IMSAI notes on CP/M System Alteration Guide

PREFACE

The Digital Research CP/M System Alteration Guide is written for those who must convert CP/M to support their peripherals before using it on their systems. Since IMSAI CP/M is supplied ready-to-run with standard IMSAI peripherals, most users will not need to read the Alteration Guide. It will, however, be of interest to those who wish to alter or add I/O drivers, and those who wish to increase their understanding of the workings of CP/M.

The following section contains notes about the differences in the IMSAI CP/M; its sections are intended to be read concurrently with the same numbered sections of the Digital Research CP/M System Alteration Guide.
IMSAI Notes on CP/M System Alteration Guide

1. Introduction

IMSAI CP/M has already been modified to work with the standard IMSAI peripherals. Further alteration will be required only if different or additional devices are to be supported.

The next two paragraphs describe the basic differences in memory and diskette organization in the IMSAI system, as these each relate to several sections of the Alteration Guide.

1.1 Memory Organization

In a 16k IMSAI CP/M system of version 1.31 or newer, the BIOS, BDOS, and CCP start at addresses 100H lower than as stated by Digital Research. The whole system is 100H bytes larger, with the added space being in the BIOS. For systems created with the CPM command for larger memories, the addresses increase by 400H for each additional K.

1.2 Diskette Organization

IMSAI CP/M diskettes have a two-sector bootstrap and initialization routine written on sectors 1 and 2 of track 0. The system itself begins at sector 3 of track 1 and is two sectors longer, extending through sector 24 of track 1.

3. Second Level System Generation

IMSAI SYSGEN version 1.31 puts the image of tracks 0 and 1 into the TPA starting at 700H (BOOT routine) with the CCP starting at 800H, the BDOS, at 1100H, and the BIOS, at 1D00H. These addresses are an even 2000H less than those at which a 16K system runs, simplifying a lot of the arithmetic described in the System Alteration Guide.

Source code for two sections of the system, BOOT and BIOS, is supplied on the distribution diskette. If you wish to modify either of these, edit and reassemble them as you would any program. Once this has been done, we suggest the following procedure for creating the modified system:
DDT SYSGEN.COM  Load SYSGEN under DDT

G103  Start SYSGEN -
      note special start address

GET SYSTEM (Y-N) Y
SOURCE ON A, TYPE RETURN
PUT SYSTEM (Y-N) N

System from disk A is now in RAM.

IBIOS.HEX  If it is desired
RE000  to replace the BIOS.

IBOOT.HEX  If it is desired
R700  to replace BOOT.

G103  Start SYSGEN again

GET SYSTEM (Y-N) N
PUT SYSTEM (Y-N) Y
DESTINATION ON B, TYPE RETURN

System from RAM is now on disk in B

The above leaves the old system running and the new
system on the diskette in drive B. To run the new
system, move the diskette to drive A and bootstrap from
it.

The above is for a 16K system. For other memory sizes,
first create a system of the desired size with the CPM
command and SYSGEN it onto a diskette. Then use a
procedure such as the above to incorporate your
modified BOOT and/or BIOS into the relocated system.
The modified BOOT or BIOS must be assembled for the
memory size in which it is to run (assembly parameter
MEMT in the current versions). The load bias remains
700H for boot but increases 400H for each additional K
for BIOS.

If you wish to make small alterations only in BIOS or
BOOT, or are testing or debugging, minor changes can be
made by patching with DDT. The procedure is as shown
above, except the alterations are made with the A
and/or S commands rather than by loading a file. The
same biases are used to translate the addresses shown
in the listings to the addresses to be used with the
DDT commands.
5. Diskette Organization

See section 1.2.

9. Reserved Locations in Page Zero

As described, plus:

0004H  Contains the drive number of the currently logged disk

0038H-003AH  Are defaulted to a jump to IMSAI BIOS' "NXM" routine (see section 11) but may be changed by program. Note, however, that programs using RST 7 will be impossible to debug with DDT.

0040H-004FH  ARE used by IMSAI BIOS and should NOT be changed by program.

10. The IMSAI BOOT

BOOT resides on sectors 1 and 2 of each CP/M system diskette. BOOT's function is to load and initialize the rest of the system. The source code for BOOT is on file BOOT.ASM on the distribution diskette and listing is given in the appendix. The programs MBOOT and LBOOT, described in the System Alteration Guide, are not used in IMSAI CP/M.

