS
                ;                               ;  "CLIST" IF SETUP PROGRAM
                ;                               ;  PARALLEL PRINTER PORT

0012 C31A0B             JMP     RTNEMPTY        ;PUNCH NOT IMPLEMENTED
0015 C31A0B             JMP     RTNEMPTY        ;READER NOT IMPLEMENTED
0018 C3F902             JMP     HOMEIT          ;MOVE HEAD TO HOME
001B C30302             JMP     SELDSK          ;SELECT DISK
001E C36D02             JMP     SETTRK          ;SET TRACK NUMBER
0021 C37302             JMP     SETSEC          ;SET SECTOR NUMBER
0024 C35502             JMP     SETDMA          ;SET DMA ADDRESS
0027 C38B02             JMP     READ            ;READ DISK
002A C39602             JMP     WRITE           ;WRITE DISK
002D C3BC0C             JMP     POLLPT          ;LIST STATUS
0030 C3D605             JMP     SECTRAN         ;SECTOR TRANSLATE

                ;       EXTENDED I/O SYSTEM JUMP VECTOR

0033 C3E90C             JMP     SELMEMORY       ; SELECT MEMORY
0036 C3CB0C             JMP     POLLDEVICE      ; POLL DEVICE
0039 C3050D             JMP     STARTCLOCK      ; START CLOCK
003C C30B0D             JMP     STOPCLOCK       ; STOP CLOCK
003F C3100D             JMP     EXITREGION      ; EXIT REGION
0042 C3170D             JMP     MAXCONSOLE      ; MAXIMUM CONSOLE NUMBER
```

```
0045 C39D12                   JMP       SYSTEMINIT          ; SYSTEM INITIALIZATION
0048 00                       NOP                           ; NO JMP HERE
0049 00                       NOP                           ; FOR MP/M DELAY
004A 00                       NOP                           ;


004B C3A102                   JMP       SETMOD              ;ROUTINE TO SET DISK MODE
004E C3EE02                   JMP       RETMOD              ;ROUTINE TO RETURN CURRENT

                              if        not mpm20
                    COLDSTART:
                    WARMSTART:
                              MVI       C,0                 ; SEE SYSTEM INIT
                                                            ; COLD & WARM START INCLUDE
                                                            ; FOR COMPATIBILITY WITH CP
                              JMP       XDOS                ; SYSTEM RESET,TERMINATE PROCESS

                    RTNEMPTY:
                              XRA       A                   ; NOT USED
                              RET                           ;
                              endif

                    LAST:
005E                          ORG       (((LAST-BASE)+0A2H) AND 0FF00H) +05EH

                    INTERUPT:
005E 470B                     DW        FLOPPY$INT          ;FLOPPY DISK INTERRUPT
0060 1C0BF                    DW        NULL$INT            ;
0062 1C0B                     DW        NULL$INT            ;
0064 1C0B                     DW        NULL$INT            ;
0066 1A0D                     DW        INT1HND             ;CTC INTERRUPT
0068 1C0B                     DW        NULL$INT            ;
006A 5E0B                     DW        HARD$INT            ;HARD DISK INTERRUPT
006C 1C0B                     DW        NULL$INT            ;
006E 1C0B                     DW        NULL$INT            ;

                              if        not mpm20
                    NULL$INT:
                              EI
                              RETI
                              endif

                              PAGE


                    ;-------------------------------------------------------------
                    ;
                    ;         WORK AND CONTROL AREAS FOR CBIOS SERVICES
                    ;
                    ;-------------------------------------------------------------

0070 FFFFFFFFFFTRK0:          DB        0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FF
```

```
007C 0408102010SELO:   DB      004H,008H,010H,020H,010H,010H,010H,020H,020
0088 0000000003MODE:   DB      000H,000H,000H,000H,003H,004H,005H,003H,004
0094 0000000000TCNT:   DB      000H,000H,000H,000H,000H,000H,000H,000H,000
00A0 0000000000PCNT:   DB      000H,000H,000H,000H,000H,000H,000H,000H,000

00AC 00           DISKNO: DB      000H                              ;CURRENT DR
00AD 00           TRAKNO: DB      000H                              ;CURRENT TR
00AE 00           HEADNO: DB      000H                              ;CURRENT HE
00AF 0000         DMAADR: DW      000H                              ;CURRENT DM
00B1 00           SECTNO: DB      000H                              ;CURRENT SE
00B2 0000         DPEPTR: DW      000H                              ;CURRENT DP
00B4 0000         DBLKAD: DW      000H                              ;CURRENT EX
00B6 0000         MPARMS: DW      000H                              ;MISC. PARA
00B8 10           HTK1:   DB      10H                               ;HARD DISK
00B9 20           HTK2:   DB      20H                               ;HARD DISK
                  ;
                  ;       PARAMETER FLAGS
                  ;
                  ;       0100H = DOUBLE HEADED DRIVES
                  ;       0200H = CENTRONICS PRINTER FOR LIST DEVICE
                  ;       0400H = FOUR DRIVE SYSTEM [ A B C D ]
                  ;--------------------------------------------------------------
                  ;
                  ;--------------------------------------------------------------
                  ;
                  ;       NOTE:
                  ;       NO CHANGES ARE TO BE MADE TO THE ABSOLUTE LOCATIONS OF
                  ;       ANY FIELDS PRIOR TO THIS POINT.  EXTERNAL PROGRAMS ARE
                  ;       DEPENDENT UPON THE LOCATION OF THE PRECEDING DATA.
                  ;
                  ;--------------------------------------------------------------

                          IF      NOT DMA
                  NMIRTN: DB      0EDH,0A2H,0EDH,045H               ;FAKE INI A
                          ENDIF

00BA C37D         DMAS1:  DB      0C3H,07DH                         ;FIRST PART
00BC 0000         DMASA:  DW      000H                              ;ADDRESS FO
00BE 0004         DMALEN: DW      1025-1                            ;LENGTH FOR

00C0 54CE68CEA5DMAS2H: DB      054H,0CEH,068H,0CEH;,0A5H,020H    ;HARD DISK

00C6 14288507     DMAS2F: DB      014H,028H,085H,007H               ;FLOPPY DISK

00CA 8ACF01CF     DMAS3:  DB      08AH,0CFH,001H,0CFH               ;LAST PART
00CE 01           DMAS3F: DB      001H                              ;001=READ,
00CF CF87                 DB      J0CFH,087H                        ;SETUP DMA,

                          PAGE

                  ;--------------------------------------------------------------
                  ;
                  ;       CONTROL BLOCKS FOR DISK DRIVER
                  ;
                  ;--------------------------------------------------------------
```

```
00D1 =           DPBASE  EQU     $                  ;START OF DISK PARAMETER BLOCK

00D1 B5010000    DPE0:   DW      XLT0,0000H         ;TRANSLATE TABLE AND WORK AREA
00D5 00000000            DW      0000H,0000H        ;SCRATCH AREA
00D9 9D12D40D            DW      DIRBUF,DPB0        ;DIR BUFF, PARM BLOCK
00DD 3E081E08            DW      CSV0,ALV0          ;CHECK VECTOR, ALLOC VECTOR

00E1 B5010000    DPE1:   DW      XLT0,0000H         ;TRANSLATE TABLE AND WORK AREA
00E5 00000000            DW      0000H,0000H        ;SCRATCH AREA
00E9 9D12D40D            DW      DIRBUF,DPB0        ;DIR BUFF, PARM BLOCK
00ED 7E085E08            DW      CSV1,ALV1          ;CHECK VECTOR, ALLOC VECTOR

00F1 B5010000    DPE2:   DW      XLT0,0000H         ;TRANSLATE TABLE AND WORK AREA
00F5 00000000            DW      0000H,0000H        ;SCRATCH AREA
00F9 9D12D40D            DW      DIRBUF,DPB0        ;DIR BUFF, PARM BLOCK
00FD BE089E08            DW      CSV2,ALV2          ;CHECK VECTOR, ALLOC VECTOR

0101 B5010000    DPE3:   DW      XLT0,0000H         ;TRANSLATE TABLE AND WORK AREA
0105 00000000            DW      0000H,0000H        ;SCRATCH AREA
0109 9D12D40D            DW      DIRBUF,DPB0        ;DIR BUFF, PARM BLOCK
010D FE089E08            DW      CSV3,ALV3          ;CHECK VECTOR, ALLOC VECTOR

                         IF      HARDSK

0111 00000000    DPE4:   DW      0000H,0000H        ;TRANSLATE TABLE AND WORK AREA
0115 00000000            DW      0000H,0000H        ;SCRATCH AREA
0119 9D12010E            DW      DIRBUF,DPB3        ;DIR BUFF, PARM BLOCK
011D 5E091E09            DW      CSV4,ALV4          ;CHECK VECTOR, ALLOC VECTOR

0121 00000000    DPE5:   DW      0000H,0000H        ;TRANSLATE TABLE AND WORK AREA
0125 00000000            DW      0000H,0000H        ;SCRATCH AREA
0129 9D12100E            DW      DIRBUF,DPB4        ;DIR BUFF, PARM BLOCK
012D 9E095E09            DW      CSV5,ALV5          ;CHECK VECTOR, ALLOC VECTOR

0131 00000000    DPE6:   DW      0000H,0000H        ;TRANSLATE TABLE AND WORK AREA
0135 00000000            DW      0000H,0000H        ;SCRATCH AREA
0139 9D121F0E            DW      DIRBUF,DPB5        ;DIR BUFF, PARM BLOCK
013D DE099E09            DW      CSV6,ALV6          ;CHECK VECTOR, ALLOC VECTOR

0141 00000000    DPE7:   DW      0000H,0000H        ;TRANSLATE TABLE AND WORK AREA
0145 00000000            DW      0000H,0000H        ;SCRATCH AREA
0149 9D12010E            DW      DIRBUF,DPB3        ;DIR BUFF, PARM BLOCK
014D 1E0ADE09            DW      CSV7,ALV7          ;CHECK VECTOR, ALLOC VECTOR

0151 00000000    DPE8:   DW      0000H,0000H        ;TRANSLATE TABLE AND WORK AREA
0155 00000000            DW      0000H,0000H        ;SCRATCH AREA
0159 9D12100E            DW      DIRBUF,DPB4        ;DIR BUFF, PARM BLOCK
015D 5E0A1E0A            DW      CSV8,ALV8          ;CHECK VECTOR, ALLOC VECTOR

0161 00000000    DPE9:   DW      0000H,0000H        ;TRANSLATE TABLE AND WORK AREA
0165 00000000            DW      0000H,0000H        ;SCRATCH AREA
0169 9D121F0E            DW      DIRBUF,DPB5        ;DIR BUFF, PARM BLOCK
016D 9E0A5E0A            DW      CSV9,ALV9          ;CHECK VECTOR, ALLOC VECTOR
```

```
0171 00000000  DPEA:   DW       0000H,0000H      ;TRANSLATE TABLE AND WORK AREA
0175 00000000          DW       0000H,0000H      ;SCRATCH AREA
0179 9D122E0E          DW       DIRBUF,DPB6      ;DIR BUFF, PARM BLOCK
017D C20A9E0A          DW       CSVA,ALVA        ;CHECK VECTOR, ALLOC VECTOR

0181 00000000  DPEB:   DW       0000H,0000H      ;TRANSLATE TABLE AND WORK AREA
0185 00000000          DW       0000H,0000H      ;SCRATCH AREA
0189 9D122E0E          DW       DIRBUF,DPB6      ;DIR BUFF, PARM BLOCK
018D E60AC20A          DW       CSVB,ALVB        ;CHECK VECTOR, ALLOC VECTOR

               ENDIF

                       if       mdisk
               ;                Virtual disk parameter header

               DPEC:   DW       0000H,0000H      ;TRANSLATE TABLE AND WORK AREA
                       DW       0000H,0000H      ;SCRATCH AREA
                       DW       DIRBUF,DPB7      ;DIR BUFF, PARM BLOCK
                       DW       CSVC,ALVC        ;CHECK VECTOR, ALLOC VECTOR
               endif

               ;---------------------------------------------------------

0191 B5010000  MODL0:  DW       XLT0,000H        ;MODEL DPE FOR MODE 0
0195 00000000          DW       000H,000H        ;
0199 9D12D40D          DW       DIRBUF,DPB0      ;

019D CF010000  MODL1:  DW       XLT1,0000H       ;MODEL DPE FOR MODE 1
01A1 00000000          DW       0000H,0000H      ;
01A5 9D12E30D          DW       DIRBUF,DPB1      ;

01A9 CF010000  MODL2:  DW       XLT2,0000H       ;MODEL DPE FOR MODE 2
01AD 00000000          DW       0000H,0000H      ;
01B1 9D12F20D          DW       DIRBUF,DPB2      ;

               ;-----------------------------------------------------

01B5 01070D1319XLT0:   DB       1,7,13,19,25,5,11,17,23,3,9,15,21
01C2 02080E141A        DB       2,8,14,20,26,6,12,18,24,4,10,16,22

               XLT1:
               XLT2:
01CF 0102030405        DB       01,02,03,04,05,06,07,08,09,10,11,12,13
01DC 0E0F101112        DB       14,15,16,17,18,19,20,21,22,23,24,25,26
01E9 1B1C1D1E1F        DB       27,28,29,30,31,32,33,34,35,36,37,38,39
01F6 28292A2B2C        DB       40,41,42,43,44,45,46,47,48,49,50,51,52

                       PAGE

               ;---------------------------------------------------------
               ;
               ;       DISK ACCESS ROUTINES
               ;
               ;---------------------------------------------------------
```

```
                    ;-----------------------------------------------------------

                    SELDSK:
0203 79                     MOV     A,C                 ;LIMIT SELECT TO REAL OPTION
0204 FE0C                   CPI     MAXDSK              ;
                            JRNC    SELERR              ;  INVALID DRIVE
0206+303A                   DB      030H,SELERR-$-1     ;---- FAKE JRNC INSTRUCTION
                    ;       MOV     A,E                 ; TEST FOR INITIAL SELECT
                    ;       ANI     1                   ;   E = 0 IS FIRST TIME
                    ;       PUSH    PSW                 ;
0208 1600                   MVI     D,0                 ;
020A 59                     MOV     E,C                 ; TRANSLATE TABLE
020B 214602                 LXI     H,DTBLT             ;  FOR LOGICAL TO PHYSICAL
020E 19                     DAD     D                   ;
020F 4E                     MOV     C,M                 ;  C = PHYSICAL DRIVE
0210 79                     MOV     A,C                 ; M translates to the 12 disk

                            if      mdisk
                            CPI     12
                            JZ      VIRTUAL
                            endif

                    ;       POP     PSW                 ; RESTORE TEST
                    ;       JRNZ    SELSDP              ; BYPASS SELECT
                    SETDSK:
0211 0600                   MVI     B,0                 ;
0213 217C00                 LXI     H,SEL0              ; BASE OF SELECT MASKS
0216 09                     DAD     B                   ;
0217 7E                     MOV     A,M                 ;  GET SELECT BYTE
0218 A7                     ANA     A                   ;   CHECK FOR VALID DRIVE
                            JRZ     SELERR              ;    DRIVE NOT CONFIGURED
0219+2827                   DB      02H,SELERR-$-1      ;---- FAKE JRZ INSTRUCTION
021B 79                     MOV     A,C                 ;
021C FE04                   CPI     4                   ; CHECK FOR FLOPPY
                            JRC     SELSDP              ;
021E+380F                   DB      038H,SELSDP-$-1     ;---- FAKE JRC INSTRUCTION
0220 7E             CHKHRD: MOV     A,M                 ; RESTORE SELECT BYTE
0221 D320                   OUT     20H                 ;
0223 C5                     PUSH    B                   ;
0224 0E01                   MVI     C,1                 ; DELAY FOR 1 MS
0226 CD8207                 CALL    DELAY               ;
0229 C1                     POP     B                   ;
022A DB24                   IN      24H                 ; CHECK FOR HARD DISK READY
022C 17                     RAL                         ;  80H = READY
                            JRNC    SELERR              ;
022D+3013                   DB      030H,SELERR-$-1     ;---- FAKE JRNC INSTRUCTION
                    SELSDP:
022F 79                     MOV     A,C                 ;

                            if      mdisk
                    VIRTUAL:
                            endif

0230 32E60A                 STA     NEWDSK              ;SAVE FOR I/O LATER
0233 2600                   MVI     H,0                 ;
```

```
0235 69                 MOV     L,C             ;COMPUTE DP HEADER ADDRESS
0236 29                 DAD     H               ;* 2
0237 29                 DAD     H               ;* 4
0238 29                 DAD     H               ;* 8
0239 29                 DAD     H               ;* 16 (DP HEADER SIZE)
023A 11D100             LXI     D,DPBASE        ;START OF DP HEADERS
023D 19                 DAD     D               ;POINT TO CORRECT ONE
023E 22B200             SHLD    DPEPTR          ;SAVE ADDRESS OF CURRENT DP
0241 C9                 RET                     ;

0242 210000     SELERR: LXI     H,0             ; INDICATE ERROR
0245 C9                 RET                     ;  AND RETURN

                ;               A,B,C,D,E,F,G,H,I,J, K, L, M,N,O
0246 0001020304DTBLT:   DB      0,1,2,3,4,5,6,7,8,9,10,11,12,0,0


                SETDMA:
0255 60                 MOV     H,B             ;TO ALLOW SAVING
0246 69                 MOV     L,C             ;
0247 22AF00             SHLD    DMAADR          ;

                        if      mpm20
025A 23                 inx     h               ;test for flush buffers
025B 7D                 mov     a,l
025C B4                 ora     h
025D C0                 rnz                     ;HL = FFFFh is flush buffer
025E 21F00A             lxi     h,hstwrt
0261 7E                 mov     a,m
0262 3600               mvi     m,0
0264 B7                 ora     a
0265 C8                 rz
0266 CD6D04             call    writehst        ;flush host write if pending
0269 B7                 ora     a
026A C8                 rz                      ;return if no error
026B E1                 pop     h
                        endif

026C C9                 ret

                SETTRK:
026D 60                 MOV     H,B             ;TO ALLOW SAVE
026E 69                 MOV     L,C             ;
026F 22E70A             SHLD    NEWTRK          ;SAVE NEXT TRACK NUMBER
0262 C9                 RET                     ;RETURN TO CALLER

                SETSEC:
0273 79                 MOV     A,C             ;FOR SAVE
0274 32E90A             STA     NEWSEC          ;
0277 C9                 RET                     ;RETURN TO CALLER

                SETDEN:
0278 117C00             LXI     D,SEL0          ;START OF SELECT/DENSITY MASK
027B 2AE60A             LHLD    NEWDSK          ;NEXT DRIVE ADDRESS
027E 2600               MVI     H,000H          ;ENSURE ZERO FOR SINGLE BYTE
```

```
0280 19                 DAD     D               ;POINT TO CORRECT MASK
0281 79                 MOV     A,C             ;ISOLATE DENSITY BIT
0282 E601               ANI     00000001B       ;
0284 4F                 MOV     C,A             ;SAVE FOR NOW
0285 7E                 MOV     A,M             ;LOAD SELECT DENSITY MASK
0286 E6FE               ANI     11111110B       ;RESET CURRENT DENSITY SETTING
0288 B1                 ORA     C               ;SET NEW VALUE
0289 77                 MOV     M,A             ;RESTORE MASK IN TABLE
028A C9                 RET                     ;RETURN TO CALLER

