MP/M II™
Operating System
SYSTEM IMPLEMENTOR'S GUIDE

Copyright © 1981
Digital Research
P.O. Box 579
801 Lighthouse Avenue
Pacific Grove, CA 93950
(408) 649-3896
TWX 910 350 5001

All Rights Reserved
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), preferably on the target MP/M II machine.
# Table of Contents

## 1 MP/M II Alteration Procedure

1.1 Preparation for MP/M II Alteration .......................... 1
1.2 Customizing the MPMLDR ........................................ 2
1.3 Customizing the XIOS .......................................... 3
1.4 Debugging an XIOS ............................................. 4
1.5 Directly Booting MP/M II ......................................... 6
   1.5.1 Preparing an MP/M II Boot Using SYSGEN ............... 7
   1.5.2 Custom Generation of an MP/M II Boot .................. 8
1.6 Loading MPM.SYS Without the MPMLDR ......................... 11
1.7 Digital Research Copyright and Trademark ................... 11
1.8 Disk Organization .............................................. 12

## 2 The MP/M II BIOS

2.1 MP/M II BIOS Overview ......................................... 15
2.2 BIOS Device Characteristics and Entry Points ............... 16
2.3 BIOS Disk Definition Tables .................................... 24
   2.3.1 Disk Parameter Table Format ............................ 24
   2.3.2 The DISKDEF Macro Library ............................... 29
2.4 External Procedure Access ..................................... 33
2.5 Blocking and Deblocking Algorithms ........................... 36
2.6 Common Memory Portion of the BNKXIOS ....................... 37

## 3 The MP/M II XIOS

3.1 MP/M II XIOS Overview ......................................... 39
3.2 MP/M XIOS Entry Points ........................................ 39
3.3 Interrupt Service Routines ..................................... 43
3.4 Time Base Management .......................................... 44
# Table of Contents

(continued)

## 4 MP/M II System File Components

4.1 System Data .................................................. 47
4.2 Customized XIOS ............................................. 48
4.3 BDOS ............................................................. 48
   4.3.1 RESBDOS .................................................. 48
   4.3.2 BNKBDOs .................................................. 49
4.4 XDOS ............................................................. 49
4.5 Resident System Processes ................................. 49
4.6 Banked Resident System Processes ....................... 49

## 5 System Generation

5.1 GENSYS Operation ............................................. 51
5.2 System Generation Parameters .............................. 52
   5.2.1 Defaults .................................................. 52
   5.2.2 Top Page of Operating System ......................... 53
   5.2.3 Number of System Consoles ............................. 53
   5.2.4 Number of Printers ..................................... 53
   5.2.5 Breakpoint RST ........................................... 53
   5.2.6 System Call User Stacks ............................... 53
   5.2.7 Z80 CPU ................................................... 54
   5.2.8 Number of Ticks / Second ............................. 54
   5.2.9 System Disk .............................................. 54
   5.2.10 Temporary File Drive .................................. 54
   5.2.11 Maximum Locked Records / Process .................. 54
   5.2.12 Total Locked Records / System ...................... 54
   5.2.13 Maximum Open Files / Process ....................... 55
   5.2.14 Total Open Files / System ........................... 55
   5.2.15 Bank Switched Memory ................................ 55
   5.2.16 Number of User Memory Segments ................... 55
   5.2.17 Common Memory Base Page ............................ 55
   5.2.18 Dsyfile Logging at Console ......................... 55
   5.2.19 Accept System Data Page Entries .................... 56
   5.2.20 Select Resident System Processes ................. 56
   5.2.21 Memory Segment Table ................................ 56
   5.2.22 Accept Memory Segment Table ....................... 57

5.3 GENSYS Execution ............................................. 57
Table of Contents
(continued)

6 MP/M Loader

6.1 MP/M Loader Operation and Display ............... 59
6.2 MPMLDR Execution ............................... 60

Appendixes

A Disk Definition Macro ............................... 61
B Sector Deblocking Algorithms ...................... 67
C Sample MP/M II Loader BIOS ....................... 75
D Simple XIOS Source Listing ......................... 77
E Sample MP/M II Banked XIOS ......................... 97
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 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 accidentally 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

All Information Presented Here is Proprietary to Digital Research
4. Exit the debugger, returning to the CCP by executing a jump to location zero.

-GO

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 GEN SYS running under CP/M, generate an 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 GEN SYS, 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 GEN SYS execution.

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

   A>DDT MPMLDR.COM
   DDT VERS 2.0
   NEXT PC
   IA00 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 GEN SYS 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.

\[-X\]

\[\ldots\ldots\ldots\ldots\ldots\ P = 09F2 \ldots\]

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

1.4 Debugging an XIOS

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 can 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.

All Information Presented Here is Proprietary to Digital Research
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 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 and then read (R) in the file with an offset of 880H bytes.

   -MPMLDR.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

All Information Presented Here is Proprietary to Digital Research
5. Return to the CP/M console command processor (CCP) by jumping to location zero.

-GO

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.

```assembly
; 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 POINT TO SCRATCH AREA
LXI H, 3380H ;SET BASE LOAD ADDRESS
MVI B, 0 ;START WITH TRACK 0
RDTRK: MVI C, 1 ;READ NEXT SECTOR
READSEC: 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
```

All Information Presented Here is Proprietary to Digital Research
; 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 ;CHECK FOR END OF TRACK
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 ;TEST FOR LAST TRACK
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 write 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>
<thead>
<tr>
<th>Record</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>First 128 bytes of the SYSDAT page</td>
</tr>
<tr>
<td>2</td>
<td>Second 128 bytes of the SYSDAT page</td>
</tr>
<tr>
<td>3-n</td>
<td>MP/M operating system in reverse order, top down.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Track#</th>
<th>Sector#</th>
<th>Page#</th>
<th>Memory Address</th>
<th>MP/M Module name</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01</td>
<td></td>
<td>(boot address)</td>
<td>Cold Start Loader</td>
</tr>
<tr>
<td>00</td>
<td>02</td>
<td>00</td>
<td>0100H</td>
<td>MPMLDR</td>
</tr>
<tr>
<td></td>
<td>03</td>
<td></td>
<td>0180H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>04</td>
<td>01</td>
<td>0200H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>05</td>
<td>02</td>
<td>0280H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>06</td>
<td>03</td>
<td>0300H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>07</td>
<td></td>
<td>0380H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>08</td>
<td>04</td>
<td>0400H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>09</td>
<td></td>
<td>0480H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>05</td>
<td>0500H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td></td>
<td>0580H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>06</td>
<td>0600H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13</td>
<td></td>
<td>0680H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>07</td>
<td>0700H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td></td>
<td>0780H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>08</td>
<td>0800H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>17</td>
<td></td>
<td>0880H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>09</td>
<td>0900H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>19</td>
<td></td>
<td>0980H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>10</td>
<td>0A00H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>21</td>
<td></td>
<td>0A80H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>11</td>
<td>0B80H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>23</td>
<td></td>
<td>0C00H</td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>25</td>
<td></td>
<td>OC80H</td>
<td>MPMLDR</td>
</tr>
<tr>
<td>00</td>
<td>26</td>
<td>12</td>
<td>0D00H</td>
<td>LDRRBOS</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td></td>
<td>0D80H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>02</td>
<td>13</td>
<td>0E00H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>03</td>
<td>14</td>
<td>0E80H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>04</td>
<td>15</td>
<td>0F00H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>05</td>
<td></td>
<td>0F80H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>06</td>
<td>16</td>
<td>1000H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>07</td>
<td></td>
<td>1080H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>08</td>
<td>17</td>
<td>1100H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>09</td>
<td></td>
<td>1180H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>18</td>
<td>1200H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td></td>
<td>1280H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>19</td>
<td>1300H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13</td>
<td></td>
<td>1380H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>20</td>
<td>1400H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td></td>
<td>1480H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td></td>
<td>1500H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>17</td>
<td></td>
<td>1580H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>21</td>
<td>1600H</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>19</td>
<td></td>
<td>1680H</td>
<td>LDRRBOS</td>
</tr>
<tr>
<td>01</td>
<td>20</td>
<td>22</td>
<td>1700H</td>
<td>LDRBIO</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td></td>
<td>1780H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>23</td>
<td>1800H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>23</td>
<td></td>
<td>1880H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>24</td>
<td>1900H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>25</td>
<td></td>
<td>1980H</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>26</td>
<td>25</td>
<td>1A00H</td>
<td>LDRBIO</td>
</tr>
</tbody>
</table>

All Information Presented Here is Proprietary to Digital Research
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 Information Presented Here is Proprietary to Digital Research
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 $\uparrow z$ (IAH) 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 or teletypes. MP/M II supports up to 16 console 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

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMONBASE</td>
<td>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.</td>
</tr>
<tr>
<td>WBOOT</td>
<td>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.</td>
</tr>
<tr>
<td>CONST</td>
<td>The CONST subroutine obtains the status of the console device specified by register D and returns OFFH 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.</td>
</tr>
<tr>
<td>CONIN</td>
<td>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.</td>
</tr>
</tbody>
</table>

All Information Presented Here is Proprietary to Digital Research
### Table 2-1. (continued)

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CONOUT</strong></td>
<td>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 <code>^Z</code> 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.</td>
</tr>
<tr>
<td><strong>LIST</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>PUNCH</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>READER</strong></td>
<td>The reader device is not implemented under MP/M II. See the note above for PUNCH.</td>
</tr>
<tr>
<td><strong>HOME</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>
Table 2-1. (continued)

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELDSK</td>
<td>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.</td>
</tr>
</tbody>
</table>

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)**

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SETTRK</td>
<td>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.</td>
</tr>
<tr>
<td>SETSEC</td>
<td>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.</td>
</tr>
<tr>
<td>SETDMA</td>
<td>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 OFFH, and all subsequent write operations get their data from 80H through OFFH, 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.</td>
</tr>
</tbody>
</table>
Thus, a call to the SETDMA subroutine is interpreted as a flush buffer call when a parameter of OFFFFH 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 OFFFFH 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:

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>non-recoverable error condition occurred</td>
</tr>
<tr>
<td>2</td>
<td>disk read/only</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no errors occurred</td>
</tr>
<tr>
<td>1</td>
<td>non-recoverable error condition occurred</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WRITE</strong></td>
<td>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 &quot;non deleted data&quot; to maintain compatibility with other CP/M and MP/M systems. WRITE returns the following error codes in register A, as shown below:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no errors occurred</td>
</tr>
<tr>
<td>1</td>
<td>non-recoverable error condition occurred</td>
</tr>
<tr>
<td>2</td>
<td>disk read/only</td>
</tr>
</tbody>
</table>

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 process is terminated or returned an error code. |
<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LISTST</td>
<td>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 OFFH 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.</td>
</tr>
<tr>
<td>SECTRAN</td>
<td>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 &quot;skew factor&quot; 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.</td>
</tr>
</tbody>
</table>

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.
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.

<table>
<thead>
<tr>
<th>Disk Parameter Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>XLT 0000 0000 0000 DIRBUF DPB CSV ALV</td>
</tr>
<tr>
<td>16b 16b 16b 16b 16b 16b 16b 16b</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XLT</td>
<td>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.</td>
</tr>
<tr>
<td>0000</td>
<td>Scratchpad values for use within the BDOS (initial value is unimportant).</td>
</tr>
<tr>
<td>DIRBUF</td>
<td>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.</td>
</tr>
<tr>
<td>DPB</td>
<td>Offset of a disk parameter block for this drive. Drives with identical disk characteristics address the same disk parameter block.</td>
</tr>
<tr>
<td>CSV</td>
<td>Offset of a scratchpad area used for software check for changed disks. This offset is different for each DPH.</td>
</tr>
<tr>
<td>ALV</td>
<td>Offset of a scratchpad area used by the BDOS to keep disk storage allocation information. This offset is different for each DPH.</td>
</tr>
</tbody>
</table>

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
0n-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.

All Information Presented Here is Proprietary to Digital Research

25
NDISKS EQU 4 ;NUMBER OF DISK DRIVES

SELDISK:
;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  ALO  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

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

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

<table>
<thead>
<tr>
<th>BLS</th>
<th>BSH</th>
<th>BLM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,024</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>2,048</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>4,096</td>
<td>5</td>
<td>31</td>
</tr>
<tr>
<td>8,192</td>
<td>6</td>
<td>63</td>
</tr>
<tr>
<td>16,384</td>
<td>7</td>
<td>127</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>BLS</th>
<th>DSM ≤ 256</th>
<th>DSM &gt; 255</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,024</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>2,048</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4,096</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>8,192</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>16,384</td>
<td>15</td>
<td>7</td>
</tr>
</tbody>
</table>

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

ALO  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 ALO, 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

<table>
<thead>
<tr>
<th>BLS</th>
<th>Directory Entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,024</td>
<td>32 times # bits</td>
</tr>
<tr>
<td>2,048</td>
<td>64 times # bits</td>
</tr>
<tr>
<td>4,096</td>
<td>128 times # bits</td>
</tr>
<tr>
<td>8,192</td>
<td>256 times # bits</td>
</tr>
<tr>
<td>16,384</td>
<td>512 times # bits</td>
</tr>
</tbody>
</table>

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 ALO are set, resulting in the values ALO = 0F0H and AL1 = 00H.
The CKS value is determined as follows: if the disk drive media is removable, then \( \text{CKS} = (\text{DRM}+1)/4 \), where DRM is the last directory entry number. If the media is fixed, then set \( \text{CKS} = 8000H \) (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 \( \text{CKS} \) bytes, which is sufficient to hold the directory check information for this particular drive. If \( \text{CKS} = (\text{DRM}+1)/4 \), then you must reserve \( (\text{DRM}+1)/4 \) bytes for directory check use. If \( \text{CKS} = 0 \), indicating no checked directory entries, or \( \text{CKS} = 8000H \), 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 \((\text{DSM}/8)+1\).

### 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 your 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,[kl6],[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 optional 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 track offset to logical track 00
kl6 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, if 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 a 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:
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 DBASE 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:

<table>
<thead>
<tr>
<th>DPH Label</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>DP0</td>
<td>XX0000H,0000H,DIRBUF,DPBO,CSV0,ALV0</td>
</tr>
<tr>
<td>DP1</td>
<td>XX0000H,0000H,DIRBUF,DPBO,CSV1,ALV1</td>
</tr>
<tr>
<td>DP2</td>
<td>XX0000H,0000H,DIRBUF,DPBO,CSV2,ALV2</td>
</tr>
<tr>
<td>DP3</td>
<td>XX0000H,0000H,DIRBUF,DPBO,CSV3,ALV3</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>XLT0</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,7,13,19,25,5,11,17,23,3,9,15,21</td>
<td></td>
</tr>
<tr>
<td>2,8,14,20,26,6,12,18,24,4,10,16,22</td>
<td></td>
</tr>
</tbody>
</table>

Following the ENDEF macro call, a number of uninitialized data areas 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:
MP/M II System Guide

2.3 BIOS Disk Definition Tables

\[ 4C72 = \text{BEGDAT EQU $} \]
\[ (\text{data areas}) \]
\[ 4DB0 = \text{ENDDAT EQU $} \]
\[ 013C = \text{DATSIZ EQU $-\text{BEGDAT}} \]

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

\[ \text{STAT d:DSK:} \]

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

\[ r: \text{128 Byte Record Capacity} \]
\[ k: \text{Kilobyte Drive Capacity} \]
\[ d: \text{32 Byte Directory Entries} \]
\[ c: \text{Checked Directory Entries} \]
\[ e: \text{Records/Extent} \]
\[ b: \text{Records/Block} \]
\[ s: \text{Sectors/Track} \]
\[ t: \text{Reserved Tracks} \]

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.

