DIGITAL RESEARCH

Post Office Box 579, Pacific Grove, California 93950, (408) 373-3403

CP/M System Alteration Guide

Version 1.3

Table of Contents

CP/M System Alteration Guide

1. INTRODUCTION

The standard CP/M system assumes operation on an Intel MDS microcomputer development system, but is designed so that the user can alter a specific set of subroutines which define the hardware operating enviornment. In this way, the user can produce a diskette which operates with a non-standard (but IBM-compatible format) drive controller and/or peripheral devices.

In order to achieve device independence, CP/M is separated into three distinct modules:

BIOS - basic I/O system which is environment dependent
BDOS - basic disk operating system which is not dependent upon the hardware configuration
CCP - the console command processor which uses the BDOS

of these modules, only the BIOS is dependent upon the particular hardware. That is, the user can "patch" the distribution version of CP/M to provide a new BIOS which provides a customized interface between the remaining CP/M modules and the user's own hardware system. The purpose of this document is to provide a step-by-step procedure for patching the new BIOS into CP/M.

The new BIOS requires some relatively simple software development and testing; the current BIOS, however, is listed in Appendix C, and can be used as a model for the customized package. A skeletal version of the BIOS is given in Appendix D which can form the base for a modified BIOS. In addition to the BIOS, the user must write a simple memory loader, called GETSYS, which brings the operating system into memory. In order to patch the new BIOS into CP/M, the user must write the reverse of GETSYS, called PUTSYS, which places an altered version of CP/M back onto the diskette. PUTSYS is usually derived from GETSYS by changing the disk read commands into disk write commands. Sample skeletal GETSYS and PUTSYS programs are described in Section 3, and listed in Appendix E. In order to make the CP/M system work automatically, the user must also supply a cold start loader, similar to the one provided wi CP/M (listed in Appendices A and B). A skeletal form of a cold start loader is given in Appendix F which can serve as a model for your loader.

2. FIRST LEVEL SYSTEM REGENERATION

The procedure to follow to patch the CP/M system is given below in several steps. Address references in each step are shown with a following "H" which denotes the hexadecimal radix, and are given for a 16K CP/M system. For larger CP/M systems, add a "bias" to each address which is shown with a "+b" following it, where b is equal to the memory size - 16K. Values for b in various standard memory sizes are

32K: b = 32K - 16K = 16K = 04000H

[illegible]	b = 48K - 16K = 32K = 08000H
62K:	b = 62K - 16K = 46K = 0B800H
64K:	b = 64K - 16K = 48K = 0C000H

(1) Review Section 4 and write a GETSYS program which reads the first two tracks of a diskette into memory. The data from the diskette must begin at location 2880H+b. Code GETSYS so that it starts at location 100H (base of the TPA), as shown in the first part of Appendix E.

(2) Test the GETSYS program by reading a blank diskette into memory, and check to see that the data has been read properly, and that the diskette has not been altered in any way by the GETSYS program.

(3) Run the GETSYS program using an initialized CP/M diskette to see if GETSYS loads CP/M starting at 2880H+b (the operating system actually starts 128 bytes later at 2900H+b).

(4) Review Section 4 and write the PUTSYS program which writes memory starting at 2880H+b back onto the first two tracks of the diskette. The PUTSYS program should be located at 200H, as shown in the second part of Appendix E.

(5) 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 using GETSYS. Test PUTSYS completely, since this program will be used to alter CP/M on disk.

(6) Study Sections 5, 6, and 7, along with the distribution version of the BIOS given in Appendix C, and write a simple version which performs a similar function for the customized environment. Use the program given in Appendix D as a model. Call this new BIOS by the name CBIOS (customized BIOS). Implement only the primitive disk operations on a single drive, and simple console input/output functions in this phase.

(7) Test CBIOS completely to ensure that it properly performs console character I/O and disk reads and writes. Be especially careful to ensure that no disk write operations occur accidently during read operations, and check that the proper track and sectors are addressed on all reads and writes. Failure to make these checks may cause distruction of the initialized CP/M system after it is patched.

(8) Referring to Figure 1 in Section 5, note that the BIOS is located between locations 3E00H+b and 3FFFH+b. Read the CP/M system using GETSYS and replace the BIOS segment by the new CBIOS developed in step (6) and tested in step (7). This replacement is done in the memory of the machine, and will be placed on the diskette in the next step.

(9) Use PUTSYS to place the patched memory image of CP/M onto the first two tracks of a blank diskette for testing.

(10) Use GETSYS to bring the copied memory image from the test diskette back into memory at 2880H+b, and check to ensure that it has loaded back properly (clear memory, if possible, before the load). Upon successful load, branch to the CCP module at location 2900H+b. The CCP will call the BDOS, which will call the CBIOS. The CBIOS will be asked to read several sectors on track 2 twice in succession, and, if successful, CP/M will type "A>".

When you make it this far, you are almost on the air. If you have trouble, use whatever debug facilities you have available to trace and breakpoint your CBIOS.

(11) Upon completion of step (10), CP/M has prompted the console for a command input. Test the disk write operation by typing

```
SAVE 1 X.COM
```

(recall that all commands must be followed by a carriage return). CP/M should respond with another prompt (after several disk accesses):

```
A>
```

If it does not, debug your disk write functions and retry.

(12) Then test the directory command by typing

```
DIR *.*
```

CP/M should respond with

```
X       COM
```

(13) Test the erase command by typing

```
ERA X.COM
```

CP/M should respond with the A prompt. When you make it this far, you have an operational system which only requires a bootstrap loader to function completely.

(14) Write a bootstrap loader which is similar to GETSYS, and place it into read-only-memory, or into track 0, sector 1 using PUTSYS (again using the test diskette, not the distribution diskette). See Sections 5 and 8 for more information on the bootstrap operation.

(15) Retest the new test diskette with the bootstrap loader installed by executing steps (11), (12), and (13). Upon completion of these tests, type a control-C (control and C keys simultaneously). The system should then execute a "warm start" which reboots the system, and types the A prompt.

(16) At this point, you probably have a good version of your customized

system on your test diskette. Use GETSYS to load CP/M from your test diskette. Remove the test diskette, place the distribution diskette (or a legal copy) into the drive, and use PUTSYS to replace the distribution version by your customized version. Do not make this replacement if you are unsure of your patch since this step destroys the system which was sent to you from Digital Research.

(17) Load your modified CP/M system and test it by typing

```
DIR *.*
```

CP/M should respond with a list of files which are provided on the initialized diskette. One such file should be the memory image for the debugger, called DDT.COM.

> NOTE: from now on, it is important that you always reboot the CP/M system when the diskette is removed and replaced by another diskette, unless the new diskette is read-only.

(18) Load and test the debugger by typing

```
DDT
```

(see the document "CP/M Dynamic Debugging Tool (DDT)" for operating information and examples). Take time to familiarize yourself with DDT; it will be your best friend in later steps.

(19) Before making further CBIOS modifications, practice using the editor (see the ED user's guide), and assembler (see the ASM user's guide). Then recode and test the GETSYS, PUTSYS, and CBIOS programs using ED, ASM, and DDT. Code and test a COPY program which does a sector-to-sector copy from one diskette to another to obtain back-up copies of the original diskette (NOTE: read your CP/M Licensing Agreement; it specifies your legal responsibilities when copying the CP/M system). Place the copyright notice

```
Copyright (c) 1976
Digital Research
```

on each copy which is made with your COPY program.

(20) Modify your CBIOS to include the extra functions for punches, readers, signon messages, and so-forth, and add the facilities for a second drive, if it exists on your system. You can make these changes with the GETSYS and PUTSYS programs which you have developed, or you can refer to the following section, which outlines CP/M facilities which will aid you in the regeneration process.

You now have a good copy of the customized CP/M system. Note that although the CBIOS portion of CP/M which you have developed belongs to you, the modified version of CP/M which you have created can be copied for your use only (again, read your Licensing Agreement), and cannot be legally copied for

anyone else's use. If you wish, you may send you name and address to Digital Research, along with a description of your hardware environment and the modifications which you have made. Digital Research will make the information available to other interested parties, and inform them of the prices and availability of your CBIOS.

It should be noted that your system remains file-compatible with all other CP/M systems, which allows transfer of non-proprietary software between users of CP/M.

3. SECOND LEVEL SYSTEM GENERATION

Now that you have the CP/M system running, you may wish to use CP/M facilities in the system regeneration process. In general, we will first get a memory image of CP/M from the first two tracks of an initialized diskette and place this memory image into a named disk file. The disk file can then be loaded, examined, patched, and replaced using the editor, assembler, debugger, and system generation program.

The SYSGEN program, supplied with your diskette, is first used to get a CP/M memory image from the first two tracks. Run the SYSGEN program as shown below

```
SYSGEN                          start the SYSGEN program
*SYSGEN VERSION 1.Ø             SYSGEN signon message
GET SYSTEM (Y/N)?Y              Answer yes to GET request
SOURCE ON B, THEN TYPE RETURN
```

at this point, place an initialized diskette into drive B and type a return (if you are operating with a single drive, answer "A" to the GET request, rather than "Y", and place the initialized diskette into drive A before typing the return). The program should respond with:

```
FUNCTION COMPLETE               Load is complete
PUT SYSTEM (Y/N)?N              Answer no to PUT request
```

system will automatically reboot at this point, with the memory image loaded into memory starting at location 9ØØH and ending at 2Ø7FH in the transient program area. The memory image for CP/M can then be saved (if you are operating with a single drive, replace your original diskette and reboot). The save operation is accomplished by typing:

```
SAVE 32 CPM.COM                 Save 2ØH = 32 pages of memory
```

The memory image created by the GET function is offset by a negative bias so that it loads into the free area of the TPA, and thus does not interfere with the operation of CP/M in higher memory. This memory image can be subsequently loaded under DDT and examined or changed in preparation for a new generation of the system. DDT is loaded with the memory image by typing

DDT CPM.COM Load DDT, then read the CPM image

DDT should respond with

```
NEXT   PC
2100  0100
```

You can then use the display and disassembly commands to examine portions of the memory image between 900H and 207FH. Note, however, that to find any particular address within the memory image, you must apply the negative bias to the CP/M address to find the actual address. Track 00, sector 01 is loaded to location 900H (you should find the cold start loader at 900H to 97FH), track 00, sector 02 is loaded into 980H (this is the base of the CCP), and so-forth through the entire CP/M system load. In a 16K system, for example, the CCP resides at the CP/M address 2900H, but is placed into memory at 980H by the SYSGEN program. Thus, the negative bias,denoted by n, satisfies

2900H + n = 980H, or n = 980H - 2900H

Assuming two's complement arithmetic, n = 0E080H, which can be checked by

2900H + 0E080H = 10980H = 0980H (ignoring high-order overflow).

Note that for larger systems, n satisfies

(2900H+b) + n = 980H, or
n = 980H - (2900H + b), or
n = 0E080H - b.

The value of n for common CP/M systems is given below

memory size	bias b	negative offset n
16K	0000H	0E080H - 0000H = 0E080H
32K	4000H	0E080H - 4000H = 0A080H
48K	8000H	0E080H - 8000H = 6080H
62K	0B800H	0E080H - 0B800H = 2880H
64K	0C000H	0E080H - 0C000H = 2080H

Assume, for example, that you want to locate the address x within the memory image loaded under DDT in a 16K system. First type

Hx,n Hexadecimal sum and difference

and DDT will respond with the value of x+n (sum) and x-n (difference). The first number printed by DDT will be the actual memory address in the image where the data or code will be found. The input

H2900,E080

for example, will produce 980H as the sum, which is where the CCP is located in the memory image under DDT.