                        if      mdisk
                MREADSECTOR:
                        call    compbank        ;compute bank
                        di
                        call    chgbank
                        lxi     b,128
                        lxi     d,localbuf
                        lhld    addroff
                        ldir                    ;block move into the dma area
                        mvi     a,02h           ; select bank 0
                        out     09h
                        ei
                        lxi     b,128
                        lhld    dmaadr
                        xchg
                        lxi     h,localbuf
                        ldir
                        xra     a
                        ret

                mbankno         db      0
                addroff         dw      0
                localbuf        ds      128

                compbank:
                        lda     newtrk
                        mov     h,a
                        ani     0fh     ;save track rem 16
                        mov     l,a
                        mov     a,h     ;restore track
                        mvi     h,0
                        ani     0f0h            ; bank is high order nibble
                        rar ! rar ! rar ! rar
                        inr     a
                        sta     mbankno         ; which bank we want

                        dad     h               ;trk 0-15
                        dad     h               ; * 2
                        dad     h               ; * 4
                        mov     e,l
                        mov     d,h
                        dad     d
                        dad     d               ; * 24:

                        lda     newsec          ; figure offset with the
                        mov     e,a
                        mvi     d,0
                        dad     d               ; add sector offset within
                        dad     h ! dad h ! dad h ! dad h! dad h ! dad h !
                        shldd   addroff         ; (track * 24 + sector) * 1
                        ret
                        endif

                READ:
                        if      mdisk
                        LDA     NEWDSK
                        CPI     12              ;VIRTUAL DISK ?
                        JZ      MREADSECTOR
                        endif

028B CDEE02             CALL    RETDMOD         ;WHAT TYPE OF I/O ??
028E FE03               CPI     003H            ;
0290 DAE405             JC      READSOFT        ;FLOPPY DISK DRIVE....
0293 C36B03             JMP     READHARD        ;HARD DISK I/O

                        if      mdisk
                mwritesector:
                        call    compbank
                        lhld    dmaadr
                        lxi     d,localbuf
                        li      b,128
                        ldir
                        di
                        call    chgbank
                        lxi     d,localbuf
                        lxi     b,128
                        lhld    addroff
                        xchg
                        ldir
                        mvi     a,02h           ; select bank 0
                        out     09h
                        ei
                        xra     a
                        ret

                chgbank:
                        lda     mbankno
                        ral
                        ral
                        ral
                        ani     018h
                        ori     memsk
                        out     009h
                        ret
                        endif

                WRITE:
                        if      mdisk
                        lda     mewdsk
                        cpi     12
```

```
                       jz      mwritesector
                       endif

0296 CDEE02            CALL RETMOD             ;WHAT TYPE OF I/O ??
0299 FE03              CPI     003H            ;
029B DAF205            JC      WRITESOFT       ;FLOPPY DISK
029E C37E03            JMP     WRITEHARD       ;HARD DISK I/O

                       PAGE

               ;-----------------------------------------------------------------
               ;
               ;       ROUTINES TO SET AND RETURN THE CURRENT DRIVE MODE
               ;
               ;-----------------------------------------------------------------

               SETMOD:
02A1 21E60A            LXI     H,NEWDSK        ; SAVE NEWDSK IN STACK
02A4 7E                MOV     A,M             ;
02A5 F5                PUSH    PSW             ;
02A6 70                MOV     M,B             ;
02A7 C5                PUSH    B               ;
               ;       MVI     E,0             ; INDICATE INITIAL SELECT
02A8 48                MOV     C,B             ; CALL DISK SELECT
02A9 CD0302            CALL    SELDSK          ;
02AC C1                POP     B               ;
02AD 7C                MOV     A,H             ; CHECK FOR BAD SELECT
02AE B5                ORA     L               ;
                       JRZ     SMERR           ;   YES - ABORT CHANGING
02AF+2832              DB      028H,SMERR-$-1  ;---- FAKE JRZ INSTRUCTION
02B1 68                MOV     L,B             ;  B AND L = DRIVE #
02B2 2600              MVI     H,000H          ;
02B4 78                MOV     A,B             ;CHECK MODE SET VALIDITY
02B5 FE04              CPI     004H            ;ONLY VALID FOR FLOPPY DISK
                       JRNC    SMERR           ;INVALID DRIVE FOR MODE SET
02B7+302A              DB      030H,SMERR-$-1  ;---- FAKE JRNC INSTRUCTION
02B9 118800            LXI     D,MODE          ;START OF MODE BYTES
02BC 19                DAD     D               ;
02BD 71                MOV     M,C             ;SAVE NEW MODE BYTE
02BE E5                PUSH    H               ;SAVE MODE BYTE ADDRESS
02BF 79                MOV     A,C             ;SETUP FOR DENSITY CHANGE
02C0 B7                ORA     A               ;
02C1 0E00              MVI     C,000H          ;ASSUME SINGLE DENSITY MODE
                       JRZ     SETSEL          ;VERIFY ASSUMPTION
02C3+2802              DB      028H,SETSEL-$-1 ;---- FAKE JRZ INSTRUCTION
02C5 0E01              MVI     C,001H          ;SET FOR DOUBLE DENSITY MODE
02C7 CD7802    SETSEL: CALL    SETDEN          ;SET DENSITY BASED ON LOW BIT
02CA E1                POP     H               ;RESTORE
02CB 6E                MOV     L,M             ;PICKUP MODE AGAIN
02CC 2600              MVI     H,000H          ;FOR SINGLE BYTE PRECISION
02CE 7D                MOV     A,L             ;SAVE MODE IN ACCUMULATOR FLAG
02CF 29                DAD     H               ;* 2
02D0 29                DAD     H               ;* 4
02D1 E5                PUSH    H               ;SAVE * 4
02D2 29                DAD     H               ;* 8
```

```
02D3 D1                 POP     D                  ;REGAIN * 4
02D4 19                 DAD     D                  ;* 12
02D5 119101             LXI     D,MODL0            ;FIRST MODEL DPE
02D8 19                 DAD     D                  ;POINT TO THIS ONE
02D9 EB               XCHG                       ;SETUP TEMPORARILY AS DESTINATION
02DA 2AB200             LHLD    DPEPTR             ;ADDRESS OF CURRENTLY SELCT DSK
02DD EB                 XCHG                       ;SETUP TO ALTER
02DE 010C00             LXI     B,12               ;LENGTH FOR MOVE
                        LDIR                       ;DO MOVE
02E1+EDB0               DB      0EDH,0B0H          ;---- FAKE LDIR INSTRUCTION
02E3 F1         SMERR:  POP     PSW                ;
02E4 E5                 PUSH    H                  ;
02E5 32E60A             STA     NEWDSK             ; RESTORE ORIGINAL NEWDSK
02E8 4F                 MOV     C,A                ;
02E9 CD0302             CALL    SELDSK             ;
02EC E1                 POP     H                  ;
02ED C9                 RET                        ;RETURN TO CALLER

                RETMOD:
02EE 118800             LXI     D,MODE             ;START OF MODE BYTES
02F1 2AE60A             LHLD    NEWDSK             ;NEXT DRIVE FOR I/O
02F4 2600               MVI     H,000H             ;RESET FOR SINGLE BYTE QUAN
02F6 19                 DAD     D                  ;POINT TO IT....
02F7 7E                 MOV     A,M                ;LOAD IT FOR CALLER
02F8 C9                 RET                        ;RETURN, WITH CURRENT MODE

                        PAGE

                ;-------------------------------------------------------------
                ;
                ;       THIS IS THE HOME DEVICE ROUTINE
                ;
                ;-------------------------------------------------------------

02F9 3AE60A     HOMEIT: LDA     NEWDSK             ; CHECK FOR FIRST HOME
02FC FE0C               CPI     12                 ; CHECK FOR VIRTUAL DISK
02FE C20803             JNZ     REALDISK
0301 AF                 XRA     A                  ; VIRTUAL DISK
0302 67                 MOV     H,A                ; SETTRACK TO ZERO
0303 6F                 MOV     L,A
0304 22E70A             SHLD    NEWTRK
0307 C9                 RET
                REALDISK:
0308 FE04               CPI     4                  ;  CHECK FOR FLOPPY
                        JRC     HOME               ;  DO NOT BYPASS FLOPPY HOME
030A+380E               DB      038H,HOME-$-1      ;---- FAKE JRC INSTRUCTION
030C 4F                 MOV     C,A                ;
030D 0600               MVI     B,0                ;POINT TO PRESENT TRACK STORAGE
030F 217000             LXI     H,TRK0             ;
0312 09                 DAD     B                  ;
0313 7E                 MOV     A,M                ; CHECK IF INITIALIZED
0314 FEFF               CPI     0FFH               ;
0316 3E00               MVI     A,0                ;
0318 C0                 RNZ                        ; YES  -  RETURN WITH NO ERROR
0319 77                 MOV     M,A                ;
```

```
                HOME:
031A 3AE60A             LDA     NEWDSK          ;GET VALUE OF DRIVE FOR HOME
031D FE04               CPI     004H            ;IS IT A HARD DISK ??
                        JRNC    HOMEHARD        ;YES, PROCESS....
031F+3022               DB      030H,HOMEHARD-$-1       ;---- FAKE JRNC INS

                HOMESOFT:
0321 CD5205             CALL    DSKSEL          ;SELECT CORRECT DRIVE (IN A REG
0324 3AF60A             LDA     ERFLAG          ;
0327 B7                 ORA     A               ;CHECK FOR ERRORS DURING SELECT
                        JRNZ    HOME1A          ;EXIT IF ERRORS
0328+2016               DB      020H,HOME1A-$-1 ;---- FAKE JRNZ INSTRUCTION
032A CDB305             CALL    POINT           ;POINT TO TRACK REGISTER SAVE
032D 3600               MVI     M,000H          ;RESET TO TRACK ZERO
032F CD1905             CALL    DBL$UPDATE      ;
0332 3E0A               MVI     A,00AH          ;HOME COMMAND....
0334 CD6307             CALL    FINTFIX         ;CLEAR ANY PENDING INTERRUPT
                ;                               ;AND ISSUE COMMAND
0337 CD3A07     HOME1:  CALL    FPYWAIT         ;WAIT UNTIL I/O COMPLETE
033A 3AFC0A             LDA     STATUS          ;PICKUP STATUS BYTE
033D E698               ANI     10011000B       ;CHECK STATUS
033F C8                 RZ                      ;RETURN WITH GOOD RESULT
0340 3E01       HOME1A: MVI     A,001H          ;SET ERROR ON HOME
0342 C9                 RET                     ;AND RETURN

                HOMEHARD:
                        IF      HARDSK
0343 CD5205             CALL    DSKSEL          ;SELECT CORRECT DRIVE (IN A
0346 CDB305             CALL    POINT           ;POINT TO SAVE AREA
0349 3600               MVI     M,000H          ;SET TO TRACK ZERO
034B EB                 XCHG                    ;POINT TO SELECT WORD
034C 7E                 MOV     A,M             ;LOAD SELECT MASK
034D E6F0               ANI     11110000B       ;RESET HEAD MASK
034F 77                 MOV     M,A             ;SAVE
0350 D320               OUT     020H            ;WRITE HEAD/SELECT MASK
0352 3E20               MVI     A,020H          ;HOME COMMAND
0354 CD2107             CALL    INTFIX          ;CLEAR ANY PENDING INTERRUPT
                ;                               ;AND ISSUE COMMAND
0357 CD1707     HOME2:  CALL    WAIT0           ;WAIT UNTIL I/O COMPLETE
035A 0E14               MVI     C,20            ;DELAY FOR 20 MILLISECONDS
035C CD8207             CALL    DELAY           ;
035F AF                 XRA     A               ;SET NEW TRACK REGISTER TO
0360 D322               OUT     022H            ;FOR CONTROLLER
                ;       LXI     H,MHM           ;***DEBUG***
                ;       CALL    MSPRT           ;***DEBUG***
0362 3AFC0A             LDA     STATUS          ;PICKUP STATUS BYTE
0365 E65D               ANI     01011101B       ;CHECK STATUS
0367 C8                 RZ                      ;
0368 3E01               MVI     A,001H          ;SET ERROR ON HOME
                        ENDIF
036A C9                 RET                     ;AND RETURN

                        PAGE

                ;-----------------------------------------------------------
```

```
                 ;
                 ;       THESE ARE THE HARD DISK UNBLOCK/REBLOCK AND READ
                 ;       AND WRITE ROUTINES CALLED BY THE BDOS SOFTWARE.
                 ;
                 ;
                 ;-------------------------------------------------------

                 READHARD:
                         IF      HARDSK
036B AF                  XRA     A                   ;RESET UNALLOCATED COUNT
036C 32F10A              STA     UNACNT              ;
036F 3E01                MVI     A,001H              ;READ THE SELECTED CP/M SECTOR
0371 32F80A              STA     READOP              ;
0374 32F70A              STA     RSFLAG              ;MUST READ DATA
0377 3E02                MVI     A,WRUAL             ;
0379 32F90A              STA     WRTYPE              ;TREAT AS UNALLOCATED
                         JR      RWOPER              ;TO PERFORM THE READ
037C+1864                DB      018H,RWOPER-$-1 ;---- FAKE JR INSTRUCTION
                         ENDIF

                 WRITEHARD:
                         IF      HARDSK
037E AF                  XRA     A                   ;WRITE THE SELECTED CP/M SECTOR
037F 32F80A              STA     READOP              ;NOT A READ OPERATION
0382 79                  MOV     A,C                 ;WRITE TYPE IS PASSED IN REG C
0383 32F90A              STA     WRTYPE              ;

                         if      mpm20
0386 E602                ani     WRUAL               ;IS IT WRITE UNALLOCATED ??
                         JRZ     CHKUNA              ;CHECK FOR UNALLOCATED
0388+2817                DB      028H,CHKUNA-$-1 ;---- FAKE JRZ INSTRUCTION
                         else
                         CPI     WRUAL               ;IS IT WRITE UNALLOCATED ??
                         JRNZ    CHKUNA              ;CHECK FOR UNALLOCATED
                         endif
                 ;
                 ;       WRITE TO UNALLOCATED, SET PARAMETERS
                 ;

038A 3E80                MVI     A,BLKSIZ/128        ;NEXT UNALLOC RECS
038C 32F10A              STA     UNACNT              ;
038F 3AE60A              LDA     MEWDSK              ;DISK FOR I/O
0392 32F20A              STA     UNADSK              ;UNADSK = NEWDSK
0395 2AE70A              LHLD    NEWTRK              ;
0398 22F30A              SHLD    UNATRK              ;UNATRK = NEWDSK
039B 3AE90A              LDA     NEWSEC              ;
039E 32F50A              STA     UNASEC              ;UNASEC = NEWSEC

                 ;
                 ;       CHECK FOR WRITE TO UNALLOCATED SECTOR
                 ;

                 CHKUNA:
03A1 3AF10A              LDA     UNACNT              ;ANY UNALLOCATED REMAIN ??
03A4 B7                  ORA     A                   ;
                         JRZ     ALLOC               ;SKIP IF NOT
```

```
03A5+2833                DB      028H,ALLOC-$-1  ;---- FAKE JRZ INSTRUCTION

                   ;
                   ;     MORE UNALLOCATED RECORDS REMAIN
                   ;

03A7 3D                  DCR     A               ;UNACNT = UNACNT - 1
03A8 32F10A              STA     UNACNT          ;
03AB 3AE60A              LDA     NEWDSK          ;SAME DISK ??
03AE 21F20A              LXI     H,UNADSK        ;
03B1 BE                  CMP     M               ;NEWDSK = UNADSK ??
                         JRNZ    ALLOC           ;SKIP IF NOT
03B2+2026                DB      020H,ALLOC-$-1  ;---- FAKE JRNZ INSTRUCTION

                   ;
                   ;     DISKS ARE THE SAME
                   ;

03B4 21F30A              LXI     H,UNATRK        ;
03B7 CD6104              CALL    NEWTRKCMP       ;NEWTRK = UNATRK ??
                         JRNZ    ALLOC           ;SKIP IF NOT
03BA+201E                DB      020H,ALLOC-$-1  ;---- FAKE JRNZ INSTRUCTION

                   ;
                   ;     TRACKS ARE THE SAME
                   ;

03BC 3AE90A              LDA     NEWSEC          ;SAME SECTOR ??
03BF 21F50A              LXI     H,UNASEC        ;
03C2 BE                  CMP     M               ;NEWSEC = UNASEC ??
                         JRNZ    ALLOC           ;SKIP IF NOT
03C3+2015                DB      020H,ALLOC-$-1  ;---- FAKE JRNZ INSTRUCTION

                   ;
                   ;     MATCH, MOVE TO NEXT SECTOR FOR FUTURE REFERENCE
                   ;

03C5 34                  INR     M               ;UNASEC = UNASEC + 1
03C6 7E                  MOV     A,M             ;END OF TRACK ??
03C7 FE80                CPI     CPMSPT          ;COUNT CP/M SECTORS
                         JRC     NOOVF           ;SKIP IF NO OVERFLOW
03C9+3809                DB      038H,NOOVF-$-1  ;---- FAKE JRC INSTRUCTION

                   ;
                   ;     OVERFLOW TO NEXT TRACK
                   ;

03CB 3600                MVI     M,000H          ;UNASEC = 0
03CD 2AF30A              LHLD    UNATRK          ;
03D0 23                  INX     H               ;
03D1 22F30A              SHLD    UNATRK          ;UNATRK = UNATRK + 1

                   ;
                   ;     MATCH FOUND, MARK AS UNNECESSARY READ
                   ;
                   NOOVF:
03D4 AF                    XRA     A                 ;ZERO TO ACCUMULATOR
03D5 32F70A                STA     RSFLAG            ;RSFLAG = 0
                           JR      RWOPER            ;TO PERFORM THE WRITE
03D8+1808                  DB      018H,RWOPER-$-1 ;---- FAKE JR INSTRUCTION

                   ;
                   ;       NOT AN UNALLOCATED RECORD, REQUIRES PRE-READ
                   ;

                   ALLOC:
03DA AF                    XRA     A                 ;ZERO TO ACCUMULATOR
03DB 32F10A                STA     UNACNT            ;UNACNT = 0
03DE 3C                    INR     A                 ;ONE TO ACCUMULATOR
03DF 32F70A                STA     RSFLAG            ;RSFLAG = 1

                   ;----------------------------------------------------------
                   ;
                   ;       THE FOLLOWING CODE IS COMMON TO BOTH READ AND WRITE
                   ;
                   ;----------------------------------------------------------

                   RWOPER:
03E2 AF                    XRA     A                 ;ZERO TO ACCUMULATOR
03E3 32F60A                STA     ERFLAG            ;NO ERRORS YET....
03E6 3AE90A                LDA     NEWSEC            ;COMPUTE HOST SECTOR
                           REPT    SECSHF            ;COMPUTE HOST SECTOR
                           ORA     A                 ;CARRY = 0
                           RAR                       ;SHIFT RIGHT
                           ENDM
03E9+B7                    ORA     A                 ;CARRY = 0
03EA+1F                    RAR                       ;SHIFT RIGHT
03EB+B7                    ORA     A                 ;CARRY = 0
03EC+1F                    RAR                       ;SHIFT RIGHT
03ED+B7                    ORA     A                 ;CARRY = 0
03EE+1F                    RAR                       ;SIFT RIGHT
03EF 32EE01                STA     NEWHST            ;HOST SECTOR TO SEEK

                   ;
                   ;       ACTIVE HOST SECTOR ??
                   ;

03F2 2AEF0A                LXI     H,STACT           ;HOST ACTIVE FLAG
03F5 7E                    MOV     A,M               ;
03F6 3601                  MVI     M,001H            ;ALWAYS BECOMES 1
03F8 B7                    ORA     A                 ;WAS IT ALREADY ?
                           JRZ     FILLHST           ;FILL HOST IF NOT
03F9+2821                  DB      028H,FILLHST-$-1          ;---- FAKE JRZ INST

                   ;
                   ;       HOST BUFFER ACTIVE, SAME AS SEEK BUFFER
                   ;