At either a cold or a warm start, track 0, sector 1 is read into RAM at location 0 and given control. This sector contains the first half of BOOT, which proceeds to read the rest of BOOT from track 0, sector 2 to location 80H, then read successive sectors to locations 2800H (in a 16K system) and up until the entire CCP, BDOS, and BIOS have been loaded.

If a disk error occurs during this bootstrap operation, BOOT displays the error code returned by the floppy disk interface in the lights, restores the drive, then retries the operation indefinitely.

After the system has been loaded, BOOT performs system initialization. Both channels of a SIO serial interface board are initialized, so that two terminals may be used on the system. The IMSAI line printer interface (LIF) is initialized. A PIC-8 board, if present, is initialized such that only interrupt 7 will be responded to. (Interrupts are not used by CP/M. This initialization was chosen for the convenience of the user who wishes to use interrupt 7 for RAM-4A memory write protect violation.) The various JMP's
required in page 0 are set up. A JMP is also put at 38H to the "NXM" entry point of the IMSAI BIOS. The IObYTE is set from the switches on a cold start or to its previous value on a warm restart. The sign-on message, whose text is in the BIOS, is printed.

After system initialization, BOOT transfers control to BIOS+0 on a cold start, or BIOS+30H on a warm restart.

11. The IMSAI BIOS

The IMSAI Basic Input-Output System's source code is on file BIOS.ASM on the distribution diskette and a listing is given in the appendix.

The IMSAI BIOS is 100H bytes longer than Digital Research's BIOS and contains considerable space for user additions.

The IMSAI BIOS generally performs the functions described in the System Alteration Guide. The entry vector is located at 3D00H rather than 3E00H in a 16k system. There are two additional entry points. The first gets control upon completion of a warm reboot and currently JMP's directly to the CCP. The second is the entry to the "NXM" routine.

The NXM routine receives control if a program JMP's to a non-existent memory address or executes an RST 7 without setting up its own JMP at 38H. When this occurs, the BIOS types

\[ \text{CRASH pppp mm} \]

where pppp is the contents of the top of the stack and mm is the contents of memory location pppp-1. If the fault causing the CRASH typeout was the execution of an RST 7, pppp would be its location, plus 1, and mm would be FF. After the CRASH typeout, the system is rebooted.

The CRASH typeout will also occur if a level 7 interrupt comes from any device, and may be used to indicate a write protect violation on RAM-4A memory boards.

The IObYTE function is implemented as described, except that a line printer driver is included but drivers for fast paper tape reader and punch are not included.
Disk errors are handled as follows:

"Not Ready" Errors:
Are retried indefinitely, so the system waits if no diskette is present in the drive being accessed.

All Other Errors:
The code returned by the floppy disk interface is displayed in the lights (and remains until another error or until another program uses the lights), the drive is restored, then the operation is retried. After 15 failures, an error return is given to the BDOS which types

PERMANENT ERROR DRIVE n

then awaits input. If a control-C is typed, the system is rebooted; any other character causes the error to be ignored.

The warm boot entry to the BIOS recieves control from the JMP at BOOT (0). This entry reads the first sector of the BOOT program to location 0, then JMP's to 3 with the current IOBYTE and logged disk values in registers.

In the IMSAI BIOS the drivers for devices "TTY:" and "CRT:" are identical except for the port accessed and one other difference: The "CRT:" driver translates the ASCII code for underscore to rubout. This is a convenience for users with Lear-Siegler ADM-3 terminals, as it eliminates the need to use the shift key when correcting input. Users with other terminals may want to remove this "feature"; the necessary change to BIOS should be evident from the comments in the listing.

Procedures were given above for incorporating a modified BIOS into the system. If you wish to make additions that require more space than is available, another two sectors (100H bytes) are available on track 1 of the disk. To use an enlarged BIOS, you must have the additional RAM above the top of the system and make the following additional changes: increase the assembly parameter LSTDMA in BOOT to cause additional sectors to be read; increase MEMT in SYSGEN (keeping SYSBOTTOM the same); and increase the number of pages specified in any GET and SAVE commands used with system images.
Table of Contents