Use the L command to disassemble portions of your CBIOS located at (3E00H+b)-n which, when you use the H command, produces an actual address of 1E80H. The disassembly command would thus be

```
L1E80
```

Terminate DDT by typing a control-c or "G0" in order to prepare the patch program. Your CBIOS, for example, can be modified using the editor, and assembled using ASM, producing a file called CBIOS.HEX which contains the Intel formatted machine code for CBIOS in "hex" format. In order to integrate your new CBIOS, return to DDT by typing

```
DDT CPM.COM                    Start DDT and load the CPM image
```

Examine the area at 1E80H where the previous version of the CBIOS resides. Then type

```
ICBIOS.HEX                     Ready the "hex" file for loading
```

Assume that your CBIOS is being integrated into a 16K CP/M system, and is thus "org'ed" at location 3E00H. In order to properly locate the CBIOS in the memory image under DDT, we must apply the negative bias n for a 16K system when loading the hex file. This is accomplished by typing

```
RE080                          Read the file with bias 0E080H
```

Upon completion of the read, re-examine the area where the CBIOS has been loaded (use a "L1E80" command), to ensure that is was loaded properly. When you are satisfied that the patch has been made, return from DDT using a control-c or "G0" command.

Now use SYSGEN to replace the patched memory image back onto a diskette (use a test diskette until you are sure of your patch), as shown in the following interaction

```
SYSGEN                         Start the SYSGEN program
*SYSGEN VERSION 1.0            Signon message from SYSGEN
GET SYSTEM (Y/N)?N             Answer no to GET request
PUT SYSTEM (Y/N)?Y             Answer yes to PUT request
DESTINATION ON B, THEN TYPE RETURN
```

Place the test diskette on drive B (if you are operating with a single drive system, answer "A" rather than "Y" to the PUT request, then remove your diskette, and replace by the test diskette), and type a return. The system will be replaced on the test diskette, and the system will automatically boot from drive A.

Test the new CP/M system, and place the Digital Research copyright notice

on the diskette, as specified in your Licensing Agreement:

4. SAMPLE GETSYS AND PUTSYS PROGRAMS

The following program provides a framework for the GETSYS and PUTSYS programs referenced in Section 2. The READSEC and WRITESEC subroutines must be inserted by the user to read and write the specific sectors.

```
;   GETSYS PROGRAM - READ TRACKS 0 AND 1 TO MEMORY AT 2880H
;   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,2880H     ;SET STACK POINTER TO SCRATCH AREA
       LXI   H, 2880H     ;SET BASE LOAD ADDRESS
       MVI   B, 0         ;START WITH TRACK 0
RDTRK:                    ;READ NEXT TRACK (INITIALLY 0)
       MVI   C,1          ;READ STARTING WITH SECTOR 1
RDSEC:                    ;READ NEXT SECTOR
       CALL  READSEC      ;USER-SUPPLIED SUBROUTINE
       LXI   D,128        ;MOVE LOAD ADDRESS TO NEXT 1/2 PAGE
       DAD   D            ;HL = HL + 128
       INR   C            ;SECTOR = SECTOR + 1
       MOV   A,C          ;CHECK FOR END OF TRACK
       CPI   27
       JC    RDSEC        ;CARRY GENERATED IF SECTOR < 27
;
;   ARRIVE HERE AT END OF TRACK, MOVE TO NEXT TRACK
       INR   B
       MOV   A,B          ;TEST FOR LAST TRACK
       CPI   2
       JC    RDTRK        ;CARRY GENERATED IF TRACK < 2
;
;   ARRIVE HERE AT END OF LOAD, HALT FOR NOW
       HLT
;
;   USER-SUPPLIED SUBROUTINE TO READ THE DISK
READSEC:
;   ENTER WITH TRACK NUMBER IN REGISTER B,
;         SECTOR NUMBER IN REGISTER C, AND
;         ADDRESS TO FILL IN HL
;
```

```
PUSH    B           ;SAVE B AND C REGISTERS
PUSH    H           ;SAVE HL REGISTERS
.....................................
perform disk read at this point, branch to
label START if an error occurs
.....................................
POP     H           ;RECOVER HL
POP     B           ;RECOVER B AND C REGISTERS
RET                 ;BACK TO MAIN PROGRAM

END     START
```

Note that this program is assembled and listed in Appendix D for reference purposes, with an assumed origin of 100H. The hexadecimal operation codes which are listed on the left may be useful if the program has to be entered through your machine's front panel switches.

The PUTSYS program can be constructed from GETSYS by changing only a few operations in the GETSYS program given above, as shown in Appendix E. The register pair HL become the dump address (next address to write), and operations upon these registers do not change within the program. The READSEC subroutine is replaced by a WRITESEC subroutine which performs the opposite function: data from address HL is written to the track given by register B and sector given by register C. It is often useful to combine GETSYS and PUTSYS into a single program during the test and development phase, as shown in the Appendix.

5. DISKETTE ORGANIZATION

The sector allocation for the distribution version of CP/M is given here for reference purposes. The first sector (see Figure 1) 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 LBOOT, has the responsibility of bringing the remaining sectors into memory starting at location 2900H+b. 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 2900H+b.

As an example, the Intel MDS hardware cold start loader brings track 0, sector 1 into absolute address 3000H. Thus, the distribution version contains two very small programs in track 0, sector 1:

MBOOT - a storage move program which moves LBOOT into place following the cold start (Appendix A)

LBOOT - the cold start boot loader (Appendix B)

Upon MDS start-up, the 128 byte segment on track 0, sector 1 is brought

into 3000H. The MBOOT program gets control, and moves the LBOOT program from location 301EH down to location 80H in memory, in order to get LBOOT out the the area where CP/M is loaded in a 16K system. Note that the MBOOT program would not be needed if the MDS loaded directly to 80H. In general, the LBOOT program could be located anywhere below the CP/M load location, but is most often located in the area between 000H and 0FFH (below the TPA).

After the move, MBOOT transfers to LBOOT at 80H. LBOOT, in turn, loads the remainder of track 0 and the initialized portion of track 1 to memory, starting at 2900H+b. The user should note that MBOOT and LBOOT are of little use in a non-MDS environment, although it is useful to study them since some of their actions will have to be duplicated in your cold start loader.

Figure 1. Diskette Allocation

Track#	Sector#	Page#	Memory Address	CP/M Module name
00	01		(boot address)	Cold Start Loader
00	02	00	2900H+b	CCP
"	03	"	2980H+b	"
"	04	01	2A00H+b	"
"	05	"	2A80H+b	"
"	06	02	2B00H+b	"
"	07	"	2B80H+b	"
"	08	03	2C00H+b	"
"	09	"	2C80H+b	"
"	10	04	2D00H+b	"
"	11	"	2D80H+b	"
"	12	05	2E00H+b	"
"	13	"	2E80H+b	"
"	14	06	2F00H+b	"
"	15	"	2F80H+b	"
"	16	07	3000H+b	"
"	17	"	3080H+b	"
"	18	08	3100H+b	"
00	19	"	3180H+b	CCP
00	20	09	3200H+b	BDOS
"	21	"	3280H+b	"
"	22	10	3300H+b	"
"	23	"	3380H+b	"
"	24	11	3400H+b	"
"	25	"	3480H+b	"
"	26	12	3500H+b	"
01	01	"	3580H+b	"
"	02	13	3600H+b	"
"	03	"	3680H+b	"
"	04	14	3700H+b	"
"	05	"	3780H+b	"

"	06	15	3800H+b	"
"	07	"	3880H+b	"
"	08	16	3900H+b	"
"	09	"	3980H+b	"
"	10	17	3A00H+b	"
"	11	"	3A80H+b	"
"	12	18	3B00H+b	"
"	13	"	3B80H+b	"
"	14	19	3C00H+b	"
"	15	"	3C80H+b	"
"	16	20	3D00H+b	"
"	17	"	3D80H+b	BDOS
01	18	21	3E00H+b	BIOS
"	19	"	3E80H+b	"
"	20	22	3F00H+b	"
01	21	"	3F80H+b	BIOS
01	22-26			(not currently used)
02-76	01-26			(directory and data)

6. THE BIOS ENTRY POINTS

The entry points into the BIOS from the cold start loader and BDOS are detailed below. Entry to the BIOS is through a "jump vector" between locations 3E00H+b and 3E2CH+b, as shown below (see also Appendices, pages C-2 and D-1). The jump vector is a sequence of 15 jump instructions which send program control to the individual BIOS subroutines. The BIOS subroutines may be empty for certain functions (i.e., they may contain a single RET operation) during regeneration of CP/M, but the entries must be present in the jump vector.

It should be noted that there is a 16 byte area reserved in page zero (see Section 9) starting at location 40H, which is available as a "scratch" area in case the BIOS is implemented in ROM by the user. This scratch area is never accessed by any other CP/M subsystem during operation.

The jump vector at 3E00H+b takes the form shown below, where the individual jump addresses are given to the left:

```
3E00H+b     JMP BOOT        ;ARRIVE HERE FROM COLD START LOAD
3E03H+b     JMP WBOOT       ;ARRIVE HERE FOR WARM START
3E06H+b     JMP CONST       ;CHECK FOR CONSOLE CHAR READY
3E09H+b     JMP CONIN       ;READ CONSOLE CHARACTER IN
3E0CH+b     JMP CONOUT      ;WRITE CONSOLE CHARACTER OUT
3E0FH+b     JMP LIST        ;WRITE LISTING CHARACTER OUT
3E12H+b     JMP PUNCH       ;WRITE CHARACTER TO PUNCH DEVICE
3E15H+b     JMP READER      ;READ READER DEVICE
```

```
3E18H+b     JMP HOME        ;MOVE TO TRACK 00 ON SELECTED DISK
3E1BH+b     JMP SELDSK      ;SELECT DISK DRIVE
3E1EH+b     JMP SETTRK      ;SET TRACK NUMBER
3E21H+b     JMP SETSEC      ;SET SECTOR NUMBER
3E24H+b     JMP SETDMA      ;SET DMA ADDRESS
3E27H+b     JMP READ        ;READ SELECTED SECTOR
3E2AH+b     JMP WRITE       ;WRITE SELECTED SECTOR
```

Each jump address corresponds to a particular subroutine which performs the specific function, as outlined below. There are three major divisions in the jump table: the system (re)initialization which results from calls on BOOT and WBOOT, simple character I/O performed by calls on CONST, CONIN, CONOUT, LIST, PUNCH, and READER, and diskette I/O performed by calls on HOME, SELDSK, SETTRK, SETSEC, SETDMA, READ, and WRITE.

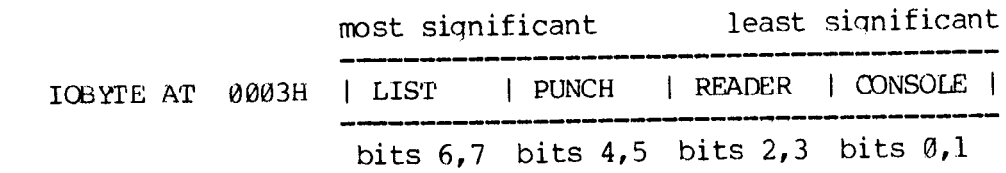
All simple character I/O operations are assumed to be performed in ASCII, upper and lower case, with high order (parity bit) set to zero. An end-of-file condition is given by an ASCII control-z (1AH). Peripheral devices are seen by CP/M as "logical" devices, and are assigned to physical devices within the BIOS. In order to operate, the BDOS needs only the CONST, CONIN, and CONOUT subroutines (LIST, PUNCH, and READER are used by PIP, but not the BDOS). Thus, the initial version of CBIOS may have empty subroutines for the remaining ASCII devices. The characteristics of each device are

CONSOLE	The principal interactive console which communicates with the operator, accessed through CONST, CONIN, and CONOUT. Typically, the CONSOLE is a device such as a CRT or Teletype.
LIST	The principal listing device, if it exists on your system, which is usually a hard-copy device, such as a printer or Teletype.
PUNCH	The principal tape punching device, if it exists, which is normally a high-speed paper tape punch or Teletype.
READER	The principal tape reading device, such as a simple optical reader or Teletype.