All Information Presented Here is Proprietary to Digital Research
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

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWTUSER</td>
<td>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.</td>
</tr>
<tr>
<td>SWTSYS</td>
<td>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.</td>
</tr>
<tr>
<td>PDISP</td>
<td>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.</td>
</tr>
<tr>
<td>XDOS</td>
<td>The XDOS entry point provides access to XDOS functions. XDOS functions are required for flag operations, queue operations and polling devices.</td>
</tr>
<tr>
<td>SYSDAT</td>
<td>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.</td>
</tr>
</tbody>
</table>
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
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.

Conditions 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 are 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, hstrak 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 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.

All Information Presented Here is Proprietary to Digital Research
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 ; IDLE 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

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELMEMORY</td>
<td>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.</td>
</tr>
</tbody>
</table>
### Table 3-1. (continued)

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>POLLDEVICE</strong></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>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 handler (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 OFFH in the accumulator if the device is ready, or OOH 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.</td>
</tr>
<tr>
<td><strong>STARTCLOCK</strong></td>
<td>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).</td>
</tr>
<tr>
<td>Subroutine</td>
<td>Function</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
</tr>
<tr>
<td>STOPCLOCK</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>EXITREGION</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>
Table 3-1. (continued)

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAXCONSOLE</td>
<td>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.</td>
</tr>
<tr>
<td>SYSTEMINIT</td>
<td>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:</td>
</tr>
<tr>
<td>C</td>
<td>MP/M debugger restart #</td>
</tr>
<tr>
<td>DE</td>
<td>MP/M entry point address for the debugger. Place a jump at the proper debugger restart location to the address contained in DE.</td>
</tr>
</tbody>
</table>
Table 3-1. (continued)

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Routine</th>
</tr>
</thead>
<tbody>
<tr>
<td>HL</td>
<td>BIOS direct jump table address. Place a jump instruction at location 0000H in each bank's base page to the address contained in HL.</td>
</tr>
<tr>
<td>IDLE</td>
<td>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:</td>
</tr>
</tbody>
</table>

    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 illustrate 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 flag #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 GEN SYS to dynamically configure the MP/M II system. The system data page can be prepared using the GEN SYS program or it can be manually prepared using DDT or SID. The Table 4-1 describes the byte assignments.

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>000-000</td>
<td>Mem$top, top page of memory</td>
</tr>
<tr>
<td>001-001</td>
<td>Nmb$cnsc, number of system consoles (TMPs)</td>
</tr>
<tr>
<td>002-002</td>
<td>Brkpt$RST, breakpoint RST #</td>
</tr>
<tr>
<td>003-003</td>
<td>Add system call user stacks, boolean</td>
</tr>
<tr>
<td>004-004</td>
<td>Bank switched, boolean</td>
</tr>
<tr>
<td>005-005</td>
<td>Z80 version, boolean</td>
</tr>
<tr>
<td>006-006</td>
<td>banked bdos, boolean</td>
</tr>
<tr>
<td>007-007</td>
<td>XIOS jump table page</td>
</tr>
<tr>
<td>008-008</td>
<td>RESBDOS base page</td>
</tr>
<tr>
<td>009-010</td>
<td>CP/NET master configuration table address</td>
</tr>
<tr>
<td>011-011</td>
<td>XDOS base page</td>
</tr>
<tr>
<td>012-012</td>
<td>RSPs (BNKXIOS top+1) base page</td>
</tr>
<tr>
<td>013-013</td>
<td>BNKXIOS base page</td>
</tr>
<tr>
<td>014-014</td>
<td>BNKBDOS base page</td>
</tr>
<tr>
<td>015-015</td>
<td>Max$mem$seg, max memory segment number</td>
</tr>
<tr>
<td>016-017</td>
<td>Initial memory segment table</td>
</tr>
<tr>
<td>048-063</td>
<td>Breakpoint vector table, filled in by debuggers</td>
</tr>
<tr>
<td>064-079</td>
<td>Reserved for MP/M II</td>
</tr>
<tr>
<td>080-095</td>
<td>System call user stack pointer table</td>
</tr>
<tr>
<td>096-119</td>
<td>Reserved for MP/M II</td>
</tr>
<tr>
<td>120-121</td>
<td>Nmb records in MPM.SYS file</td>
</tr>
<tr>
<td>122-122</td>
<td># ticks/sec</td>
</tr>
<tr>
<td>123-123</td>
<td>System Drive</td>
</tr>
<tr>
<td>124-124</td>
<td>Common Memory Base Page</td>
</tr>
<tr>
<td>125-125</td>
<td>Number of RSPs</td>
</tr>
<tr>
<td>126-127</td>
<td>Listcp array Address</td>
</tr>
<tr>
<td>128-143</td>
<td>Subflg, submit flag array</td>
</tr>
</tbody>
</table>

All Information Presented Here is Proprietary to Digital Research
Table 4-1. (continued)

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

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 named 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 BDOS/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 OFFH. The name provided in the process descriptor is used to obtain the banked portion which has a file type of BRS.
The second part of a banked system process is the actual code piece for the process. The rules for the BRS portion are as follows:

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

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

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

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

OA>gensys

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

Default entries are shown in (paren). Default base is 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 (CO) ?
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:

All Information Presented Here is Proprietary to Digital Research
5.1 System Generation Overview

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
BNKBDDS SPR 9500H 2300H
BNKXDDS 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 issues involved in answering each of the GEN
SYS queries shown in the example above.

5.2.1 Defaults

The GEN
SYS program displays default entry values within parentheses. The base is hex unless a # character precedes the value to indicate a decimal base. The initial prompt determines if the internal GEN
SYS defaults are to be used, or those of the most recently 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 from 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.

All Information Presented Here is Proprietary to Digital Research
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.

All Information Presented Here is Proprietary to Digital Research
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 a "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 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.

All Information Presented Here is Proprietary to Digital Research
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 banked 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 nonexistent 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 can 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 banks. 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
280 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
BNKBDO S 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

------------------
MP/M II Sys 7500H 8B00H 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 for 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
def

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,[kl6],[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 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 dir elements to checksum
  ofs is the number of tracks to skip (word)
  kl6 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

All Information Presented Here is Proprietary to Digital Research
; the value of "begdat" at the end of assembly defines the
; beginning of the uninitialized 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

disks set nd ;; for later reference

dpb&dn equ $ ; base of disk parameter blocks
;; generate the nd elements

dsknxt set 0
  rept nd
    dskhdr &dsknxt
  endm
  dsknxt set dsknxt+l
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,kl6
;
; generate the set statements for later tables
 ckysz 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/l28 ; ;number of sectors/block
 blkshf set 0 ; ;counts right 0's in blkval
 blkmsk set 0 ; ;fills with 1's from right
 rept 16 ; ;once for each bit position
 if blkval=l
 exitm
 endif

;; otherwise, high order 1 not found yet
 blkshf set blkshf+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=l
 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)

All Information Presented Here is Proprietary to Digital Research
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
ddna

; ; ; generate equ $%
dpw %sectors,<;sec per track>
ddb %blkshf,<;block shift>
ddb %blkmsk,<;block mask>
ddb %extmsk,<;extnt mask>
ddw %(dks)-l,<;disk size-l>
ddw %(dir)-l,<;directory max>
ddb %dirblk shr 8,<;alloc0>
ddb %dirblk and 0ffh,<;alloc1>
if nul cmp
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 equ 0 ;;no xlate table
else
if skf = 0
xlt&dn equ 0 ;;no xlate table
else

;;; generate the translate table
nxstsec set 0 ;;next sector to fill
nxstbas set 0 ;;moves by one on overflow
gcd ;; gcdn = gcd(sectors,skew)
neltst set sectors/gcdn

;; neltst is number of elements to generate
;; before we overlap previous elements
neltst set neltst ;;counter
xlt&dn equ $ ;;translate table
rept sectors ;;once for each sector

All Information Presented Here is Proprietary to Digital Research
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 nelstst
endif
defds lab,spa.ce
lab: ds space
endm

defds macro lb,dn,val
  defds lb&dn,%val&dn
endm

ldp macro lb,dn,val
  defds lb&dn,%val&dn
endm

endif

generate the necessary ram data areas
begdat equ $

endir

$-begdat

$-begdat

$-begdat

$-begdat

$-begdat
Appendix B
Sector Deblocking Algorithms for MP/M II

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

All Information Presented Here is Proprietary to Digital Research
The BDOS entry points given below show the code which is relevant to deblocking only.

DISKDEF macro, or hand coded tables go here

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

All Information Presented Here is Proprietary to Digital Research

69
; 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 lhdl 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 ; l to accum
00B2 327101 sta rsflag ; rsflag = 1
;
; ******************************************************
; * Common code for READ and WRITE follows *
; *
**MP/M II System Guide Appendix B Sector Deblocking**

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

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
00B7 326801  rept sechsf
00BC+B7  ora a ; carry = 0
00BD+1F  rar ; shift right
00BE+B7  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

00C3 216901  lxi h, hstact ; host active flag
00C6 7E  mov a, m
00C7 3601  mvi m, l ; always becomes 1
00C9 B7  ora a ; was it already?
00CA CAF100  jz filhst ; fill host if not
00CD 3A6001  lxi h, hstdsk
00D0 216401  cmp m
00D3 BE  jnz nomatch

00D4 C2EA00 ; host buffer active, same as seek buffer?
00D7 216501  lxi h, hsttrk
00DA CD5201  call sektrkcmp ; sektrk = hsttrk?
00DD C2EA00  jnz nomatch

00E0 3A6801  lxi h, hstsec ; sekhst = hstsec?
00E3 216701  cmp m
00E6 BE  jz match ; skip if match

00E7 CA0E01 ; same disk, same track, same buffer?
00EE C45E01 ; proper disk, but not correct sector

00F1 3A6001  lxi h, hstdsk
00F4 326401  sta hstdsk
00F7 2A6101  lhl1d sektrk
00FA 226501  shld hsttrk
00FD 3A6801  lxi h, sekhst
0100 326701  sta hstsec
0103 3A7101  lxi h, hrflag ; need to read?
```

All Information Presented Here is Proprietary to Digital Research
match:

; copy data to or from buffer

010E 3A6301
    lda sekssec ; mask 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 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
    lhlh 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,l
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

All Information Presented Here is Proprietary to Digital Research

72
MP/M II System Guide

Appendix B Sector Deblocking

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
0159 13  inx  d
015A 23  inx  h
015B 1A  ldax  d
015C 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  *
;************************************************************************************************************

sekdsk:  ds 1  ;seek disk number
sektrk:  ds 2  ;seek track number
seksec:  ds 1  ;seek sector number
hstdsk:  ds 1  ;host disk number

All Information Presented Here is Proprietary to Digital Research
hsttrak: ds 2 ;host track number
hstsec: ds 1 ;host sector number

sekhst: ds 1 ;seek shr secshf
hstact: ds 1 ;host active flag
hstwrt: ds 1 ;host written flag

unacnt: ds 1 ;unalloc rec cnt
unadsk: ds 1 ;last unalloc disk
unatrk: ds 2 ;last unalloc track
unasec: ds 1 ;last unalloc sector

erflag: ds 1 ;error reporting
rsflag: ds 1 ;read sector flag
readop: ds 1 ;1 if read operation
wrtype: ds 1 ;write operation type
dmaadr: ds 2 ;last dma address
hstbuf: ds hstsz ;host buffer

;******************************************************
;* The ENDEF macro invocation goes here *
;******************************************************
end
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:
1734 C9 ret ; crt: input
crtout:
1735 C9 ret

All Information Presented Here is Proprietary to Digital Research
crtst:  ; crt: status
1736 C9  ret

ttyin:  ; tty: input
1737 C9  ret

ttyout:  ; tty: output
1738 C9  ret

1739 C9  lptout:  ; lpt: output
ret

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

end
Appendix D
Simple 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

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 ;cold start
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

All Information Presented Here is Proprietary to Digital Research
MP/M II System Guide

Appendix D Simple XIOS Source

0033 C30C02 jmp semory ; select memory
0036 C3F301 jmp polldevice ; poll device
0039 C30D02 jmp startclock ; start clock
003C C31302 jmp stopclock ; stop clock
003F C31802 jmp exitregion ; exit region
0042 C31F02 jmp maxconsole ; maximum console numb
0045 C32202 jmp systeminit ; system initializatio
0048 00 db 0 ; force use of interna
        jmp idle ; idle procedure

    commonbase:
0049 C35A00 jmp coldstart
004C C30000 swtuser: jmp $-$
004F C30000 swtsys: jmp $-$
0052 C30000 pdis: jmp $-$
0055 C30000 xdos: jmp $-$
0058 0000 sysdat: dw $-

coldstart:
warmstart:
005A 0E00 mvi c,0
005C C35500 jmp xdos ; system reset, termin

; I/O handlers
;
; MP/M II V2.0 Console Bios
;
0003 = nmbcns equ 3 ; number of consoles
0083 = poll equ 131 ; X DOS poll function
0086 = makeque equ 134 ; X DOS make queue function
0089 = readque equ 137 ; X DOS read queue function
008B = writeque equ 139 ; X DOS write queue function
008D = xdelay equ 141 ; X DOS delay function
0090 = create equ 144 ; X DOS 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 = plc13 equ 4 ; poll console in #2 (Port 3)
        if debug
0005 = plc0 equ 5 ; poll console in #0
        endif

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

All Information Presented Here is Proprietary to Digital Research

78
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
0071 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 handler
; 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
; h = double table index for addr
; d = address
; return addr points to jump to handler
0083 87     add  a ; double table index for addr
0084 E1     pop  h ; return addr points to jump to handler
0085 5F     mov  e,a
0086 1600   mvi  d,0
0088 19     dad  d ; add table index * 2 to tbl b
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 = stso equ data0+1
0041 = cd0 equ stso
0048 = data1 equ 48h
0049 = stsl equ data1+1
0049 = cd1 equ stsl
0050 = data2 equ 50h
0051 = st2 equ data2+1

All Information Presented Here is Proprietary to Digital Research
MP/M II System Guide

Appendix D Simple XIOS Source

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:

008E 3AAF00
0091 B7
0092 3E00
0094 C0

if ldcmd

lda pt0cntr
ora a
mvi a,0

rnz
endif

0095 DB41
0097 E602
0099 C8
009A 3EFF
009C C9

in sts0
ani 2
rz
mvi a,0ffh
ret

pt0in:

009D 21AF00
00A0 7E
00A1 B7
00A2 CAB600
00A5 35
00A6 2AB000
00A9 7E
00AA 23
00AB 22B000
00AE C9

ldcmand

lxh h,pt0cntr
mov a,m
ora a
jz ldcmd0empty
dcr m
lhld pt0ptr
mov a,m
inx h
shld pt0ptr
ret

pt0cntr:

00AF 04
00B0 B200
00B2 746F6420

db ldcmd0empty-pt0ldcmd

pt0ptr:

dw pt0ldcmd

pt0ldcmd:

db 'tod'

ldcmd0empty:

endif

00B6 0E83
00B8 B005
00BA CD5500
00BD DB40
00BF E67F
00C1 C9

mvi c,poll
mvi e,plci0
call xdos
in data0
ani 7fh
ret

else

pt0st:

; return 0ffh if ready,
; 000h if not

All Information Presented Here is Proprietary to Digital Research
lda c0inmsgcnt
ora a
rz
mvi a,0ffh
ret
;
; Console #0 Input
;
c0inpd:
dw c2inpd ; 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 'c0inqque ' ; 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
c0inquqcb:
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
MP/M II System Guide

Appendix D Simple XIOS Source