03FB 3AE60A                LDA     NEWDSK
```

```
03FE 21EA0A             LXI     H,HSTDSK        ;SAME DISK ??
0401 BE                 CMP     M               ;NEWDSK = HSTDSK ??
                        JRNZ    NOMATCH         ;
0402+2011               DB      020H,NOMATCH-$-1        ;---- FAKE JRNX INST

               ;
               ;        SAME DISK, SAME TRACK ??
               ;

0404 21EB0A             LXI     H,HSTTRK        ;
0407 CD6104             CALL    NEWTRKCMP       ;NEWTRK = HSTTRK ??
                        JRNZ    NOMATCH         ;
040A+2009               DB      020H,NOMATCH-$-1        ;---- FAKE JRNZ INST

               ;
               ;        SAME DISK, SAME TRACK, SAME BUFFER ??
               ;

040C 3AEE0A             LDA     NEWHST          ;
040F 21ED0A             LXI     H,HSTSEC        ;NEWHST = HSTSEC ??
0412 BE                 CMP     M               ;
                        JRZ     MATCH           ;SKIP IF MATCH
0413+2824               DB      028H,MATCH-$-1  ;---- FAKE JRZ INSTRUCTION

               ;
               ;        PROPER DISK, BUT NOT CORRECT SECTOR
               ;

               NOMATCH:
0415 3AF00A             LDA     HSTWRT          ;HOST WRITTEN ??
0418 B7                 ORA     A               ;
0419 C46D04             CNZ     WRITEHST        ;CLEAR HOST BUFFER

               ;
               ;        MAY HAVE TO FILL HOST BUFFER
               ;

               FILLHST:
041C 3AE60A             LDA     NEWDSK          ;
041F 32EA0A             STA     HSTDSK          ;
0422 2AE70A             LHLD    NEWTRK          ;
0425 22EB0A             SHLD    HSTTRK          ;
0428 3AEE0A             LDA     NEWHST          ;
042B 32ED0A             STA     HSTSEC          ;
042E 3AF70A             LDA     RSFLAG          ;NEED TO READ ??
0431 B7                 ORA     A               ;
0432 C47F04             CNZ     READHST         ;YES, IF 1
0435 AF                 XRA     A               ;ZERO TO ACCUMULATOR
0436 32F00A             STA     HSTWRT          ;NO PENDING WRITE

               MATCH:
0439 3AE90A             LDA     NEWSEC          ;MASK BUFFER NUMBER
043C 3607               ANI     SECMSK          ;LEAST SIGNIF BITS
043E 6F                 MOV     L,A             ;READY TO SHIFT
043F 2600               MVI     H,000H          ;DOUBLE COUNT
```

```
                                REPT    7
                                DAD     H               ;
                                ENDM
0441+29                         DAD     H               ;
0442+29                         DAD     H               ;
0443+29                         DAD     H               ;
0444+29                         DAD     H               ;
0445+29                         DAD     H               ;
0446+29                         DAD     H               ;
0447+29                         DAD     H               ;

                        ;
                        ;       HL NOW HAS RELATIVE HOST BUFFER ADDRESS
                        ;

0448 119C0E                     LXI     D,HSTBUF        ;
044B 19                         DAD     D               ;HL = HOST ADDRESS
044C EB                         XCHG                    ;NOW IN DE
044D 2AAF00                     LHLD    DMAADR          ;GET/PUT CP/M DATA
0450 EB                         XCHG                    ;SET FOR Z80 LDIR INSTRUCTION
                        ;       LXI     B,128           ;LENGTH OF MOVE
0451 3AF80A                     LDA     READOP          ;WHICH WAY ??
0454 B7                         ORA     A               ;
0455 C23D0E                     JNZ     RWMOVE          ;SKIP IF READ
                        ;
                        ;       WRITE OPERATION, MARK AND SWITCH DIRECTION
                        ;

0458 3E01                       MVI     A,001H          ;
045A 32F00A                     STA     HSTWRT          ;HSTWRT = 1
045D EB                         XCHG                    ;SWAP DIRECTION
045E C33D0E                     jmp     rwmove
                                endif


                                PAGE

                        ;--------------------------------------------------------------
                        ;
                        ;       UTILITY SUBROUTINE FOR 16 BIT COMPARE
                        ;
                        ;--------------------------------------------------------------

                                IF      HARDSK
                        NEWTRKCMP:
0461 EB                         XCHG                    ;HL = .UNATRK OR .HSTTRK
0462 21E70A                     LXI     H,NEWTRK        ;
0465 1A                         LDAX    D               ;LOW BYTE COMPARE
0466 BE                         CMP     M               ;SAME ??
0467 C0                         RNZ                     ;RETURN IF NOT
0468 13                         INX     D               ;TO CHECK HIGH BYTE
0469 23                         INX     H               ;
046A 1A                         LDAX    D               ;
046B BE                         CMP     M               ;SETS FLAGS
```

```
046C C9                 RET                     ;

                        PAGE

                ;-------------------------------------------------------------
                ;
                ;       WRITEHST PERFORMS THE PHYSICAL WRITE TO THE HOST DISK
                ;       READHST PERFORMS THE PHYSICAL READ FROM THE HOST DISK
                ;
                ;       HSTDSK = HOST DISK NUMBER
                ;       HSTTRK = HOST TRACK NUMBER
                ;       HSTSEC = HOST SECTOR NUMBER
                ;       RETURN ERROR FLAG IN ERFLAG
                ;
                ;-------------------------------------------------------------

                WRITEHST:
046D E305               MVI     A,005H          ;SETUP DMA FOR WRITE .
046F 32CE00             STA     DMAS3F          ;
0472 3E02               MVI     A,002H          ;WRITE COMMAND
0474 32FA0A             STA     CMD             ;SAVE FOR LATER
0477 219B03             LXI     H,HSTBUF-1      ;WRITE MUST WRITE CONTROL BLOCK
047A 22BC00             SHLD    DMASA           ;
                        JR      HRW0            ;
047D+1810               DB      018H,HR20-$-1   ;---- FAKE JR INSTRUCTION -

                READHST:
047F 3E01               MVI     A,001H          ;SETUP DMA FOR READ
0481 32CD00             STA     DMAS3F          ;
0484 3E04               MVI     A,004H          ;READ COMMAND
0486 32FA0A             STA     CMD             ;SAVE FOR LATER
0489 219C0E             LXI     H,HSTBUF        ;READ ONLY DATA BYTES
048C 22BC00             SHLD    DMASA           ;

                HRW0:
048F 3E05               MVI     A,05            ;FIVE RETRIES
0491 32020B             STA     T$RETRIES       ;SETUP TEMPORARY RETRIES COUNTER
0494 3EFF               MVI     A,0FFH          ;INIT TOGGLE SO THAT NO HOME
0496 32030B             STA     HOME$TOGGLE     ;ALTERNATE RETRIES WILL BE ATTEMPTED
                ;                               ;OTHER RETRIES WILL BE DONE

                HRW1:
0499 3AED0A             LDA     HSTSEC          ;HOST SECTOR NUMBER
049C 32B100             STA     SECTNO          ;SAVE SECTOR NUMBER
049F 3AEA0A             LDA     HSTDSK          ;PICKUP DRIVE ID FOR SELECT DRIVE
04A2 CD5205             CALL    DSKSEL          ;SELECT CORRECT DRIVE FOR INDEX
04A5 CDB305             CALL    POINT           ;POINT TO TRACK REGISTER SAVE
04A8 EB                 XCHG                    ;POINT TO SELECT MASK
04A9 3EF0               MVI     A,11110000B     ;TO REMOVE CURRENT HEAD SELECT
04AB A6                 ANA     M               ;
04AC 77                 MOV     M,A             ;
04AD E5                 PUSH    H               ;SAVE MASK ADDRESS
04AE CD3205             CALL    SETHED          ;COMPUTE CORRECT HEAD NUMBER
04B1 7D                 MOV     A,L             ;TRACK NUMBER AFTER HEAD CALCUL
04B2 32AD00             STA     TRAKNO          ;
```

```
04B5 E1                 POP     H               ;RESTORE MASK ADDRESS
04B6 3AAE00             LDA     HEADNO          ;TO OR IN NEW HEAD NUMBER
04B9 B6                 ORA     M               ;
04BA 77                 MOV     M,A             ;SAVE NEW DRIVE/HEAD SELECT
04BB E67F               ANI     07FH            ; MASK OFF LARGE DRIVE FLAG
04BD D320               OUT     020H            ;WRITE IT TO SELECT NEW HEAD
04BF 0E01               MVI     C,1             ;DELAY FOR 1 MILLISECOND
04C1 CD8207             CALL    DELAY           ;

                HRW2:
04C4 CDB305             CALL    POINT           ;IS A SEEK NECESSARY ??
04C7 3AAD00             LDA     TRAKNO          ;CHECK
04CA BE                 CMP     M               ;WELL ??
                        JRZ     HRW5            ;NO SEEK NECESSARY...
04CB+2814               DB      028H,HRW5-$-1   ;---- FAKE JRZ INSTRUCTION

                HRW3:
04CD D322               OUT     022H            ;WRITE NEW TRACK NUMBER
04CF 46                 MOV     B,M             ;SAVE TEMPORARILY
04D0 77                 MOV     M,A             ;UPDATE TRACK REGISTER SAVE
04D1 78                 MOV     A,B             ;OLD TRACK NUMBER
04D2 D321               OUT     021H            ;TO OLD TRACK REGISTER
04D4 3E10               MVI     A,010H          ;SEEK COMMAND
04D6 CD2107             CALL    INTFIX          ;CLEAR ANY PENDING INTERRUP
                ;                               ;AND ISSUE COMMAND
04D9 CD1707     HRW4:   CALL    WAIT0           ;WAIT FOR I/O
04DC 0E14               MVI     C,20            ;DELAY AFTER SEEK FOR 20 MILLI
04DE CD8207             CALL    DELAY           ;

                HRW5:
04E1 3AB100             LDA     SECTNO          ;SET SECTOR
04E4 D321               OUT     021H            ;

                HRW6:
04E6 21BA00             LXI     H,DMAS1         ;SETUP DMA FOR HARD DISK I/O
04E9 010006             LXI     B,0600H         ;
                        OUTIR                   ;
04EC+EDB3               DB      05DH,0B3H       ;---- FAKE OTIR INSTRUCTION
04EE 21C000             LXI     H,DMAS2H        ;
04F1 010006             LXI     B,0600H         ;
                        OUTIR                   ;
04F4+EDB3               DB      0EDH,0B3H       ;---- FAKE OTIR INSTRUCTION
04F6 21CA00             LXI     H,DMAS3         ;
04F9 010007             LXI     B,0700H         ;
                        OUTIR                   ;
04FC+EDB3               DB      0EDH,0B3H       ;---- FAKE OTIR INSTRUCTION

04FE 3AFA0A             LDA     CMD             ;PICKUP I/P COMMAND
050A CD2107             CALL    INTFIX          ;CLEAR ANY PENDING INTERRUPT
                ;                               ;AND ISSUE COMMAND
0504 CD1707     HRW7:   CALL    WAIT0           ;WAIT FOR COMPLETION

0507 3E5D               MVI     1,01011101B     ;SETUP STATUS AND MASK
0509 32FB0A             STA     MASK            ;SAVE FOR STATUS CHECK
```

```
050C CDAE06                 CALL    CHECK$STAT      ;CHECK STATUS FROM I/O
050F C8                     RZ                      ;OK ??

0510 3A030B                 LDA     HOME$TOGGLE     ;
0513 2F                     CMA                     ;CHANGE TOGGLE SO THAT HOME
0514 32030B                 STA     HOME$TOGGLE     ;

                            JR      HRW1            ;RETRY I/O
0517+1880                   DB      018H,HRW1-$-1   ;---- FAKE JR INSTRUCTION -
                            ENDIF

                            PAGE

                    ;--------------------------------------------------------
                    ;
                    ;       DOUBLE SIDED TRACK REGISTER UPDATE ROUTINE
                    ;
                    ;--------------------------------------------------------


                    DBL$UPDATE:
0519 3AB600                 LDA     MPARMS          ;CHECK FOR DOUBLE SIDED DRIVE
051C E601                   ANI     1               ;  IS FLAG SET
051E C8                     RZ                      ;   NO - SO RETURN
051F 3AAC00                 LDA     DISKNO          ;CURRENT DISK DRIVE
0522 FE04                   CPI     004H            ;IS IT A FLOPPY
0524 D0                     RNC                     ;NO, RETURN WITHOUT UPDATE
0525 E602                   ANI     00000010B       ;IS THIS DRIVE 2 OR 3 ??
0527 7E                     MOV     A,M             ;WE WERE CALLED WITH (HL) P
                            JRZ     DBL$LOW         ;IT MUST BE DRIVE ZRO OR O
0528+2804                   DB      028H,DBL$LOW-$-1        ;---- FAKE JRZ INSTR
052A 2B                     DCX     H               ;BACKUP TO OTHER SIDE POINT
052B 2B                     DCX     H               ;
                            JR      DBL$SAVE        ;
052C+1802                   DB      018H,DBL$SAVE-$-1       ;---- FAKE JR INSTR

                    DBL$LOW:
052E 23                     INX     H               ;BUMP UP TO DRIVE TWO OR THREE
052F 23                     INX     H               ;

                    DBL$SAVE:
0530 77                     MOV     M,A             ;UPDATE OTHER SIDE REGISTER
0531 C9                     RET                     ;


                            PAGE

                    ;--------------------------------------------------------
                    ;
                    ;       ROUTINE TO COMPUTE HEAD NUMBER FROM TRACK NUMBER
                    ;       TRACK NUMBER IS IN HL ON ENTRY
                    ;
                    ;--------------------------------------------------------
                            IF      HARDSK
                    SETHED:
```

```
0532 2AEB0A          LHLD    HSTTRK          ;CP/M TRACK NUMBER (0-800)
0535 E680            ANI     80H             ; CHECK FOR LARGE DRIVE
0537 7D              MOV     A,L             ;LOW ORDER
                     JRZ     SETH14          ;  SMALL DRIVE
0538+2806            DB      028H,SETH14-$-1 ;---- FAKE JRZ INSTRUCTION
053A E607            ANI     00000111B       ;GET TRACK MOD 8 (HEAD NUMBER
053C 0E03            MVI     C,3             ;LIMIT LOOP FOR DIVIDE BY EIGHT
                     JR      SETDVD          ;
053E+1804            DB      018H,SETDVD-$-1 ;---- FAKE JR INSTRUCTION -
0540 E603    SETH14: ANI     00000011B       ;GET TRACK MOD 4 (HEAD NUMB
0542 0E02            MVI     C,2             ;LIMIT LOOP FOR DIVIDE BY FIVE
0544 32AE00  SETDVD: STA     HEADNO          ;SAVE AS HEAD NUMBER
0547 B7      SHD1:   ORA     A               ;ENSURE CARRY IS ZERO
0548 7C              MOV     A,H             ;FOR SHIFT
0549 1F              RAR                     ;ONE BIT
054A 67              MOV     H,A             ;
054B 7D              MOV     A,L             ;LOW ORDER
054C 1F              RAR                     ;CARRY PARTICIPATES FROM HIBYTE
054D 6F              MOV     L,A             ;
054E 0D              DCR     C               ;END OF DIVIDE YET ??
                     JRNZ    SHD1            ;NO, CONTINUE
054F+20F6            DB      020H,SHD1-$-1   ;---- FAKE JRNZ INSTRUCTION
0551 C9              RET                     ;RETURN TO CALLER, TRACK IN
                     ENDIF

                     PAGE

             ;------------------------------------------------------------
             ;
             ;       DISK DRIVE SELECT ROUTINE
             ;               ON ENTRY, THE ACCUMULATOR CONTAINS THE DRIVE
             ;               RETURNS CARRY SET FOR HARD DISK SELECTED
             ;               RETURNS CARRY RESET FOR FLOPPY DISK SELECTED
             ;
             ;------------------------------------------------------------

             DSKSEL:
0552 FE04            CPI     004H            ;IS IT HARD DISK ??
                     JRNC    SELHARD         ;YES, GO PROCESS....
0554+3045            DB      030H,SELHARD-$-1        ;---- FAKE JRNC INST

             SELSOFT:
0556 21AC00          LXI     H,DISKNO        ;CURRENT DRIVE NUMBER
0559 BE              CMP     M               ;SAME DRIVE AS LAST TIME ??
                     JRZ     SLS3            ;YES, DONT BOTHER WITH UNLOCK
055A+2819            DB      028H,SLS3-$-1   ;---- FAKE JRZ INSTRUCTION
055C 77              MOV     M,A             ;UPDATE WITH CURRENT DRIVE

             ;------------------------------------------------------------
             ;
             ;       WE WILL NOW FORCE THE HEAD TO UNLOAD PRIOR TO THE SEEK
             ;       TO ENSURE THAT WHEN WE RETURN TO THIS DISK WE WILL
             ;       LOAD AND WAIT FOR THE HEAD TO SETTLE.
             ;
             ;------------------------------------------------------------
```

```
                    SLS1:
055D DB04                   IN      004H            ;ENSURE FLOPPY PORT NOT BUSY
055F 1F                     RAR                     ;
                            JRC     SLS1            ;
0560+38FB                   DB      038H,SLS1-$-1   ;---- FAKE JRC INSTRUCTION
0562 DB05                   IN      005H            ;READ THE TRACK REGISTER
0564 D307                   OUT     007H            ;ENSURE WE DONT MOVE THE HEAD

0566 3E12                   MVI     A,012H          ;SEEK AND UNLOAD HEAD
0568 CD6307                 CALL    FINTFIX         ;CLEAR ANY PENDING INTERRUPT
                    ;                               ;  AND ISSUE COMMAND
056B CD3A07         SLS2:   CALL    FPYWAIT         ;WAIT HERE FOR INTERRUPT
056E 3AFC0A                 LDA     STATUS          ;HOW DID THE I/O GO?
0571 E698                   ANI     10011000B       ;  CHECK
                            JRNZ    SLSERR          ;EXIT IF ERROR
0573+2020                   DB      020H,SLS344-$-1 ;---- FAKE JRNZ INSTRUCTION

                    ;----------------------------------------------------------
                    ;
                    ;       WE WILL NOW LOAD THE SELECT MASK AND SELECT THE DRIVE
                    ;       EVEN IF ITS THE SAME DRIVE BECAUSE THE DENSITY MAY
                    ;       HAVE CHANGED.
                    ;
                    ;----------------------------------------------------------

                    SLS3:
0575 CDB305                 CALL    POINT           ;POINT TO TRACK SAVE AREA
0578 EB                     XCHG                    ;POINT TO SELECT MASK
0579 3AAD00                 LDA     TRAKNO          ;NEXT TRACK FOR I/O
057C FE02                   CPI     002H            ;IS IT TRACK ZERO OR ONE
057E 3EFF                   MVI     A,11111111B     ;ASSUME NO....
                            JRNC    SLS4            ;VERIFY ASSUMPTION
0580+3002                   DB      030H,SLS4-$-1   ;---- FAKE JRNC INSTRUCTION
0582 3EFE                   MVI     A,11111110B     ;FORCE SINGLE DENSITY FOR O

                    SLS4:
0584 A6                     ANA     M               ;LOAD MASK AND CORRECT IF NECESSARY
0585 D308                   OUT     008H            ;SELECT IT
0587 DB04                   IN      004H            ;IS DRIVE READY?
0589 17                     RAL                     ;
                            JRC     SLSERR          ;IF NOT...BRANCH
058A+3809                   DB      038H,SLSERR-$-1 ;---- FAKE JRC INSTRUCTION
058C EB                     XCHG                    ;RESTORE TRACK REGISTER ADDRESS
058D 7E                     MOV     A,M             ;PICK UP TRACK NUMBER
058E D305                   OUT     005H            ;GIVE IT TO CONTROLLER
0590 AF                     XRA     A               ;ENSURE CARRY IS RESET
0591 32F60A                 STA     ERFLAG          ;ALSO ZERO ERROR INDICATOR
0594 C9                     RET