Note that a single peripheral can be assigned as the LIST, PUNCH, and READER device simultaneously. If no peripheral device is assigned as the LIST, PUNCH, or READER device, the CBIOS created by the user should give an appropriate error message so that the system does not "hang" if the device is accessed by PIP or some other user program.

For added flexibility, the user can optionally implement the "iobyte" function which allows reassignment of physical and logical devices. The

iobyte function creates a mapping of logical to physical devices which can be altered during CP/M processing. The definition of the iobyte function corresponds to the Intel standard as follows: a single location in memory (currently location 0003H) is maintained, called IOBYTE, which defines the logical to physical device mapping which is in effect at a particular time. The mapping is performed by splitting the IOBYTE into four distinct fields of two bits each, called the CONSOLE, READER, PUNCH, and LIST fields, as shown below



The value in each field can be in the range 0-3, defining the assigned source or destination of each logical device. The values which can be assigned to each field are given below

CONSOLE field (bits 0,1)
- 0 - console is assigned to the Teletype device (TTY)
- 1 - console is assigned to the CRT device (CRT)
- 2 - batch mode: use the READER as the CONSOLE input, and the LIST device as the CONSOLE output
- 3 - user defined console device

READER field (bits 2,3)
- 0 - READER is the Teletype device
- 1 - READER is the high-speed reader device (RDR)
- 2 - user defined reader # 1
- 3 - user defined reader # 2

PUNCH field (bits 4,5)
- 0 - PUNCH is the Teletype device
- 1 - PUNCH is the high speed punch device (PUN)
- 2 - user defined punch # 1
- 3 - user defined punch # 2

LIST field (bits 6,7)
- 0 - LIST is the Teletype device
- 1 - LIST is the CRT device
- 2 - LIST is the line printer device
- 3 - user defined list device

Note again that the implementation of the IOBYTE is optional, and affects only the organization of your CBIOS. No CP/M systems use the IOBYTE (although they tolerate the existence of the IOBYTE at location 0003H), except for PIP which allows access to the TTY: and CRT: devices. If you do not implement the IOBYTE, you cannot access these physical devices through PIP. In any case, the IOBYTE implementation should be omitted until your basic CBIOS is fully

implemented and tested; then add the IOBYTE to increase your facilities.

Disk I/O is always performed through a sequence of calls on the various disk access subroutines which 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 on 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 called before the read and write operations are performed. Note, however, that the BIOS does not attempt error recovery when a read or write fails, but instead reports the error condition to the BDOS. The BDOS then retries the read or write, assuming the track and sector address remain the same. The HOME subroutine may be called during error recovery, following by a re-seek of the particular track and sector. 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.

The exact responsibilites of each entry point subroutine are given below:

BOOT — The BOOT entry point gets control from the cold start loader and is responsible for basic system initialization, including sending a signon message (which can be omitted in the first version). If the IOBYTE function is implemented, it must be set at this point. The various system parameters which are set by the WBOOT entry point must be initialized, and control is transferred to the CCP at 2900H+b for further processing.

WBOOT — The WBOOT entry point gets control when a warm start occurs. A warm start is performed whenever a user program branches to location 0000H, or when the CPU is reset from the front panel. The CP/M system must be loaded from the first two tracks of drive A up to, but not including, the BIOS (or CBIOS, if you have completed your patch). System parameters must be initialized as shown below:

location 0,1,2	set to JMP WBOOT for warm starts (0000H: JMP 3E03H+b)
location 3	set initial value of IOBYTE, if implemented in your CBIOS
location 5,6,7	set to JMP BDOS, which is the primary entry point to CP/M for transient programs. (0005H: JMP 3206H+b)

(see Section 9 for complete details of page zero use)

Upon completion of the initialization, the WBOOT program must branch to the CCP at 2900H+b to (re)start the system. Upon entry to the CCP, register C is set to the drive to select after system initialization (normally drive A is selected by setting register C to zero).

CONST — Sample the status of the currently assigned console device and return a 0FFH in register A if a character is ready to read, and 00H in register A if no console characters are ready.

CONIN — Read the next console character into register A, and set the parity bit (high order bit) to zero. If no console character is ready, wait until a character is typed before returning.

CONOUT — Send the character from register C to the console output device. The character is in ASCII, with high order parity bit set to zero. You may want to include a time-out 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 which cause your console device to react in a strange way (a control-z causes the Lear Seigler terminal to clear the screen, for example).

LIST — Send the character from register C to the currently assigned listing device. The character is in ASCII with zero parity.

PUNCH — Send the character from register C to the currently assigned punch device. The character is in ASCII with zero parity.

READER — Read the next character from the currently assigned reader device into register A with zero parity (high order bit must be zero), an end of file condition is reported by returning an ASCII control-z (1AH).

HOME — Return the disk head of the currently selected disk (initially disk A) 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.

SELDSK — Select the disk drive given by register C for further operations, where register C contains 0 for drive A, and 1 for drive B (the standard CP/M distribution version supports a maximum of two drives). If your system has only one drive, you may wish to give an error message at the console, and terminate execution. You can, if you wish, type a message at the console to switch diskettes to simulate a two drive

---. In this case, you must keep account of the current drive and type an appropriate message when the drive changes.

SETTRK — Register C 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 C can take on values in the range 0-76 corresponding to valid track numbers.

SETSEC — Register C contains the sector number (1 through 26) for subsequent disk accesses on the currently selected drive. You can choose to send this information to the controller at this point, or instead delay sector selection until the read or write operation occurs.

SETDMA — Registers B and C (high order 8 bits in B, low order 8 bits in C) contain the DMA (direct 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 fill their data into 80H through 0FFH, and all subsequent write operations get their data from 80H through 0FFH, until the next call to SETDMA occurs. The initial DMA address is assumed to be 80H. 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 CBIOS which you construct uses the 128 byte area starting at the selected DMA address for the memory buffer during the I/O operation.

READ — Assuming the drive has been selected, the track has been set, the sector has been set, and the DMA address has been specified, this subroutine attempts to read the selected sector. The read operation may involve several retries (10 is a good number) if errors occur during the read operation. If the read is completed correctly, the READ subroutine should return a 00 in register A. If the read cannot be performed, a 01 should be returned: in this case CP/M prints the message

PERM ERROR DISK x

where x is the disk number.

it will be useful to have the additional error responses.

WRITE — Write the data from the currently selected DMA address to the currently selected drive, track, and sector. The data should be marked as "non deleted data" to maintain compatibility with other CP/M systems. The error codes given in the READ command are returned in register A, with error recovery attempts as described above.

7. A SAMPLE BIOS

The program shown in Appendix D can serve as a basis for your first BIOS. The simplest functions are assumed in this BIOS, so that you can enter it through the front panel, if absolutely necessary. Note that the user must alter and insert code into the subroutines for CONST, CONIN, CONOUT, READ, WRITE, and WAITIO subroutines. Storage is reserved for user-supplied code in these regions. The scratch area reserved in page zero (see Section 9) for the BIOS is used in this program, so that it could be implemented in ROM, if desired.

Once operational, this skeletal version can be enhanced to print the initial sign-on message and perform better error recovery. The subroutines for LIST, PUNCH, and READER can be filled-out, and the IOBYTE function can be implemented.

8. A SAMPLE COLD START LOADER

The program shown in Appendix E can serve as a basis for your cold start loader. The disk read function must be supplied by the user, and the program must be loaded somehow starting at location 0000. Note that space is reserved for your patch so that the total amount of storage required for the cold start loader is 128 bytes. Eventually, you will probably want to get this loader onto the first disk sector (track 0, sector 1), and cause your controller to load it into memory automatically upon system start-up. Alternatively, you may wish to place the cold start loader into ROM, and place it above the CP/M system. In this case, it will be necessary to originate the program at a higher address, and key-in a jump instruction at system start-up which branches to the loader. Subsequent warm starts will not require this key-in operation, since the entry point 'WBOOT' gets control, thus bringing the system in from disk automatically. Note also that the skeletal cold start loader has minimal error recovery, which may be enhanced on later versions.

9. RESERVED LOCATIONS IN PAGE ZERO

Main memory page zero, between locations 00H and 0FFH, contains several segments of code and data which are used during CP/M processing. The code and

ta areas are given below for reference purposes.

Locations from	to	Contents
0000H	0002H	Contains a jump instruction to the warm start entry point at location 3E03H+b. This allows a simple programmed restart (JMP 0000H) or manual restart from the front panel.
0003H	0003H	Contains the Intel standard IOBYTE, which is optionally included in the user's CBIOS, as described in Section 6.
0004H	0004H	(not currently used - reserved)
0005H	0007H	Contains a jump instruction to the BDOS, and serves two purposes: JMP 0005H provides the primary entry point to the BDOS, as described in the manual "CP/M Interface Guide," and LHLD 0006H brings the address field of the instruction to the HL register pair. This value is the lowest address in memory used by CP/M (assuming the CCP is being overlayed). Note that the DDT program will change the address field to reflect the reduced memory size in debug mode.
0008H	0027H	(interrupt locations 1 through 5 not used)
0030H	0037H	(interrupt location 6, not currently used - reserved)
0038H	003AH	Contains a jump instruction into the DDT program when running in debug mode for programmed breakpoints, but is not otherwise used by CP/M.
003BH	003FH	(not currently used - reserved)
0040H	004FH	16 byte area reserved for scratch by CBIOS, but is not used for any purpose in the distribution version of CP/M
0050H	005BH	(not currently used - reserved)
005CH	007CH	default file control block produced for a transient program by the Console Command Processor.
007DH	007FH	(not currently used - reserved)
0080H	00FFH	default 128 byte disk buffer (also filled with the command line when a transient is loaded under the CCP).

Note that this information is set-up for normal operation under the CP/M system, but can be overwritten by a transient program if the BDOS facilities are not required by the transient. If, for example, a particular program