```assembly
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 stso
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 stso
00D8 E601 ani 01h
00DA C8 rz
00DB 3EFF mvi a, 00ffh
00DD C9 ret

;
; Line Printer Driver: TI 810 Serial Printer
;
; initflag:
00DE 00 db 0 ; printer initialization flag

list:

ptlout: ; List Output

00DF 3ADE00 lda initflag
00E2 B7 ora a

All Information Presented Here is Proprietary to Digital Research

82
MP/M II System Guide Appendix D Simple XIOS Source

```
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:
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:
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 dw 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
```

All Information Presented Here is Proprietary to Digital Research
0151 C7C7C7C7C7  dw 0c7c7h,0c7c7h,0c7c7h
0157 7F01  dw c2inp  ; starting address

   c2inq:
0159 0000  dw 0  ; q1
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
0179 00  db 0

   c2inuqcb:
017A 5901  dw c2inq  ; pointer
017C 7E01  dw char2in  ; msgadr
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 0E88  mvi c,writeque
0190 117501  lxi d,c2inuqcb
0193 CD5500  call xdos  ; write c2in queue
0196 C38701  jmp c2inloop

   pt2in:  ; return character in reg A
0199 0E89  mvi c,readque
019B 17A01  lxi d,c2inuqcb
019E CD5500  call xdos  ; read from c2 in queu
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

All Information Presented Here is Proprietary to Digital Research
MP/M II System Guide

Appendix D Simple XIOS Source

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

All Information Presented Here is Proprietary to Digital Research
MP/M II System Guide  Appendix D  Simple XIOS Source

01EA C9         ret 
; Poll Console #2 (Port 3) Output 
; polco3:         ; return 0ffh if ready, 
                  ; 000h if not 

01EB DB59       in  st3s3 
01ED E601       anl  01h 
01EF C8         rz  
01F0 3EFF       mvi  a,0ffh 
01F2 C9         ret  

; MP/M II V2.0  Xios 
; 
polldevice:     ; Reg C = device # to be poll 
                  ; 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:          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 

All Information Presented Here is Proprietary to Digital Research
; startclock:
; will cause flag #1 to be set
; at each system time unit tick
020D 3EFF mvi a,Offh
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 di
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,intnhnd
022A 223900 shld 0039h ; JMP INTHND at 0038H
022D OE90 mvi c,create
if debug
022F 111101 lxi d,c2inpd
else
0231 CD5500 lxi d,c0inpd
endif
call xdos
0232 CD5500 lxi intmsk
0235 3A3004
Idle procedure
; idle:
    ret
; or-
    ei
    hlt
    ret
; for full interrupt s

; MP/M II V2.0 Interrupt Handlers

inthnd:
    ; Interrupt handler entry poin
    ; All interrupts gen a RST 7
    ; Location 0038H contains a j
    ; 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          anl   01000000b ; test & jump if clk i
0264 C28F02        jnz  clk60hz
0267 DB80           in   stat       ; read disk status por

All Information Presented Here is Proprietary to Digital Research
MP/M II System Guide

Appendix D Simple XIOS Source

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 in

0275 C3B502  jmp  intdone

diskintr:
0278 AF  xra  a
0279 D380  out  cmdl
027B 1E05  mvi  e,5
027D C38702  jmp  concmn

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

con2in:
in data2
sta ch2in
mvi e,8
; jmp concmn ; set flag #8

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

clock60hz:
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 tic

notickn:
029D 210004  lxi  h,cnt60
02A0 35  dcr  m ; dec 60 tick cntr
02A1 C2AD02  jnz  notlsec

All Information Presented Here is Proprietary to Digital Research
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 preemp ; clear preempted flag
02B9 Cl pop b
02BA D1 pop d
02BB 2A2B04 lhld svdsp
02BE F9 sphl         ; restore stk ptr
02BF F1 pop psw
02C0 2A2D04 lhld svdret
02C3 E5 push h
02C4 2A2904 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 dispatche
; 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 00 position of current driv
call headload
; h,l point to word with track for selected disk
home1:
02CA CDDA03 mvi m,00          ; set current track ptr back to
02CD 3600 in stat          ; read fdc status
02CF DB80 ani 4          ; test track 0 bit
02D1 E604 rz          ; return if at 0
MP/M II System Guide

Appendix D Simple XIOS Source

02D4 37 stc ;direction=out
02D5 CDC203 call step ;step one track
02D8 C3CD02 jmp homel ;loop

; seldsk: ;drive number in c
02DB 210000 lx a, c ;a is disk number 0 ... ndisks
02DE 79 cpi ndisks ;less than ndisks?
02EF 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 lx 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
030E 79 mov a,c ;step will update trk indicator
030F BE cmp m ;are we where we want to be
0310 C20B03 jnz settkx ;not yet

; have stepped enough
0313 3E14 mvi a, 20d
; seekrt: ;need 10 msec delay for final step time and head settle
0314 08 call delay
; ret ;end of settrk routine

All Information Presented Here is Proprietary to Digital Research

91
delay: ;delay for c[A] X .5 milliseconds
0315 C5  push    b
0316 0E86 mvi    c,dlycnst ;constant adjusted to .5 ms
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 saved 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 below
; buf with trk,sctr,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  jmp  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 numb
; will result in head unloaded
; error, so don't check
0374 77  mov  m,a  ; fill in track

All Information Presented Here is Proprietary to Digital Research
<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0375 2B</td>
<td>dcx h</td>
</tr>
<tr>
<td>0376 C1</td>
<td>pop b</td>
</tr>
<tr>
<td>0377 79</td>
<td>mov a,c</td>
</tr>
<tr>
<td>0378 4E</td>
<td>mov c,m</td>
</tr>
<tr>
<td>0379 77</td>
<td>mov m,a</td>
</tr>
<tr>
<td>037A 7C</td>
<td>mov a,h</td>
</tr>
<tr>
<td>037B D381</td>
<td>out haddr ; high byte</td>
</tr>
<tr>
<td>037D 7D</td>
<td>mov a,l</td>
</tr>
<tr>
<td>037E D382</td>
<td>out laddr ; low byte</td>
</tr>
<tr>
<td>0380 78</td>
<td>mov a,b</td>
</tr>
<tr>
<td>0381 D380</td>
<td>out cmdl ; start disk read/write</td>
</tr>
</tbody>
</table>

rwwait:

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0383 C5</td>
<td>push b</td>
</tr>
<tr>
<td>0384 D5</td>
<td>push d</td>
</tr>
<tr>
<td>0385 E5</td>
<td>push h</td>
</tr>
<tr>
<td>0386 0E84</td>
<td>mvi c,flagwait</td>
</tr>
<tr>
<td>0388 1E05</td>
<td>mvi e,5</td>
</tr>
<tr>
<td>038A CD5500</td>
<td>call xdos ; wait for disk intrpt</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>038D E1</td>
<td>pop h</td>
</tr>
<tr>
<td>038E D1</td>
<td>pop d</td>
</tr>
<tr>
<td>038F C1</td>
<td>pop b</td>
</tr>
<tr>
<td>0390 71</td>
<td>mov m,c</td>
</tr>
<tr>
<td>0391 23</td>
<td>inx h</td>
</tr>
<tr>
<td>0392 72</td>
<td>mov m,d</td>
</tr>
<tr>
<td>0393 23</td>
<td>inx h</td>
</tr>
<tr>
<td>0394 73</td>
<td>mov m,e</td>
</tr>
<tr>
<td>0395 DB80</td>
<td>in stat ; test for errors</td>
</tr>
<tr>
<td>0397 E6F0</td>
<td>ani 0f0h</td>
</tr>
<tr>
<td>0399 C8</td>
<td>rz a will be 0 if no errors</td>
</tr>
</tbody>
</table>

; error from disk

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>039A F5</td>
<td>push psw ; save error condition</td>
</tr>
<tr>
<td>039B 213B04</td>
<td>lxi h,errors</td>
</tr>
<tr>
<td>039E 35</td>
<td>dcr m</td>
</tr>
<tr>
<td>039F C2A603</td>
<td>jnz redo ; not ten yet, do a retry</td>
</tr>
</tbody>
</table>

; we have too many errors, print out hex number for last received error type, cpm will print perm error message

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>03A2 F1</td>
<td>pop psw ; get code</td>
</tr>
<tr>
<td>03A3 3E01</td>
<td>mvi a,l</td>
</tr>
<tr>
<td>03A5 C9</td>
<td>ret redo:</td>
</tr>
</tbody>
</table>

; b still has read/write flag

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>03A6 F1</td>
<td>pop psw ; get error code</td>
</tr>
<tr>
<td>03A7 E6E0</td>
<td>ani 0e0h ; retry if not track error</td>
</tr>
<tr>
<td>03A9 C24C03</td>
<td>jnz tryagn ;</td>
</tr>
</tbody>
</table>

; was a track error so need to reseek

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>03AC C5</td>
<td>push b ; save read/write indicator</td>
</tr>
<tr>
<td>03AD 113204</td>
<td>lxi d,track</td>
</tr>
<tr>
<td>03B0 2A3904</td>
<td>lhld diskno ; selected disk</td>
</tr>
</tbody>
</table>

All Information Presented Here is Proprietary to Digital Research
03B3 19  dad  d ;point to correct trk indicato
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
dostep:
03C6 3E04  mvi  a,04h ;set direction = in
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
03D2 C31503  jmp  delay
; ;
; 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 select 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

All Information Presented Here is Proprietary to Digital Research
MP/M II System Guide  Appendix D  Simple XIOS Source

03EF B8  cmp    b       ;are we on the same drive
03F0 C2F803 jnz    needly
 ;we are on the same drive
 ;is the head loaded?
03F3 DB80  in     stat
03F5 E680  ani    80h
03F7 C8    rz     ;already loaded

03F8 AF  xra   a
03F9 D380  out   cmd1  ;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
 ; 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 intr
 else
 intmsk:  db    54h  ; intrpt msk, enables clk intr
 endif
0431 00   preemp: db    0       ; preempted boolean
 ;
scrat:  db    0       ; start of scratch area
0432 00   track:  db    0       ; current trk on drive 0
0433 00   trkl:  db    0       ; current trk on drive 1
0434 00   trk2:  db    0
0435 00   trk3:  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+1005F04 DW CSV0,ALVO ;CHECK, ALLOC VECTORS

All Information Presented Here is Proprietary to Digital Research
MP/M II System Guide

Appendix D Simple XIOS Source

044F+00000000 DPE1: DW XLT1,0000H ;TRANSLATE TABLE
0453+00000000 DW 0000H,0000H ;SCRATCH AREA
0457+70045F04 DW DIRBUF,DPB0 ;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 ;DIRECTORY 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
04F0+ ALVO: 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

All Information Presented Here is Proprietary to Digital Research
Appendix E
Sample MP/M II Banked XIOS

ALTOS COMPUTER SYSTEMS
2360 BERING DRIVE
SAN JOSE, CALIFORNIA 95131

Copyright 1980, ALTOS COMPUTER SYSTEMS

This program is a copyright program product of
ALTOS COMPUTER SYSTEMS and is distributed to the
owners of ALTOS SUN SERIES 8000 computers for
use on those systems 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 2.0
. Add support for Hard disk drives
. Add support for disk MODE selection
. Provide compatibility MODE for 1.4 operation
. Remove CTC/1791 counter reset
. CORRECT HARD DISK SEEK PROBLEM
. Add code to recover from WD1791 going to sl
. Initialize parallel port for Centronics pri

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 IBM single density
. 1 ALTOS double density Version 2.0

All Information Presented Here is Proprietary to Digital Research
MP/M II System Guide

MP/M Banked XIOS

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

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

MACLIB  DISKDEF
MACLIB  Z80S

FFFF = TRUE EQU OFFFFH ;VALUE FOR TRUE
0000 = FALSE EQU NOT TRUE ;VALUE FOR FALSE

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

1700 = ldrbiosbase equ 1700h ;

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

;-------------------------------------------------------------
; 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 THE ALTOS DOUBLE DENSIT
; AND THE ALTOS 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 TRAC
0008 = HSTBLK EQU HSTSIZ/128 ;CP/M SECTORS PER HOST BUFF
0080 = CPMSPT EQU HSTBLK * HSTSPT ;CP/M SECTORS PER TRACK
0007 = SECMASK EQU HSTBLK - 1 ;SECTOR MASK
0003 = SECSHF EQU 3 ;LOG2(HHSTBLK)

All Information Presented Here is Proprietary to Digital Research
THE FOLLOWING EQUATES APPLY TO THE RELOCATABILITY OF THE CBION AND SHOULD NOT BE USER ALTERED.

FFFF = RELOC EQU TRUE ;RELOCATABLE VERSION ??

if mdisk
maxdsk equ 13
ELSE
endif

000C = MAXDSK EQU 12 ;MAXIMUM NUMBER OF LOGICAL
ELSE
MAXDSK EQU 4 ;MAXIMUM NUMBER OF LOGICAL
ENDIF

if RELOC
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 = PLC00 EQU PLLPT+1 ;POLL CONSOLE OUT #0 (CRT:)
0002 = PLC01 EQU PLC00+1 ;POLL CONSOLE OUT #1 (CRT:)
0003 = PLC02 EQU PLC01+1 ;POLL CONSOLE OUT #2 (CRT:)
0004 = PLC03 EQU PLC02+1 ;POLL CONSOLE OUT #3 (CRT:)
0005 = PLC10 EQU PLC03+1 ;POLL CONSOLE IN #0 (CRT:)
0006 = PLC11 EQU PLC10+1 ;POLL CONSOLE IN #1 (CRT:)

All Information Presented Here is Proprietary to Digital Research
MP/M II System Guide

Appendix E MP/M Banked XIOS

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
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 HOMEBIT ;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 POLLEDVICE ; 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

All Information Presented Here is Proprietary to Digital Research

102
MP/M II System Guide

Appendix E  MP/M Banked XIOS

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 P

RTNEMPTY:
XRA A ; NOT USED
RET
endif

LAST:
ORG (((LAST-BASE)+0A2H) AND OFF00H) +05EH

INTERUPT:
005E 470B  DW  FLOPPY$INT ; FLOPPY DISK INTERR
0060 1C0B  DW  NULL$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 INTERRUP
006C 1C0B  DW  NULL$INT 
006E 1C0B  DW  NULL$INT 

if not mpm20
NULL$INT:
EI
RETI
endif

PAGE

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

0070 FFFFFFFFFFFFFTRKO:  DB  OFFH,OFFH,OFFH,OFFH,OFFH,OFFH,OFFH,OFFH

All Information Presented Here is Proprietary to Digital Research
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
ANY FIELDS PRIOR TO THIS POINT. EXTERNAL PROGRAMS A
DEPENDENT UPON THE LOCATION OF THE PRECEEDING DATA.