0595 AF             SLSERR: XRA     A               ;ENSURE CARRY IS RESET
0596 3C                     INR     A               ;SET TO 1 FOR ERROR FLAG
0597 32F60A                 STA     ERFLAG          ;SHOW ERROR
059A C9                     RET
```

```
                ;-------------------------------------------------------------
                ;
                ;       THIS ROUTINE SETS UP THE HARD DISK BY SELECTING THE
                ;       DRIVE AND RELOADING THE HEAD AND TRACK REGISTERS IN
                ;       HARD DISK CONTROLLER READY FOR I/O LATER.
                ;
                ;-------------------------------------------------------------

                SELHARD:
                        IF      HARDSK
059B 21AC00             LXI     H,DISKNO        ;CURRENT DRIVE SELECTED
059E BE                 CMP     M               ;SAME ??
059F C8                 RZ                      ;YES, NO NEW SELECT NECESSARY
05A0 77                 MOV     M,A             ;UPDATE DISKNO

                SLH1:
05A1 CDB305             CALL    POINT           ;TRACK SAVE REGISTER
05A4 EB                 XCHG                    ;POINT TO SELECT MASK
05A5 7E                 MOV     A,M             ;LOAD DRIVE/HEAD VALUE
05A6 D320               OUT     020H            ;WRITE IT TO SELECT PORT
05A8 EB                 XCHG                    ;REGAIN ADDRESS OF TRACK REGSTR
05A9 7E                 MOV     A,M             ;LOAD OLD TRACK NUMBER
05AA D322               OUT     022H            ;WRITE IT TO OLD TRACK REGISTER
05AC 0E14               MVI     C,20            ;DELAY FOR 20 MILLISECONDS
05AE CD8207             CALL    DELAY           ;
05B1 37                 STC                     ;SET CARRY TO SHOW HARD DISK
                        ENDIF
05B2 C9                 RET                     ;RETURN TO CALLER

                        PAGE

                ;-------------------------------------------------------------
                ;
                ;       SUBROUTINE TO POINT TO CURRENT TRACK REGISTER SAVE
                ;
                ;-------------------------------------------------------------

                POINT:
05B3 2AAC00             LHLD    DISKNO          ;PICKUP CURRENT DISK
05B6 7D                 MOV     A,L             ;
05B7 2600               MVI     H,0             ;RESET HIGH ORDER HALF
05B9 117000             LXI     D,TRK0          ;LOAD TRACK POINTER
05BC 19                 DAD     D               ;POINT TO CURRENT TRACK PTR
05BD 54                 MOV     D,H             ; DE = TRACK
05BE 5D                 MOV     E,L             ;
05BF 010C00             LXI     B,12            ;
05C2 09                 DAD     B               ; HL = SELECT
                        IF      HARDSK
05C3 FE04               CPI     4               ;
                        JRC     PNTFN           ; FLOPPY DISK
05C5+380D               DB      038H,PNTFN-$-1    ;---- FAKE JRC INSTRUCTION
F5C7 3E10               MVI     A,10H           ;
05C9 A6                 ANA     M               ; CHECK DRIVE SELECT
                        JRZ     PNTH2           ;   MUST BE DRIVE # 2
05CA+2805               DB      028H,PNTH2-$-1  ;---- FAKE JRZ INSTRUCTION
```

```
05CC 11B800             LXI     D,HTK1          ; POINT TO DRIVE 1
                        JR      PNTFN           ;
05CF+1803               DB      018H,PNTFN-$-1  ;---- FAKE JR INSTRUCTION -
05D1 11B900     PNTH2:  LXI     D,HTK2          ; POINT TO DRIVE 2
                        ENDIF
05D4 EB         PNTFN:  XCHG                    ; SWITCH
05D5 C9                 RET                     ; HL = TRACK    DE = SELECT

                ;---------------------------------------------------------------
                ;
                ;       ROUTINE TO TRANSLATE SECTOR NUMBER
                ;
                ;---------------------------------------------------------------

                SECTRAN:
05D6 EB                 XCHG                    ;TABLE ADDRESS IS IN DE (NO
05D7 7C                 MOV     A,H             ;IS THERE A TABLE ADDRESS ?
05D8 B5                 ORA     L               ;
                        JRZ     STRN2           ;NO, JUST RETURN ENTERED QUEUE
05D9+2807               DB      028H,STRN2-$-1  ;---- FAKE JRZ INSTRUCTION

                STRN1:
05DB 0600               MVI     B,000H          ;ENSURE OK FOR SINGLE BYTE
05DD 09                 DAD     B               ;ADD SECTOR NUMBER
05DE 6E                 MOV     L,M             ;LOAD TRANSLATED VALUE
05DF 2600               MVI     H,000H          ;
05E1 C9                 RET                     ;NEW VALUE RETURNED IN HL

                STRN2:
05E2 09                 DAD     B               ;RETURN SAME VALUE AS ENTERED
05E3 C9                 RET                     ;

                ;---------------------------------------------------------------
                ;
                ;       ROUTINES TO DO FLOPPY I/O
                ;
                ;---------------------------------------------------------------

                READSOFT:
05E4 3E9F               MVI     A,09FH          ;MASK FOR READ STATUS
05E6 32FB0A             STA     MASK            ;
05E9 3E01               MVI     A,001H          ;SETUP DMA FOR READ
05EB 32CE00             STA     DMAS3F          ;
05EE 3E8C               MVI     A,08CH          ;READ COMMAND
                        JR      SRW1            ;
05F0+180F               DB      018H,SRW1-$-1   ;---- FAKE JR INSTRUCTION -

                WRITESOFT:
05F2 3EFF               MVI     A,0FFH          ;MASK FOR WRITE STATUS
05F4 32FB0A             STA     MASK            ;
05F7 CD6B0E             CALL    MVDTB           ;
05FA 3E05               MVI     A,005H          ;SETUP DMA FOR WRITE
05FC 32CE00             STA     DMAS3F          ;
05FF 3EAC               MVI     A,0ACH          ;WRITE COMMAND
```

```
                SRW1:
0601 32FA0A             STA     CMD             ;
0604 211D13             LXI     H,FPYBUF        ;
0607 22BC00             SHLD    DMASA           ;
060A 3AE60A             LDA     NEWDSK          ;
060D CD5205             CALL    DSKSEL          ;SELECT DRIVE FOR I/O
0610 3AF60A             LDA     ERFLAG          ;CHECK FOR SELECT ERROR
0613 B7                 ORA     A               ;
0614 C0                 RNZ                     ;RETURN IF ERROR

                SRW2:
0615 3E0A               MVI     A,10            ;SET NUMBER OF TRIALS
0617 32020B             STA     T$RETRIES       ;SAVE FOR RETRY ROUTINE
061A AF                 XRA     A
061B 32030B             STA     HOME$TOGGLE     ;FORCE HOME PRIOR TO EACH RETRY

                LOAD$HEAD:
061E DB08               IN      008H            ;IS HEAD LOADED ??
0620 E602               ANI     00000010B       ;CHECK IT....
                        JRNZ    REMOVE$LD       ;YES, ITS LOADED, DONT RELOAD
0622+201F               DB      020H,REMOVE$LD-$-1      ;---- FAKE JRNZ INST
0624 DB05               IN      005H            ;DUMMY SEEK TO START HEAD LOAD
0626 D307               OUT     007H            ;KEEP IT SHORT....
0628 3E1A               MVI     A,01AH          ;START HEAD LOADING
062A CD6307             CALL    FINTFIX         ;CLEAR ANY PENDING INTERRUPT
                ;                               ;AND ISSUE COMMAND
062D CD3A07     LDH1:   CALL    FPYWAIT         ;WAIT FOR I/O TO COMPLETE
0630 3AFC0A             LDA     STATUS          ;HOW DID IT GO?
0633 E698               ANI     10011000B       ;CHECK
                        JRNZ    CHECKIT         ;DO NOT GO ON IF ERROR
0635+2044               DB      020H,CHECKIT-$-1        ;---- FAKE JRNZ INS

0637 0E10               MVI     C,16            ;WAIT HERE FOR 16 MS
0639 CD8207             CALL    DELAY           ;CALL WAIT ROUTINE
063C CDB305             CALL    POINT           ;REESTABLISH TRACK REGISTER
063F 36FE               MVI     M,254           ;ENSURE FURTHER SEEK AND DELAY
                        JR      TRKTST          ;
0641+1807               DB      018H,TRKTST-$-1 ;---- FAKE JR INSTRUCTION -

                REMOVE$LD:
0643 21FA0A             LXI     H,CMD           ;POINT TO I/O COMMAND
0646 3EFB               MVI     A,11111011B     ;REMOVE HEAD LOAD BIT
0648 A6                 ANA     M               ;DO IT....
0649 77                 MOV     M,A             ;SAVE IT BACK INTO CMD

                TRKTST:
064A CDB305             CALL    POINT           ;RESTORE TRACK REGISTER POINTER
064D 3AE70A             LDA     NEWTRK          ;GET NEW TRACK NUMBER
0650 32AD00             STA     TRAKNO          ;SAVE IN COMMON PLACE
0653 BE                 CMP     M               ;SAME AS LAST TIME ??
                        JRZ     FSECSET         ;YES, DONT BOTHER WITH SEEK
0654+281A               DB      028H,FSECSET-$-1        ;---- FAKE JRZ INST
0656 77                 MOV     M,A             ;SAVE IT
0657 D307               OUT     007H            ;ALSO SEND IT TO CONTROLLER
```

```
0659 CD1905             CALL    DBL$UPDATE      ;DOUBLE SIDED SUPPORT

               FLOPPY$SEEK:
065C 3E1A               MVI     A,01AH          ;SEEK COMMAND WITH HEAD LOAD
065E CD6307             CALL    FINTFIX         ;CLEAR ANY PENDING INTERRUPT
               ;                                ;AND ISSUE COMMAND
0661 CD3A07    FPS1:    CALL    FPYWAIT         ;WAIT FOR I/O TO COMPLETE
0664 3AFC0A             LDA     STATUS          ;HOW DID IT GO?
0667 E698               ANI     10011000B       ;CHECK
                        JRNZ    CHECKIT         ;DO NOT GO ON IF ERROR
0669+2010               DB      020H,CHECKIT-$-1        ;---- FAKE JRNZ INS

066B 0E10               MVI     C,16            ;SET FOR 16 MS DELAY
066D CD8207             CALL    DELAY           ;
               FSECSET:
0670 3AE90A             LDA     NEWSEC          ;SET SECTOR
0673 32B100             STA     SECTNO          ;SAVE IN COMMON PLACE
0676 D306               OUT     006H            ;

0678 CD8706             CALL    FLOPPYIO        ;DO I/O
               CHECKIT:
067B CDAE06             CALL    CHECK$STAT      ;CHECK STATUS OF I/O
067E 3AF60A             LDA     ERFLAG          ;SETUP TO RETURN TO BDOS
0681 CC7E0E             CZ      MVDFB           ;
0684 C8                 RZ                      ;EITHER OK OR PERMANENT ERROR
                        JR      LOAD$HEAD       ;ERROR, JUST RETRY THIS SAME
0685+1897               DB      018H,LOAD$HEAD-$-1      ;---- FAKE JR INSTR

                        PAGE

               ;---------------------------------------------------------------
               ;
               ;        THIS IS THE ROUTINE THAT DOES THE FLOPPY DISK I/O
               ;
               ;---------------------------------------------------------------

               FLOPPYIO:
                        IF      NOT DMA
                        LXI     H,066H          ;MOVE DATA FROM 066H TO SAVE
                        LXI     D,SAVE1         ;
                        LXI     B,004H          ;
                        LDIR                    ;MOVE IT

                        LXI     H,NMIRTN        ;SET NMI ROUTINE TO NMI ADDRESS
                        LXI     D,066H          ;
                        LXI     B,004H          ;
                        LDIR                    ;MOVE IT

                        LDA     CMD             ;IS IT A WRITE ??
                        ANI     20H             ;
                        JZ      FRD             ;NO, LEAVE INI CMD IN LOW MEMRY
                        LXI     H,067H          ;POINT TO COMMAND AREA
                        MVI     M,0A3H          ;MAKE IT AN OTI CMD....
               FRD      EQU     $               ;LABEL
                        ENDIF
```

```
                           IF      DMA
0687 21BA00                LXI     H,DMAS1          ;INITIALIZE DMA
068A 010006                LXI     B,0600H          ;
                           OUTIR                    ;WRITE TO DMA
068D+EDB3                  DB      0EDH,0B3H        ;---- FAKE OTIR INSTRUCTION
068F 21C600                LXI     H,DMAS2F         ;
0692 010004                LXI     B,0400H          ;
                           OUTIR                    ;WRITE TO DMA
0695+EDB3                  DB      0EDH,0B3H        ;---- FAKE OTIR INSTRUCTION
0697 21CA00                LXI     H,DMAS3          ;
069A 010007                LXI     B,0700H          ;
                           OUTIR                    ;WRITE TO DMA
069D+EDB3                  BD      0EDH,0B3H        ;---- FAKE OTIR INSTRUCTION
                           ENDIF

069F 0E07                  MVI     C,007H           ;PORT ADDRESS FOR I/O
06A1 211D13                LXI     H,FPYBUF         ;DMA ADDRESS
06A4 3AFA0A                LDA     CMD              ;I/O COMMAND
06A7 CD6307                CALL    FINTFIX          ;CLEAR ANY PENDING INTERRUPT
                   ;                                ;AND ISSUE COMMAND
06AA CD3A07        FWT1:   CALL    FPYWAIT          ;WAIT HERE FOR I/O TO COMPLETE

                           IF      NOT DMA
                           LXI     H,SAVE1          ;SETUP TO REPLACE DATA
                           LXI     D,066H           ;COPIED FROM NMI LOCATION
                           LXI     B,004H           ;
                           LDIR                     ;MOVE IT....
                           ENDIF

06AD C9                    RET                      ;RETURN, I/O COMPLETED

                   ;-------------------------------------------------------------
                   ;
                   ;       WE WILL NOW CHECK THE STATUS OF THE I/O OPERATION
                   ;               RETURN WITH CONDITION CODE ZERO = NO RETRY
                   ;               RETURN WITH CONDITION CODE NON ZERO = RETRY
                   ;
                   ;-------------------------------------------------------------

                   CHECK$STAT:
06AE 21F60A                LXI     H,ERFLAG         ;POINT TO ERROR INDICATOR
06B1 3600                  MVI     M,000H           ;ASSUME OK
06B3 21FC0A                LXI     H,STATUS         ;CHECK STATUS
06B6 3AFB0A                LDA     MASK             ;MASK FOR UNWANTED BIT REMOVAL
06B9 A6                    ANA     M                ;
06BA 77                    MOV     M,A              ;SAVE CLEANED STATUS
06BB C8                    RZ                       ;OK, SO RETURN

                   CHKS0:
06BC CDEE02                CALL    RETMOD           ;
06BF FE03                  CPI     003H             ;HARD DISK ??
06C1 21FC0A                LXI     H,STATUS         ;
06C4 7E                    MOV     A,M              ;RELOAD STATUS BYTE
                           JRNC    CHKS2            ;YES, CHECK FOR DRIVE READY
```

```
06C5+3006                     DB      030H,CHKS2-$-1   ;---- FAKE JRNC INSTRUCTION

                   CHKS1:
06C7 FE80                     CPI     080H             ;IS FLOPPY DISK NOT READY ?
                              JRZ     BADIO            ;YES, DONT BOTHER WITH RETRY
06C9+283E                     DB      028H,BADIO-$-1   ;---- FAKE JRZ INSTRUCTION
                              JR      CHKS3            ;GO TO BAD MESSAGE ROUTINE
06CB+1819                     DB      018H,CHKS3-$-1   ;---- FAKE JR INSTRUCTION -

                   CHKS2:
06CD FE00                     CPI     000H             ;IS HARD DISK NOT READY ??
                              JRZ     BADIO            ;YES, BYPASS ERROR MESSAGE
06CF+2837                     DB      028H,BADIO-$-1   ;---- FAKE JRZ INSTRUCTION
06D1 E640                     ANI     01000000B        ;IS IT WRITE FAULT ??
                              JRZ     CHKS3            ;NO, CONTINUE ON
06D3+2811                     DB      028H,CHKS3-$-1   ;---- FAKE JRZ INSTRUCTION
06D5 CDB305                   CALL    POINT            ;POINT TO TRACK REGISTER
06D8 EB                       XCHG                     ;POINT TO SELECT MASK
06D9 7E                       MOV     A,M              ;
06DA F640                     ORI     01000000B        ;TURN ON WRITE FAULT CLEAR
06DC D320                     OUT     020H             ;
06DE 7E                       MOV     A,M              ;RESET CLEAR
06DF D320                     OUT     020H             ;
06E1 0E14                     MVI     C,20             ;DELAY JUST TO BE SAFE
06E3 CD8207                   CALL    DELAY            ;

                   CHKS3:
06E6 3A030B                   LDA     HOME$TOGGLE
06E9 B7                       ORA     A                ;IS A HOME NEEDED ON THIS RETRY
                              JRNZ    CHKS4            ;
06EA+200B                     DB      020H,CHKS4-$-1   ;---- FAKE JRNZ INSTRUCTION

06EC 3AFC0A                   LDA     STATUS           ;SAVE STATUS OVER HOME
06EF F5                       PUSH    PSW              ;
06F0 CD1A03                   CALL    HOME             ;RESET DEVICE TO HOME
06F3 F1                       POP     PSW              ;
06F4 32FC0A                   STA     STATUS           ;SAVE FOR ERROR MESSAGE

                   CHKS4:
06F7 119400                   LXI     D,TCNT           ;BUMP TEMP ERROR COUNT
06FA CD0F07                   CALL    ADDERRORS        ;
06FD 21020B                   LXI     H,T$RETRIES      ;PICKUP RETRY COUNT
0700 35                       DCR     M                ;DECREMENT COUNT OF RETRIES
0701 C0                       RNZ                      ;

0702 11A000                   LXI     D,PCNT           ;BUMP PERMANENT ERROR COUNT
0705 CD0F07                   CALL    ADDERRORS        ;

                   BADIO:
0708 21F60A                   LXI     H,ERFLAG         ;SET PERMANENT ERROR
070B 3601                     MVI     M,001H           ;DO IT....
070D AF                       XRA     A                ;RESET TO PRECLUDE RETRIES
070E C9                       RET                      ;RETURN TO CALLER

                   ADDERRORS:
```

```
070F 2AAC00             LHLD    DISKNO          ;BUMP COUNT OF DISK ERRORS
0712 2600               MVI     H,000H          ;
0714 19                 DAD     D               ;POINT TO ERROR REGISTER
0715 34                 INR     M               ;
0716 C9                 RET                     ;

                        PAGE

                ;--------------------------------------------------
                ;
                ;       THIS IS HARD DISK WAIT ENTRY
                ;
                ;--------------------------------------------------
                ;

                WAIT0:
0717 C5                 PUSH    B               ; SAVE RETRY COUNT
0718 0E84               MVI     C,FLAGWT        ; FUNCTION FLAG WAIT
071A 1E05               MVI     E,HDFLAG        ; DEVICE IS HARD DISK
071C CD100B             CALL    XDOS
071F C1                 POP     B               ; RESTORE RETRY COUNTER IN