performs only simple I/O and must begin execution at location 0, it can be first loaded into the TPA, using normal CP/M facilities, with a small memory move program which gets control when loaded (the memory move program must get control from location 100H, which is the assumed beginning of all transient programs). The move program can then proceed to move the entire memory image down to location 0, and pass control to the starting address of the memory load. Note that if the BIOS is overwritten, or if location 0 (containing the warm start entry point) is overwritten, then the programmer must bring the CP/M system back into memory with a cold start sequence.

```
                ; MDS LOADER MOVE PROGRAM, PLACES COLD START BOOT AT BOOTB
                ;
3000              ORG     3000H     ;WE ARE LOADED HERE ON COLD START
0080 =          BOOTB     EQU       80H       ;START OF COLD BOOT PROGRAM
0080 =          BOOTL     EQU       80H       ;LENGTH OF BOOT
D900 =          MBIAS     EQU       900H-$    ;BIAS TO ADD DURING LOAD
0078 =          BASE      EQU       078H      ;'BASE' USED BY DISK CONTROLLER
0079 =          RTYPE     EQU       BASE+1    ;RESULT TYPE
007B =          RBYTE     EQU       BASE+3    ;RESULT TYPE
                ;
00FF =          BSW       EQU       0FFH      ;BOOT SWITCH
                ;
                ; CLEAR DISK STATUS
3000 DB79         IN      RTYPE
3002 DB7B         IN      RBYTE
                ;
                COLDSTART:
3004 DBFF         IN      BSW
3006 E602         ANI     2H        ;SWITCH ON?
3008 C20430       JNZ     COLDSTART
                ;
300B 211E30       LXI     H,BOOTV   ;VIRTUAL BASE
300E 0680         MVI     B,BOOTL   ;LENGTH OF BOOT
3010 118000       LXI     D,BOOTB   ;DESTINATION OF BOOT
3013 7E         MOVE:     MOV       A,M
3014 12           STAX    D         ;TRANSFERRED ONE BYTE
3015 23           INX     H
3016 13           INX     D
3017 05           DCR     B
3018 C21330       JNZ     MOVE
301B C38000       JMP     BOOTB     ;TO BOOT SYSTEM
                ;
                BOOTV:    ;BOOT LOADER PLACE HERE AT SYSTEM GENERATION
089E =          LBIAS     EQU       $-80H+MBIAS       ;COLD START BOOT BEGINS AT 80H
301E              END
```

```
                ; MDS COLD START LOADER FOR CP/M
0000 =          FALSE       EQU     0
FFFF =          TRUE        EQU     NOT FALSE
0000 =          TESTING     EQU     FALSE     ;IF TRUE, THEN GO TO MON80 ON ERRORS
                ;
0010 =          MSIZE       EQU     16        ;MEMORY SIZE IN KILOBYTES
2000 =          CBASE       EQU     (MSIZE-8)*1024      ;CPM BASE ADDRESS BIAS BEYOND 8K
2900 =          BDOSB       EQU     CBASE+900H          ;BASE OF DOS LOAD
3206 =          BDOS        EQU     CBASE+1206H         ;ENTRY TO DOS FOR CALLS
4000 =          BDOSE       EQU     MSIZE*1024          ;END OF DOS LOAD
3E00 =          BOOT        EQU     BDOSE-2*256         ;COLD START ENTRY POINT
3E03 =          RBOOT       EQU     BOOT+3              ;WARM START ENTRY POINT
                ;
0080              ORG       80H       ;LOADED DOWN FROM HARDWARE BOOT AT 3000H
                ;
1700 =          BDOSL       EQU     BDOSE-BDOSB
0002 =          NTRKS       EQU     2         ;NUMBER OF TRACKS TO READ
002E =          BDOSS       EQU     BDOSL/128 ;NUMBER OF SECTORS IN DOS
0019 =          BDOS0       EQU     25        ;NUMBER OF BDOS SECTORS ON TRACK 0
0015 =          BDOS1       EQU     BDOSS-BDOS0         ;NUMBER OF SECTORS ON TRACK 1
                ;
F800 =          MON80       EQU     0F800H    ;INTEL MONITOR BASE
FF0F =          RMON80      EQU     0FF0FH    ;RESTART LOCATION FOR MON80
0078 =          BASE        EQU     078H      ;'BASE' USED BY CONTROLLER
0079 =          RTYPE       EQU     BASE+1    ;RESULT TYPE
007B =          RBYTE       EQU     BASE+3    ;RESULT BYTE
007F =          RESET       EQU     BASE+7    ;RESET CONTROLLER
                ;
0078 =          DSTAT       EQU     BASE      ;DISK STATUS PORT
0079 =          LOW         EQU     BASE+1    ;LOW IOPB ADDRESS
007A =          HIGH        EQU     BASE+2    ;HIGH IOPB ADDRESS
0003 =          RECAL       EQU     3H        ;RECALIBRATE SELECTED DRIVE
0004 =          READF       EQU     4H        ;DISK READ FUNCTION
0100 =          STACK       EQU     100H      ;USE END OF BOOT FOR STACK
                ;
                RSTART:
0080 310001       LXI       SP,STACK;IN CASE OF CALL TO MON80
                ; CLEAR THE CONTROLLER
0083 D37F         OUT       RESET     ;LOGIC CLEARED
                ;
                ;
0085 0602         MVI       B,NTRKS   ;NUMBER OF TRACKS TO READ
0087 21B700       LXI       H,IOPB0
                ;
                START:
                ;
                ; READ FIRST/NEXT TRACK INTO BDOSB
008A 7D           MOV       A,L
```

```
---- --79          OUT       LOW
008D 7C            MOV       A,H
008E D37A          OUT       HIGH
0090 DB78        WAIT0:      IN        DSTAT
0092 E604          ANI       4
0094 CA9000        JZ        WAIT0
                   ;
                   ; CHECK DISK STATUS
0097 DB79          IN        RTYPE
0099 E603          ANI       11B
009B FE02          CPI       2
                   ;
                   IF        TESTING
                   CNC       RMON80    ;GO TO MONITOR IF 11 OR 10
                   ENDIF
                   IF        NOT TESTING
009D D28000        JNC       RSTART    ;RETRY THE LOAD
                   ENDIF
                   ;
00A0 DB7B          IN        RBYTE     ;I/O COMPLETE, CHECK STATUS
                   ; IF NOT READY, THEN GO TO MON80
00A2 17            RAL
00A3 DC0FFF        CC        RMON80    ;NOT READY BIT SET
00A6 1F            RAR                 ;RESTORE
00A7 E61E          ANI       11110B    ;OVERRUN/ADDR ERR/SEEK/CRC/XXXX
                   ;
                   IF        TESTING
                   CNZ       RMON80    ;GO TO MONITOR
                   ENDIF
                   IF        NOT TESTING
00A9 C28000        JNZ       RSTART    ;RETRY THE LOAD
                   ENDIF
                   ;
                   ;
00AC 110700        LXI       D,IOPBL   ;LENGTH OF IOPB
00AF 19            DAD       D         ;ADDRESSING NEXT IOPB
00B0 05            DCR       B         ;COUNT DOWN TRACKS
00B1 C28A00        JNZ       START
                   ;
                   ;
                   ; JMP TO BOOT TO PRINT INITIAL MESSAGE, AND SET UP JMPS
00B4 C3003E        JMP       BOOT
                   ;
                   ; PARAMETER BLOCKS
00B7 80          IOPB0:      DB        80H       ;IOCW, NO UPDATE
00B8 04            DB        READF     ;READ FUNCTION
00B9 19            DB        BDOS0     ;# SECTORS TO READ ON TRACK 0
00BA 00            DB        0         ;TRACK 0
00BB 02            DB        2         ;START WITH SECTOR 2 ON TRACK 0
00BC 0029          DW        BDOSB     ;START AT BASE OF BDOS
```

```
0007 =          IOPBL       EQU         $-IOPB0
                ;
00BE 80         IOPB1:      DB          80H
00BF 04           DB        READF
00C0 15           DB        BDOS1       ;SECTORS TO READ ON TRACK 1
00C1 01           DB        1           ;TRACK 1
00C2 01           DB        1           ;SECTOR 1
00C3 8035         DW        BDOSB+BDOS0*128     ;BASE OF SECOND READ
                ;
00C5              END
```

```
                ; MDS I/O DRIVERS FOR CP/M
                ; VERSION 1.1  OCTOBER, 1976
                ;
                ; COPYRIGHT (C) 1976
                ; DIGITAL RESEARCH
                ; BOX 579, PACIFIC GROVE CA.
                ;
                ;
                ;
0010 =          MSIZE   EQU     16          ;MEMORY SIZE IN KILOBYTES
000B =          VERS    EQU     11          ;CPM VERSION NUMBER
3E00 =          PATCH   EQU     MSIZE*1024-2*256    ;BASE OF THIS MODULE (ABOVE DOS)
                ;
3E00             ORG    PATCH
2000 =          CBASE   EQU     (MSIZE-8)*1024      ;BIAS FOR SYSTEMS LARGER THAN 8K
2900 =          CPMB    EQU     CBASE+900H          ;BASE OF CPM (CONSOLE PROCESSOR ENTRY
3206 =          BDOS    EQU     CBASE+1206H         ;BASIC DOS (RESIDENT PORTION)
1500 =          CPML    EQU     $-CPMB    ;LENGTH (IN BYTES) OF CPM SYSTEM
002A =          NSECTS  EQU     CPML/128  ;NUMBER OF SECTORS TO LOAD
E080 =          LBIAS   EQU     980H-CPMB ;LOADER BIAS VALUE USED IN SYSGEN
0002 =          OFFSET  EQU     2         ;NUMBER OF DISK TRACKS USED BY CP/M
0080 =          BUFF    EQU     80H       ;DEFAULT BUFFER ADDRESS
000A =          RETRY   EQU     10        ;MAX RETRIES ON DISK I/O BEFORE ERROR
                ;
                ; PERFORM FOLLOWING FUNCTIONS
                ; BOOT      COLD START
                ; WBOOT     WARM START (SAVE I/O BYTE)
                ; (BOOT AND WBOOT ARE THE SAME FOR MDS)
                ; CONST     CONSOLE STATUS
                ;           REG-A = 00 IF NO CHARACTER READY
                ;           REG-A = FF IF CHARACTER READY
                ; CONIN     CONSOLE CHARACTER IN (RESULT IN REG-A)
                ; CONOUT    CONSOLE CHARACTER OUT (CHAR IN REG-C)
                ; LIST      LIST OUT (CHAR IN REG-C)
                ; PUNCH     PUNCH OUT (CHAR IN REG-C)
                ; READER    PAPER TAPE READER IN (RESULT TO REG-A)
                ; HOME      MOVE TO TRACK 00
                ;
                ; (THE FOLLOWING CALLS SET-UP THE IO PARAMETER BLOCK FOR THE
                ; MDS, WHICH IS USED TO PERFORM SUBSEQUENT READS AND WRITES)
                ; SELDSK    SELECT DISK GIVEN BY REG-C (0,1,2...)
                ; SETTRK    SET TRACK ADDRESS (0,...76) FOR SUBSEQUENT READ/WRITE
                ; SETSEC    SET SECTOR ADDRESS (1,...,26) FOR SUBSEQUENT READ/WRITE
                ; SETDMA    SET SUBSEQUENT DMA ADDRESS (INITIALLY 80H)
                ;
                ; (READ AND WRITE ASSUME PREVIOUS CALLS TO SET UP THE IO PARAMETERS)
                ; READ      READ TRACK/SECTOR TO PRESET DMA ADDRESS