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

00BA C37D DMAS1: DB 0C3H,07DH ;FIRST PART
00BC 0000 DMASA: DW 000H ;ADDRESS FOR
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 DIS

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

CONTROL BLOCKS FOR DISK DRIVER

All Information Presented Here is Proprietary to Digital Research
All Information Presented Here is Proprietary to Digital Research

105
MP/M II System Guide

Appendix E MP/M Banked XIOS

0171 00000000 DPEA: DW 0000H,0000H ;TRANSLATE TABLE AND WORK A
0175 00000000 DW 0000H,0000H ;SCRATCH AREA
0179 9D122E0E DW DIRBUF,DPB6 ;DIR BUFF, PARM BLOCK
017D C20A9E0A DW CSV,ALVA ;CHECK VECTOR, ALLOC VECTOR
0181 00000000 DPEB: DW 0000H,0000H ;TRANSLATE TABLE AND WORK A
0185 00000000 DW 0000H,0000H ;SCRATCH AREA
0189 9D122E0E DW DIRBUF,DPB6 ;DIR BUFF, PARM BLOCK
018D E60AC20A DW CSV,ALVB ;CHECK VECTOR, ALLOC VECTOR

ENDIF

if mdisk

; Virtual disk parameter header

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

-----------------------------------------------------

0191 B5010000 MODL0: DW XLTO,0000H ;MODEL DPE FOR MODE 0
0195 00000000 DW 0000H,000H ;
0199 9D12D40D DW DIRBUF,DPBO ;

019D CF010000 MODL1: DW XLTI,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 01070D1319XLTO: 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

XLTI:
XLT2:

01CF 012030405 DB 01,02,03,04,05,06,07,08,09,10,11,12,13
01DC 0EOF101112 DB 14,15,16,17,18,19,20,21,22,23,24,25,26
01E9 1B1CID1E1F 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

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

All Information Presented Here is Proprietary to Digital Research

106
;-----------------------------------------------------------------------

SELDSDK:

0203 79 MOV A,C ;LIMIT SELECT TO REAL OPTION
0204 FEOC CPI MAXDSK ;
0206+303A JRNC SELERR ; INVALID DRIVE
0208 1600 DB 030H,SELERR-$-1 ;----- FAKE JRNC INSTRUCTION
020A 59 MOV A,E ; TEST FOR INITIAL SELECT
020B 214602 ANI 1 ; E = 0 IS FIRST TIME
020E 19 PUSH PSW ;
020F 4E MOV D,0 ;
0210 79 MOV E,C ; TRANSLATE TABLE

if mdisk
CPI 12 ; FOR LOGICAL TO PHYSICAL
JZ VIRTUAL
endif

; POP PSW ; RESTORE TEST
; JRNC 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
0219+2827 JRZ SELERR ; DRIVE NOT CONFIGURED
021B 79 MOV A,E ;
021C FE04 CPI 4 ; CHECK FOR FLOPPY
021E+380F JRC SELSDP ;
0220 7E DB 038H,SELERR-$-1 ;----- FAKE JRC INSTRUCTION
0221 D320 MOV A,M ; RESTORE SELECT BYTE
0229 C5 OUT 20H ;
0224 0E01 PUSH B ; DELAY FOR 1 MS
0226 CD8207 CALL DELAY ;
0229 C1 POP B ;
022A DB24 RAL ; CHECK FOR HARD DISK READY
022C 17 JRNC SELERR ; 80H = READY
022D+3013 DB 030H,SELERR-$-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 ;

All Information Presented Here is Proprietary to Digital Research
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

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
0256 69      MOV     L,C        ;
0257 22AF00   SHLD    DMAADDR  ;
025A 23      inx    h         ;test for flush buffers
025B 7D      mov    a,l
025C B4      ora     h
025D C0      rnz    h
025E 21F00A   lxil    h,hstwrt ;HL = FFFFh is flush buffer
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
026B E1      pop     h
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
0272 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,SELO    ;START OF SELECT/DENSITY MA
027B 2AE60A   LHLD    NEWDSK    ;NEXT DRIVE ADDRESS
027E 2600    MVI     H,000H    ;ENSURE ZERO FOR SINGLE BYT
0280 19  DAD  D  ;POINT TO CORRECT MASK
0281 79  MOV  A,C  ;ISOLATE DENSITY BIT
0282 E601  ANI  0000001B
0284 4F  MOV  C,A  ;SAVE FOR NOW
0285 7E  MOV  A,M  ;LOAD SELECT DENSITY MASK
0286 E6FE  ANI  1111110B  ;RESET CURRENT DENSITY SETT
0288 B1  ORA  C  ;SET NEW VALUE
0289 77  MOV  M,A  ;RESTORE MASK IN TABLE
028A C9  RET  ;RETURN TO CALLER

if

MREADSECTOR:
call  compbank  ;compute bank
di
call  chgbank
mov  b,128
mov  d,localbuf
lad  addroff
ldir ;block move into the dma ar
mov  a,02h ; select bank 0
out  09h
ei
mov  b,128
lad  dmaadr
xchg
mov  h,localbuf
ldir
xor  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
mov  h,0
ani  0f0h ; bank is high order nibble
rar  ! rar  ! rar  ! rar
inc  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 within the

All Information Presented Here is Proprietary to Digital Research

109
MP/M II System Guide

Appendix E MP/M Banked XIOS

 mov e,a
 mvi d,0
 dad d  ; add sector offset within
 dad h ! dad h ! dad h ! dad h ! dad h ! dad h !
 shld addroff  ; (track * 24 + sector) * 1
 ret
 endif

READ:

if mdisk
 lda newdsk
 cpi 12
 jmp MREADSECTOR
 endif

028B CDEE02  call RETMOD  ; WHAT TYPE OF I/O ??
028E FE03  cpi 003H  ;
0290 DAE405  jc READSOFT  ; FLOPPY DISK DRIVE....
0293 C3B03  jmp READHARD  ; HARD DISK I/O

if mdisk

mwritesector:
call compbank
lhlh d,localbuf
lxi b,128
ldir
di
call chgbank
lxi d,localbuf
lxi b,128
lhlh 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 newdsk
cpi 12

All Information Presented Here is Proprietary to Digital Research
ROUTINES TO SET AND RETURN THE CURRENT DRIVE MODE

SETMOD:

<table>
<thead>
<tr>
<th>Code</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>02A1</td>
<td>21E60A</td>
<td>LXI H,NEWDSK ; SAVE NEWDSK IN STACK</td>
</tr>
<tr>
<td>02A4</td>
<td>7E</td>
<td>MOV A,M ;</td>
</tr>
<tr>
<td>02A5</td>
<td>F5</td>
<td>PUSH PSW ;</td>
</tr>
<tr>
<td>02A6</td>
<td>70</td>
<td>MOV M,B ;</td>
</tr>
<tr>
<td>02A7</td>
<td>C5</td>
<td>PUSH B ;</td>
</tr>
<tr>
<td>02A8</td>
<td>48</td>
<td>MVI E,0 ; INDICATE INITIAL SELECT</td>
</tr>
<tr>
<td>02A9</td>
<td>CD0302</td>
<td>CALL SELDSK ;</td>
</tr>
<tr>
<td>02AC</td>
<td>C1</td>
<td>POP B ;</td>
</tr>
<tr>
<td>02AD</td>
<td>7C</td>
<td>MOV A,H ; CHECK FOR BAD SELECT</td>
</tr>
<tr>
<td>02AE</td>
<td>B5</td>
<td>ORA L ;</td>
</tr>
<tr>
<td>02AF+2832</td>
<td>DB 028H,SMERR-$-1</td>
<td>---- FAKE JRZ INSTRUCTION</td>
</tr>
<tr>
<td>02B1</td>
<td>68</td>
<td>MOV L,B ; B AND L = DRIVE #</td>
</tr>
<tr>
<td>02B2</td>
<td>2600</td>
<td>MVI H,000H ; CHECK MODE SET VALIDITY</td>
</tr>
<tr>
<td>02B4</td>
<td>78</td>
<td>MOV A,B ;ONLY VALID FOR FLOPPY DISK</td>
</tr>
<tr>
<td>02B5</td>
<td>FE04</td>
<td>CPI 004H ;INVALID DRIVE FOR MODE SET</td>
</tr>
<tr>
<td>02B7+302A</td>
<td>DB 030H,SMERR-$-1</td>
<td>---- FAKE JRNC INSTRUCTION</td>
</tr>
<tr>
<td>02B9</td>
<td>118800</td>
<td>LXI D,MODE ;START OF MODE BYTES</td>
</tr>
<tr>
<td>02BC</td>
<td>19</td>
<td>DAD D ;</td>
</tr>
<tr>
<td>02BD</td>
<td>71</td>
<td>MOV M,C ;SAVE NEW MODE BYTE</td>
</tr>
<tr>
<td>02BE</td>
<td>E5</td>
<td>PUSH H ;SAVE MODE BYTE ADDRESS</td>
</tr>
<tr>
<td>02BF</td>
<td>79</td>
<td>MOV A,C ;SETUP FOR DENSITY CHANGE</td>
</tr>
<tr>
<td>02C0</td>
<td>B7</td>
<td>ORA A ;</td>
</tr>
<tr>
<td>02C1</td>
<td>0E00</td>
<td>MVI C,00OH ;ASSUME SINGLE DENSITY MODE</td>
</tr>
<tr>
<td>02C3+2802</td>
<td>DB 028H,SETSEL-$-1</td>
<td>---- FAKE JRZ INSTRUCTION</td>
</tr>
<tr>
<td>02C5</td>
<td>0E01</td>
<td>MVI C,001H ;SET FOR DOUBLE DENSITY MOD</td>
</tr>
<tr>
<td>02C7</td>
<td>CD7802</td>
<td>SETSEL: CALL SETDEN ;SET DENSITY BASED ON LOW B</td>
</tr>
<tr>
<td>02CA</td>
<td>E1</td>
<td>POP H ;RESTORE</td>
</tr>
<tr>
<td>02CB</td>
<td>6E</td>
<td>MOV L,M ;PICKUP MODE AGAIN</td>
</tr>
<tr>
<td>02CC</td>
<td>2600</td>
<td>MVI H,000H ;FOR SINGLE BYTE PRECISION</td>
</tr>
<tr>
<td>02CE</td>
<td>7D</td>
<td>MOV A,L ;SAVE MODE IN ACCUMULATOR F</td>
</tr>
<tr>
<td>02CF</td>
<td>29</td>
<td>DAD H ;* 2</td>
</tr>
<tr>
<td>02D0</td>
<td>29</td>
<td>DAD H ;* 4</td>
</tr>
<tr>
<td>02D1</td>
<td>E5</td>
<td>PUSH H ;SAVE * 4</td>
</tr>
<tr>
<td>02D2</td>
<td>29</td>
<td>DAD H ;* 8</td>
</tr>
</tbody>
</table>

All Information Presented Here is Proprietary to Digital Research

111
MP/M II System Guide

Appendix E  MP/M Banked XIOS

02D3 D1 POP D ;REGAIN * 4
02D4 19 DAD D ;* 12
02D5 119101 LXI D,MODLO ;FIRST MODEL DPE
02D8 19 DAD D ;POINT TO THIS ONE
02D9 EB XCHG ;SETUP TEMPORARILY AS DESTI
02DA 2AB200 LHLD DPEPTR ;ADDRESS OF CURRENTLY SELEC
02DD EB XCHG ;SETUP TO ALTER
02DE 010C00 LXI B,12 ;LENGTH FOR MOVE
02E0 00 EB 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,M obstacle ;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 ; VIRTUAL DISK
0301 AF XRA A ; SET TRACK TO ZERO
0302 67 MOV H,A
0303 6F MOV L,A
0304 22E70A SHLD NEWTRK
0307 C9 RET

REALDISK:

0308 FE04 CPI 4 ; CHECK FOR FLOPPY
030A+380E JRC HOME ; DO NOT BYPASS FLOPPY HOM
030C 4F DB 038H,HOME-$-1 ;---- FAKE JRC INSTRUCTION
030D 0600 MOV C,A ;
030F 217000 LXI H,TRK0 ;
0312 09 DAD B ;
0313 7E MOV A,M ; CHECK IF INITIALIZED
0314 FEFF CPI 0PFH ;
0316 3E00 MOV A,0 ;
0318 C0 RNZ ; YES - RETURN WITH NO ER
0319 77 MOV M,A ;

All Information Presented Here is Proprietary to Digital Research

112
HOME:
031A 3AE60A LDA NEWDSK ;GET VALUE OF DRIVE FOR HOM
031D FE04 CPI 004H ;IS IT A HARD DISK ??
031F+3022 JRNC HOMEHARD ;YES, PROCESS....
030H,HOMEHARD-$-1 ;---- FAKE JRNC INS

HOMESOFT:
0321 CD5205 CALL DSKSEL ;SELECT CORRECT DRIVE (IN A
0324 3AF60A LDA ERFLAG ;
0327 B7 ORA A ;CHECK FOR ERRORS DURING SE
0328+2016 JRNZ HOME1A ;EXIT IF ERRORS
032A CD8305 CALL DB 020H,HOME1A-$-1 ;---- FAKE JRNZ INSTRUCTION
032D 3600 MVI M,000H ;POINT TO TRACK REGISTER SA
032F CD1905 CALL DBL$UPDATE ;
0332 3E0A MVI A,00AH ;HOME COMMAND....
0334 CD6307 CALL FINTFIX ;CLEAR ANY PENDING INTERRUP
030H,HOME2C-$-1 ;AND ISSUE COMMAND
0337 CD3A07 HOME1: CALL FPYWAIT ;WAIT UNTIL I/O COMPLETE
033A 3APC0A LDA STATUS ;PICKUP STATUS BYTE
033D E698 ANI 1011000B ;CHECK STATUS
033F C8 RZ ;RETURN WITH GOOD ESULT
0340 3E01 HOME1A: MVI A,001H ;SET ERROR ON HOME
0342 C9 RET ;AND RETURN....