                ;       READ OR WRITE IS OK, ACCUMULATOR CONTAINS ZERO

0720 C9                 RET

                ;--------------------------------------------------
                ;
                ;       THE FOLLOWING CODE GUARANTEES THAT HARD DISK FLAG
                ;       INTERRUPT AS IT APPEARS THAT WE OCCASIONALLY GET
                ;       FLAG SET AS A RETRY OF AN INTERRUPT FROM THE HARD
                ;       DISK, WHEN WE DO NOT EXPECT IT.
                ;
                ;--------------------------------------------------

                INTFIX:
0721 F5                 PUSH    PSW
0722 C5                 PUSH    B
0723 D5                 PUSH    D
0724 E5                 PUSH    H

0725 0E85               MVI     C,FLAGST
0727 1E05               MVI     E,HDFLAG
0729 CD100B             CALL    XDOS            ;EITHER FLAG 5 WILL BE SET
                ;                               ;IT IS ALREADY SET - IN WHICH
                ;                               ;THIS REQUEST WILL BE IGNORED

072C 0E84               MVI     C,FLAGWT
072E 1E05               MVI     E,HDFLAG
0730 CD100B             CALL    XDOS            ;NOW CLEAR THE FLAG

0733 E1                 POP     H
0734 D1                 POP     D
```

```
0735 C1                 POP     B
0736 F1                 POP     PSW                 ;RESTORE REGISTERS

0737 D323               OUT     023H                ;ISSUE COMMAND TO HARD DISK

0739 C9                 RET

                        PAGE

                ;------------------------------------------------------------
                ;
                ;       THIS IS FLOPPY DISK WAIT ENTRY
                ;
                ;------------------------------------------------------------

                FPYWAIT:
073A C5                 PUSH    B           ;SAVE RETRY COUNT
073B E5                 PUSH    H
073C 0E84               MVI     C,FLAGWT            ; FUNCTION IS FLAG WAIT
073E 1E06               MVI     E,FPYFLAG           ; WAIT FOR FLOPPY
0740 CD100B             CALL    XDOS
0743 F5                 PUSH    PSW
0744 DAD00D             LDA     FPYTIME             ;DID WD1791 GO TO SLEEP?
0747 B7                 ORA     A                   ;
                        JRNZ    NOFPYRST            ;IF STILL AWAKE, SKIP RESET
0748+2015               DB      020H,NOFPYRST-$-1       ;---- FAKE JRNZ INS

074A DB09               IN      009H                ;GET CURRENT BANK NUMBER
074C E618               ANI     00011000B           ;REMOVE OTHER INTO
074E D309               OUT     009H                ;RESET WD1791
0750 0E01               MVI     C,1                 ;DELAY 1 MILLISEC
0752 CD8207             CALL    DELAY               ;
0755 F602               ORI     00000010B           ;END RESET
0757 D309               OUT     009H                ;
0759 3AE60A             LDA     NEWDSK              ;MAKE SURE CURRENT DISK AND
075C 32AC00             STA     DISKNO              ; THE SAME
                NOFPYRST:
075F F1                 POP     PSW
0760 E1                 POP     H
0761 C1                 POP     B                   ;RESTORE ENTRY COUNT IN <C>


0762 C9                 RET

                ;------------------------------------------------------------
                ;
                ;       THE FOLLOWING CODE GUARANTEES THAT FLOPPY DISK FLAG
                ;
                ;------------------------------------------------------------

                FINTFIX:
0763 F5                 PUSH    PSW
0764 C5                 PUSH    B
0765 D5                 PUSH    D
0766 E5                 PUSH    H
```

```
0767 0E85             MVI     C,FLAGST
0769 1E06             MVI     E,FPYFLAG
076B CD100B           CALL    XDOS

076E 0E84             MVI     C,FLAGWT
0770 1E06             MVI     E,FPYFLAG
0772 CD100B           CALL    XDOS

0775 210301           LXI     H,00103H        ;SET TIME OUT INDICATOR ON
0778 22D00D           SHLD    FPYTIME         ; TIME TO BE BETWEEN 2 AND 3

077B E1               POP     H
077C D1               POP     D
077D C1               POP     B
077E F1               POP     PSW

077F D304             OUT     004H            ;ISSUE COMMAND TO FLOPPY DISK

0781 C9               RET

                      if      not mpm20
               FPYTIME:
                      DW      0

               FPYTCNT:
                      DW      0
                      endif

                      PAGE

               ;------------------------------------------------------------
               ;
               ;      THIS IS THE DELAY ROUTINE. IT WILL LOOP HERE FOR THE
               ;      NUMBER OF MILLISECONDS SPECIFIED IN REGISTER C.
               ;
               ;------------------------------------------------------------

               DELAY:
0782 0664      DEL1:  MVI     B,100           ;FORCE DELAY FOR 1 MILLISEC
0784 00        DEL2:  NOP                     ;INSTRUCTIONS TO FILL IN TIME
0785 29               DAD     H               ;
0786 29               DAD     H               ;
0787 05               DCR     B               ;AT ONE MILLISECOND YET ??
0788 C28407           JNZ     DEL2            ;NO, KEEP ON LOOPING
078B 0D               DCR     C               ;END OF REQUESTED INTERVAL
078C C28207           JNZ     DEL1            ;NO, KEEP ON
078F C9               RET                     ;RETURN TO CALLER


               ;**********************************************
               ;*     NOTE:THE INITIALIZATION CODE WILL BE
               ;*     OVERWRITTEN BY DIRBUF & FPYBUF
               ;**********************************************
```

```
                         if      not mpm20
                DIRBUF   EQU     $
                         endif


                ;-------------------------------------------------------
                ;
                ;        DISK CONFIGURATION TABLE
                ;
                ;-------------------------------------------------------

                         IF      HARDSK
                                                              ;    PIN C

0790 0000000000DSCNO:    DB      00H,00H,00H,00H,00H,00H,00H,00H
0798 1000000000          DB      10H,00H,00H,00H,00H,00H,10H,00H       ;
07A0 9090900000          DB      90H,90H,90H,00H,00H,00H,00H,00H       ;
07A8 0000000000          DB      00H,00H,00H,00H,00H,00H,00H,00H
07B0 1000002000          DB      10H,00H,00H,20H,00H,00H,10H,20H       ;
07B8 0000000000          DB      00H,00H,00H,00H,00H,00H,00H,00H
07C0 9090902000          DB      90H,90H,90H,20H,00H,00H,00H,20H       ;
07C8 909090A0A0          DB      90H,90H,90H,0A0H,0A0H,0A0H,0H,0H      ;
                         ENDIF
                ;-------------------------------------------------------
                ;
                ;        SET UP DISK CONFIGURATION
                ;
                ;        [ THIS CODE EXECUTED ONLY ONCE ]
                ;
                ;-------------------------------------------------------
                ;
07D0 217E00     SDCONF:  LXI     H,SEL0+2         ;POINT TO DRIVE C:
07D3 3AB600              LDA     MPARMS           ;
07D6 E605                ANI     05H              ; TEST FOR FOUR FLOPPIES
07D8 C3DE07              JMP     SDDBL            ;  YES  SKIP THE ZAP
07DB 77                  MOV     M,A              ;
07DC 23                  INX     H                ; ZAP C: AND D:
07DD 77                  MOV     M,A              ;
                SDDBL:
07DE 118000              LXI     D,SEL0+4         ;POINT TO DRIVE E:
                         IF      HARDSK
07E1 DB25                IN      025H             ;READ CONFIGURATION PORT
07E3 E607                ANI     07H              ;STRIP OFF HIGH PART
07E5 17                  RAL                      ;
07E6 17                  RAL                      ;
07E7 17                  RAL                      ;
07E8 0600                MVI     B,0              ;
07EA 4F                  MOV     C,A              ;POINT TO CONFIGURATION TAB
07EB 219007              LXI     H,DSCNO          ;
07EE 09                  DAD     B                ;  INDEX TO RIGHT ENTRY
07EF 0608                MVI     B,8              ;
07F1 7E         SDL1:    MOV     A,M              ; CHANGE ALL SELECT MASKS
07F2 12                  STAX    D                ;
07F3 13         SDOK:    INX     D                ; NEXT
```

```
07F4 23                      INX     H                  ;  DRIVE
                             DJNZ    SDL1               ;
07F5+10FA                    DB      010H,SDL1-$-1      ;---- FAKE DJNZ INSTRUCTION
                             ENDIF
                             IF      NOT HARDSK
                             XCHG                       ;
                             MVI     B,8                ;
                             XRA     A                  ;
                    SDL2:    MOV     M,A                ;ZAP ALL HARD DRIVES
                             INX     H                  ;
                             DJNZ    SDL2               ;
                             ENDIF

07F7 C9                      RET

07F8 =              INITEND  EQU     $

07F8 E5             XETMOD:  PUSH    H                  ;SAVE MODE BYTE ADDRESS
07F9 79                      MOV     A,C                ;SETUP FOR DENSITY CHANGE
07FA B7                      ORA     A                  ;
07FB 0E00                    MVI     C,000H             ;ASSUME SINGLE DENSITY MODE
                             JRZ     XETSEL             ;VERIFY ASSUMPTION
07FD+2802                    DB      028H,XETSEL-$-1    ;---- FAKE JRZ INSTRUCTION
07FF 0E01                    MVI     C,001H             ;SET FOR DOUBLE DENSITY MODE
0801 CD7802         XETSEL:  CALL SETDEN                ;SET DENSITY BASED ON LOW BIT
0804 E1                      POP     H                  ;RESTORE
0805 6E                      MOV     L,M                ;PICKUP MODE AGAIN
0806 2600                    MVI     H,000H             ;FOR SINGLE BYTE PRECISION
0808 7D                      MOV     A,L                ;SAVE MODE IN ACCUMULATOR F
0809 29                      DAD     H                  ;* 2
080A 29                      DAD     H                  ;* 4
080B E5                      PUSH    H                  ;SAVE * 4
080C 29                      DAD     H                  ;* 8
080D D1                      POP     D                  ;REGAIN * 4
080E 19                      DAD     D                  ;* 12
080F 119101                  LXI     D,MODL0            ;FIRST MODEL DPE
0812 19                      DAD     D                  ;POINT TO THIS ONE
0813 EB                      XCHG                       ;SETUP TEMPORARILY AS DESTINATION
0814 2AB200                  LHLD    DPEPTR             ;ADDRESS OF CURRENTLY SELECT DRIVE
0817 EB                      XCHG                       ;SETUP TO ALTER
0818 010C00                  LXI     B,12               ;LENGTH FOR MOVE
                             LDIR                       ;DO MOVE
081B+EDB0                    DB      0EDH,0B0H          ;---- FAKE LDIR INSTRUCTION
081D C9                      RET                        ;RETURN TO CALLER

                             PAGE

                    ;-----------------------------------------------------------
                    ;
                    ;        THE FOLLOWING AREA CONTAINS THE DISK/WORK SAVE AREA
                    ;        USED BY THE CBIOS IN THE NORMAL COURSE OF ACTIVITY.
                    ;
                    ;-----------------------------------------------------------
```

```
                        if      mpm20
                ;tempbuf        equ     (dirbuf-base)+128
                        else
                TEMPBUF EQU     (DIRBUF-BASE)+256
                        ORG TEMPBUF+((INITEND-BASE)/TEMPBUF)*((INITEND-BASE
                        endif

081E =          BEGDAT  EQU     $                               ;START OF BDOS AREA
                ;DIRBUF:DS      128                     ;OVERLAYS  SYSTEMINIT CODE
081E            ALV0:   DS      32
083E            CSV0:   DS      32
085E            ALV1:   DS      32
087E            CSV1:   DS      32
089E            ALV2:   DS      32
08BE            CSV2:   DS      32
08DE            ALV3:   DS      32
08FE            CSV3:   DS      32
                        IF      HARDSK
091E            ALV4:   DS      64
095E            CSV4:   DS      0
095E            ALV5:   DS      64
099E            CSV5:   DS      0
099E            ALV6:   DS      64
09DE            CSV6:   DS      0
09DE            ALV7:   DS      64
0A1E            CSV7:   DS      0
0A1E            ALV8:   DS      64
0A5E            CSV8:   DS      0
0A5E            ALV9:   DS      64
0A9E            CSV9:   DS      0
0A9E            ALVA:   DS      36
0AC2            CSVA:   DS      0
0AC2            ALVB:   DS      36
0AE6            CSVB:   DS      0
                        endif

                        if      mdisk
                ALVC:   DS      32                              ;VIRTUAL DISK
                CSVC:   DS      0
                        endif

                        if      not mpm20
                        if      hardsk
                        DS      1                               ;MUST PRECEDE HSTBUF
                HSTBUF: DS      1024                            ;HOST BUFFER AREA
                        DS      1                               ;MUST FOLLOW HSTBUF
                        ENDIF

                FPYBUF  EQU     DIRBUF+128                      ; FLOPPY I/O BUFFER
                        endif

0AE6            NEWDSK: DS      1                               ;SEEK DISK NUMBER
0AE7            NEWTRK: DS      2                               ;SEEK TRACK NUMBER
0AE9            NEWSEC: DS      1                               ;SEEK SECTOR NUMBER
```

```
0AEA            HSTDSK: DS      1                       ;HOST DISK NUMBER
0AEB            HSTTRK: DS      2                       ;HOST TRACK NUMBER
0AED            HSTSEC: DS      1                       ;HOST SECTOR NUMBER

0AEE            NEWHST: DS      1                       ;SEEK SHR SECSHF
0AEF            HSTACT: DS      1                       ;HOST ACTIVE FLAG
0AF0            HSTWRT: DS      1                       ;HOST WRITTEN FLAG

0AF1            UNACNT: DS      1                       ;UNALLOCATED RECORD
0AF2            UNADSK: DS      1                       ;LAST UNALLOCATED DISK
0AF3            UNATRK: DS      2                       ;LAST UNALLOCATED TRACK
0AF5            UNASEC: DS      1                       ;LAST UNALLOCATED SECTR

0AF6            ERFLAG: DS      1                       ;ERROR REPORTING
0AF7            RSFLAG: DS      1                       ;READ SECTOR FLAG
0AF8            READOP: DS      1                       ;1 IF READ OPERATION
0AF9            WRTYPE: DS      1                       ;WRITE OPERATION TYPE

0AFA 00         CMD:    DB      0                       ;COMMANDS FOR NEXT
0AFB 00         MASK:   DB      0                       ;STATUS MASKS BUFFER
0AFC 00         STATUS: DB      0                       ;STATUS SAVE LOCATION

0AFD 00000000   SAVE1:  DB      000H,000H,000H,000H     ;SAVE AREA FOR NMI
0B0A 00         P$RETRIES: DB   000H                    ;COUNTER FOR PERMANENT
0B02 00         T$RETRIES: DB   000H                    ;COUNTER FOR TEMPORARY
                HOME$TOGGLE:
0B03 00                 DB      000H                    ;INDICATOR TO TELL
                ;                                       ;.. IF HOME SHOULD


                        page

                        if      mpm20

                ; ***********************************************************
                ; *
                ; *          M P / M   2 . 0   C O M M O N   B A S E
                ; *
                ; ***********************************************************

                commonbase:
0B04 C3150B             jmp     coldstart
0B07 C30000     swtuser:jmp     $-$
0B0A C30000     swtsys: jmp     $-$
0B0D C30000     pdisp:  jmp     $-$
0B10 C30000     xdos:   jmp     $-$
0B13 0000       sysdat: dw      $-$
                COLDSTART:
                WARMSTART:
0B15 0E00               MVI     C,0             ; SEE SYSTEM INIT
                                                ; COLD & WARM START INCLUDE
                                                ; FOR COMPATIBILITY WITH CP
0B17 C3100B             JMP     XDOS            ; SYSTEM RESET, TERMINATE P
```

```
                  rtnempty:
0B1A AF                     xra     a
0B1B C9                     ret

                  NULL$INT:
0B1C FB                     EI
                            RETI
0B1D+ED4D                   DB      0EDH,04DH         ;---- FAKE RETI INSTRUCTION
                            endif


                  ;-----------------------------------------------------------
                  ;
                  ;         CENTRONICS PRINTER ROUTINE (WITH SEPARATE BUSY TEST
                  ;
                  ;-----------------------------------------------------------

                  CNSTAT:
0B1F 3E01                   MVI     A,001H            ;TO SET STROBE HIGH
0B21 D310                   OUT     010H              ;
0B23 DB10                   IN      010H              ;READ PRINTER STATUS
0B25 E620                   ANI     020H              ;REMOVE ALL BUT BUSY BIT
0B27 3EFF                   MVI     A,0FFH            ;ASSUME NOT BUSY
0B29 C8                     RZ                        ;CHECK ASSUMPTION
0B2A AF                     XRA     A                 ;SET TO SHOW STILL BUSY
0B2B C9                     RET                       ;
                  ;
                  CLIST:
0B2C CD1F0B                 CALL    CNSTAT            ;IS PRINTER READY NOW?
0B2F B7                     ORA     A
                            JRNZ    CLIST1            ;IF READY, SKIP POLL
0B30++2009                  DB      020H,CLIST1-$-1 ;---- FAKE JRNZ INSTRUCTION

0B32 C5                     PUSH    B                 ;
0B33 0E83                   MVI     C,POLL            ; POLL DEVICE
0B35 1E00                   MVI     E,PLLPT           ;  PRINTER
0B37 CD100B                 CALL    XDOS              ;WAIT FOR PRINTER TO FREE UP
0B3A C1                     POP     B                 ;

                  CLIST1:
0B3B 79                     MOV     A,C               ;CHARACTER TO PRINT
0B3C D311                   OUT     011H              ;WRITE IT TO DATA PORT
0B3E 3E00                   MVI     A,000H            ;TO FORCE STROBE LOW
0B40 D310                   OUT     010H              ;
0B42 3E01                   MVI     A,001H            ;TO FORCE STROBE HIGH
0B44 D310                   OUT     010H              ;
0B46 C9                     RET                       ;

                            PAGE

                  ;-----------------------------------------------------------
                  ;
                  ;         DISK INTERRUPT ROUTINE
                  ;
                  ;-----------------------------------------------------------
```

```
                    FLOPPY$INT:
0B47 22C80D                 SHLD    SVDHL
0B4A 21500B                 LXI     H,FDINTH
0B4D C37F0D                 JMP     INTINIT
                    FDINTH:
0B50 DB04                   IN      004H            ;GET STATUS
0B52 32FC0A                 STA     STATUS          ;SAVE FOR I/O ROUTINE
0B55 3E00                   MVI     A,0             ;STOP TIMING OF RESPONSE TO
0B57 32D10D                 STA     FPYTIME+1       ;
0B5A 1E06                   MVI     E,FPYFLAG       ;SHOW I/O COMPLETED
                            JR      HDSTFLG
0B5C+1813                   DB      018H,HSDTFLG-$-1         ;---- FAKE JR INSTR

                    HARD$INTH:
0B5E 22C80D                 SHLD    SVDHL
0B61 21670B                 LXI     H,HDINTH
0B64 C37F0D                 JMP     INTINIT

0B67 DB24                   IN      024H            ;GET STATUS
0B69 32FC0A                 STA     STATUS          ;SAVE FOR CHECK LATER

0B6C AF                     XRA     A
0B6D D323                   OUT     023H            ;RESET INTERRUPT BY RELOADING

0B6F 1E05                   MVI     E,HDFLAG        ;SHOW I/O COMPLETED
                    HDSTFLG:
0B71 0E85                   MVI     C,FLAGST
0B73 CD100B                 CALL    XDOS
0B76 C3670D                 JMP     INTDONE

                            PAGE

                    ;------------------------------------------------------------
                    ;
                    ;       CONSOLE DISPLAY ROUTINES
                    ;
                    ;------------------------------------------------------------