                ; WRITE     WRITE TRACK/SECTOR FROM PRESET DMA ADDRESS
```

```
                        ;
                        ; JUMP VECTOR FOR INDIVIUAL ROUTINES
3E00 C3443E             JMP     BOOT
3E03 C3543E             WBOOTE: JMP     WBOOT
3E06 C3073F             JMP     CONST
3E09 C30A3F             JMP     CONIN
3E0C C3103F             JMP     CONOUT
3E0F C3293F             JMP     LIST
3E12 C32C3F             JMP     PUNCH
3E15 C32F3F             JMP     READER
3E18 C3323F             JMP     HOME
3E1B C3373F             JMP     SELDSK
3E1E C3503F             JMP     SETTRK
3E21 C3553F             JMP     SETSEC
3E24 C35A3F             JMP     SETDMA
3E27 C3603F             JMP     READ
3E2A C3693F             JMP     WRITE
                        ;
                        ;
                        ; END OF CONTROLLER - INDEPENDENT CODE, THE REMAINING SUBROUTINES
                        ; ARE TAILORED TO THE PARTICULAR OPERATING ENVIRNOMENT, AND MUST
                        ; BE ALTERED FOR ANY SYSTEM WHICH DIFFERS FROM THE INTEL MDS.
                        ;
                        ; THE FOLLOWING CODE ASSUMES THE MDS MONITOR EXISTS AT 0F800H
                        ; AND USES THE I/O SUBROUTINES WITHIN THE MONITOR
                        ;
                        ; WE ALSO ASSUME THE MDS SYSTEM HAS TWO DISK DRIVES AVAILABLE
0002 =          NDISKS  EQU     2               ;NUMBER OF DRIVES AVAILABLE
00FD =          REVRT   EQU     0FDH            ;INTERRUPT REVERT PORT
00FC =          INTC    EQU     0FCH            ;INTERRUPT MASK PORT
00F3 =          ICON    EQU     0F3H            ;INTERRUPT CONTROL PORT
007E =          INTE    EQU     01111$1110B             ;ENABLE RST 0(WARM BOOT), RST 7 (MONI
                        ;
                        ; MDS MONITOR EQUATES
F800 =          MON80   EQU     0F800H  ;MDS MONITOR
FF0F =          RMON80  EQU     0FF0FH  ;RESTART MON80 (DISK SELECT ERROR)
F803 =          CI      EQU     0F803H  ;CONSOLE CHARACTER TO REG-A
F806 =          RI      EQU     0F806H  ;READER IN TO REG-A
F809 =          CO      EQU     0F809H  ;CONSOLE CHAR FROM C TO CONSOLE OUT
F80C =          PO      EQU     0F80CH  ;PUNCH CHAR FROM C TO PUNCH DEVICE
F80F =          LO      EQU     0F80FH  ;LIST FROM C TO LIST DEVICE
F812 =          CSTS    EQU     0F812H  ;CONSOLE STATUS 00/FF TO REGISTER A
                        ;
                        ; DISK PORTS AND COMMANDS
0078 =          BASE    EQU     78H     ;BASE OF DISK COMMAND IO PORTS
0078 =          DSTAT   EQU     BASE    ;DISK STATUS (INPUT)
0079 =          RTYPE   EQU     BASE+1  ;RESULT TYPE (INPUT)
007B =          RBYTE   EQU     BASE+3  ;RESULT BYTE (INPUT)
                        ;
0079 =          LOW     EQU     BASE+1  ;IOPB LOW ADDRESS (OUTPUT)
```

```
                HIGH    EQU     BASE+2  ;IOPB HIGH ADDRESS (OUTPUT)
                ;
0004 =          READF   EQU     4H      ;READ FUNCTION
0006 =          WRITF   EQU     6H      ;WRITE FUNCTION
0003 =          RECAL   EQU     3H      ;RECALIBRATE DRIVE
0004 =          IORDY   EQU     4H      ;I/O FINISHED MASK
000D =          CR      EQU     0DH     ;CARRIAGE RETURN
000A =          LF      EQU     0AH     ;LINE FEED
                ;
                SIGNON: ;SIGNON MESSAGE: XXK CP/M VERS Y.Y
3E2D 0D0A0A       DB    CR,LF,LF
3E30 3136         DB    MSIZE/10+'0',MSIZE MOD 10 + '0'
3E32 4B2043502F   DB    'K CP/M VERS '
3E3E 312E31       DB    VERS/10+'0','.',VERS MOD 10+'0'
3E41 0D0A00       DB    CR,LF,0
                ;
                BOOT:   ;PRINT SIGNON MESSAGE AND GO TO DOS
3E44 310001       LXI   SP,BUFF+80H
3E47 212D3E       LXI   H,SIGNON
3E4A CD723F       CALL  PRMSG     ;PRINT MESSAGE
3E4D AF           XRA   A         ;CLEAR ACCUMULATOR
3E4E 32D33F       STA   DISKT     ;SELECT DISK 0 ON ENTRY
3E51 C3A63E       JMP   GOCPM     ;GO TO CP/M
                ;
                ;
                WBOOT:; LOADER ON TRACK 0, SECTOR 1, WHICH WILL BE SKIPPED FOR WARM
                ; READ CP/M FROM DISK - ASSUMING THERE IS A 128 BYTE COLD START
                ; START.
                ;
3E54 318000       LXI   SP,BUFF   ;USING DMA - THUS 80 THRU FF AVAILABLE FOR STACK
3E57 3AD23F       LDA   DISKN     ;CURRENTLY LOGGED DISK, RETURN TO DISKN IF NOT 0
3E5A 32D33F       STA   DISKT     ;STORE INTO DISK TEMP SINCE WE BOOT OFF OF 0
                ;
3E5D 0E0A         MVI   C,RETRY   ;MAX RETRIES
3E5F C5           PUSH  B
                WBOOT0: ;ENTER HERE ON ERROR RETRIES
3E60 010029       LXI   B,CPMB    ;SET DMA ADDRESS TO START OF DISK SYSTEM
3E63 CD5A3F       CALL  SETDMA
3E66 0E02         MVI   C,2       ;START READING SECTOR 2
3E68 CD553F       CALL  SETSEC
3E6B 0E00         MVI   C,0       ;START READING TRACK 0
3E6D CD503F       CALL  SETTRK
3E70 0E00         MVI   C,0       ;START WITH DISK 0
3E72 CD373F       CALL  SELDSK    ;CHANGES DISKN TO 0
                ;
                ; READ SECTORS, COUNT NSECTS TO ZERO
3E75 C1           POP   B         ;10-ERROR COUNT
3E76 062A         MVI   B,NSECTS
                RDSEC:  ;READ NEXT SECTOR
3E78 C5           PUSH  B         ;SAVE SECTOR COUNT
```

```
3E79 CD603F     CALL    READ
3E7C C2E03E     JNZ     BOOTERR   ;RETRY IF ERRORS OCCUR
3E7F 2AD93F     LHLD    IOD       ;INCREMENT DMA ADDRESS
3E82 118000     LXI     D,128     ;SECTOR SIZE
3E85 19         DAD     D         ;INCREMENTED DMA ADDRESS IN HL
3E86 44         MOV     B,H
3E87 4D         MOV     C,L       ;READY FOR CALL TO SET DMA
3E88 CD5A3F     CALL    SETDMA
3E8B 3AD83F     LDA     IOS       ;SECTOR NUMBER JUST READ
3E8E FE1A       CPI     26        ;READ LAST SECTOR?
3E90 DA9C3E     JC      RD1
                ; MUST BE SECTOR 26, ZERO AND GO TO NEXT TRACK
3E93 3AD73F     LDA     IOT       ;GET TRACK TO REGISTER A
3E96 3C         INR     A
3E97 4F         MOV     C,A       ;READY FOR CALL
3E98 CD503F     CALL    SETTRK
3E9B AF         XRA     A         ;CLEAR SECTOR NUMBER
3E9C 3C         RD1:    INR     A         ;TO NEXT SECTOR
3E9D 4F         MOV     C,A       ;READY FOR CALL
3E9E CD553F     CALL    SETSEC
3EA1 C1         POP     B         ;RECALL SECTOR COUNT
3EA2 05         DCR     B         ;DONE?
3EA3 C2783E     JNZ     RDSEC
                ;
                ; DONE WITH THE LOAD, RESET DEFAULT BUFFER ADDRESS
                GOCPM:  ;(ENTER HERE FROM COLD START BOOT)
                ; ENABLE RST0 AND RST7
3EA6 F3         DI
3EA7 3E12       MVI     A,12H     ;INITIALIZE COMMAND
3EA9 D3FD       OUT     REVRT
3EAB AF         XRA     A
3EAC D3FC       OUT     INTC      ;CLEARED
3EAE 3E7E       MVI     A,INTE    ;RST0 AND RST7 BITS ON
3EB0 D3FC       OUT     INTC
3EB2 AF         XRA     A
3EB3 D3F3       OUT     ICON      ;INTERRUPT CONTROL
                ;
                ; SET DEFAULT BUFFER ADDRESS TO 80H
3EB5 018000     LXI     B,BUFF
3EB8 CD5A3F     CALL    SETDMA
                ;
                ; RESET MONITOR ENTRY POINTS
3EBB 3EC3       MVI     A,JMP
3EBD 320000     STA     0
3EC0 21033E     LXI     H,WBOOTE
3EC3 220100     SHLD    1         ;JMP WBOOT AT LOCATION 00
3EC6 320500     STA     5
3EC9 210632     LXI     H,BDOS
3ECC 220600     SHLD    6         ;JMP BDOS AT LOCATION 5
3ECF 323800     STA     7*8       ;JMP TO MON80 (MAY HAVE BEEN CHANGED BY DDT)
```

```
3ED2 2100F8      LXI     H,MON80
3ED5 223900      SHLD    7*8+1
                 ; LEAVE IOBYTE SET
                 ; PREVIOUSLY SELECTED DISK WAS B, SEND PARAMETER TO CPM
3ED8 3AD33F      LDA     DISKT
3EDB 4F          MOV     C,A        ;LOOKS LIKE A SINGLE PARAMETER TO CPM
3EDC FB          EI
3EDD C30029      JMP     CPMB
                 ;
                 ; ERROR CONDITION OCCURRED, PRINT MESSAGE AND RETRY
                 BOOTERR:
3EE0 C1          POP     B          ;RECALL COUNTS
3EE1 0D          DCR     C
3EE2 CAE93E      JZ      BOOTER0
                 ; TRY AGAIN
3EE5 C5          PUSH    B
3EE6 C3603E      JMP     WBOOT0
                 ;
                 BOOTER0:
                 ; OTHERWISE TOO MANY RETRIES
3EE9 21F23E      LXI     H,BOOTMSG
3EEC CD7F3F      CALL    ERROR
3EEF C3543E      JMP     WBOOT      ;FOR ANOTHER TRY
                 ;
                 BOOTMSG:
3EF2 2A43414E4E  DB      '*CANNOT BOOT SYSTEM*',0
                 ;
                 ;
                 CONST:   ;CONSOLE STATUS TO REG-A
                 ; (EXACTLY THE SAME AS MDS CALL)
3F07 C312F8      JMP     CSTS
                 ;
                 CONIN:   ;CONSOLE CHARACTER TO REG-A
3F0A CD03F8      CALL    CI
3F0D E67F        ANI     7FH        ;REMOVE PARITY BIT
3F0F C9          RET
                 ;
                 CONOUT:  ;CONSOLE CHARACTER FROM C TO CONSOLE OUT
                 ; SAME AS MDS CALL, BUT WAIT FOR SLOW CONSOLES ON LINE FEED
3F10 79          MOV     A,C        ;GET CHARACTER TO ACCUM
3F11 FE0A        CPI     LF         ;END OF LINE?
3F13 F5          PUSH    PSW        ;SAVE CONDITION FOR LATER
3F14 CD09F8      CALL    CO         ;SEND THE CHARACTER (MAY BE LINE FEED)
3F17 F1          POP     PSW
3F18 C0          RNZ                ;RETURN IF IT WASN'T A LINE FEED
                 ;
                 ; WAIT 13 CHARACTER TIMES (AT 2400 BAUD) FOR LINE FEED TO HAPPEN
                 ; (THIS WORKS OUT TO ABOUT 50 MILLISECS)
3F19 0632        MVI     B,50       ;NUMBER OF MILLISECS TO WAIT
3F1B 0EB6        T1:     MVI     C,182      ;COUNTER TO CONTROL 1 MILLISEC LOOP
```

```
3F1D 0D        T2:       DCR     C          ;1 CYCLE = .5 USEC
3F1E C21D3F      JNZ     T2      ;10 CYCLES= 5.5 USEC
                ;                ----------
                ;                =    5.5 USEC PER LOOP*  182 = 1001 USEC
3F21 05          DCR     B
3F22 C21B3F      JNZ     T1      ;FOR ANOTHER LOOP
3F25 C9          RET
                ;
                ;
3F26 C309F8      JMP     CO
                ;
                LIST:     ;LIST DEVICE OUT
                ; (EXACTLY THE SAME AS MDS CALL)