Section                                      Page
1. INTRODUCTION                               1
2. FIRST LEVEL SYSTEM REGENERATION           1
3. SECOND LEVEL SYSTEM REGENERATION          5
4. SAMPLE GETSYS AND PUTSYS PROGRAMS         8
5. DISKETTE ORGANIZATION                     9
6. THE BIOS ENTRY POINTS                     11
7. A SAMPLE BIOS                             17
8. A SAMPLE COLD START LOADER                17
9. RESERVED LOCATIONS IN PAGE ZERO           17

Appendix
A. THE MDS LOADER MOVE PROGRAM
B. THE MDS COLD START LOADER
C. THE MDS BASIC I/O SYSTEM (BIOS)
D. A SKELETAL CBIOS
E. A SKELETAL GETSYS/PUTSYS PROGRAM
F. A SKELETAL COLD START LOADER
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 environment. 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 with 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: \[ b = 32K - 16K = 16K = 04000H \]
(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 CB IOS (customized BIOS). Implement only the primitive disk operations on a single drive, and simple console input/output functions in this phase.

(7) Test CB IOS 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 accidentally 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 destruction 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 CB IOS 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 BIOS 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 BIOS 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 BIOS 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 BIOS 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 your 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
*SYSGEN VERSION 1.0
GET SYSTEM (Y/N)?Y
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
PUT SYSTEM (Y/N)?N
```

The system will automatically reboot at this point, with the memory image loaded into memory starting at location 900H and ending at 207FH 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 20H = 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 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

\[ 2900H + n = 980H, \text{ or } n = 980H - 2900H \]

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

\[ (2900H+b) + n = 980H, \text{ or } n = 980H - (2900H + b), \text{ or } n = 0E080H - b. \]

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

<table>
<thead>
<tr>
<th>Memory size</th>
<th>Bias ( b )</th>
<th>Negative offset ( n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>16K</td>
<td>0000H</td>
<td>0E080H - 0000H = 0E080H</td>
</tr>
<tr>
<td>32K</td>
<td>4000H</td>
<td>0E080H - 4000H = 0A080H</td>
</tr>
<tr>
<td>48K</td>
<td>8000H</td>
<td>0E080H - 8000H = 6000H</td>
</tr>
<tr>
<td>62K</td>
<td>0B800H</td>
<td>0E080H - 0B800H = 2880H</td>
</tr>
<tr>
<td>64K</td>
<td>0C000H</td>
<td>0E080H - 0C000H = 2080H</td>
</tr>
</tbody>
</table>

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

\[ \text{H}x,n \]

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

\[ \text{H}2900,0E080 \]
for example, will produce 9800H 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 CB IOS located at (3E00H+b) 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 CB IOS, for example, can be modified using the editor, and assembled using ASM, producing a file called CB IOS.HEX which contains the Intel formatted machine code for CB IOS in "hex" format. In order to integrate your new CB IOS, 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 CB IOS resides. Then type:

ICBIOS.HEX

Ready the "hex" file for loading

Assume that your CB IOS is being integrated into a 16K CP/M system, and is thus "org'ed" at location 3E00H. In order to properly locate the CB IOS 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:

RE0800

Read the file with bias 0E080H

Upon completion of the read, re-examine the area where the CB IOS has been loaded (use a "L1E80" command), to ensure that it 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 SYS GEN 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:

Copyright (c), 1976
Digital Research

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:
MVI C, 1 ;READ NEXT TRACK (INITIALLY 0)
;READ STARTING WITH SECTOR 1
RDSEC:
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

FOP H ;RECOVER HL
FOP 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 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 800H and 0FFH (below the TPA).

After the move, MBX1L 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

<table>
<thead>
<tr>
<th>Track#</th>
<th>Sector#</th>
<th>Page#</th>
<th>Memory Address</th>
<th>CP/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>2900H+b</td>
<td>CCP</td>
</tr>