                    ;
                    CONST:                     ; CONSOLE STATUS
0B79 CD9A0B                 CALL    PTBLJMP    ; COMPUTE AND JUMP TO HNDLR
0B7C AD0B                   DW      PT0ST      ; CONSOLE #0 STATUS ROUTINE
0B7E EC0B                   DW      PT1ST      ; CONSOLE #1 STATUS ROUTINE
0B80 2B0C                   DW      PT2ST      ; CONSOLE #2 STATUS ROUTINE
0B82 6A0C                   DW      PT3ST      ; CONSOLE #3 STATUS ROUTINE

                    CONIN:                     ; CONSOLE INPUT
0B84 CD9A0B                 CALL    PTBLJMP    ; COMPUTE AND JUMP TO HNDLR
0B87 B80B                   DW      PT0IN      ; CONSOLE #0 INPUT
0B89 F70B                   DW      PT1IN      ; CONSOLE #1 INPUT
0B8B 360C                   DW      PT2IN      ; CONSOLE #2 INPUT
0B8D 750C                   DW      PT3IN      ; CONSOLE #3 INPUT

                    CONOUT:                    ; CONSOLE OUTPUT
```

```
0B8F CD9A0B            CALL    PTBLJMP ; COMPUTE AND JUMP TO HNDLR
0B92 CA0B              DW      PT0OUT  ; CONSOLE #0 OUTPUT
0B94 090C              DW      PT1OUT  ; CONSOLE #1 OUTPUT
0B96 480C              DW      PT2OUT  ; CONSOLE #2 OUTPUT
0B98 870C              DW      PT3OUT  ; CONSOLE #3 OUTPUT

               ;
               PTBLJMP:                ; COMPUTE AND JUMP TO HANDLR
                                       ; D = CONSOLE #
                                       ; DO NOT DESTROY <D>
0B9A 7A                MOV     A,D
0B9B FE04              CPI     NMBCNS
                       JRC     TBLJMP
0B9D+3803              DB      038H<TBLJMP-$-1 ;---- FAKE JRC INSTRUCTION
0B9F F1                POP     PSW     ; THROW AWAY TABLE ADDRESS
0BA0 AF                XRA     A
0BA1 C9                RET
               TBLJMP:                 ;COMPUTE AND JUMP TO HANDLER
                                       ; A = TABLE INDEX
0BA2 87                ADD     A       ; DOUBLE TABLE INDEX FOR ADR OFFST
0BA3 E1                POP     H       ; RETURN ADR POINTS TO JUMP TBL
0BA4 5F                MOV     E,A
0BA5 1600              MVI     D,0
0BA7 19                DAD     D       ; ADD TABLE INDEX * 2 TO TBL BASE
0BA8 5E                MOV     E,M     ; GET HANDLER ADDRESS
0BA9 23                INX     H
0BAA 56                MOV     D,M
0BAB EB                XCHG
0BAC E9                PCHL            ; JUMP TO COMPUTED CNS HANDLER


                       PAGE

               ;-------------------------------------------------------------
               ;
               ;       SERIAL PORT ADDRESS EQUATES
               ;
               ;-------------------------------------------------------------

001C =         DATA0   EQU     01CH            ;CONSOLE #0 DATA
001D =         STS0    EQU     DATA0+1         ;CONSOLE #0 STATUS
002C =         DATA1   EQU     02CH            ;CONSOLE #1 DATA
002D =         STS1    EQU     DATA1+1         ;CONSOLE #1 STATUS
002E =         DATA2   EQU     02EH            ;CONSOLE #2 DATA
002F =         STS2    EQU     DATA2+1         ;CONSOLE #2 STATUS
002A =         DATA3   EQU     02AH            ;CONSOLE #3 DATA
002B =         STS3    EQU     DATA3+1         ;CONSOLE #3 STATUS
001E =         LPTPRT0 EQU     01EH            ;PRINTER #0 DATA
001F =         LPTSTS0 EQU     LPTPRT0+1       ;PRINTER #0 STATUS
0028 =         LPTPRT1 EQU     028H            ;PRINTER #1 DATA
0029 =         LPTSTS1 EQU     LPTPRT1+1       ;PRINTER #1 STATUS

                       PAGE
```

```
                    ;-----------------------------------------------------------------
                    ;
                    ;       POLL CONSOLE  # 0   INPUT
                    ;
                    ;-----------------------------------------------------------------

                    POLCI0:
                    PT0ST:                                  ; TEST CONSOLE STATUS
0BAD AF                     XRA     A                       ;  RETURN  0FFH IF READY
0BAE D31D                   OUT     STS0                    ;          0FFH IF NOT
0BB0 DB1D                   IN      STS0                    ;
0BB2 E601                   ANI     1                       ;  RX CHAR ?
0BB4 C8                     RZ                              ;   NO
0BB5 3EFF                   MVI     A,0FFH                  ;   YES  -  SET FLAG
0BB7 C9                     RET                             ;

                    ;
                    ;-----------------------------------------------------------------
                    ;
                    ;       CONSOLE  # 0  INPUT
                    ;
                    ;-----------------------------------------------------------------
                    ;
                    PT0IN:                                  ; RETURN CHAR IN REG A
0BB8 CDAD0B                 CALL    POLCI0                  ;IS IT READY NOW?
0BBB B7                     ORA     A                       ;
                            JRNZ    PT0IN1                  ;IF READY, SKIP POLL
0BBC+2007                   DB      020H,PT0IN1-$-1 ;---- FAKE JRNZ INSTRUCTION
0BBE 0E83                   MVI     C,POLL                  ;
0BC0 1E05                   MVI     E,PLCI0                 ; POLL CONSOLE #0 INPUT
0BC2 CD100B                 CALL    XDOS                    ;
0BC5 DB1C           PT0IN1: IN      DATA0                   ; READ CHARACTER
0BC7 E67F                   ANI     7FH                     ; STRIP PARITY
0BC9 C9                     RET                             ;
                    ;
                    ;
                    ;-----------------------------------------------------------------
                    ;
                    ;       CONSOLE  # 0  OUTPUT
                    ;
                    ;-----------------------------------------------------------------
                    ;
                    PT0OUT:                                 ;REG C = CHAR TO OUTPUT
0BCA CDDD0B                 CALL    POLCO0                  ;IS IT READY NOW?
0BCD C7                     ORA     A                       ;
                            JRNZ    PT0OUT1                 ;IF READY, SKIP POLL
0BCE+2009                   DB      020H,PT0OUT1-$-1        ;---- FAKE JRNZ INS
0BD0 C5                     PUSH    B                       ;
0BD1 0E83                   MVI     C,POLL                  ;
0BD3 1E01                   MVI     E,PLCO0                 ;
0BD5 CD100B                 CALL    XDOS                    ; POLL CONSOLE #0 OUTPUT
0BD8 C1                     POP     B                       ;
                    PT0OUT1:
0BD9 79                     MOV     A,C                     ;
0BDA D31C                   OUT     DATA0                   ; TRANSMIT CHARACTER
```

```
0BDC C9                   RET                         ;
                    ;
                    ;
                    ;-------------------------------------------------------------
                    ;
                    ;     POLL CONSOLE  # 0  OUTPUT
                    ;
                    ;-------------------------------------------------------------
                    ;
                    POLCO0:                           ; RETURN 0FFH IF READY
0BDD 3E10                 MVI     A,10H               ;        000H IF NOT
0BDF D31D                 OUT     STS0                ; RESET INT BIT
0BE1 DB1D                 IN      STS0                ; READ STATUS
0BE3 E60C                 ANI     0CH                 ; MASK FOR DTR AND TXE
0BE5 FE0C                 CPI     0CH                 ; MUST HAVE BOTH
0BE7 3E00                 MVI     A,0                 ;
0BE9 C0                   RNZ                         ; RETURN NOT READY
0BEA 3D                   DCR     A                   ;CHANGE "A" TO 0FFH
0BEB C9                   RET                         ; RETURN READY

                          PAGE


                    ;-------------------------------------------------------------
                    ;
                    ;     POLL CONSOLE  # 1   INPUT
                    ;
                    ;-------------------------------------------------------------

                    POLCI1:
                    PT1ST:                            ; TEST CONSOLE STATUS
0BEC AF                   XRA     A                   ;  RETURN  0FFH IF READY
0BED D32D                 OUT     STS1                ;          000H IF NOT
0BEF DB2D                 IN      STS1                ;
0BF1 E601                 ANI     1                   ;  RX CHAR ?
0BF3 C8                   RZ                          ;   NO
0BF4 3EFF                 MVI     A,0FFH              ;   YES  -  SET FLAG
0BF6 C9                   RET                         ;
                    ;
                    ;-------------------------------------------------------------
                    ;
                    ;     CONSOLE  # 1   INPUT
                    ;
                    ;-------------------------------------------------------------
                    ;
                    PT1IN:                            ; RETURN CHAR IN REG A
0BF7 CD3C0B               CALL    POLCI1              ;READY NOW?
0BFA B7                   ORA     A                   ;
                          JRNZ    PT1IN1              ; IF READY, SKIP POLL
0BFB+2007                 DB      020H,PT1IN1-$-1     ;---- FAKE JRNZ INSTRUCTION
0BFD 0E83                 MVI     C,POLL              ;
0BFF 1E06                 MVI     E,PLCI1             ; POLL CONSOLE #1 INPUT
0C01 CD100B               CALL    XDOS                ;
0C04 DB2C           PT1IN1: IN    DATA1               ; READ CHARACTER
0C06 E67F                 ANI     7F                  ; STRIP PARITY
```

```
0C08 C9                 RET                     ;
                ;
                ;
                ;-----------------------------------------------------------------
                ;
                ;       CONSOLE  # 1  OUTPUT
                ;
                ;-----------------------------------------------------------------
                ;
                PT1OUT:                         ; REG C = CHAR TO OUTPUT
0C09 CD1C0C             CALL    POLC01          ;ARE WE READY NOW?
0C0C B7                 ORA     A               ;
                        JRNZ    PT1OUT1         ;IF READY, SKIP POLL
0C0D+2009               DB      020H,PT1OUT1-$-1        ;---- FAKE JRNZ INS
0C0F C5                 PUSH    B               ;
0C10 0E83               MVI     C,POLL          ;
0C12 1E02               MVI     E,PLCO1         ;
0C14 CD100B             CALL    XDOS            ; POLL CONSOLE #1 OUTPUT
0C17 C1                 POP     B               ;
                PT1OUT1:
0C18 79                 MOV     A,C             ;
0C19 D32C               OUT     DATA1           ; TRANSMIT CHARACTER
0C1B C9                 RET                     ;
                ;
                ;
                ;-----------------------------------------------------------------
                ;
                ;       POLL CONSOLE  # 1  OUTPUT
                ;
                ;-----------------------------------------------------------------
                ;
                POLCO1:                         ; RETURN 0FFH IF READY
0C1C 3E10               MVI     A,10H           ;        000H IF NOT
0C1E D32D               OUT     STS1            ; RESET INT BIT
0C20 DB2D               IN      STS1            ; READ STATUS
0C22 E60C               ANI     0CH             ; MASK FOR DTR AND TXE
0C24 FE0C               CPI     0CH             ; MUST HAVE BOTH
0C26 3E00               MVI     A,0             ;
0C28 C0                 RNZ                     ; RETURN NOT READY
0C29 3D                 DCR     A               ;CHANGE "A" TO 0FFH
0C2A C9                 RET                     ; RETURN READY

                        PAGE

                ;-----------------------------------------------------------------
                ;
                ;       POLL CONSOLE  # 2  INPUT
                ;
                ;-----------------------------------------------------------------

                POLCI2:
                PT2ST:                          ; TEST CONSOLE STATUS
0C2B AF                 XRA     A               ;  RETURN  0FFH IF READY
```

```
0C2C D32F               OUT     STS2            ;        000H IF NOT
0C2E DB2F               IN      STS2            ;
0C30 E601               ANI     1               ; RX CHAR ?
0C32 C8                 RZ                      ;  NO
0C33 3EFF               MVI     A,0FFH          ;  YES  -  SET FLAG
0C35 C9                 RET                     ;
                ;
                ;-----------------------------------------------------------------
                ;
                ;       CONSOLE  # 2  INPUT
                ;
                ;-----------------------------------------------------------------
                ;
                PT2IN:                          ; RETURN CHAR IN REG A
0C36 CD2B0C             CALL    POLCI2          ;READY NOW?
0C39 B7                 ORA     A               ;
                        JRNZ    PT2IN1          ;IF READY, SKIP POLL
0C3A+2007               DB      020H,PT2IN1-$-1 ;---- FAKE JRNZ INSTRUCTION
0C3C 0E83               MVI     C,POLL          ;
0C3E 1E07               MVI     E,PLCI2         ; POLL CONSOLE #2 INPUT
0C40 CD100B             CALL    XDOS            ;
0C43 DB2E       PT2IN1: IN      DATA2           ; READ CHARACTER
0C45 E67F               ANI     7F              ; STRIP PARITY
0C47 C9                 RET                     ;
                ;
                ;
                ;-----------------------------------------------------------------
                ;
                ;       CONSOLE  # 2   OUTPUT
                ;
                ;-----------------------------------------------------------------
                ;
                PT2OUT:                         ; REG C = CHAR TO OUTPUT
0C48 CD5B0C             CALL    POLCO2          ;READY NOW?
0C4B B7                 ORA     A               ;
                        JRNZ    PT2OUT1         ;IF READY, SKIP POLL
0C4C+2009               DB      020H,PR2OUT1-$-1        ;---- FAKE JRNZ INS
0C4E C5                 PUSH    B               ;
0C4F 0E83               MVI     C,POLL          ;
0C51 1E03               MVI     E,PLCO2         ;
0C53 CD100B             CALL    XDOS            ; POLL CONSOLE #2 OUTPUT
0C56 C1                 POP     B               ;
                PT2OUT1:
0C57 79                 MOV     A,C             ;
0C58 D32E               OUT     DATA2           ; TRANSMIT CHARACTER
0C5A C9                 RET                     ;
                ;
                ;
                ;-----------------------------------------------------------------
                ;
                ;       POLL CONSOLE  # 2   OUTPUT
                ;
                ;-----------------------------------------------------------------
                ;
                POLCO2:                         ; RETURN 0FFH IF READY
```

```
0C5B 3E10               MVI     A,10H           ;         000H IF NOT
0C5D D32F               OUT     STS2            ; RESET INT BIT
0C5F DB2F               IN      STS2            ; READ STATUS
0C61 E60C               ANI     0CH             ; MASK FOR DTR AND TXE
0C63 FE0C               CPI     0CH             ; MUST HAVE BOTH
0C65 3E00               MVI     A,0             ;
0C67 C0                 RNZ                     ; RETURN NOT READY
0C68 3D                 DCR     A               ;CHANGE "A" TO 0FFH
0C69 C9                 RET                     ; RETURN READY

                        PAGE

                ;-----------------------------------------------------------
                ;
                ;       POLL CONSOLE  # 3  INPUT
                ;
                ;-----------------------------------------------------------

                POLCI3:
                PT3ST:                          ; TEST CONSOLE STATUS
0C6A AF                 XRA     A               ;  RETURN  0FFH IF READY
0C6B D32B               OUT     STS3            ;          000H IF NOT
0C6D DB2B               IN      STS3            ;
0C6F E601               ANI     1               ;  RX CHAR ?
0C71 C8                 RZ                      ;   NO
0C72 3EFF               MVI     A,0FFH          ;   YES  -  SET FLAG
0C74 C9                 RET
                ;
                ;-----------------------------------------------------------
                ;
                ;       CONSOLE  # 3  INPUT
                ;
                ;-----------------------------------------------------------
                ;
                PT3IN:                          ; RETURN CHAR IN REG A
0C75 CD6A0C             CALL    POLCI3          ;READY NOW?
0C78 B7                 ORA     A               ;
                        JRNZ    PT3IN1          ;IF READY, SKIP POLL
0C79+2007               DB      020H,PT3IN1-$-1 ;--- FAKE JRNZ INSTRUCTION
0C7B 0E83               MVI     C,POLL          ;
0C7D 1E08               MVI     E,PLCI3         ; POLL CONSOLE #3 INPUT
0C7F CD100B             CALL    XDOS            ;
0C82 DB2A       PT3IN1: IN      DATA3           ; READ CHARACTER
0C84 E67F               ANI     7FH             ; STRIP PARITY
0C86 C9                 RET                     ;
                ;
                ;
                ;-----------------------------------------------------------
                ;
                ;       CONSOLE  # 3  OUTPUT
                ;
                ;-----------------------------------------------------------
                ;
                PT3OUT:                              ; REG C = CHAR TO OUTPUT
0C87 CD9A0C             CALL    POLCO3               ;READY NOW?
0C8A B7                 ORA     A                    ;
                        JRNZ    PT3OUT1              ;IF READY, SKIP POLL
0C8B+2009               DB      020H,PT3OUT1-$-1          ;---- FAKE JRNZ INS
0C8D C5                 PUSH    B                    ;
0C8E 0E83               MVI     C,POLL               ;
0C90 1E04               MVI     E,PLCO3              ;
0C92 CD100B             CALL    XDOS                 ; POLL CONSOLE #3 OUTPUT
0C95 C1                 POP     G                    ;
                PT3OUT1:
0C96 79                 MOV     A,C                  ;
0C97 D32A               OUT     DATA3                ; TRANSMIT CHARACTER
0C99 C9                 RET                          ;
                ;
                ;
                ;-------------------------------------------------------------
                ;
                ;       POLL CONSOLE  # 3   OUTPUT
                ;
                ;-------------------------------------------------------------
                ;
                POLCO3:                              ; RETURN 0FFH IF READY
0C9A 3E10               MVI     A,10H                ;         000H IF NOT
0C9C D32B               OUT     STS2                 ; RESET INT BIT
0C9E DB2B               IN      STS2                 ; READ STATUS
0CA0 E60C               ANI     0CH                  ; MASK FOR DTR AND TXE
0CA2 FE0C               CPI     0CH                  ; MUST HAVE BOTH
0CA4 3E00               MVI     A,0                  ;
0CA6 C0                 RNZ                          ; RETURN NOT READY
0CA7 3D                 DCR     A                    ;CHANGE "A" TO 0FFH
0CA8 C9                 RET                          ; RETURN READY

                        PAGE



                ;-------------------------------------------------------------
                ;
                ;       LINE PRINTER  # 0  DRIVER
                ;
                ;-------------------------------------------------------------
                ;
                LIST:                                ;LIST OUTPUT #0
0CA9 CDBC0C             CALL    POLLPT
0CAC B7                 ORA     A                    ;IS PRINTER READY NOW?
                        JRNZ    LIST1                ;IF READY, SKIP POLL
0CAD+2009               DB      020H,LIST1-$-1       ;---- FAKE JRNZ INSTRUCTION

0CAF C5                 PUSH    B
0CB0 0E83               MVI     C,POLL               ; POLL PRINTER STATUS
0CB2 1E00               MVI     E,PLLPT              ;
0CB4 CD100B             CALL    XDCS                 ;
0CB7 C1                 POP     B                    ;
```

```
                LIST1:
0CB8 79                 MOV     A,C             ; CHARACTER TO PRINT
0CB9 D31E               OUT     LPTPRT0         ;
0CBB C9                 RET
                ;
                ;-----------------------------------------------------------
                ;
                ;       POLL PRINTER OUTPUT
                ;
                ;-----------------------------------------------------------
                ;
                POLLPT:                         ; RETURN 0FFH IF READY
0CBC 3E10               MVI     A,10H           ;        000H IF NOT
0CBE D31F               OUT     LPTSTS0         ; RESET INT BIT
0CC0 DB1F               IN      LPTSTS0         ; READ STATUS
0CC2 E60C               ANI     0CH             ; MASK FOR DTR AND TXE
0CC4 FE0C               CPI     0CH             ; MUST HAVE BOTH
0CC6 3E00               MVI     A,0             ;
0CC8 C0                 RNZ                     ; RETURN NOT READY
0CC9 3D                 DCR     A               ;CHANGE "A" TO 0FFH
0CCA C9                 RET                     ; RETURN READY
                ;
                        PAGE