HOMEHARD:
0343 CD5205 IF HARDSK ;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 E6P0 ANI 1111000B ;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 INTERRUP
0357 CD1707 HOME2: CALL WAIT0 ;AND ISSUE COMMAND
035A 0E14 MVI C,20 ;WAIT UNTIL I/O COMPLETE
035C CD8207 CALL DELAY ;DELAY FOR 20 MILLISECONDS
035F AF XRA A ;SET NEW TRACK REGISTER TO
0360 D322 OUT 022H ;FOR CONTROLLER
0362 3APC0A LDA STATUS ;FOR CONTROLLER
0365 E65D ANI 01011101B ;CHECK STATUS
0367 C8 RZ ;
0368 3E01 MVI A,001H ;SET ERROR ON HOME
036A C9 RET ;AND RETURN.

PAGE

All Information Presented Here is Proprietary to Digital Research
THESE ARE THE HARD DISK UNBLOCK/REBLOCK AND READ AN
Routines Called by the BDOS Software.

READHARD:

```assembly
036B AF
  IF HARDSK
    XRA A
    ;RESET UNALLOCATED COUNT
  036C 32F10A
    STA UNACNT
  036F 3E01
    MVI A,001H
    ;READ THE SELECTED CP/M SEC
  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 -
    ENDI
```

WRITEHARD:

```assembly
037E AF
  IF HARDSK
    XRA A
    ;WRITE THE SELECTED CP/M SEC
  037F 32F80A
    STA READOP
    ;NOT A READ OPERATION
  0382 79
    MOV A,C
    ;WRITE TYPE IS PASSED IN RE
  0383 32F90A
    STA WRTYPE
    
    if mpm20
      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 NEWDSK
    ;DISK FOR I/O
  0392 32F20A
    STA UNADSK
    ;UNADSK = NEWDSK
  0395 2AE70A
    LHLD NEWTRK
    ;
  0398 22F30A
    SHLD UNATRK
    ;UNATRK = NEWTRK
  039B 3AE90A
    LDA NEWSEC
    ;
  039E 32F50A
    STA UNASEC
    ;UNASEC = NEWSEC
    ; CHECK FOR WRITE TO UNALLOCATED SECTOR
    ;
```

CHKUNA:

```assembly
03A1 3AF10A
  LDA UNACNT
  ;ANY UNALLOCATED REMAIN ??
03A4 B7
  ORA A
  ;
```

All Information Presented Here is Proprietary to Digital Research

114
JRZ ALLOC ;SKIP IS NOT
DB 028H,ALLOC-$-1 ;----- FAKE JRZ INSTRUCTION

; MORE UNALLOCATED RECORDS REMAIN
;

DCR A ;UNACNT = UNACNT - 1
STA UNACNT ;
LDA NEWDSK ;SAME DISK ??
LXI H,UNADSK ;
CMP M ;NEWDSK = UNADSK ??
JRNZ ALLOC ;SKIP IF NOT
DB 020H,ALLOC-$-1 ;----- FAKE JRNZ INSTRUCTION

; DISKS ARE THE SAME
;

LXI H,UNATRK ;
CALL NEWTRKCMP ;NEWTRK = UNATRK ??
JRNZ ALLOC ;SKIP IF NOT
DB 020H,ALLOC-$-1 ;----- FAKE JRNZ INSTRUCTION

; TRACKS ARE THE SAME
;

LDA NEWSEC ;SAME SECTOR ??
LXI H,UNASEC ;
CMP M ;NEWSEC = UNASEC ??
JRNZ ALLOC ;SKIP IF NOT
DB 020H,ALLOC-$-1 ;----- FAKE JRNZ INSTRUCTION

; MATCH, MOVE TO NEXT SECTOR FOR FUTURE REFERENCE
;

INR M ;UNASEC = UNASEC + 1
MOV A,M ;END OF TRACK ??
CPI CPMSPT ;COUNT CP/M SECTORS
JRC NOOVF ;SKIP IF NO OVERFLOW
DB 038H,NOOVF-$-1 ;----- FAKE JRC INSTRUCTION

; OVERFLOW TO NEXT TRACK
;

MVI M,000H ;UNASEC = 0
LHLD UNATRK ;
INX H ;
SHLD UNATRK ;UNATRK = UNATRK + 1

; MATCH FOUND, MARK AS UNNECESSARY READ

All Information Presented Here is Proprietary to Digital Research

115
NOOVF:

03D4 AF  XRA A ;ZERO TO ACCUMULATOR
03D5 32F70A STA RSFLAG ;RSFLAG = 0
03D7+1808 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....
03E5 3AE90A LDA NEWSEC ;COMPUTE HOST SECTOR
03E9+B7 REPT SECSHF ;COMPUTE HOST SECTOR
03EA+1F ORA A ;CARRY = 0
03EB+B7 RAR ;SHIFT RIGHT
03EC+1F ORA A ;CARRY = 0
03ED+B7 RAR ;SHIFT RIGHT
03EE+1F ORA A ;CARRY = 0
03EF 32EE0A STA NEWHST ;HOST SECTOR TO SEEK

ACTIVE HOST SECTOR ??

03F2 21EF0A LXI H,HSTACT ;HOST ACTIVE FLAG
03F5 7E MOV A,M ;
03F6 3601 MVI M,001H ;ALWAYS BECOMES 1
03F8 B7 ORA A ;WAS IT ALREADY ??
03F9+2821 JRZ FILLHST ;FILL HOST IF NOT
03FA+2821 DB 028H,FILLHST-$-1 ;---- FAKE JRZ INST

HOST BUFFER ACTIVE, SAME AS SEEK BUFFER

All Information Presented Here is Proprietary to Digital Research 116
03FB 3AE60A LDA NEWDSK 
03FE 21BA0A LXI H,HSTDSK ;SAME DISK ??
0401 BE CMP M ;NEWDSK = HSTDSK ??
0402+2011 JRNZ NOMATCH 
DB 020H,NOMATCH-$-1 ;---- FAKE JRNZ INS

; SAME DISK, SAME TRACK ??

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

; SAME DISK, SAME TRACK, SAME BUFFER ??

040C 3AAE0A 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 3AAE0A 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 E607 ANI SECMSK ;LEAST SIGNIF BITS
043E 6F MOV L,A ;READY TO SHIFT

All Information Presented Here is Proprietary to Digital Research
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 ;

All Information Presented Here is Proprietary to Digital Research
WRITEHST:

046D 3E05 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 219B0E LXI H,HSTBUF-1 ;WRITE MUST WRITE CONTROL B
047A 22BC00 SHLD DMASA ;
JR HRW0 ;
047D+1810 DB 018H,HRW0-$-1 ;---- FAKE JR INSTRUCTION -

READHST:

047F 3E01 MVI A,001H ;SETUP DMA FOR READ
0481 32CE00 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$RETRIES ;SETUP TEMPORARY RETRIES CO
0494 3EFF MVI A,0FFH ;INIT TOGGLE SO THAT NO HOM
0496 32030B STA HOME$TOGGLE ;ALTERNATE RETRIES WILL BE
;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
04A2 CD5205 CALL DSKSEL ;SELECT CORRECT DRIVE FOR I
04A4 CDB305 CALL POINT ;POINT TO TRACK REGISTER SA
04A8 EB XCHG ;POINT TO SELECT MASK
04A9 3EF0 MVI A,11110000B ;TO REMOVE CURRENT HEAD SEL
04AB A6 ANA M ;
04AC 77 MOV M,A ;
04AD E5 PUSH H ;SAVE MASK ADDRESS
04AE CD3205 CALL SETHED ;COMPUTE CORRECT HEAD NUMBE
04B1 7D MOV A,L ;TRACK NUMBER AFTER HEAD CA

All Information Presented Here is Proprietary to Digital Research
MP/M II System Guide

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 HEA
04BF EB01 MVI C,1; DELAY FOR 1 MILLISECOND
04C1 CD8207 CALL DELAY;

04C4 CDB305 CALL POINT; IS A SEEK NECESSARY ??
04C7 3AA00 LDA TRAKNO; CHECK
04CA BE CMP M; WELL ??
04CB+2814 JRZ HRW5; NO SEEK NECESSARY...

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
HRW2:
04D9 CD1707 CALL WAIT0; AND ISSUE COMMAND
04DC EB14 MVI C,20; WAIT FOR I/O
04DE CD8207 CALL DELAY;

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

04E6 21BA00 LXI H,DMAS1; SETUP DMA FOR HARD DISK I/
04E9 010006 LXI B,0600H;
04EC+EDB3 OUTIR;
04EE 21C000 LXI H,DMAS2H;
04F1 010006 LXI B,0600H;
04F4+EDB3 OUTIR;
04F6 21CA00 LXI H,DMAS3;
04F9 010007 LXI B,0700H;
04FC+EDB3 OUTIR;
04FE 3AA00 LDA CMD; PICKUP I/O COMMAND
0501 CD2107 CALL INTFIX; CLEAR ANY PENDING INTERRUP
0504 CD1707 HRW7: CALL WAIT0; WAIT FOR COMPLETION
0507 3E5D MVI A,01011101B; SETUP STATUS AND MASK

All Information Presented Here is Proprietary to Digital Research
MP/M II System Guide

Appendix E MP/M Banked XIOS

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$TOGGLE ;CHANGE TOGGLE SO THAT HOME
0513 2F CMA ;
0514 32030B STA HOME$TOGGLE ;

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

Page

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

DBL$UPDATE:
0519 3AB600 LDA MPARMS ;CHECK FOR DOUBLE SIDED DRI
051C E601 ANI 1 ; IS FLAG SET
051E C8 RZ ; NO - SO RETURN
051F 3AAC00 LDA DISKNO ;CURRENT DISK DRIVE
0522 FE04 CPI 004H ;IS IT A FLOPPY
0524 D0 RNC ;NO, RETURN WITHOUT UPDATE
0525 E602 ANI 00000010B ;IS THIS DRIVE 2 OR 3 ??
0527 7E MOV A,M ;WE WERE CALLED WITH (HL) P
0528+2804 JRZ DBL$LOW ;IT MUST BE DRIVE ZERO OR 0
052A 2B DCX H ;BACKUP TO OTHER SIDE POINT
052B 2B DCX H ;
052C+1802 JR DBL$SAVE ;
DB 018H,DBL$SAVE-$-1 ;------ FAKE JR INSTR

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

SETED:

All Information Presented Here is Proprietary to Digital Research

121
MP/M II System Guide

Appendix E  MP/M Banked XIOS

0532 2AEB0A   LHLD   HSTRTRK   ;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-$-1   ; FAKE JRZ INSTRUCTION
053A E607     ANI     00000111B   ; GET TRACK MOD 8 (HEAD NUMB
053C 0E03     MVI     C,3     ; LIMIT LOOP FOR DIVIDE BY E
              JR      SETDVD;
053E+1804     DB      018H,SETDVD-$-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 F
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 HI
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
              ENDF

PAGE

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

DSKSEL:

0552 FE04     CPI     004H     ; IS IT HARD DISK ??
0554+3045     JRNC    SELHARD     ; YES, GO PROCESS....

SELHARD:

0556 21AC00    LXI     H,DISKNO     ; CURRENT DRIVE NUMBER
0559 BE       CMP     M     ; SAME DRIVE AS LAST TIME ??
              JRZ     SLS3     ; YES, DONT BOTHER WITH UNLO
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 S
; TO ENSURE THAT WHEN WE RETURN TO THIS DISK WE WILL
; LOAD AND WAIT FOR THE HEAD TO SETTLE.
;
;-----------------------------------------------------

All Information Presented Here is Proprietary to Digital Research

122
MP/M II System Guide

Appendix E MP/M Banked XIOS

SLS1:

055D DB04 IN 004H ;ENSURE FLOPPY PORT NOT BUS
055F 1F RAR ;
JRC SLS1 ;
0560+38FB DB 038H,SLS1-$-1 ;---- FAKE JRC INSTRUCTION
0562 DB05 IN 005H ;READ THE TRACK REGISTER
0564 D307 OUT 007H ;ENSURE WE DONT MOVE THE HE
0566 3E12 MVI A,012H ;SEEK AND UNLOAD HEAD
0568 CD6307 CALL FINTFIX ;CLEAR ANY PENDING INTERRUP
; AND ISSUE COMMAND
056B CD3A07 SLS2: CALL FPYWAIT ;WAIT HERE FOR INTERRUPT
056E 3AF0A LDA STATUS ;HOW DID THE I/O GO?
0571 E698 ANI 10011000B ; CHECK
JRNZ SLSERR ;EXIT IF ERROR
0573+2020 DB 020H,SLSERR-$-1 ;---- FAKE JRNC INSTRUCTION

;--------------------------------------------------------
; WE WILL NOW LOAD THE SELECT MASK AND SELECT THE DRI
; 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....
JRNCA SLS4 ;VERIFY ASSUMPTION
0580+3002 DB 030H,SLS4-$-1 ;---- FAKE JRC INSTRUCTION
0582 3EFFE MVI A,11111110B ;FORCE SINGLE DENSITY FOR 0

SLS4:

0584 A6 ANA M ;LOAD MASK AND CORRECT IF N
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 ADD
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

All Information Presented Here is Proprietary to Digital Research

123
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:

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

SLHL:

| 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 RE |
| 05A9 7E | MOV A,M ;LOAD OLD TRACK NUMBER |
| 05AA D322 | OUT 022H ;WRITE IT TO OLD TRACK REGI |
| 05AC 0E14 | MVI C,20 ;DELAY FOR 20 MILLISECONDS |
| 05AE CD8207 | CALL DELAY ; |
| 05B1 37 | STC ;SET CARRY TO SHOW HARD DIS |
| 05B2 C9 | ENDIF ;RETURN TO CALLER |

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 |
| 05C3 FE04 | CPI 4 ; |
| 05C5+380D | DB 038H,PNTFN-$-1 ; ----- FAKE JRC INSTRUCTION |
| 05C7 3B10 | MVI A,10H ; |
| 05C9 A6 | ANA M ; CHECK DRIVE SELECT |
| 05CA+2805 | JRZ PNTTH2 ; MUST BE DRIVE # 2 |

All Information Presented Here is Proprietary to Digital Research
MP/M II System Guide

Appendix E MP/M Banked XIOS

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

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

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 QU
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 ENTER
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
05F0+180F   JR      SRW1  ;
              DB      018H,SRW1-$-1  ;---- FAKE JR INSTRUCTION -

WRIT SOFT:
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

All Information Presented Here is Proprietary to Digital Research

125
MP/M II System Guide

Appendix E  MP/M Banked XIOS

SRW1:

0601 32FA0A STA CMD ;
0604 211D13 LXI H,FPHYBUF ;
0607 22BC00 SHLD DMASA ;
060A 3AE60A LDA NEWDSK ;
060D CD5205 CALL DSKSEL ;SELECT DRIVE FOR I/O
0610 3A60A 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$RETRIES ;SAVE FOR RETRY ROUTINE
061A AF STA A
061B 32030B STA HOME$TOGGLE ;FORCE HOME PRIOR TO EACH R

LOAD$HEAD:

061E DB08 IN 008H ;IS HEAD LOADED ??
0620 E602 ANI 0000010B ;CHECK IT....
0622+201F JRNZ REMOVE$LD ;YES, ITS LOADED, DONT RELO
0624 DB05 IN 005H ;DUMMY SEEK TO START HEAD L
0626 D307 OUT 007H ;KEEP IT SHORT....
0628 3E1A MVI A,01AH ;START HEAD LOADING
062A CD6307 CALL FINTFIX ;CLEAR ANY PENDING INTERRUP
0630 3E0A MVI A,10 ;SET NUMBER OF TRIALS
0632 E602 IN 008H ;IS HEAD LOADED ??
0634 CD3A07 CALL PPYWAIT ;WAIT FOR I/O TO COMPLETE
0636 3AFC0A LDA STATUS ;HOW DID IT GO?
0638 E698 ANI 10011000B ;CHECK
063A CD3A07 CALL CHECKIT ;DO NOT GO ON IF ERROR
063C 007H IKEEP IT SHORT ••••
063D+2044 DB 020H,CHECKIT-$-1 ;---- FAKE JRNZ INS

LDH1:

0641 0E10 MVI C,16 ;WAIT HERE FOR 16 MS
0643 CD8207 CALL DELAY ;CALL WAIT ROUTINE
0645 3DB305 CALL POINT ;REESTABLISH TRACK REGISTER
0646 36FE MVI M,254 ;ENSURE FURTHER SEEK AND DE
064F 1807 DB 018H,TRKTST-$-1 ;---- FAKE JR INSTRUCTION -

REMOVE$LD:

064A 21FA0A LXI H,CMD ;POINT TO I/O COMMAND
064B 3EBF MVI A,1111101B ;REMOVE HEAD LOAD BIT
064D 06 A6 CALL M ;DO IT....
064F 77 MOV M,A ;SAVE IT BACK INTO CMD

TRKTST:

0650 3DB305 CALL POINT ;RESTORE TRACK REGISTER POI
0652 3AE70A LDA NEWTRK ;GET NEW TRACK NUMBER
0655 32AD00 STA TRAKNO ;SAVE IN COMMON PLACE
0657 BE CMP M ;SAME AS LAST TIME ??
0657+281A JRZ FSECSET ;YES, DONT BOTHER WITH SEEK
0657 77 MOV M,A ;SAVE IT
0657 D307 OUT 007H ;ALSO SEND IT TO CONTROLLER

All Information Presented Here is Proprietary to Digital Research

126
All Information Presented Here is Proprietary to Digital Research

127
MP/M II System Guide

Appendix E MP/M Banked XIOS

0687 21BA00 IF DMA
0688 010006 LXI H,DMAS1 ;INITIALIZE DMA
068D+EDB3 OUTIR ;WRITE TO DMA
068F 21C600 DB 0EDH,0B3H ;---- FAKE OTIR INSTRUCTION
0692 010004 LXI H,DMAS2F ;
0695+EDB3 OUTIR ;WRITE TO DMA
0697 21CA00 DB 0EDH,0B3H ;---- FAKE OTIR INSTRUCTION
069A 010007 LXI B,0700H ;
069D+EDB3 OUTIR ;WRITE TO DMA
069F 0E07 DB 0EDH,0B3H ;---- FAKE OTIR INSTRUCTION
06A1 211D13 LXI H,FPYBUF ;DMA ADDRESS
06A4 3AFA0A LDA CMD ;I/O COMMAND
06A7 CD6307 CALL FINTFIX ;CLEAR ANY PENDING INTERRUPT
06AA CD3A07 FWT1: CALL FPYWAIT ;WAIT HERE FOR I/O TO COMPLETE
06AD C9 IF NOT DMA
06B1 3600 LXI H,SAVE1 ;SETUP TO REPLACE DATA
06B3 21FC00 LXI D,066H ;COPIED FROM NMI LOCATION
06B6 3AFB0A LXI B,004H ;
06B9 A6 LDIR ;MOVE IT....
06BB 77 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 REMO
06B9 A6 ANA M ;
06BA 77 MOV M,A ;SAVE CLEANED STATUS
06BB C8 RZ ;OK, SO RETURN

CHKS0:

06BC CDEE02 CALL RETMOD ;
06BF FEO3 CPI 003H ;HARD DISK ??
06C1 21FC0A LXI H,STATUS ;
06C4 7E MOV A,M ;RELOAD STATUS BYTE
JRNC CHKS2 ;YES, CHECK FOR DRIVE READY

All Information Presented Here is Proprietary to Digital Research

128
MP/M II System Guide

Appendix E MP/M Banked XIOS

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 RETR
06C9+283D 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 HOMES$TOGGLE ;IS A HOME NEEDED ON THIS R
06E9 B7 ORA A ;
06EA+200B JRNZ CHKS4 ;
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:

All Information Presented Here is Proprietary to Digital Research

129
System Appendix E MP/M Banked XIOS

PAGE

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 ;

// 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 I
;
IT APPEARS THAT WE OCCASIONALLY GET FLAG SET AS A R
;
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
;
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

All Information Presented Here is Proprietary to Digital Research
0735 C1  POP  B  ;RESTORE REGISTERS
0736 F1  POP  PSW
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 3AD00D LDA  FPYTIME  ;DID WD1791 GO TO SLEEP?
0747 B7  ORA  A
0748+2015 JRNZ  NOPYRST  ;IF STILL AWAKE, SKIP RESET
0749+2015 DB  020H,NOPYRST-$-1  ;---- FAKE JRNZ INS
074A DB09 IN  009H  ;GET CURRENT BANK NUMBER
074C E618 ANI  00011000B  ;REMOVE OTHER INFO
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

NOPYRST:
075F F1  POP  PSW
0760 E1  POP  H
0761 C1  POP  B  ;RESTORE RETRY COUNT IN <C>
0762 C9  RET

; THE FOLLOWING CODE GUARANTEES THAT FLOPPY DISK FLAG
;

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

All Information Presented Here is Proprietary to Digital Research
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

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

077F D304 OUT 004H ;ISSUE COMMAND TO FLOPPY DI
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 TIE
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
;*************************************************************************
### DISK CONFIGURATION TABLE

```assembly
<table>
<thead>
<tr>
<th>ADDRESS (hex)</th>
<th>CONFIGURATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0790</td>
<td>DB 00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H</td>
</tr>
<tr>
<td>0798</td>
<td>DB 10H,00H,00H,00H,00H,00H,00H,00H,10H,00H,00H,00H,00H,00H,00H,00H</td>
</tr>
<tr>
<td>07A0</td>
<td>DB 90H,90H,90H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H</td>
</tr>
<tr>
<td>07A8</td>
<td>DB 00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H</td>
</tr>
<tr>
<td>07B0</td>
<td>DB 10H,00H,00H,00H,20H,00H,00H,00H,10H,20H</td>
</tr>
<tr>
<td>07B8</td>
<td>DB 00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H</td>
</tr>
<tr>
<td>07C0</td>
<td>DB 90H,90H,90H,20H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H</td>
</tr>
<tr>
<td>07C8</td>
<td>DB 90H,90H,90H,0A0H,0A0H,0A0H,0A0H,00H,00H,00H</td>
</tr>
</tbody>
</table>
```

### SET UP DISK CONFIGURATION

```
[ THIS CODE EXECUTED ONLY ONCE ]
```

```assembly
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 ;
07DE 118000 SDDBL: LXI D,SEL0+4 ;POINT TO DRIVE E:
07E1 DB25 IF HARDSK ;READ CONFIGURATION PORT
07E3 E607 IN 025H ;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 SDLL: MOV A,M ;CHANGE ALL SELECT MASKS
07F2 12 STAX D ;
07F3 13 SDOK: INX D ;NEXT
```

All Information Presented Here is Proprietary to Digital Research

133
07F4 23  INX  H  ; DRIVE
   DJNZ  SDL1  ;
07F5+10FA  DB  010H,SDL1-$-1  ;---- FAKE DJNZ INSTRUCTION
   ENDF  
   IF  NOT HARDSK
   XCHG
   MVI  B,8  ;
   XRA  A  ;
   SDL2:  MOV  M,A  ; ZAP ALL HARD DRIVES
   INX  H  ;
   DJNZ  SDL2  ;
   ENDF  
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 MOD
0801 CD7802  XETSEL:  CALL  SETDEN  ; SET DENSITY BASED ON LOW B
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,MODLO  ; FIRST MODEL DPE
0812 19  DAD  D  ; POINT TO THIS ONE
0813 EB  XCHG  ; SETUP TEMPORARILY AS DESTI
0814 2AB200  LHLD  DPEPTR  ; ADDRESS OF CURRENTLY SELECTED
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

-----------------------------
THE FOLLOWING AREA CONTAINS THE DISK/WORK SAVE AREA
USED BY THE CB IOS IN THE NORMAL COURSE OF ACTIVITY.
-----------------------------

All Information Presented Here is Proprietary to Digital Research

134
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
099E
ALV5: DS 64
099E
CSV5: DS 0
099E
ALV6: DS 64
099E
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
CSV9: DS 0
0AC2
ALVB: DS 36
0AE6
CSVB: DS 0
endif

if mdisk

ALVC: DS 32
CSVC: DS 0
;if not mpm20
endif
if not mpm20
if hardsk
DS 1
HSTBUF: DS 1024
ENDIF

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

0AE6
NEWDSK: DS 1
0AE7
NEWTRK: DS 2
0AE9
NEWSEC: DS 1

All Information Presented Here is Proprietary to Digital Research

135
## MP/M II System Guide

### Appendix E MP/M Banked XIOS

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00AE0</td>
<td>HSTDSK: DS</td>
<td>1</td>
</tr>
<tr>
<td>00AE1</td>
<td>HSTTRK: DS</td>
<td>2</td>
</tr>
<tr>
<td>00AE2</td>
<td>HSTSEC: DS</td>
<td>1</td>
</tr>
<tr>
<td>00AE3</td>
<td>NEWHST: DS</td>
<td>1</td>
</tr>
<tr>
<td>00AE4</td>
<td>HSTACT: DS</td>
<td>1</td>
</tr>
<tr>
<td>00AE5</td>
<td>HSTWRT: DS</td>
<td>1</td>
</tr>
<tr>
<td>00AE6</td>
<td>UNACNT: DS</td>
<td>1</td>
</tr>
<tr>
<td>00AE7</td>
<td>UNADSK: DS</td>
<td>1</td>
</tr>
<tr>
<td>00AE8</td>
<td>UNATRKR: DS</td>
<td>2</td>
</tr>
<tr>
<td>00AE9</td>
<td>UNASEC: DS</td>
<td>1</td>
</tr>
<tr>
<td>00AEF</td>
<td>ERFLAG: DS</td>
<td>1</td>
</tr>
<tr>
<td>00AF0</td>
<td>RSFLAG: DS</td>
<td>1</td>
</tr>
<tr>
<td>00AF1</td>
<td>READOP: DS</td>
<td>1</td>
</tr>
<tr>
<td>00AF2</td>
<td>WRTYPE: DS</td>
<td>1</td>
</tr>
<tr>
<td>00AF3</td>
<td>CMD: DB</td>
<td>0</td>
</tr>
<tr>
<td>00AF4</td>
<td>MASK: DB</td>
<td>0</td>
</tr>
<tr>
<td>00AF5</td>
<td>STATUS: DB</td>
<td>0</td>
</tr>
<tr>
<td>00AF6</td>
<td>SAVE1: DB</td>
<td>0000H</td>
</tr>
<tr>
<td>00AF7</td>
<td>P$RETRIES: DB</td>
<td>0000H</td>
</tr>
<tr>
<td>00AF8</td>
<td>T$RETRIES: DB</td>
<td>0000H</td>
</tr>
<tr>
<td>00AF9</td>
<td>HOME$TOGGLE: DB</td>
<td>0000H</td>
</tr>
<tr>
<td>0100</td>
<td>COLDSTART:</td>
<td>$-$$</td>
</tr>
<tr>
<td>0107</td>
<td>WARMSTART:</td>
<td>$-$$</td>
</tr>
<tr>
<td>010B</td>
<td>xdos:</td>
<td>$-$$</td>
</tr>
<tr>
<td>010C</td>
<td>sysdat:</td>
<td>$-$$</td>
</tr>
</tbody>
</table>

```assembly

commonbase:

jmp  coldstart
jmp  $-$$
jmp  $-$$
jmp  $-$$
jmp  $-$$
jmp  $-$$

COLDSTART:

MVI  C,0

JMP  XDOS

```

---

All Information Presented Here is Proprietary to Digital Research

136
MP/M II System Guide  Appendix E  MP/M Banked XIOS

rtnempty:
0B1A AF  xra a
0B1B C9  ret

NULL$INT:
0B1C FB  EI
          RETI
0B1D+ED4D  DB  OEDH,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 CD10B  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 B00  MVI E,PLLPT ; PRINTER
0B37 CD10B  CALL XDOS ;WAIT FOR PRINTER TO FREE U
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

All Information Presented Here is Proprietary to Digital Research

137
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,FPPYFLAG ;SHOW I/O COMPLETED
JR HDSTFLG
0B5C+1813 DB 018H,HDSTFLG-$-1 ;---- FAKE JR INSTR

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

HDINTH:
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 RELOADI
0B6F 1E05 MVI E,HDFLAG ;SHOW I/O COMPLETED

HDSTFLG:
0B71 0E85 MVI C,FLAGST
0B73 CD100B CALL XDOS
0B76 C3670D JMP INTDONE

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

All Information Presented Here is Proprietary to Digital Research
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 HANDLER
; D = CONSOLE #
; DO NOT DESTROY <D>

0B9A 7A MOV A, D
0B9B FE04 CPI NMBCN5
0B9D+3803 JRC TBLJMP
0B9F F1 DB 038H, TBLJMP-5-1 ; ---- FAKE JRC INSTRUCTION
0BA0 AF POP PSW ; THROW AWAY TABLE ADDRESS
0BA1 C9 XRA A

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

;-----------------------------------------------------
; 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 = LPTPRTO EQU 01EH ; PRINTER #0 DATA
001F = LPTSTS0 EQU LPTPRTO+1 ; PRINTER #0 STATUS
0028 = LPTPRT1 EQU 028H ; PRINTER #1 DATA
0029 = LPTSTS1 EQU LPTPRT1+1 ; PRINTER #1 STATUS

All Information Presented Here is Proprietary to Digital Research
;-----------------------------
;
;   POLL CONSOLE # 0 INPUT
;
;-----------------------------

POLCIO:

PTOST:

XRA A ; TEST CONSOLE STATUS
OUT STS0 ; RETURN OFFH IF READY
IN STS0 ; 000H IF NOT

ANI 1 ; RX CHAR ?
RZ ; NO

MVI A, OFFH ; YES - SET FLAG
RET

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

; CONSOLE # 0 INPUT
;
;-----------------------------

PTOIN:

CALL POLCIO ; RETURN CHAR IN REG A
ORA A ; IS IT READY NOW?
JRNZ PT0IN1 ; IF READY, SKIP POLL
DB 020H,PT0IN1-$-1 ;----- FAKE JRNZ INSTRUCTION