3F29 C30FF8      JMP     LO
                ;
                PUNCH:    ;PUNCH DEVICE OUT
                ; (EXACTLY THE SAME AS MDS CALL)
3F2C C30CF8      JMP     PO
                ;
                READER:   ;READER CHARACTER IN TO REG-A
                ; (EXACTLY THE SAME AS MDS CALL)
3F2F C306F8      JMP     RI
                ;
                HOME:     ;MOVE TO HOME POSITION
                ; TREAT AS TRACK 00 SEEK
3F32 0E00        MVI     C,0
3F34 C3503F      JMP     SETTRK
                ;
                SELDSK:   ;SELECT DISK GIVEN BY REGISTER C
                ; CP/M HAS CHECKED FOR DISK SELECT 0 OR 1, BUT WE MAY HAVE
                ; A SINGLE DRIVE MDS SYSTEM, SO CHECK AGAIN AND GIVE ERROR
                ; BY CALLING MON80
3F37 79          MOV     A,C
3F38 FE02        CPI     NDISKS    ;TOO LARGE?
3F3A D40FFF      CNC     RMON80    ;GIVES #ADDR MESSAGE AT CONSOLE
3F3D 32D23F      STA     DISKN     ;SELECT DISK N
                ;
3F40 17          RAL
3F41 17          RAL
3F42 17          RAL
3F43 17          RAL
3F44 E610        ANI     10000B ,  ;UNIT NUMBER IN POSITION
3F46 4F          MOV     C,A       ;SAVE IT
3F47 21D53F      LXI     H,IOF     ;IO FUNCTION
3F4A 7E          MOV     A,M
3F4B E6CF        ANI     11001111B ;MASK OUT DISK NUMBER
3F4D B1          ORA     C         ;MASK IN NEW DISK NUMBER
3F4E 77          MOV     M,A       ;SAVE IT IN IOPB
3F4F C9          RET
                ;
```

```
                ;
                SETTRK:     ;SET TRACK ADDRESS GIVEN BY C
3F50 21D73F       LXI       H,IOT
3F53 71           MOV       M,C
3F54 C9           RET
                ;
                SETSEC:     ;SET SECTOR NUMBER GIVEN BY C
3F55 21D83F       LXI       H,IOS
3F58 71           MOV       M,C
3F59 C9           RET
                ;
                SETDMA:     ;SET DMA ADDRESS GIVEN BY REGS B,C
3F5A 69           MOV       L,C
3F5B 60           MOV       H,B
3F5C 22D93F       SHLD      IOD
3F5F C9           RET
                ;
                READ:       ;READ NEXT DISK RECORD (ASSUMING DISK/TRK/SEC/DMA SET)
3F60 0E04         MVI       C,READF   ;SET TO READ FUNCTION
3F62 CD903F       CALL      SETFUNC
3F65 CD993F       CALL      WAITIO    ;PERFORM READ FUNCTION
3F68 C9           RET                 ;MAY HAVE ERROR SET IN REG-A
                ;
                ;
                WRITE:      ;DISK WRITE FUNCTION
3F69 0E06         MVI       C,WRITF
3F6B CD903F       CALL      SETFUNC   ;SET TO WRITE FUNCTION
3F6E CD993F       CALL      WAITIO
3F71 C9           RET                 ;MAY HAVE ERROR SET
                ;
                ;
                ; UTILITY SUBROUTINES
                PRMSG:      ;PRINT MESSAGE AT H,L TO 0
3F72 7E           MOV       A,M
3F73 B7           ORA       A         ;ZERO?
3F74 C8           RZ
                ; MORE TO PRINT
3F75 E5           PUSH      H
3F76 4F           MOV       C,A
3F77 CD09F8       CALL      CO
3F7A E1           POP       H
3F7B 23           INX       H
3F7C C3723F       JMP       PRMSG
                ;
                ERROR:      ;ERROR MESSAGE ADDRESSES BY H,L
3F7F CD723F       CALL      PRMSG
                ; ERROR MESSAGE WRITTEN, WAIT FOR RESPONSE FROM CONSOLE
3F82 CD0A3F       CALL      CONIN
3F85 0E0D         MVI       C,CR      ;CARRIAGE RETURN
3F87 CD103F       CALL      CONOUT
```

```
3F8A 0E0A       MVI     C,LF        ;LINE FEED
3F8C CD103F     CALL    CONOUT
3F8F C9         RET                 ;MAY BE RETURNING FOR ANOTHER RETRY
                ;
                SETFUNC:
                ; SET FUNCTION FOR NEXT I/O (COMMAND IN REG-C)
3F90 21D53F     LXI     H,IOF       ;IO FUNCTION ADDRESS
3F93 7E         MOV     A,M         ;GET IT TO ACCUMULATOR FOR MASKING
3F94 E6F8       ANI     11111000B ;REMOVE PREVIOUS COMMAND
3F96 B1         ORA     C           ;SET TO NEW COMMAND
3F97 77         MOV     M,A         ;REPLACED IN IOPB
3F98 C9         RET
                ;
                WAITIO:
3F99 0E0A       MVI     C,RETRY     ;MAX RETRIES BEFORE PERM ERROR
                REWAIT:
                ; START THE I/O FUNCTION AND WAIT FOR COMPLETION
3F9B DB79       IN      RTYPE
3F9D DB7B       IN      RBYTE       ;CLEARS THE CONTROLLER
                ;
3F9F 3ED4       MVI     A,IOPB AND 0FFH     ;LOW ADDRESS FOR IOPB
3FA1 D379       OUT     LOW                 ;TO THE CONTROLLER
3FA3 3E3F       MVI     A,IOPB SHR 8        ;HIGH ADDRESS FOR IOPB
3FA5 D37A       OUT     HIGH                ;TO THE CONTROLLER, STARTS OPERATION
                ;
3FA7 DB78       WAIT0:  IN      DSTAT               ;WAIT FOR COMPLETION
3FA9 E604       ANI     IORDY               ;READY?
3FAB CAA73F     JZ      WAIT0
                ;
                ; CHECK IO COMPLETION OK
3FAE DB79       IN      RTYPE               ;MUST BE I/O COMPLETE (00) UNLINKED
                ; 00 UNLINKED I/O COMPLETE,  01 LINKED I/O COMPLETE (NOT USED)
                ; 10 DISK STATUS CHANGED     11 (NOT USED)
3FB0 FE02       CPI     10B                 ;READY STATUS CHANGE?
3FB2 CAC63F     JZ      WREADY
                ;
                ; MUST BE 00 IN THE ACCUMULATOR
3FB5 B7         ORA     A
3FB6 C2CB3F     JNZ     WERROR              ;SOME OTHER CONDITION, RETRY
                ;
                ; CHECK I/O ERROR BITS
3FB9 DB7B       IN      RBYTE
3FBB 17         RAL
3FBC DAC63F     JC      WREADY              ;UNIT NOT READY
3FBF 1F         RAR
3FC0 E6FE       ANI     11111110B ;ANY OTHER ERRORS?  (DELETED DATA OK)
3FC2 C2CB3F     JNZ     WERROR
                ;
                ; READ OR WRITE IS OK, ACCUMULATOR CONTAINS ZERO
3FC5 C9         RET
```

```
                        ;
                        WREADY:     ;NOT READY, TREAT AS ERROR FOR NOW
3FC6 DB7B                 IN        RBYTE     ;CLEAR RESULT BYTE
3FC8 C3CB3F               JMP       TRYCOUNT
                        ;
                        WERROR:     ;RETURN HARDWARE MALFUNCTION (CRC, TRACK, SEEK, ETC.)
                        ; THE MDS CONTROLLER HAS RETURNED A BIT IN EACH POSITION
                        ; OF THE ACCUMULATOR, CORRESPONDING TO THE CONDITIONS:
                        ; 0          - DELETED DATA (ACCEPTED AS OK ABOVE)
                        ; 1          - CRC ERROR
                        ; 2          - SEEK ERROR
                        ; 3          - ADDRESS ERROR (HARDWARE MALFUNCTION)
                        ; 4          - DATA OVER/UNDER FLOW (HARDWARE MALFUNCTION)
                        ; 5          - WRITE PROTECT (TREATED AS NOT READY)
                        ; 6          - WRITE ERROR (HARDWARE MALFUNCTION)
                        ; 7          - NOT READY
                        ; (ACCUMULATOR BITS ARE NUMBERED 7 6 5 4 3 2 1 0)
                        ;
                        ; IT MAY BE USEFUL TO FILTER OUT THE VARIOUS CONDITIONS,
                        ; BUT WE WILL GET A PERMANENT ERROR MESSAGE IF IT IS NOT
                        ; RECOVERABLE.  IN ANY CASE, THE NOT READY CONDITION IS
                        ; TREATED AS A SEPARATE CONDITION FOR LATER IMPROVEMENT
                        TRYCOUNT:
                        ; REGISTER C CONTAINS RETRY COUNT, DECREMENT 'TIL ZERO
3FCB 0D                   DCR       C
3FCC C29B3F               JNZ       REWAIT    ;FOR ANOTHER TRY
                        ;
                        ; CANNOT RECOVER FROM ERROR
3FCF 3E01                 MVI       A,1       ;ERROR CODE
3FD1 C9                   RET
                        ;
                        ;
                        ; DATA AREAS (MUST BE IN RAM)
3FD2 00                 DISKN:      DB        0         ;CURRENT DISK
3FD3 00                 DISKT:      DB        0         ;TEMP FOR CURRENT DISK DURING WARM START
                        IOPB:       ;IO PARAMETER BLOCK
3FD4 80                   DB        80H       ;NORMAL I/O OPERATION
3FD5 04                 IOF:        DB        READF     ;IO FUNCTION, INITIAL READ
3FD6 01                 ION:        DB        1         ;NUMBER OF SECTORS TO READ
3FD7 02                 IOT:        DB        OFFSET    ;TRACK NUMBER
3FD8 01                 IOS:        DB        1         ;SECTOR NUMBER
3FD9 8000               IOD:        DW        BUFF      ;IO ADDRESS
                        ;
                        ;
3FDB                      END
```

```
                ; SKELETAL CBIOS FOR FIRST LEVEL OF CP/M ALTERATION
                ;
                ; NOTE :  MSIZE DETERMINES WHERE THIS CBIOS IS LOCATED
0010 =          MSIZE     EQU     16        ;CP/M VERSION MEMORY SIZE IN KILOBYTES
3E00 =          PATCH     EQU     MSIZE*1024-2*256    ;START OF THE CBIOS PATCH
                ;
                ; WE WILL USE THE AREA RESERVED STARTING AT LOCATION
                ; 40H IN PAGE 0 FOR HOLDING THE VALUES OF:
                ;           TRACK   =  LAST SELECTED TRACK
                ;           SECTOR  =  LAST SELECTED SECTOR
                ;           DMAAD   =  LAST SELECTED DMA ADDRESS
                ;           DISKNO  =  LAST SELECTED DISK NUMBER
                ; (NOTE THAT ALL ARE BYTE VALUES EXCEPT FOR DMAAD)
                ;
0040 =          SCRAT     EQU     40H                 ;BASE OF SCRATCH AREA (FROM 40H TO 4FH
0040 =          TRACK     EQU     SCRAT               ;CURRENTLY SELECTED TRACK
0041 =          SECTOR    EQU     SCRAT+1             ;CURRENTLY SELECTED SECTOR
0042 =          DMAAD     EQU     SCRAT+2             ;CURRENT DMA ADDRESS
0046 =          DISKNO    EQU     DMAAD+4   ;CURRENT DISK NUMBER
                ;
                ;
3E00             ORG      PATCH     ;ORGIN OF THIS PROGRAM
0000 =          CBASE     EQU     (MSIZE-16)*1024     ;BIAS FOR SYSTEMS LARGER THAN 16K
2900 =          CPMB      EQU     CBASE+2900H         ;BASE OF CP/M (= BASE OF CCP)
3206 =          BDOS      EQU     CBASE+3206H         ;BASE OF RESIDENT PORTION OF CP/M
1500 =          CPML      EQU     $-CPMB              ;LENGTH OF THE CP/M SYSTEM IN BYTES