                ;
                ;  MP/M 1.0  EXTENDED I/O SYSTEM
                ;
                ;
                POLLDEVICE:
                                        ; REG C = DEVICE # TO BE POLLED
                                        ; RETURN 0FFH IF READY,
                                        ;        000H IF NOT
0CCB 79                 MOV     A,C
0CCC FE09               CPI     NMBDEV
                        JRC     DEVOK
0CCE+3802               DB      038H,DEVOK-$-1  ;---- FAKE JRC INSTRUCTION
0CD0 3E09               MVI     A,NMBDEV; IF DEV # >= NMBDEV,
                                        ; SET TO NMBDEV
                DEVOK:
0CD2 CDA20B             CALL    TBLJMP  ; JUMP TO DEV POLL CODE

                DEVTBL:
0CD5 BC0C               DW      POLLPT  ; POLL PRINTER OUTPUT - THIS WILL POLL
                ;                       ;  SPECIFIED PARALLEL PORT FOR PRINTER
0CD7 DD0B               DW      POLLC0  ; POLL CONSOLE #0 OUTPUT
0CD9 1C0C               DW      POLCO1  ; POLL CONSOLE #1 OUTPUT
0CDB 5B0C               DW      POLCO2  ; POLL CONSOLE #2 OUTPUT
0CDD 9A0C               DW      POLCO3  ; POLL CONSOLE #3 OUTPUT
0CDF AD0B               DW      POLCI0  ; POLL CONSOLE #0 INPUT
0CE1 EC0B               DW      POLCI1  ; POLL CONSOLE #1 INPUT
0CE3 2B0C               DW      POLCI2  ; POLL CONSOLE #2 INPUT
0CE5 6A0C               DW      POLCI3  ; POLL CONSOLE #3 INPUT
0009 =          NMBDEV  EQU     ($-DEVTBL)/2
0CE7 1A0B               DW      RTNEMPTY; BAD DEVICE HANDLER
```

```
                                  PAGE

                         ; SELECT / PROTECT MEMORY

                         SELMEMORY:
                                                 ; REG BC = ADR OF MEM DESCRIPTOR
                                                 ; BC -> BASE   1 BYTE,
                                                 ;       SIZE   1 BYTE,
                                                 ;       ATTRIB 1 BYTE,
                                                 ;       BANK   1 BYTE.
                                                 ;
                                                 ;    BIOS TABLE MODIFIED
0CE9 FE20                         CPI     20H    ;
0CEB CAEB0C                       JZ      $
0CEE 210300                       LXI     H,3    ; POINT TO BANK
0CF1 09                           DAD     B      ;
0CF2 7E                           MOV     A,M    ;  GET IT
0CF3 32030D                       STA     BANKNO ; SAVE BANK NUMBER
0CF6 17                           RAL            ;
0CF7 17                           RAL            ;
0CF8 17                           RAL            ;
0CF9 E618                         ANI     018H   ; MASK FOR PIO
0CFB F602                         ORI     MEMSK  ;
0CFD 32040D                       STA     CURMEM ; STORE CURRENT BANK MASK
0D00 D309                         OUT     009H   ; SET PIO
0D02 C9                           RET

0D03 00                  BANKNO:  DB      0      ; LAST SELECTED MEMORY BANK NUMBER
0D04 00                  CURMEM:  DB      0      ; LAST SELECTED MEMORY BANK MASK


                         ; START CLOCK

                         STARTCLOCK:
                                                 ; WILL CAUSE FLAG #1 TO BE SET
                                                 ;  AT EACH SYSTEM TIME UNIT TICK
0D05 3EFF                         MVI     A,0FFH
0D07 32CE0D                       STA     TICKN
0D0A C9                           RET

                         ; STOP CLOCK

                         STOPCLOCK:
                                                 ; WILL STOP FLAG #1 SETTING AT
                                                 ;  SYSTEM TIME UNIT TICK
0D0B AF                           XRA     A
0D0C 32CE0D                       STA     TICKN
0D0F C9                           RET

                         ; EXIT REGION

                         EXITREGION:
```

```
                                                ; EI IF NOT PREEMPTED
0D10 3ACF0D                     LDA     PREEMP
0D13 B7                         ORA     A
0D14 C0                         RNZ
0D15 FB                         EI
0D16 C9                         RET

                        ; MAXIMUM CONSOLE NUMBER

                        MAXCONSOLE:
0D17 3E04                       MVI     A,NMBCNS
0D19 C9                         RET

                        ;  MP/M 1.0   INTERRUPT HANDLERS

008E =                  DSPTCH  EQU     142

                        INT1HND:
                                                ; INTERRUPT 1 HANDLER ENTRY POINT
                                                ;
                        T20MS:
0D1A 22C80D                     SHLD    SVDHL
0D1D 21220D                     LXI     H,TIMERINT
                                JR      INTINIT
0D20+185D                       DB      018H,INTINIT-$-1        ;---- FAKE JR INSTR
                        TIMERINT:
0D22 3ACE0D                     LDA     TICKN
0D25 B7                         ORA     A               ; TEST TICKN, INDICATES
                                                        ;  DELAYED PROCESS(ES)
                                JRZ     NOTICKN
0D26+2807                       DB      028H,NOTICKN-$-1        ;---- FAKE JRZ INST
0D28 0E85                       MVI     C,FLAGST
0D2A 1E01                       MVI     E,1
0D2C CD100B                     CALL    XDOS            ; SET FLAG #1 EACH TICK
                        NOTICKN:
0D2F 219D0D                     LXI     H,CNTX
0D32 35                         DCR     M               ; DEC TICK CNTR
                                JRNZ    NOT1SEC
0D33+2032                       DB      020H,NOT1SEC-$-1        ;---- FAKE JRNZ INST
0D35 3E7D                       MVI     A,125
0D37 2B                         DCX     H
0D38 96                         SUB     M
0D39 77                         MOV     M,A             ; *** TOGGLE COUNT 62 <-> 6
0D3A 23                         INX     H
0D3B 77                         MOV     M,A             ; *** ACTUAL #/SEC = 62.5
0D3C 0E85                       MVI     C,FLAGST
0D3E 1E02                       MVI     E,2
0D40 CD100B                     CALL    XDOS            ; SET FLAG #2 @ 1 SEC
0D43 2AD00D                     LHLD    FPYTIME         ;IS FLOPPY TIME CHECK IN EF
0D46 7C                         MOV     A,H             ;
0D47 B7                         ORA     A               ;
                                JRZ     NOT1SEC         ;IF NOT IN EFFECT, FINISH
0D48+281D                       DB      028H,NOT1SEC-$-1        ;---- FAKE JRZ INST
0D4A 2D                         DCR     L               ;SUBTRACT A SECOND
0D4B 22D00D                     SHLD    FPYTIME         ;SAVE FOR NEXT TIME
```

```
                              JRNZ    NOT1SEC          ;IF NOT TOO LONG, FINISH
0E4E+2017                     DB      020H,NOT1SEC-$-1       ;---- FAKE JRNZ INS
0D50 65                       MOV     H,L              ;ZERO OUT INDICATOR
0D51 22D00D                   SHLD    FPYTIME          ;PREVENT RE-ENTRY OF THIS ROUTINE
0D54 0E85                     MVI     C,FLAGST         ;
0D56 1E06                     MVI     E,FPYFLAG        ;
0D58 CD100B                   CALL    XDOS             ;CAUSE I/O FOR FLOPPY TO CONTINUE
0D5B 3E90                     MVI     A,10010000B
0D5D 32FC0A                   STA     STATUS           ;SHOW ERROR IN FLOPPY I/O
0D60 2AD20D                   LHLD    FPYTCNT
0D63 23                       INX     H                ;COUNT TIMES WD1791 GOES TO
0D64 22D20D                   SSLD    FPYTCNT          ;

                      NOT1SEC:

                      INTDONE:
0D67 AF                       XRA     A
0D68 32CF0D                   STA     PREEMP  ; CLEAR PREEMPTED FLAG
0D6B C1                       POP     B
0D6C D1                       POP     D
0D6D 2ACA0D                   LHLD    SVDSP
0D70 F9                       SPHL                     ; RESTORE STK PTR
0D71 F1                       POP     PSW
0D72 2ACC0D                   LHLD    SVDRET
0D75 E5                       PUSH    H
0D76 210D0B                   LXI     H,PDISP          ; MP/M DISPATCH
0D79 E5                       PUSH    H                ;  PUT ON STACK FOR RETURN
0D7A 2AC80D                   LHLD    SVDHL

                      ; THE FOLLOWING DISPATCH CALL WILL FORCE ROUND ROBIN
                      ;  SCHEDULING OF PROCESSES EXECUTING AT THE SAME PRIORITY
                      ;  EACH 1/32ND OF A SECOND.
                      ; NOTE: INTERRUPTS ARE NOT ENABLED UNTIL THE DISPATCHER
                      ;  RESUMES THE NEXT PROCESS.  THIS PREVENTS INTERRUPT
                      ;  OVER-RUN OF THE STACKS WHEN STUCK OR HIGH FREQUENCY
                      ;  INTERRUPTS ARE ENCOUNTERED.

                              RETI                     ; DISPATCH
0D7D+ED4D                     DB      0EDH,04DH        ;---- FAKE RETI INSTRUCTION

                      INTINIT:
D7F 22C60D                    SHLD    ADRINTHD
0D82 E1                       POP     H
0D83 22CC0D                   SHLD    SVDRET
0D86 F5                       PUSH    PSW
0D87 210000                   LXI     H,0
0D8A 39                       DAD     SP
0D8B 22CA0D                   SHLD    SVDSP            ; SAVE USERS STK PTR
0D8E 31C60D                   LXI     SP,LSTINTSTK     ; LCL STK FOR INTR HNDL
0D91 D5                       PUSH    D
0D92 C5                       PUSH    B

0D93 3EFF                     MVI     A,0FFH
0D95 32CF0D                   STA     PREEMP           ; SET PREEMPTED FLAG
0D98 2AC60D                   LHLD    ADRINTHD
```

```
0D9B E9                 PCHL            ;JUMP TO INTERRUPT HANDLER

                ;
                ; BIOS DATA SEGMENT
                ;
0D9C 3E         TOGCNT: DB      62      ; TOGGLE COUNTER 62 <-> 63
0D9D 3E         CNTX:   DB      62      ; TICK CNTR TO 1 SEC
                INTSTK:                 ; LOCAL INTRPT STK
0D9E C7C7C7C7C7         DW      0C7C7H,0C7C7H,0C7C7H,0C7C7H,0C7C7H
0DA8 C7C7C7C7C7         DW      0C7C7H,0C7C7H,0C7C7H,0C7C7H,0C7C7H
0DB2 C7C7C7C7C7         DW      0C7C7H,0C7C7H,0C7C7H,0C7C7H,0C7C7H
0DBC C7C7C7C7C7         DW      0C7C7H,0C7C7H,0C7C7H,0C7C7H,0C7C7H
                LSTINTSTK:
0DC6 0000       ADRINTHD: DW    0       ; INTERRUPT HANDLER ADDRESS
0DC8 0000       SVDHL:  DW      0       ; SAVED REGS HL DURING INT HNDL
0DCA 0000       SVDSP:  DW      0       ; SAVED SP DURING INT HNDL
0DCC 0000       SVDRET: DW      0       ; SAVED RETURN DURING INT HNDL
0DCE 00         TICKN:  DB      0       ; TICKING BOOLEAN,TRUE = DELAYED
0DCF 00         PREEMP: DB      0       ; PREEMPTED BOOLEAN

                        if      mpm20
                FPYTIME:
0DD0 0000               DW      0

                FPYTCNT:
0DD2 0000               DW      0
                        endif

                        PAGE

                ;-----------------------------------------------------------
                ;
                ;       THESE ARE THE DISK TYPE DEFINITION BLOCKS
                ;       EACH OF WHICH CORRESPONDS TO A PARTICULAR MODE.
                ;
                ;-----------------------------------------------------------

0DD4 =          DPB0:   EQU     $                   ;VERSION 2.0, SINGLE DENSITY
0DD4 1A00               DW      26                  ;SECTORS PER TRACK
0DD6 03                 DB      3                   ;BLOCK SHIFT
0DD7 07                 DB      7                   ;BLOCK SHIFT MASK
0DD8 00                 DB      0                   ;EXTENT MASK
0DD9 F200               DW      242                 ;DISK SIZE MINUS 1
0DDB 3F00               DW      63                  ;DIRECTORY MAX
0DDD C0                 DB      192                 ;ALLOC0
0DDE 00                 DB      0                   ;ALLOC1
0DDF 1000               DW      16                  ;CHECK AREA SIZE
0DE1 0200               DW      2                   ;OFFSET TO START TRACK

0DE3 =          DPB1:   EQU     $                   ;VERSION 2.0, DOUBLE DENSITY
0DE3 3400               DW      52                  ;SECTORS PER TRACK
0DE5 04                 DB      4                   ;BLOCK SHIFT
0DE6 0F                 DB      15                  ;BLOCK SHIFT MASK
```

```
ODE7 01                 DB      1               ;EXTENT MASK
0D38 F200               DW      242             ;DISK SIZE MINUS 1
ODEA 7F00               DW      127             ;DIRECTORY MAX
ODEC C0                 DB      192             ;ALLOC0
ODED 00                 DB      0               ;ALLOC1
ODEE 2000               DW      32              ;CHECK AREA SIZE
ODF0 0200               DW      2               ;OFFSET TO START TRACK

ODF2 =          DPB2:   EQU     $               ;DOUBLE DENSITY
ODF2 3000               DW      48              ;SECTORS PER TRACK
ODF4 04                 DB      4               ;BLOCK SHIFT
ODF5 0F                 DB      15              ;BLOCK SHIFT MASK
ODF6 00                 DB      0               ;EXTENT MASK (1.4 COMPATIBLE
ODF7 E000               DW      224             ;DISK SIZE MINUS 1
ODF9 5F00               DW      95              ;DIRECTORY MAX
ODFB C0                 DB      192             ;ALLOC0
ODFC 00                 DB      0               ;ALLOC1
ODFD 1800               DW      24              ;CHECK AREA SIZE
ODFF 0200               DW      2               ;OFFSET TO START TRACK

                        IF      HARDSK

                        if      mpm20
                DPB3:   DISKDEF 3,0,127,,16384,512,512,0,1,,0
0E01+=          DPB3    EQU     $               ;DISK PARM BLOCK
0E01+8000               DW      128             ;SEC PER TRACK
0E03+07                 DB      7               ;BLOCK SHIFT
0E04+7F                 DB      127             ;BLOCK MASK
0E05+07                 DB      7               ;EXTNT MASK
0E06+FF01               DW      511             ;DISK SIZE-1
0E08+FF01               DW      511             ;DIRECTORY MAX
0E0A+80                 DB      128             ;ALLOC0
0E0B+00                 DB      0               ;ALLOC1
0E0C+0080               DW      8000H+CKSZ              ;PERMANENT DISK WITH
0E0E+0100               DW      1               ;OFFSET
0000+=          XLT3    EQU     0               ;NO XLATE TABLE

                DPB4:   DISKDEF 4,0,127,,16384,512,512,0,513,,0
0E10+=          DPB4    EQU     $               ;DISK PARM BLOCK
0E10+8000               DW      128             ;SEC PER TRACK
0E12+07                 DB      7               ;BLOCK SHIFT
0E13+7F                 DB      127             ;BLOCK MASK
0E14+07                 DB      7               ;EXTNT MASK
0E15+FF01               DW      511             ;DISK SIZE-1
0E17+FF01               DW      511             ;DIRECTORY MAX
0E19+80                 DB      128             ;ALLOC0
0E1A+00                 DB      0               ;ALLOC1
0E1B+0080               DW      8000H+CKSZ              ;PERMANENT DISK WITH
0E1D+0102               DW      1               ;OFFSET
0000+=          XLT4    EQU     0               ;NO XLATE TABLE

                DPB5:   DISKDEF 5,0,127,,16384,512,512,0,1025,,0
0E1F+=          DPB5    EQU     $               ;DISK PARM BLOCK
0E1F+8000               DW      128             ;SEC PER TRACK
0E21+07                 DB      7               ;BLOCK SHIFT
```

```
0E22+7F                 DB      127                     ;BLOCK MASK
0E23+07                 DB      7                       ;EXTNT MASK
0E24+FF01               DW      511                     ;DISK SIZE-1
0E26+FF01               DW      511                     ;DIRECTORY MAX
0E28+80                 DB      128                     ;ALLOC0
0E29+00                 DB      0                       ;ALLOC1
0E2A+0080               DW      8000H+CKSZ                      ;PERMANENT DISK WITH
0E2C+0104               DW      1                       ;OFFSET
0000+=          XLT5    EQU     0                       ;NO XLATE TABLE

                DPB6:   DISKDEF 6,0,127,,16384,288,512,0,513,,0
0E2E+=          DPB6    EQU     $                       ;DISK PARM BLOCK
0E2E+8000               DW      128                     ;SEC PER TRACK
0E30+07                 DB      7                       ;BLOCK SHIFT
0E31+7F                 DB      127                     ;BLOCK MASK
0E32+07                 DB      7                       ;EXTNT MASK
0E33+1F01               DW      287                     ;DISK SIZE-1
0E35+FF01               DW      511                     ;DIRECTORY MAX
0E37+80                 DB      128                     ;ALLOC0
0E38+00                 DB      0                       ;ALLOC1
0E39+0080               DW      8000H+CKSZ                      ;PERMANENT DISK WITH
0E3B+0102               DW      513                     ;OFFSET
                XLT6    EQU     0                       ;NO XLATE TABLE
                        else
                DPB3:   DISKDEF 3,0,127,,16384,512,512,0,1

                DPB4:   DISKDEF 4,0,127,,16384,512,512,0,513

                DPB5:   DISKDEF 5,0,127,,16384,512,512,0,1025

                DPB6:   DISKDEF 6,0,127,,16384,288,512,0,513
                        endif

                        ENDIF

                        if      mdisk
                DPB7:   EQU     $                       ;VIRTUAL DISK
                        DW      24                      ;SECTORS PER TRACK
                        DB      3                       ;BLOCK SHIFT
                        DB      7                       ;BLOCK SHIFT MASK
                        DB      0                       ;EXTENT MASK
                        DW      142                     ;DISK SIZE MINUS 1
                        DW      63                      ;DIRECTORY MAX
                        DB      0C0H                    ;ALLOC0
                        DB      0                       ;ALLOC1
                        DW      0                       ;CHECK AREA SIZE
                        DW      0                       ;OFFSET TO START TRACK
                        endif

                        page

                ;
                ;       MOVE SUBROUTINE
                ;
```

```
                       if      hardsk
               RWMOVE:
0E3D D5                push    d
0E3E E5                push    h
0E3F CD070B            call    swtuser          ;switch in user bank
0E42 E1                pop     h
0E43 D1                pop     d
0E44 018000            lxi     b,128
                       LDIR                     ;MOVE DATA TO/FROM BUFFER
0E47+EDB0              DB      0EDH,0B0H        ;---- FAKE LDIR INSTRUCTION
0E49 CD0A0B            call    swtsys           ;switch system back in
               ;
               ;       DATA HAS BEEN MOVED TO/FROM HOST BUFFER
               ;

0E4C 3AF90A            LDA     WRTYPE           ;WRITE TYPE ??