MVI C,POLL ; POLL CONSOLE #0 INPUT
MVI E,PLCIO
CALL XDOS

IN DATA0 ; READ CHARACTER
ANI 7FH ; STRIP PARITY
RET

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

; CONSOLE # 0 OUTPUT
;
;-----------------------------

PT0OUT:

CALL POLCO0 ; REG C = CHAR TO OUTPUT
ORA A ; IS IT READY NOW?
JRNZ PT0OUT1 ; IF READY, SKIP POLL
DB 020H,PT0OUT1-$-1 ;----- FAKE JRNZ INS

MVI C,POLL
MVI E,PLCIO
CALL XDOS ; POLL CONSOLE #0 OUTPUT
POP B

PT0OUT1:

MOV A,C
OUT DATA0 ; TRANSMIT CHARACTER

All Information Presented Here is Proprietary to Digital Research
MP/M II System Guide

Appendix E MP/M Banked XIOS

0BDC C9 RET
;
;
---------------------------------------------------------------------
;
POLL CONSOLE # 0 OUTPUT
;
---------------------------------------------------------------------
;
POLCO0:
0BDD 3E10 MVI A,10H ; RETURN OFFH IF READY
0BDF D31D OUT STS0 ; 000H IF NOT
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 OFFH
0BEB C9 RET ; RETURN READY

PAGE

---------------------------------------------------------------------
;
POLL CONSOLE # 1 INPUT
;
---------------------------------------------------------------------
;
POLC11:
PT1ST:
0BEC AF XRA A ; TEST CONSOLE STATUS
0BED D32D OUT STS1 ; RETURN OFFH IF READY
0BEF DB2D IN STS1 ; 000H IF NOT
0BF1 E601 ANI 1 ; RX CHAR ?
0BF3 C8 RZ ; NO
0BF4 3EFF MVI A,OFFH ; YES - SET FLAG
0BF6 C9 RET ;

---------------------------------------------------------------------
;
CONSOLE # 1 INPUT
;
---------------------------------------------------------------------
;
PT1IN:
0BF7 CDE00B CALL POLC11 ; RETURN CHAR IN REG A
0BFA B7 ORA A ; READY NOW?
0BF8+2007 JRNZ PT1IN1 ; IF READY, SKIP POLL
0BF9 0E83 DB 020H,PT1IN1-$-1 ; ---- FAKE JRNZ INSTRUCTION
0BF0 1806 MVI C,POLL ;
0BF1 0C01 DB100B MVI E,PLC11 ; POLL CONSOLE #1 INPUT
0BF3 C8 0C04 DB2C CALL XDOS ;
0BF5 0C06 E67F PT1IN1: IN DATAL ; READ CHARACTER
0BF7 0C07 7FH ; STRIP PARITY

All Information Presented Here is Proprietary to Digital Research

141
MP/M II System Guide

Appendix E  MP/M Banked XIOS

0C08 C9  RET
;
;
-----------------------------------------------------
;
CONSOLE # 1 OUTPUT
;
-----------------------------------------------------
;
PT1OUT: ; REG C = CHAR TO OUTPUT
0C09 CD1C0C  CALL  POLCOL  ; ARE WE READY NOW?
0C0C B7  ORA A
0C0D+2009  JRNZ PT1OUT1  ; IF READY, SKIP POLL
0C0F C5  DB 020H,PT1OUT1-$-1  ; ---- FAKE JRNZ INS
0C10 0E83  PUSH B
0C12 1E02  MVI C,POLL
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
;
-----------------------------------------------------
;
POLCOL: ; 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 6E0C  ANI 0CH  ; MASK FOR DTR AND TXE
0C24 FE0C  CPI 0CH  ; MUST HAVE BOTH
0C26 3E00  MVI A,0
0C28 C0  RNZ
0C29 3D  DCR A  ; CHANGE "A" TO 0FFH
0C2A C9  RET  ; RETURN READY

PAGE

-----------------------------------------------------
;
POLL CONSOLE # 2 INPUT
;
-----------------------------------------------------
;
POLC12:
PT2ST: ; TEST CONSOLE STATUS
0C2B AF  XRA A  ; RETURN 0FFH IF READY

All Information Presented Here is Proprietary to Digital Research

142
CONSOLE # 2 INPUT

; RETURN CHAR IN REG A

0C36 CD2BOC       CALL POLCI2 ; READY NOW?
0C39 B7           ORA A ;
0C3A+2007          JRNZ PT2IN1 ; IF READY, SKIP POLL
0C3C 0E83         DB 020H,PT2IN1-$-1 ;----- FAKE JRNZ INSTRUCTION
0C3E 1E07         MVI E,PCLI2 ; POLL CONSOLE #2 INPUT
0C40 CD100B   CALL XDOS ;
0C43 DB2E     PT2IN1: IN DATA2 ; READ CHARACTER
0C45 B67F       ANI 7FH ; STRIP PARITY
0C47 C9         RET ;

CONSOLE # 2 OUTPUT

; REG C = CHAR TO OUTPUT

0C48 CD5BOC       CALL POLCO2 ; READY NOW?
0C4B B7           ORA A ;
0C4C+2009          JRNZ PT2OUT1 ; IF READY, SKIP POLL
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 ;
0C57 79          MOV A,C ;
0C58 D32E       OUT DATA2 ; TRANSMIT CHARACTER
0C5A C9         RET ;

POLL CONSOLE # 2 OUTPUT

POLCO2: ; RETURN OFFH IF READY
MP/M II System Guide

Appendix E MP/M Banked XIOS

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 OFFH
0C69 C9 RET ; RETURN READY

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

POLCI3:
PT3ST: ; TEST CONSOLE STATUS

0C6A AP XRA A ; RETURN OFFH 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,OFFH ; 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 ;
0C79+2007 JRNZ PT3IN1 ; IF READY, SKIP POLL
0C7B 0E83 DB 020H,PT3IN1-$-1 ;---- FAKE JRNZ INSTRUCTION
0C7D 1E08 MVI C,POLL ;
0C7F CD100B MVI E,PLCI3 ; POLL CONSOLE #3 INPUT
0C82 DB2A CALL XDOS ;
0C84 B67F PT3IN1 : IN DATA3 ; READ CHARACTER
0C86 C9 ANI 7FH ; STRIP PARITY

-----------------
;
;
CONSOLE # 3 OUTPUT
;
-----------------

All Information Presented Here is Proprietary to Digital Research
PT3OUT:

CALL POLCO3 ; REG C = CHAR TO OUTPUT
ORA A
JRNZ PT3OUT1 ; READY NOW?

DB 020H,PT3OUT1-$-1 ; IF READY, SKIP POLL

PUSH B
MV1 C,POLL
MV1 E,PLOCO3
CALL XDOS ; POLL CONSOLE #3 OUTPUT
POP B

PT3OUT1:

MOV A,C
OUT DATA3 ; TRANSMIT CHARACTER

POLL CONSOLE #3 OUTPUT

POLCO3:

MVI A,10H ; RETURN 0FFH IF READY
OUT STS3 ; 000H IF NOT
IN STS3 ; RESET INT BIT

ANI 0CH ; READ STATUS
CPI 0CH ; MASK FOR DTR AND TXE

MVI A,0 ; MUST HAVE BOTH
RNZ ; RETURN NOT READY
OCR A ; CHANGE "A" TO OFFH
RET ; RETURN READY

PAGE

LINE PRINTER #0 DRIVER

LIST:

CALL POLLPT ; LIST OUTPUT #0
ORA A ; IS PRINTER READY NOW?
JRNZ LIST1 ; IF READY, SKIP POLL

DB 020H,LIST1-$-1 ; FAKE JRNZ INSTRUCTION

PUSH B
MV1 C,POLL ; POLL PRINTER STATUS
MV1 E,PLLPT
CALL XDOS
POP B

All Information Presented Here is Proprietary to Digital Research
LIST1:

0CB8 79 MOV A,C ; CHARACTER TO PRINT
0CB9 D31E OUT LPTPRT0 ;
0CBB C9 RET ;

; POLL PRINTER OUTPUT ;

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

POLLPT: ; RETURN OFFH 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 OFFH
0CCA C9 RET ; RETURN READY

; PAGE

; MP/M 1.0 EXTENDED I/O SYSTEM

;

POLLDEVICE: ; REG C = DEVICE # TO BE POLLED

0CCB 79 MOV A,C ; RETURN OFFH IF READY,
0CCC FE09 CPI NMBDEV ; 000H IF NOT
0CCE+3802 JRC DEVOK
0CD0 3E09 DB 038H,DEVOK-$-1 ;----- FAKE JRC INSTRUCTION

DEVOK: CALL TBLJMP ; JUMP TO DEV POLL CODE

DEVTBL: DW POLLPT ; POLL PRINTER OUTPUT - THIS WILL P

; SPECIFIED PARALLEL PORT FOR PRIN

0CD5 BC0C DW POLCO0 ; POLL CONSOLE #0 OUTPUT
0CD7 DD0B DW POLCO1 ; POLL CONSOLE #1 OUTPUT
0CD9 1C0C DW POLCO2 ; POLL CONSOLE #2 OUTPUT
0CDB 5B0C DW POLCO3 ; POLL CONSOLE #3 OUTPUT
0CDD 9A0C DW POLCI0 ; POLL CONSOLE #0 INPUT
0CDF AD0B DW POLCI1 ; POLL CONSOLE #1 INPUT
0CE1 EC0B DW POLCI2 ; POLL CONSOLE #2 INPUT
0CE3 2B0C DW POLCI3 ; POLL CONSOLE #3 INPUT
0CE5 6A0C NMBDEV EQU ($-DEVTBL)/2

All Information Presented Here is Proprietary to Digital Research
MP/M II System Guide

Appendix E MP/M Banked XIOS

OCE7 1A0B DW RTNEMPTY; BAD DEVICE HANDLER

PAGE

; SELECT / PROTECT MEMORY

SELMEMORY:

; SELMEMORY:
; REG BC = ADR OF MEM DESCRIPTOR
; BC -> BASE 1 BYTE,
; SIZE 1 BYTE,
; ATTRIB 1 BYTE,
; BANK 1 BYTE.
; BIOS TABLE MODIFIED

OCE9 FE20 CPI 20H ;
OCEB CAEB0C JZ $ ;
OCEE 210300 LXI H,3 ; POINT TO BANK
OCF1 09 DAD B ;
OCF2 7E MOV A,M ; GET IT
OCF3 32030D STA BANKNO ; SAVE BANK NUMBER
OCF6 17 RAL ;
OCF7 17 RAL ;
OCF8 17 RAL ;
OCF9 E618 ANI 018H ; MASK FOR PIO
OCFB F602 ORI MEMSK ;
OCFD 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 3EFE MVI A, OFFH
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

All Information Presented Here is Proprietary to Digital Research

147
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-$-1 ;---- FAKE JRNZ INS
0D35 3E7D MVI A,125
0D37 2B DCX H
0D38 96 SUB M
0D39 77 MOV M,A ; *** TOGGLE COUNT 62 <-> 6
0D3A 23 INX H
0D3B 77 MOV M,A ; *** ACTUAL #/SEC = 62.5
0D3C 0E85 MVI C,FLAGST
0D3E 1E02 MVI E,2
0D40 CD100B CALL XDOS ; SET FLAG #2 @ 1 SEC
0D43 2AD00D LHLD FPYTIME ; IS FLOPPY TIME CHECK IN EF
0D46 7C MOV A,H ;
0D47 B7 ORA A ;
JRZ NOT1SEC ; IF NOT IN EFFECT, FINISH
0D48+281D DB 028H,NOT1SEC-$-1 ;---- FAKE JRZ INST
0D4A 2D DCR L ; SUBTRACT A SECOND
All Information Presented Here is Proprietary to Digital Research
PAGE

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

0DDB 0000  
0DDB =  
DPB0: EQU $                  ;VERSION 2.0, IBM SINGLE DE
0DDB 1A00  
0DDB DW  26                        ;SECTORS PER TRACK
0DDB 03   
0DDB DB  3                          ;BLOCK SHIFT
0DDB 07   
0DDB DB  7                          ;BLOCK SHIFT MASK
0DDB 00   
0DDB DB  0                          ;EXTENT MASK
0DDB F200  
0DDB DW  242                        ;DISK SIZE MINUS 1
0DDB 3F00  
0DDB DW  63                          ;DIRECTORY MAX
0DDB C0   
0DDB DB  192                          ;ALLOC0
0DDE 00   
0DDE DB  0                          ;ALLOC1
0DDE 1000  
0DDE DW  16                          ;CHECK AREA SIZE
0DDE 0200  
0DDE DW  2                            ;OFFSET TO START TRACK

0DE3 =  
DPB1: EQU $                  ;VERSION 2.0, IBM DOUBLE DE
0DE3 3400  
0DE3 DW  52                        ;SECTORS PER TRACK
0DE5 04   
0DE5 DB  4                          ;BLOCK SHIFT

All Information Presented Here is Proprietary to Digital Research
```plaintext
0DE6_0F        DB 15 ; BLOCK SHIFT MASK
0DE7_01        DB 1  ; EXTENT MASK
0DE8_F200      DW 242 ; DISK SIZE MINUS 1
0DEA_7F00      DW 127 ; DIRECTORY MAX
0DEC_C0        DB 192 ; ALLOC0
0DED_00        DB 0  ; ALLOC1
0DEE_2000      DW 32  ; CHECK AREA SIZE
0DF0_0200      DW 2  ; OFFSET TO START TRACK

0DF2  =  DPB2:  EQU $  ; VERSION 1.4 ALTOS DOUBLE D
0DF2_3000      DW 48  ; SECTORS PER TRACK
0DF4_04        DB 4  ; BLOCK SHIFT
0DF5_0F        DB 15 ; BLOCK SHIFT MASK
0DF6_00        DB 0  ; EXTENT MASK (1.4 COMPATABLE
0DF7_E000      DW 224 ; DISK SIZE MINUS 1
0DF9_5F00      DW 95  ; DIRECTORY MAX
0DFB_C0        DB 192 ; ALLOC0
0DFC_00        DB 0  ; ALLOC1
0DFD_1800      DW 24  ; CHECK AREA SIZE
0DFF_0200      DW 2  ; OFFSET TO START TRACK

IF HARDSK

if mpm20

0E01_+=  DPB3:  DISKDEF 3,0,127,,16384,512,512,0,1,,0
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_FF00     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 WIT
0E0E_0100     DW 1  ; OFFSET
0000_+        XLT3 EQU 0  ; NO XLTATE TABLE

0E10_+=  DPB4:  DISKDEF 4,0,127,,16384,512,512,0,513,,0
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 WIT
0E1D_0102     DW 513 ; OFFSET
0000_+        XLT4 EQU 0  ; NO XLTATE TABLE

0E1F_+=  DPB5:  DISKDEF 5,0,127,,16384,512,512,0,1025,,0
0E1F_+8000    DW 128 ; SEC PER TRACK

All Information Presented Here is Proprietary to Digital Research

151
```
MP/M II System Guide

Appendix E MP/M Banked XIOS

MOVE SUBROUTINE

All Information Presented Here is Proprietary to Digital Research
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 ??