002A =          NSECTS    EQU     CPML/128  ;NUMBER OF SECTORS TO LOAD ON WARM START
                ;
                ; JUMP VECTOR FOR INDIVIDUAL SUBROUTINES
3E00 C32D3E      JMP      BOOT                ;COLD START
                WBOOTE:
3E03 C3303E      JMP      WBOOT               ;WARM START
3E06 C3993E      JMP      CONST               ;CONSOLE STATUS
3E09 C3AC3E      JMP      CONIN               ;CONSOLE CHARACTER IN
3E0C C3BF3E      JMP      CONOUT              ;CONSOLE CHARACTER OUT
3E0F C3D13E      JMP      LIST                ;LIST CHARACTER OUT
3E12 C3D33E      JMP      PUNCH               ;PUNCH CHARACTER OUT
3E15 C3D53E      JMP      READER              ;READER CHARACTER OUT
3E18 C3DA3E      JMP      HOME                ;MOVE HEAD TO HOME POSITION
3E1B C3E03E      JMP      SELDSK              ;SELECT DISK
3E1E C3F53E      JMP      SETTRK              ;SET TRACK NUMBER
3E21 C30A3F      JMP      SETSEC              ;SET SECTOR NUMBER
3E24 C31F3F      JMP      SETDMA              ;SET DMA ADDRESS
3E27 C3353F      JMP      READ                ;READ DISK
3E2A C3483F      JMP      WRITE               ;WRITE DISK
                ;
                ;
                ; INDIVIDUAL SUBROUTINES TO PERFORM EACH FUNCTION
```

```
                BOOT:   ;SIMPLEST CASE IS TO JUST PERFORM PARAMETER INITIALIZATION
3E2D C3793E       JMP   GOCPM             ;INITIALIZE AND GO TO CP/M
                ;
                WBOOT:  ;SIMPLEST CASE IS TO READ THE DISK UNTIL ALL SECTORS LOADED
3E30 318000       LXI   SP,80H            ;USE SPACE BELOW BUFFER FOR STACK
3E33 0E00         MVI   C,0               ;SELECT DISK 0
3E35 CDE03E       CALL  SELDSK
3E38 CDDA3E       CALL  HOME              ;GO TO TRACK 00
                ;
3E3B 062A         MVI   B,NSECTS  ;B COUNTS THE NUMBER OF SECTORS TO LOAD
3E3D 0E00         MVI   C,0               ;C HAS THE CURRENT TRACK NUMBER
3E3F 1602         MVI   D,2               ;D HAS THE NEXT SECTOR TO READ
                ; NOTE THAT WE BEGIN BY READING TRACK 0, SECTOR 2 SINCE SECTOR 1
                ; CONTAINS THE COLD START LOADER, WHICH IS SKIPPED IN A WARM START
3E41 210029       LXI   H,CPMB            ;BASE OF CP/M (INITIAL LOAD POINT)
                LOAD1:  ;LOAD ONE MORE SECTOR
3E44 C5           PUSH  B         ;SAVE SECTOR COUNT, CURRENT TRACK
3E45 D5           PUSH  D         ;SAVE NEXT SECTOR TO READ
3E46 E5           PUSH  H         ;SAVE DMA ADDRESS
3E47 4A           MOV   C,D       ;GET SECTOR ADDRESS TO REGISTER C
3E48 CD0A3F       CALL  SETSEC    ;SET SECTOR ADDRESS FROM REGISTER C
3E4B C1           POP   B         ;RECALL DMA ADDRESS TO B,C
3E4C C5           PUSH  B         ;REPLACE ON STACK FOR LATER RECALL
3E4D CD1F3F       CALL  SETDMA    ;SET DMA ADDRESS FROM B,C
                ;
                ; DRIVE SET TO 0, TRACK SET, SECTOR SET, DMA ADDRESS SET
3E50 CD353F       CALL  READ
3E53 FE00         CPI   00H       ;ANY ERRORS?
3E55 C2303E       JNZ   WBOOT     ;RETRY THE ENTIRE BOOT IF AN ERROR OCCURS
                ;
                ; NO ERROR, MOVE TO NEXT SECTOR
3E58 E1           POP   H         ;RECALL DMA ADDRESS
3E59 118000       LXI   D,128     ;DMA=DMA+128
3E5C 19           DAD   D         ;NEW DMA ADDRESS IS IN H,L
3E5D D1           POP   D         ;RECALL SECTOR ADDRESS
3E5E C1           POP   B         ;RECALL NUMBER OF SECTORS REMAINING, AND CURRENT TRK
3E5F 05           DCR   B         ;SECTORS=SECTORS-1
3E60 CA793E       JZ    GOCPM     ;TRANSFER TO CP/M IF ALL HAVE BEEN LOADED
                ;
                ; MORE SECTORS REMAIN TO LOAD, CHECK FOR TRACK CHANGE
3E63 14           INR   D
3E64 7A           MOV   A,D       ;SECTOR=27?, IF SO, CHANGE TRACKS
3E65 FE1B         CPI   27
3E67 DA443E       JC    LOAD1     ;CARRY GENERATED IF SECTOR<27
                ;
                ; END OF CURRENT TRACK, GO TO NEXT TRACK
3E6A 1601         MVI   D,1       ;BEGIN WITH FIRST SECTOR OF NEXT TRACK
3E6C 0C           INR   C         ;TRACK=TRACK+1
                ;
                ; SAVE REGISTER STATE, AND CHANGE TRACKS
```

```
3E6D C5          PUSH    B
3E6E D5          PUSH    D
3E6F E5          PUSH    H
3E70 CDF53E      CALL    SETTRK    ;TRACK ADDRESS SET FROM REGISTER C
3E73 E1          POP     H
3E74 D1          POP     D
3E75 C1          POP     B
3E76 C3443E      JMP     LOAD1     ;FOR ANOTHER SECTOR
                 ;
                 ; END OF LOAD OPERATION, SET PARAMETERS AND GO TO CP/M
                 GOCPM:
3E79 3EC3        MVI     A,0C3H    ;C3 IS A JMP INSTRUCTION
3E7B 320000      STA     0         ;FOR JMP TO WBOOT
3E7E 21033E      LXI     H,WBOOTE  ;WBOOT ENTRY POINT
3E81 220100      SHLD    1         ;SET ADDRESS FIELD FOR JMP AT 0
                 ;
3E84 320500      STA     5         ;FOR JMP TO BDOS
3E87 210632      LXI     H,BDOS    ;BDOS ENTRY POINT
3E8A 220600      SHLD    6         ;ADDRESS FIELD OF JUMP AT 5 TO BDOS
                 ;
3E8D 018000      LXI     B,80H     ;DEFAULT DMA ADDRESS IS 80H
3E90 CD1F3F      CALL    SETDMA
                 ;
3E93 FB          EI                ;ENABLE THE INTERRUPT SYSTEM
                 ; FUTURE VERSIONS OF CCP WILL SELECT THE DISK GIVEN BY REGISTER
                 ; C UPON ENTRY, HENCE ZERO IT IN THIS VERSION OF THE BIOS FOR
                 ; FUTURE COMPATIBILITY.
3E94 0E00        MVI     C,0       ;SELECT DISK ZERO AFTER INITIALIZATION
3E96 C30029      JMP     CPMB      ;GO TO CP/M FOR FURTHER PROCESSING
                 ;
                 ;
                 ; SIMPLE I/O HANDLERS (MUST BE FILLED IN BY USER)
                 ; IN EACH CASE, THE ENTRY POINT IS PROVIDED, WITH SPACE RESERVED
                 ; TO INSERT YOUR OWN CODE
                 ;
                 CONST:    ;CONSOLE STATUS, RETURN 0FFH IF CHARACTER READY, 00H IF NOT
3E99             DS      10H       ;SPACE FOR STATUS SUBROUTINE
3EA9 3E00        MVI     A,00H
3EAB C9          RET
                 ;
                 CONIN:    ;CONSOLE CHARACTER INTO REGISTER A
3EAC             DS      10H       ;SPACE FOR INPUT ROUTINE
3EBC E67F        ANI     7FH       ;STRIP PARITY BIT
3EBE C9          RET
                 ;
                 CONOUT: ;CONSOLE CHARACTER OUTPUT FROM REGISTER C
3EBF 79          MOV     A,C       ;GET TO ACCUMULATOR
3EC0             DS      10H       ;SPACE FOR OUTPUT ROUTINE
3ED0 C9          RET
                 ;
```

```
                LIST:   ;LIST CHARACTER FROM REGISTER C
3ED1 79           MOV   A,C       ;CHARACTER TO REGISTER A
3ED2 C9           RET             ;NULL SUBROUTINE
                ;
                PUNCH:  ;PUNCH CHARACTER FROM REGISTER C
3ED3 79           MOV   A,C       ;CHARACTER TO REGISTER A
3ED4 C9           RET             ;NULL SUBROUTINE
                ;
                ;
                READER: ;READ CHARACTER INTO REGISTER A FROM READER DEVICE
3ED5 3E1A         MVI   A,1AH     ;ENTER END OF FILE FOR NOW (REPLACE LATER)
3ED7 E67F         ANI   7FH       ;REMEMBER TO STRIP PARITY BIT
3ED9 C9           RET
                ;
                ;
                ; I/O DRIVERS FOR THE DISK FOLLOW
                ; FOR NOW, WE WILL SIMPLY STORE THE PARAMETERS AWAY FOR USE
                ; IN THE READ AND WRITE SUBROUTINES
                ;
                HOME:   ;MOVE TO THE TRACK 00 POSITION OF CURRENT DRIVE
                ; TRANSLATE THIS CALL INTO A SETTRK CALL WITH PARAMETER 00
3EDA 0E00         MVI   C,0       ;SELECT TRACK 0
3EDC CDF53E       CALL  SETTRK
3EDF C9           RET             ;WE WILL MOVE TO 00 ON FIRST READ/WRITE
                ;
                SELDSK: ;SELECT DISK GIVEN BY REGISTER C
3EE0 79           MOV   A,C
3EE1 324600       STA   DISKNO
3EE4              DS    10H       ;SPACE FOR DISK SELECTION ROUTINE
3EF4 C9           RET
                ;
                SETTRK: ;SET TRACK GIVEN BY REGISTER C
3EF5 79           MOV   A,C
3EF6 324000       STA   TRACK
3EF9              DS    10H       ;SPACE FOR TRACK SELECT
3F09 C9           RET
                ;
                SETSEC: ;SET SECTOR GIVEN BY REGISTER C
3F0A 79           MOV   A,C
3F0B 324100       STA   SECTOR
3F0E              DS    10H       ;SPACE FOR SECTOR SELECT
3F1E C9           RET
                ;
                SETDMA: ;SET DMA ADDRESS GIVEN BY REGISTERS B AND C
3F1F 69           MOV   L,C       ;LOW ORDER ADDRESS
3F20 60           MOV   H,B       ;HIGH ORDER ADDRESS
3F21 224200       SHLD  DMAAD     ;SAVE THE ADDRESS
3F24              DS    10H       ;SPACE FOR SETTING THE DMA ADDRESS
3F34 C9           RET
                ;
```

```
                READ:       ;PERFORM READ OPERATION (USUALLY THIS IS SIMILAR TO WRITE
                ; SO WE WILL ALLOW SPACE TO SET UP READ COMMAND, THEN USE
                ; COMMON CODE IN WRITE)
3F35              DS        10H       ;SET UP READ COMMAND
3F45 C3583F       JMP       WAITIO    ;TO PERFORM THE ACTUAL I/O
                ;
                WRITE:      ;PERFORM A WRITE OPERATION
3F48              DS        10H       ;SET UP 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, WE HAVE SAVED THE DISK NUMBER IN 'DISKNO' (0,1)
                ;                                THE TRACK NUMBER IN 'TRACK' (0-76)
                ;                                THE SECTOR NUMBER IN 'SECTOR' (1-26)
                ;                                THE DMA ADDRESS IN 'DMAAD' (0-65535)
                ; ALL REMAINING SPACE FROM $ THROUGH MSIZE*1024-1 IS AVAILABLE:
00A7 =          LEFT        EQU       (MSIZE*1024-1)-$    ;SPACE REMAINING IN CBIOS
                ;
3F58 3E01         MVI       A,1       ;ERROR CONDITION
3F5A C9           RET                 ;REPLACED WHEN FILLED-IN
3F5B              END
```

```
                ; COMBINED GETSYS AND PUTSYS PROGRAMS FROM SECTION 4
                ;
                ; START THE PROGRAMS AT THE BASE OF THE TRANSIENT PROGRAM AREA
0100              ORG       100H