                       if      mpm20
0E4F E601              ani     WRDIR            ;TO DIRECTORY ??
                       JRZ     RWEND            ;NO, JUST END UP HERE
0E51+280D              DB      028H,RWEND-$-1   ;---- FAKE JRZ INSTRUCTION
                       else
                       CPI     WRDIR            ;TO DIRECTORY ??
                       JRNZ    RWEND            ;NO, JUST END UP HERE
                       endif

               ;
               ;       CLEAR HOST BUFFER FOR DIRECTORY WRITE
               ;

0E53 3AF60A            LDA     ERFLAG           ;CHECK PRIOR TO DIR ACTIVITY
0E56 B7                ORA     A                ;ERRORS ??
                       JRNZ    RWEND            ;SKIP IF SO....
0E57+2007              DB      020H,RWEND-$-1   ;---- FAKE JRNZ INSTRUCTION
0E59 AF                XRA     A                ;ZERO TO ACCUMULATOR
0E5A 32F00A            STA     HSTWRT           ;BUFFER WRITTEN
0E5D CD6D04            CALL    WRITEHST         ;

               RWEND:
0E60 3AF60A            LDA     ERFLAG           ;
0E63 B7                ORA     A                ;IF ERRORS, RESET SO NO MATCH
0E64 C8                RZ                       ;NONE, JUST RETURN
0E65 21EA0A            LXI     H,HSTDSK         ;
0E68 36FF              MVI     M,0FFH           ;CANT POSSIBLY MATCH, MUST ERROR
                       ENDIF
0E6A C9                RET                      ;

               MVDTB:
0E6B 2AAF00            LHLD    DMAADR           ; MOVE DATA TO FLOPPY BUFFER
0E6E E5                push    h
0E6F CD070B            call    swtuser ;switch in user bank,
0E72 E1                pop     h        ;   cannot access non-common BNKXIOS
0E73 111D13            LXI     D,FPYBUF         ;
0E76 018000            LXI     B,128            ; 128 BYTES
                       LDIR                     ;
```

```
0E79+EDB0              DB      0EDH              ;---- FAKE LDIR INSTRUCTION
0E7B C30A0B            jmp     swtsys            ;switch system back in
               ;       RET                       ;

0E7E F5        MVDFB:  PUSH    PSW               ; MOVE DATA FROM FLOPPY BUFFER
0E7F 3AFA0A            LDA     CMD               ;
0E82 E620              ANI     20H               ; CHECK FOR READ
                       JRNZ    MVDFX             ;  NO - BYPASS MOVE
0E84+2013              DB      020H,MVDFX-$-1    ;---- FAKE JRNZ INSTRUCTION
0E86 2AAF00            LHLD    DMAADR            ;
0E89 E5                push    h
0E8A CD070B            call    swtuser ;switch in user bank,
0E8D D1                pop     d        ;    cannot access non-common BNKXIOS
0E8E 211D13            LXI     H,FPYBUF          ;
0E91 018000            LXI     B,128             ; 128 BYTES
                       LDIR                      ;
0E94+EDB0              DB      0EDH,0B0H         ;---- FAKE LDIR INSTRUCTION
0E96 CD0A0B            call    swtsys            ;switch system back in
0E99 F1        MVDFX:  POP     PSW               ;
0E9A C9                RET                       ;

                       IF      HARDSK

0E9B                   DS      1                          ;MUST PRECEDE HSTBUF
0E9C           HSTBUF: DS      1024                       ;HOST BUFFER AREA
129C                   DS      1                          ;MUST FOLLOW HSTBUF
                       ENDIF

                       PAGE

               ;-----------------------------------------------------------------
               ;
               ;       INITIALIZE MP/M: REAL TIME CLOCK & DISKS
               ;
               ;-----------------------------------------------------------------

                       if      mpm20
129D =         dirbuf  equ     $
131D =         fpybuf  equ     dirbuf+128
                       endif

               SYSTEMINIT:
                       ; C  = BREAKPOINT RESTART NUMBER
                       ; DE = BREAKPOINT RESTART HANDLER ADDRESS
                       ; HL = DIRECT XIOS INTERCEPT JUMP TABLE ADDRESS

129D 225E13            SHLD    SVDJT
12A0 69                MOV     L,C
12A1 2600              MVI     H,0
12A3 29                DAD     H
12A4 29                DAD     H
12A5 29                DAD     H       ;HL = RESTART JUMP ADDRESS
12A6 226013            SHLD    SVDBPA

                       if      not mdisk
```

```
12A9 2A130B              lhd      sysdat
12AC 2E0F                mvi      l,15       ;hl =  .nmbmemsegs
12AE 46                  mov      b,m        ;b = nmbmemsegs
                test$bank$setup$loop:
12AF 23                  inx      h
12B0 23                  inx      h
12B1 23                  inx      h
12B2 23                  inx      h          ;hl = .memseg(i).bank
12B3 7E                  mov      a,m
12B4 B7                  ora      a
12B5 C2BF12              jnz      bank$setup
12B8 05                  dcr      b
12B9 C2AF12              jnz      test$bank$setup$loop
12BC C3CE12              jmp      after$bank$setup
                bank$setup:
12BF 3E1A                MVI      A,01AH              ; SELECT BANK 3
12C1 CD4813              CALL     STMVTR              ; SET UP VECTORS
12C4 3E12                MVI      A,012H              ; SELECT BANK 2
12C6 CD4813              CALL     STMVTR              ; SET UP VECTORS
12C9 3E0A                MVI      A,00AH              ; SELECT BANK 1
12CB CD4813              CALL     STMVTR              ; SET UP VECTORS
                after$bank$setup:
                         else
                         mvi      a,lah               ; bank 3 select for directory
                         out      09h
                         lxi      h,0bffeh
                         mvi      a,0e5h
                         cmp      m
                         inx      h
                         jrnz     fill
                         cmp      m
                         urz      dontfill
                fill:
                         mov      m,a                 ;set directory initialized
                         dcx      h
                         mov      m,a

                         lxi      b,07ffh             ;first 2 k of bank one gets
                         lxi      h,0
                         lxi      d,1
                         mvi      a,0ah               ; select bank 1
                         out      09h
                         mvi      m,0e5h
                         ldir
                dontfill:
                         endif
12CE 3E02                MVI      A,002H              ; SELECT BANK 0
12D0 CD4813              CALL     STMVTR              ; SET UP VECTORS

12D3 213717              lxi      h,ldrbiosbase+density$mask$offset
                ;;;;;     LXI      H,1737H             ; MOVE PARAMETERS CHANGED BY
12D6 117C00              LXI      D,SEL0              ;       THE SET UP PROGRAM
12D9 010400              LXI      B,4                 ;  4 SELECT MASKS
                         LDIR                         ;
12DC+EDB0                DAB      0EDH,0B0H           ;---- FAKE LDIR INSTRUCTION
```

```
12DE 118800             LXI     D,MODE              ;
12E1 0J10400            LXI     B,4                 ;  4 MODE BYTES
                        LDIR                        ;
12E4+EDB0               DB      0EDH,0B0H           ;—— FAKE LDIR INSTRUCTION
12E6 2ABB17             lhld    ldrbiosbase+misc$params$offset
                        LHLD    17BBH               ; GET MISC. PARAMETERS
12E9 22B600             SHLD    MPARMS              ;
12EC 3AB600             LDA     MPARMS              ; NOW TEST FOR CENTRONICS PRNTR
12EF E602               ANI     2                   ;
                        JRZ     PRTOK               ;  NO  -  LEAVE SERIAL
12F1+2814               DB      028H,PRTOK-$-1      ;—— FAKE JRZ INSTRUCTION
12F3 212C0B             LXI     H,CLIST             ;
12F6 221000             SHLD    WBOTE+13            ; CHANGE PRINTER ROUTINE
12F9 211F0B             LXI     H,CNSTAT            ;   AND STATUS CHECK
12FC 22D50C             SHLD    DEVTBL              ;
12FF 3E03               MVI     A,003H              ;INITIALIZE PARALLEL PORT
120A D313               OUT     013H
1303 3E0F               MVI     A,00FH              ;
1305 D313               OUT     013H

              PRTOK:
1307 010300             LXI     5,003H              ;SET THE MODE FOR DRIVES INIT
              MODESET:
130A CD2F02             CALL    SELSDP              ;SELECT DRIVE FOR MODESET
130D 218800             LXI     H,MODE              ;
1310 09                 DAD     B                   ;POINT TO CORRECT MODE BYTE
1311 C5                 PUSH    B                   ;SAVE COUNT OF DRIVES
1312 41                 MOV     B,C                 ; B = DRIVE #
1313 4E                 MOV     C,M                 ;
1314 CDF807             CALL    XETMOD              ;SET MOVE
1317 C1                 POP     B                   ;
1318 0D                 DCR     C                   ;END OF LIST YET ??
1319 F20A13             JP      MODESET             ;SET MODE FOR ALL DRIVES
131C CDD007             CALL    SDCONF              ;SET DISK CONFIGURATION

131F 018000             LXI     B,80H
1322 CD5502             CALL    SETDMA              ;SET DMA ADDRESS

1325 E5                 push    h

                        if      mpm20
1326 2A130B             lhld    sysdat
1329 2E07               mvi     l,7
132B 7E                 mov     a,m
                        else
                        lxi     h,INTERUPT
                        mov     a,h
                        endif

132C E1                 pop     h
132D ED47               DB      0EDH,047H            ;—— FAKE STAI INSTRUCTION

132F 3E60               MVI     A,60H                ; SET VECTOR FOR CTC
1331 D330               OUT     30H                  ; CTC CHANNEL 0
1333 3EA7               MVI     A,0A7H               ; RESET  /  LOAD TIME CONSTANT
```

```
1335 D333               OUT     33H             ; CHANNEL 3
1337 3EFA               MVI     A,250           ;  TIME CONSTANT
1339 D333               OUT     033H            ;

                        IF      HARDSK
133B AF                 XRA     A               ;ZERO ACCUMULATOR
133C 32EF0A             STA     HSTACT          ;SET HOST BUFFER INACTIVE
133F 32F10A             STA     UNACNT          ;SET UNALLOCATED COUNT TO ZERO

1342 219B0E             LXI     H,HSTBUF-1      ;SETUP WRITE CONTROL BYTE
1345 360D               MVI     M,0D            ;
                        ENDIF

1347 C9                 RET                     ;

                STMVTR:
1348 D309               OUT     MEMPORT
134A 3EC3               MVI     A,0C3H          ; SET VECTORS FOR BDOS
134C 320000             STA     0               ;  JMP INSTRUCTION
134F 2A5E13             LHLD    SVDJT           ;
1352 220100             SHLD    1
1355 2A6013             LHLD    SVDBPA
1358 77                 MOV     M,A
1359 23                 INX     H
135A 73                 MOV     M,E
135B 23                 INX     H
135C 72                 MOV     M,D
135D C9                 RET                     ;

135E            SVDJT:  DS      2       ; SAVED DIRECT JUMP TABLE ADDRESS
1360            SVDBPA: DS      2       ; SAVED BREAK POINT ADDRESS

                        if      mpm20
1362 =          xiosend equ     $
139D =          fdbuf   equ     (dirbuf-base)+256
139D                    org fdbuf+((xiosend-base)/fdbuf)*((xiosend-base)-fd
139D 00                 db      0
                        endif


139E                    END


070F ADDERRORS  0DC6 ADRINTHD   13CE AFTERBANKS 03DA ALLOC
081E ALV0       0853 ALV1       089E ALV2       08DE ALV3
091E AVL4       095E ALV5       099E ALV6       09DE ALV7
0A1E ALV8       0A5E ALV9       0A9E ALVA       0AC2 ALVB
0708 BADIO      0D03 BANKNO     12BF BANKSETUP  0000 BASE
081E BEGDAT     4000 BLKSIZ     067B CHECKIT    06AE CHECKSTAT
0220 CHKHRD     06BC CHKS0      06C7 CHKS1      06CD CHKS2
06E6 CHKS3      06F7 CHKS4      03A1 CHKUNA     0B2C CLIST
0B3B CLIST1     0AFA CMD        0B1F CNSTAT     0D9D CNTX
0B15 COLDSTART  0B04 COMMONBASE 0B84 CONIN      0B8F CONOUT
```

```
0B79 CONST       0080 CPMSPT      083E CSV0        087E CSV1
08BE CSV2        08FE CSV3        095E CSV4        099E CSV5
09DE CSV6        0A1E CSV7        0A5E CSV8        0A9E CSV9
0AC2 CSVA        0AE6 CSVB        0D04 CURMEM      001C DATA0
002C DATA1       002E DATA2       002A DATA3       00B4 DBLKAD
052E DBLLOW      0530 DBLSAVE     0519 DBLUPDATE   0782 DEL1
0784 DEL2        0782 DELAY       0037 DENSITYMAS  0CD2 DEVOK
0CD5 DEVTBL      129D DIRBUF      00AC DISKNO      FFFF DMA
00AF DMAADR      00BE DMALEN      00BA DMAS1       00C6 DMAS2F
00C0 DMAS2H      00CA DMAS3       00CE DMAS3F      00BC DMASA
0DD4 DPB0        0DE3 DPB1        0DF2 DPB2        0E01 DPB3
0E10 DPB4        0E1F DPB5        0E2E DPB6        00D1 DPBASE
00D1 DPE0        00E1 DPE1        00F1 DPE2        0101 DPE3
0111 DPE4        0121 DPE5        0131 DPE6        0141 DPE7
0151 DPE8        0161 DPE9        0171 DPEA        0181 DPEB
00B2 DPEPTR      0790 DSCNO       0552 DSKSEL      008E DSPTCH
0246 DTBLT       0AF6 ERFLAG      0D10 EXITREGION  0000 FALSE
139D FDBUF       0B50 FDINTH      041C FILLHST     0763 FINTFIX
0085 FLAGST      0084 FLAGWT      0B47 FLOPPYINT   0687 FLOPPYIO
065C FLOPPYSEEK  0661 FPS1        131D FPYBUF      0006 FPYFLAG
0DD2 FPYTCNT     0DD0 FPYTIME     073A FPYWAIT     0760 FSECSET
06AA FWT1        0B53 HARDINT     FFFF HARDSK      0005 HDFLAG
0B67 HDINTH      0B71 HDSTFLG     00AE HEADNO      031A HOME
0337 HOME1       0340 HOME1A      0357 HOME2       0343 HOMEHARD
02F9 HOMEIT      0321 HOMESOFT    0B03 HOMETOGGLE  048F HRW0
0499 HRW1        04C4 HRW2        04CD HRW3        04D9 HRW4
04E1 HRW5        04E6 HRW6        0504 HRW7        0AEF HSTACT
0008 HSTBLK      0E9C HSTBUF      0AEA HSTDSK      0AED HSTSEC
0400 HSTSIZ      0010 HSTSPT      0AEB HSTTRK      0AF0 HSTWRT
00B8 HTK1        00B9 HTK2        07F8 INITEND     0D1A INT1HND
0D67 INTDONE     005E INTERUPT    0721 INTFIX      0D7F INTINIT
0D9E INTSTK      0051 LAST        062D LDH1        1700 LDRBIOSBAS
0CA9 LIST        0CB8 LIST1       061E LOADHEAD    001E LPTPRT0
0028 LPTPRT1     001F LPTSTS0     0029 LPTSTS1     0DC6 LSTINTSTK
0AFB MASK        0439 MATCH       0D17 MAXCONSOLE  000C MAXDSK
0000 MDISK       0009 MEMPORT     0002 MEMSK       00BB MISCPARAMS
0088 MODE        130A MODESET     0191 MODL0       019D MODL1
01A9 MODL2       00B6 MPARM       FFFF MPM20       0E7E MVDFB
0E99 MVDFX       0E6B MVDTB       0AE6 NEWDSK      0AEE NEWHST
0AE9 NEWSEC      0AE7 NEWTRK      0461 NEWTRKCMP   0004 NMBCNS
0009 NMBDEV      075F NOFPYRST    0415 NOMATCH     03D4 NOOVF
0D67 NOT1SEC     0D2F NOTICKN     0B1C NULLINT     00A0 PCNT
0B0D PDISP       0005 PLCI0       0006 PLCI1       0007 PLCI2
0008 PLCI3       0001 PLCO0       0002 PLCO1       0003 PLCO2
0004 PLCO3       0000 PLLPT       05D4 PNTFN       05D1 PNTH2
05B3 POINT       0BAD POLCI0      0BEC POLCI1      0C2B POLCI2
0C6A POLCI3      0BDD POLCO0      0C1C POLCO1      0C5B POLCO2
0C9A POLCO3      0083 POLL        0CCB POLLDEVICE  0CBC POLLPT
0DCF PREEMP      0B01 PRETRIES    1307 PRTOK       0BB8 PT0IN
0BC5 PT0IN1      0BCA PT0OUT      0BD9 PT0OUT1     0BAD PT0ST
0BF7 PT1IN       0C04 PT1IN1      0C09 PT1OUT      0C18 PT1OUT1
0BEC PT1ST       0C36 PT2IN       0C43 PT2IN1      0C48 PT2OUT
0C57 PT2OUT1     0C2B PT2ST       0C75 PT3IN       0C82 PT3IN1
0C87 PT3OUT      0C96 PT3OUT1     0C6A PT3ST       0B9A PTBLJMP
028B READ        036B READHARD    047F READHST     0AF8 READOP
```

```
05E4 READSOFT   0308 REALDISK   FFFF RELOC      0643 REMOVELD
02EE RETMOD     0AF7 RSFLAG     0B1A RTNEMPTY   0E60 RWEND
0E3D RWMOVE     03E2 RWOPER     0AFD SAVEI      07D0 SDCONF
07DE SDDBL      07F1 SDL1       07F3 SDOK       0007 SECMSK
0003 SECSHF     00B1 SECTNO     05D6 SECTRAN    007C SEL0
0203 SELDSK     0242 SELERR     059B SELHARD    0CE9 SELMEMORY
022F SELSDP     0556 SELSOFT    0278 SETDEN     0255 SETDMA
0211 SETDSK     0544 SETDVD     0540 SETH14     0532 SETHED
02A1 SETMOD     0273 SETSEC     02C7 SETSEL     026D SETTRK
0547 SHD1       05A1 SLHI       055D SLS1       056B SLS2
0575 SLS3       0584 SLS4       0595 SLSERR     02E3 SMERR
0601 SRW1       0615 SRW2       0D05 STARTCLOCK 0AFC STATUS
1348 STMVTR     0D0B STOPCLOCK  05DB STRN1      05E2 STRN2
001D STS0       002D STS1       002F STS2       002B STS3
1360 SVDBPA     0DC8 SVDHL      135E SVDJT      0DCC SVDRET
0DCA SVDSP      0B0A SWTSYS     0B07 SWTUSER    0B13 SYSDAT
129D SYSTEMINIT 0D1A T20MS      0BA2 TBLJMP     0094 TCNT
12AF TESTBANKSE 0DCE TICKN      0D22 TIMERINT   0D9C TOGCNT
00AD TRAKNO     0B02 TRETRIES   0070 TRK0       064A TRKTST
FFFF TRUE       0AF1 UNACNT     0AF2 UNADSK     0AF5 UNASEC
0AF3 UNATRK     0717 WAIT0      0B15 WARMSTART  0003 WBOTE
0000 WRALL      0001 WRDIR      0296 WRITE      037E WRITEHARD
046D WRITEHST   05F2 WRITESOFT  0AF9 WRTYPE     0002 WRUAL
0B10 XDOS       07F8 XETMOD     0801 XETSEL     1362 XIOSEND
01B5 XLT0       01CF XLT1       01CE XLT2       0000 XLT3
0000 XLT4       0000 XLT5       0000 XLT6
```

INDEX