0E4F E601 if mpm20
        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 ACTIVIT
0E56 B7    ORA A ;ERRORS ??
0E57+2007 JRNZ RWEND ;SKIP IF SO....
0E59 AF    DB 020H,RWEND-$-1 ;----- FAKE JRNZ INSTRUCTION
0E5A 32F00A else
        XRA A ;ZERO TO ACCUMULATOR
        JRNZ RWEND ;NO, JUST END UP HERE
endif

0E5D CD6D04 CALL WRITEHST ;

RWEND:

0E60 3AF60A LDA ERFLAG ;
0E63 B7    ORA A ;IF ERRORS, RESET SO NO MAT
0E64 C8    RZ ;NONE, JUST RETURN
0E65 21EA0A LXI H,HSTDSK ;
0E68 36FF   MVI M,OFFH ;CANT POSSIBLY MATCH, MUST
0E6A C9    ENDF
0E6B 2AA0F00 RET ;

MVDTB:

0E6B 2AA0F00 LHLH DMAADR ; MOVE DATA TO FLOPPY BUFFE
0E6E E5    push h
0E6F CD070B call swtuser ;switch in user bank,
0E72 E1    pop h ; cannot access non-common BNKXIO
0E73 111D13 LXI D,FPYBUF ;
0E76 018000 LXI B,128 ; 128 BYTES

All Information Presented Here is Proprietary to Digital Research

153
MP/M II System Guide

Appendix E MP/M Banked XIOS

All Information Presented Here is Proprietary to Digital Research

154
12A6 226013     SHLD     SVDBPA
  if not mdisk
12A9 2A130B     lhl     sysdat
12AC 2E0F       mvi     1,15 ; 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 directo
  out    09h
  lxi    h,0bffe
  mvi    a,0e5h
  cmp    m
  inx    h
  jrnz   fill
  cmp    m
dontfill:
  jrz    dontfill
  fill:
  mov    m,a ; set directory initialized
  dcx    h
  mov    m,a
  lxi    b,07ff
  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
12D6 117C00     LXI    H,1737H ; MOVE PARAMETERS CHANGED B
                 LXI    D,SEL0 ; THE SETUP PROGRAM

All Information Presented Here is Proprietary to Digital Research

155
Appendix E  MP/M Banked XIOS

```
12D9 010400  LXI B,4 ; 4 SELECT MASKS
       LDIR ;
12DC+EDB0  DB 0EDH,0B0H ;---- FAKE LDIR INSTRUCTION
12DE 118800  LXI D,MODE ;
12E1 010400  LXI B,4 ; 4 MODE BYTES
       LDIR ;
12E4+EDB0  DB 0EDH,0B0H ;---- FAKE LDIR INSTRUCTION
12E6 2ABB17  lhl d lrdbiosbase+misc$params$offset
            LHLD 17BBH ; GET MISC. PARAMETERS
12E9 22B600  SHLD MPARMS ;
12EC 3AB600  LDA MPARMS ; NOW TEST FOR CENTRONICS P
12EF E602  ANI 2 ;
12F1+2814  DB 028H,PRTOK-$-1 ;---- FAKE JRZ INSTRUCTION
12F3 21C0B  LXI H,CLIST ;
12F6 221000  SHLD WBOTE+13 ; CHANGE PRINTER ROUTINE
12F9 21F00B  LXI H,CNSTAT ; AND STATUS CHECK
12FC 22D50C  SHLD DEVTLB ;
12FF 3E03  MVI A,003H ;INITIALIZE PARALLEL PORT
1301 D313  OUT 013H ;
1303 3E0F  MVI A,00FH ;
1305 D313  OUT 013H ;

PRTOK:
1307 010300  LXI B,003H ;SET THE MODE FOR DRIVES IN
   MODESET:
130A CD2F02  CALL SELSDP ;SELECT DRIVE FOR MODESET
130D 218000  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 MODE
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 SEDDMA ;SET DMA ADDRESS
1325 E5  push h

     if mpm20
1326 2A130B  lhl d sysdat
1329 2E07  mvi l,7
132B 7E  mov a,m
else
132C E1  pop h
132D ED47  DB 0EDH,047H ;---- FAKE STAI INSTRUCTION
```

All Information Presented Here is Proprietary to Digital Research

156
MP/M II System Guide

Appendix E  MP/M Banked XIOS

132F 3E60  MVI A,60H ; SET VECTOR FOR CTC
1331 D330  OUT 30H ; CTC CHANNEL 0
1333 3EA7  MVI A,0A7H ; RESET / LOAD TIME CONST
1335 D333  OUT 33H ; CHANNEL 3
1337 3EFA  MVI A,250 ; TIME CONSTANT
1339 D333  OUT 033H ;

133B AF  IF HARDSK
133C 32EF0A  STA HSTACT ;ZERO ACCUMULATOR
133F 32F10A  STA UNACNT ;SET HOST BUFFER INACTIVE
1342 219B0E  XRA A ;SET UNALLOCATED COUNT TO 2
1345 360D  STA HSTBUF-1 ; SETUP WRITE CONTROL BYTE F
1347 C9  ENDIF

1348 D309  OUT MEMPORT
134A 3EC3  STA 0 ; JMP INSTRUCTION
134C 320000  LHLDSVDJT ;
134F 2A5E13  SHLD 1
1352 220100  LHLD SVDJP A
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 12CE AFTERBANKS 03DA ALLOC
081E ALV0 085E ALV1 089E ALV2 08DE ALV3
091E ALV4 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 BLK512 067B CHECKIT 06AE CHECKSTAT
0220 CHKHRD 06BC CHKS0 06C7 CHKS1 06CD CHKS2

All Information Presented Here is Proprietary to Digital Research

157
All Information Presented Here is Proprietary to Digital Research
<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP/M II System Guide</td>
<td>Appendix E MP/M Banked XIOS</td>
<td></td>
</tr>
<tr>
<td>0C57 PT2OUT1</td>
<td>0C2B PT2ST</td>
<td>0C75 PT3IN</td>
</tr>
<tr>
<td>0C87 PT3OUT</td>
<td>0C96 PT3OUT1</td>
<td>0C6A PT3ST</td>
</tr>
<tr>
<td>028B READ</td>
<td>036B READHARD</td>
<td>047F READHST</td>
</tr>
<tr>
<td>05E4 READSOFT</td>
<td>0308 REALDISK</td>
<td>FFFF RELOC</td>
</tr>
<tr>
<td>02EE RETMOD</td>
<td>0AF7 RSFLAG</td>
<td>0B1A RTNEMPTY</td>
</tr>
<tr>
<td>0E3D RWMOVE</td>
<td>03E2 RWOPER</td>
<td>0AFD SAVE1</td>
</tr>
<tr>
<td>07DE SDDBL</td>
<td>07F1 SDL1</td>
<td>07F3 SDOK</td>
</tr>
<tr>
<td>0003 SECISHF</td>
<td>00B1 SECTNO</td>
<td>05D6 SECTRAN</td>
</tr>
<tr>
<td>0203 SELDSK</td>
<td>0242 SELERR</td>
<td>059B SELHARD</td>
</tr>
<tr>
<td>022F SELSDP</td>
<td>0556 SELSOFT</td>
<td>0278 SETDEN</td>
</tr>
<tr>
<td>0211 SETDSK</td>
<td>0544 SETDVD</td>
<td>0540 SETH14</td>
</tr>
<tr>
<td>02A1 SETMOD</td>
<td>0273 SETSEC</td>
<td>02C7 SETSEL</td>
</tr>
<tr>
<td>0547 SHD1</td>
<td>05A1 SLH1</td>
<td>055D SLS1</td>
</tr>
<tr>
<td>0575 SLS3</td>
<td>0584 SLS4</td>
<td>0595 SLSERR</td>
</tr>
<tr>
<td>0601 SNW1</td>
<td>0615 SRW2</td>
<td>0D05 STARTCLOCK</td>
</tr>
<tr>
<td>1348 STMVTR</td>
<td>0D0B STOPCLOCK</td>
<td>05DB STRN1</td>
</tr>
<tr>
<td>001D STS0</td>
<td>002D STS1</td>
<td>002F STS2</td>
</tr>
<tr>
<td>1360 SVDBPA</td>
<td>0DC8 SVDHL</td>
<td>135E SVDJT</td>
</tr>
<tr>
<td>0DCA SVDTP</td>
<td>080A SWT3SYS</td>
<td>0807 SWTUSER</td>
</tr>
<tr>
<td>129D SYSTEMINIT</td>
<td>0D1A T20MS</td>
<td>0BA2 TBLJMP</td>
</tr>
<tr>
<td>12AF TESTBANKSE</td>
<td>0DCE TICKN</td>
<td>0D22 TIMERINT</td>
</tr>
<tr>
<td>00AD TRAKNO</td>
<td>0B02 TRETries</td>
<td>0070 TRK0</td>
</tr>
<tr>
<td>FFFF TRUE</td>
<td>0AF1 UNACNT</td>
<td>0AF2 UNADSK</td>
</tr>
<tr>
<td>0AF3 UNATRKG</td>
<td>0717 WAIT0</td>
<td>0B15 WARMSTART</td>
</tr>
<tr>
<td>0000 WRALL</td>
<td>0001 WRDIR</td>
<td>0296 WRITE</td>
</tr>
<tr>
<td>046D WRITEHST</td>
<td>05F2 WRITESOFT</td>
<td>0AF9 WRTYPE</td>
</tr>
<tr>
<td>0B10 XDOS</td>
<td>07F8 XETMOD</td>
<td>0801 XETSEL</td>
</tr>
<tr>
<td>01B5 XLT0</td>
<td>01CF XLT1</td>
<td>01CF XLT2</td>
</tr>
<tr>
<td>0000 XLT4</td>
<td>0000 XLT5</td>
<td>0000 XLT6</td>
</tr>
</tbody>
</table>

All Information Presented Here is Proprietary to Digital Research
Index

A

ALL field, 27
ALV, 25
attribute byte, 56

B

bank byte, 57
bank-switched memory, 55
banked resident system
process, 49
Basic Disk Operating System
(BDOS), 15, 48
BIOS, 15
BIOS devices, 16
BIOS disk definition, 29
BIOS Disk Definition Tables,
24
BIOS entry point, 15
BIOS jump table, 16
BIOS subroutines, 15, 17, 19, 20, 22, 23
BIOS WRITE entry point, 36
BLM field, 27
BLM values, 27
blocking and deblocking, 36
BLS, 28
BNKBDOS, 47, 48
BNKBDOS.SPR, 49
BNKXDIS, 47
BNKXIOS, 1, 3
BNKXIOS.SPR, 48
BOOT, 12
booting MP/M II, 6
bootstrap operation, 13
breakpoint, 6
breakpoint restart number, 53
BSH field, 27
BSH values, 27

C

cold start loader, 6, 7
cold start loader, 6, 7
common memory base page, 55
COMMONBASE subroutine, 33, 37
CONIN, 17
CONOUT, 18
consoles, 16
CONST, 17
copyright, 11
CSV, 25
custom generation of an MP/M
II boot, 8
custom resident system
processes, 49
customized XIOS, 47

d

dayfile logging, 55
ddt, 2, 4, 7, 47
debugging a resident system
process, 4
debugging an XIOS, 4, 5, 6
default entry values, 52
delay list, 44
dir parameter, 31
DIRBUF, 25
directory check information,
29
directory entries, 28
disk I/O, 15, 16
disk organization, 12, 13
Disk Parameter Block (DPB),
26, 29
Disk Parameter Headers (DPHs),
24, 25, 32
Disk Parameter Table Format,
24
DISKDEF, 24, 29, 30
disks, 16
dks parameter, 31
DMA address, 16
dn parameter, 31
DPB, 25
DPBASE, 25
dph labels, 32
DRM, 28
DRM labels, 32
DSM field, 27

E

END statement, 30
ENDEF macro, 30, 32
execution address, 5, 7, 11
EXITREGION, 41
EXM field, 27
EXM values, 28
Extended Input/Output System
(XIOS), 39
external procedure access
code, 34
external procedure entry
points, 33
F
Flag Set, 44
flags, 43
fsc parameter, 31
G
GENSYS, 51, 57
GETSYS, 6, 8, 9
H
header block, 32
HOME, 18
I
IDLE process, 43
Intel MDS-800, 7, 12
interrupt service routines, 43
interrupts, 3
J
jump addresses, 15
jump instructions, 15
jump vector, 15, 33
K
kl6 parameter, 31
L
LDRBIOS, 1, 2, 3
licensing agreement, 11
LIST, 18
list devices, 16
LISTST, 23
load address, 7
loading MPM.SYS, 11
locked records, 54
logical devices, 15
logical drives, 30
lsc parameter, 31
M
MAC, 29
macro statements, 29
MAXCONSOLE, 42
memory map, 5, 56
memory segment table, 56
MP/M II Boot, 7
MP/M II nucleus, 49
MPM.SYS, 47
MPM.SYS file, 11
MPMLDR, 2, 6
MPMLDR display, 58
O
OFF field, 27, 29
offs parameter, 31
one second flag, 44
P
page address, 11
page relocatable files, 49
PDISP entry point, 35, 44
POLLDEVICE, 40
prm parameter, 31
process termination, 15
PUNCH, 18
PUTSYS, 6, 8, 10
R
READ, 21
READER, 18
real time, 44
RESBDOS, 47, 48
RESBDOS.SPR, 48
resident system process, 47,
49, 56
restart number, 4
RESXIOS, 1, 3
RESXIOS.SPR, 48
RET instruction, 15
RETI instruction, 44
RMAC, 29
S
sector allocation, 12
SECTRAN, 23
SEKDSDK entry point, 36
SELDSDK, 2, 16, 19, 25
SELMEMORY, 40
SETDMA, 20
SETSEC, 20
SETTRK, 20
SID, 2, 4, 7, 47
skew factor, 23
skew factor parameter, 32
skf parameter, 31
SPT field, 27
STARTCLOCK, 40
STAT, 33
STOPCLOCK, 41
SUBMIT, 54
subroutine, 1
SWTSYS, 22
SWTSYS entry point, 35
SWTUSER, 22
SWTUSER entry point, 35
SYSDAT entry, 35
SYSDAT page, 11
SYSGEN, 6, 7, 8
system data page, 35, 47, 56
system tick, 44, 45
SYSTEMINIT, 42

T
temporary file drive, 54
Terminal Message Processes (TMPs), 53
time bases, 44
TMP, 47
top page of operating system, 53
trademark, 11
translation table, 32
translation vectors, 26

U
user memory segments, 55

W
WBOOT, 17
WRITE, 22
WRITESEC subroutines, 9

X
XDOS, 15, 47, 53
XDOS entry point, 35
XDOS.SPR, 49
XIOS, 1, 3, 15, 48
XIOS entry points, 15, 39

Z
Z80 CPU, 54