0010 =          MSIZE       EQU       16        ;SIZE OF MEMORY IN KILOBYTES
                ; BIAS IS THE AMOUNT TO ADD TO ADDRESSES FOR SYSTEMS LARGER THAN 16K
                ; (REFERRED TO AS 'B' THROUGHOUT THE TEXT)
0000 =          BIAS        EQU       (MSIZE-16)*1024
                ;
                ; GETSYS PROGRAM - READ TRACKS 0 AND 1 TO MEMORY AT 2880H+BIAS
                ; REGISTER            USE
                ;   A                 (SCRATCH REGISTER)
                ;   B                 TRACK COUNT (0...76)
                ;   C                 SECTOR COUNT (1...26)
                ;   D,E               (SCRATCH REGISTER PAIR)
                ;   H,L               LOAD ADDRESS
                ;   SP                SET TO STACK ADDRESS
                ;
                GSTART:                                   ;START OF THE GETSYS PROGRAM
0100 318028       LXI       SP,2880H+BIAS     ;SET STACK POINTER TO SCRATCH AREA
0103 218028       LXI       H,2880H+BIAS      ;SET BASE LOAD ADDRESS
0106 0600         MVI       B,0               ;START WITH TRACK 00
                RDTRK:                                    ;READ FIRST (NEXT) TRACK
0108 0E01         MVI       C,1               ;READ STARTING WITH SECTOR 1
                RDSEC:
010A CD0003       CALL      READSEC           ;READ NEXT SECTOR
010D 118000       LXI       D,128             ;CHANGE LOAD ADDRESS TO NEXT 1/2 PAGE
0110 19           DAD       D                 ;HL=HL+128 TO NEXT ADDRESS
0111 0C           INR       C                 ;SECTOR=SECTOR+1
0112 79           MOV       A,C               ;CHECK FOR END OF TRACK
0113 FE1B         CPI       27
0115 DA0A01       JC        RDSEC             ;CARRY GENERATED IF C<27
                ;
                ; ARRIVE HERE AT END OF TRACK, MOVE TO NEXT TRACK
0118 04           INR       B                 ;TRACK=TRACK+1
0119 78           MOV       A,B               ;CHECK FOR LAST TRACK
011A FE02         CPI       2                 ;TRACK=2?
011C DA0801       JC        RDTRK             ;CARRY GENERATED IF TRACK < 2
                ;
                ; ARRIVE HERE AT END OF LOAD, HALT FOR NOW
011F FB           EI
0120 76           HLT
                ;
                ; PUTSYS PROGRAM - PLACE MEMORY STARTING AT 2880H+BIAS BACK TO TRACKS
                ; 0 AND 1.  START THIS PROGRAM ON THE NEXT PAGE
0200              ORG       (S+100H) AND 0FF00H
                ; REGISTER                 USE
                ;   A                     (SCRATCH REGISTER)
```

```
                ;   B                           TRACK COUNT (0,1)
                ;   C                           SECTOR COUNT (1...26)
                ;   D,E                         (SCRATCH REGISTER PAIR)
                ;   H,L                         DUMP ADDRESS
                ;   SP                          SET TO STACK ADDRESS
                ;
                PSTART:             ;START OF THE PUTSYS PROGRAM
0200 318028       LXI     SP,2880H+BIAS     ;SET STACK POINTER TO SCRATCH AREA
0203 218028       LXI     H,2880H+BIAS      ;SET BASE DUMP ADDRESS
0206 0600         MVI     B,0               ;START WITH TRACK 0
                WRTRK:                                ;WRITE FIRST (NEXT) TRACK
0208 0E01         MVI     C,1               ;START WRITING AT SECTOR 1
                WRSEC:                                ;WRITE FIRST (NEXT) SECTOR
020A CD8003       CALL    WRITESEC  ;PERFORM THE WRITE
020D 118000       LXI     D,128             ;MOVE DUMP ADDRESS TO NEXT 1/2 PAGE
0210 19           DAD     D                 ;HL=HL+128
0211 0C           INR     C                 ;SECTOR=SECTOR+1
0212 79           MOV     A,C               ;CHECK FOR END OF TRACK
0213 FE1B         CPI     27                ;SECTOR=27?
0215 DA0A02       JC      WRSEC             ;CARRY GENERATED IF SECTOR < 27
                ;
                ; ARRIVE HERE AT END OF TRACK, MOVE TO NEXT TRACK
0218 04           INR     B                 ;TRACK=TRACK+1
0219 78           MOV     A,B               ;TEST FOR LAST TRACK
021A FE02         CPI     2                 ;TRACK=2?
021C DA0802       JC      WRTRK             ;CARRY GENERATED IF TRACK < 2
                ;
                ; ARRIVE HERE AT END OF DUMP, HALT FOR NOW
021F FB           EI
0220 76           HLT
                ;
                ;
                ; USER-SUPPLIED SUBROUTINES FOR SECTOR READ AND SECTOR WRITE
                ;
                ; MOVE TO NEXT PAGE FOR READSEC AND WRITESEC
0300              ORG     ($+100H) AND 0FF00H
                ;
                READSEC:            ;READ THE NEXT SECTOR
                ; TRACK TO READ IS IN REGISTER B
                ; SECTOR TO READ IS IN REGISTER C
                ; BRANCH TO LABEL GSTART IF ERROR OCCURS
                ; READ 128 BYTES OF DATA TO ADDRESS GIVEN BY H,L
0300 C5           PUSH    B
0301 E5           PUSH    H
                ; ** PLACE READ OPERATION HERE **
0302 E1           POP     H
0303 C1           POP     B
0304 C9           RET
                ;
                ; MOVE TO NEXT 1/2 PAGE FOR WRITESEC SUBROUTINE
```

```
0380                ORG     ($ AND 0FF00H) + 80H
                    WRITESEC:           ;WRITE THE NEXT SECTOR
                    ; TRACK TO WRITE IS IN REGISTER B
                    ; SECTOR TO WRITE IS IN REGISTER C
                    ; BRANCH TO LABEL PSTART IF ERROR OCCURS
                    ; WRITE 128 BYTES OF DATA FROM ADDRESS GIVEN BY H,L
0380 C5             PUSH    B
0381 E5             PUSH    H
                    ; ** PLACE WRITE OPERATION HERE **
0382 E1             POP     H
0383 C1             POP     B
0384 C9             RET
                    ;
                    ; END OF GETSYS/PUTSYS PROGRAM
0385                END
```

```
                ; THIS IS A SAMPLE COLD START LOADER WHICH, WHEN MODIFIED, RESIDES
                ; ON TRACK 00, SECTOR 01 (THE FIRST SECTOR ON THE DISKETTE).  WE
                ; ASSUME THAT THE CONTROLLER HAS LOADED THIS SECTOR INTO MEMORY
                ; UPON SYSTEM STARTUP (THIS PROGRAM CAN BE KEYED-IN, OR EXIST IN
                ; A PAGE OF READ-ONLY MEMORY BEYOND THE ADDRESS SPACE OF THE CP/M
                ; VERSION YOU ARE RUNNING).  THE COLD START LOADER BRINGS THE CP/M
                ; SYSTEM INTO MEMORY AT 'LOADP' (NOMINALLY 2900H) + 'BIAS' WHERE
                ; THE BIAS VALUE ACCOUNTS FOR MEMORY SYSTEMS LARGER THAN 16K, AND
                ; CP/M VERSIONS WHICH HANDLE THE LARGER MEMORY SPACE.  IN A 16K
                ; SYSTEM, THE VALUE OF BIAS IS 0000H.  AFTER LOADING THE CP/M SYS-
                ; TEM, THE COLD START LOADER BRANCHES TO THE 'BOOT' ENTRY POINT OF
                ; THE BIOS, WHICH BEGINS AT 'BIOS' + 'BIAS'.  THE COLD START LOADER
                ; IS NOT USED AGAIN UNTIL THE SYSTEM IS POWERED UP AGAIN, AS LONG
                ; AS THE BIOS IS NOT OVERWRITTEN.
                ;
                ; THE ORGIN IS 0, ASSUMING THE CONTROLLER LOADS THE COLD START
                ; PROGRAM AT THE BASE OF MEMORY.  THIS ORIGIN MUST BE IN HIGH
                ; MEMORY (BEYOND THE END OF THE BIOS) IF THE COLD START LOADER
                ; IS IMPLEMENTED IN READ-ONLY-MEMORY.
0000              ORG     0000H       ;BASE OF MEMORY
0010 =          MSIZE     EQU     16          ;MEMORY SIZE IN KILOBYTES
0000 =          BIAS      EQU     (MSIZE-16)*1024     ;BIAS TO ADD TO LOAD ADDRESSES
2900 =          LOADP     EQU     2900H       ;LOAD POINT FOR CP/M SYSTEM
3E00 =          BIOS      EQU     3E00H       ;BASIC I/O SYSTEM (2 PAGES = 512 BYTES)
3E00 =          BOOT      EQU     BIOS        ;COLD START ENTRY POINT IN BIOS
1700 =          SIZE      EQU     BIOS+512-LOADP      ;SIZE OF THE CP/M SYSTEM TO LOAD
002E =          SECTS     EQU     SIZE/128    ;NUMBER OF SECTORS TO LOAD
                ;
                ; BEGIN THE LOAD OPERATION
0000 010200     COLD:     LXI     B,2                 ;CLEAR B TO 0, SET C TO SECTOR 2
0003 162E         MVI     D,SECTS     ;NUMBER OF SECTORS TO LOAD IS IN D
0005 210029       LXI     H,LOADP+BIAS                ;LOAD POINT IN H,L
                ;
                LSECT:      ;LOAD NEXT SECTOR
                ; INSERT INLINE CODE AT THIS POINT TO READ ONE 128 BYTE SECTOR
                ; FROM TRACK GIVEN BY REGISTER B,
                ;     SECTOR GIVEN BY REGISTER C,
                ; INTO ADDRESS GIVEN BY REGISTER PAIR H,L
                ; BRANCH TO LOCATION 'COLD' IF A READ ERROR OCCURS
                ;
                ; ************************************************************
                ; USER SUPPLIED READ OPERATION GOES HERE
                ; ************************************************************
                ; (SPACE IS RESERVED FOR YOUR PATCH)
0008 C36B00       JMP     PASTPATCH ;REMOVE THIS JUMP WHEN PATCHED
000B              DS      60H
                ;
                PASTPATCH:
```

```
                ; GO TO NEXT SECTOR IF LOAD IS INCOMPLETE
006B 15           DCR     D                   ;SECTS=SECTS-1
006C CA003E       JZ      BOOT+BIAS           ;GO TO BOOT LOADER AT 3E00H+BIAS
                ;
                ; MORE SECTORS TO LOAD
                ; USE SP FOR SCRATCH REGISTER TO HOLD LOAD ADDRESS INCREMENT
006F 318000       LXI     SP,128
0072 39           DAD     SP                  ;HL=HL+128 TO NEXT LOAD ADDRESS
                ;
0073 0C           INR     C                   ;SECTOR=SECTOR+1
0074 79           MOV     A,C                 ;MOVE SECTOR COUNT TO A FOR COMPARE
0075 FE1B         CPI     27                  ;END OF CURRENT TRACK?
0077 DA0800       JC      LSECT               ;CARRY GENERATED IF SECTOR < 27
                ;
                ; END OF TRACK, MOVE TO NEXT TRACK
007A 0E01         MVI     C,1                 ;SECTOR=1
007C 04           INR     B                   ;TRACK=TRACK+1
007D C30800       JMP     LSECT               ;FOR ANOTHER SECTOR
                ;
0080              END
```