<tr>
<td>&quot;</td>
<td>03</td>
<td>&quot;</td>
<td>2980H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>04</td>
<td>01</td>
<td>2A00H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>05</td>
<td>&quot;</td>
<td>2A80H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>06</td>
<td>02</td>
<td>2B00H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>07</td>
<td>&quot;</td>
<td>2B80H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>08</td>
<td>03</td>
<td>2C00H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>09</td>
<td>&quot;</td>
<td>2C80H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>10</td>
<td>04</td>
<td>2D00H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>11</td>
<td>&quot;</td>
<td>2D80H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>12</td>
<td>05</td>
<td>2E00H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>13</td>
<td>&quot;</td>
<td>2E80H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>14</td>
<td>06</td>
<td>2F00H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>15</td>
<td>&quot;</td>
<td>2F80H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>16</td>
<td>07</td>
<td>3000H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>17</td>
<td>&quot;</td>
<td>3080H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>18</td>
<td>08</td>
<td>3100H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>00</td>
<td>19</td>
<td>&quot;</td>
<td>3180H+b</td>
<td>CCP</td>
</tr>
<tr>
<td>00</td>
<td>20</td>
<td>09</td>
<td>3200H+b</td>
<td>BDOS</td>
</tr>
<tr>
<td>&quot;</td>
<td>21</td>
<td>&quot;</td>
<td>3280H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>22</td>
<td>10</td>
<td>3300H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>23</td>
<td>&quot;</td>
<td>3380H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>24</td>
<td>11</td>
<td>3400H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>25</td>
<td>&quot;</td>
<td>3480H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>26</td>
<td>12</td>
<td>3500H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>&quot;</td>
<td>3580H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>02</td>
<td>13</td>
<td>3600H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>03</td>
<td>&quot;</td>
<td>3680H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>04</td>
<td>14</td>
<td>3700H+b</td>
<td>&quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>05</td>
<td>&quot;</td>
<td>3780H+b</td>
<td>&quot;</td>
</tr>
</tbody>
</table>
### 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:

<table>
<thead>
<tr>
<th>01</th>
<th>18</th>
<th>21</th>
<th>3E00H+b</th>
<th>BIOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td></td>
<td></td>
<td>3E80H+b</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>22</td>
<td>3F00H+b</td>
<td></td>
</tr>
</tbody>
</table>

| 01 | 21 | 3F80H+b | BIOS |

<table>
<thead>
<tr>
<th>01</th>
<th>22-26</th>
<th>(not currently used)</th>
</tr>
</thead>
<tbody>
<tr>
<td>02-76</td>
<td>01-26</td>
<td>(directory and data)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>3E00H+b</td>
<td>JMP BOOT</td>
<td>;ARRIVE HERE FROM COLD START LOAD</td>
</tr>
<tr>
<td>3E03H+b</td>
<td>JMP WBOOT</td>
<td>;ARRIVE HERE FOR WARM START</td>
</tr>
<tr>
<td>3E06H+b</td>
<td>JMP CONST</td>
<td>;CHECK FOR CONSOLE CHAR READY</td>
</tr>
<tr>
<td>3E09H+b</td>
<td>JMP CONIN</td>
<td>;READ CONSOLE CHARACTER IN</td>
</tr>
<tr>
<td>3E0CH+b</td>
<td>JMP CONOUT</td>
<td>;WRITE CONSOLE CHARACTER OUT</td>
</tr>
<tr>
<td>3E0FH+b</td>
<td>JMP LIST</td>
<td>;WRITE LISTING CHARACTER OUT</td>
</tr>
<tr>
<td>3E12H+b</td>
<td>JMP PUNCH</td>
<td>;WRITE CHARACTER TO PUNCH DEVICE</td>
</tr>
<tr>
<td>3E15H+b</td>
<td>JMP READER</td>
<td>;READ READER DEVICE</td>
</tr>
</tbody>
</table>
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 (lAH). 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:

<table>
<thead>
<tr>
<th>Address</th>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3E18H+b</td>
<td>JMP HOME</td>
<td>MOVE TO TRACK 00 ON SELECTED DISK</td>
</tr>
<tr>
<td>3E1BH+b</td>
<td>JMP SELDSK</td>
<td>SELECT DISK DRIVE</td>
</tr>
<tr>
<td>3E1EH+b</td>
<td>JMP SETTRK</td>
<td>SET TRACK NUMBER</td>
</tr>
<tr>
<td>3E21H+b</td>
<td>JMP SETSEC</td>
<td>SET SECTOR NUMBER</td>
</tr>
<tr>
<td>3E24H+b</td>
<td>JMP SETDMA</td>
<td>SET DMA ADDRESS</td>
</tr>
<tr>
<td>3E27H+b</td>
<td>JMP READ</td>
<td>READ SELECTED SECTOR</td>
</tr>
<tr>
<td>3E2AH+b</td>
<td>JMP WRITE</td>
<td>WRITE SELECTED SECTOR</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>most significant</th>
<th>least significant</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBYTE AT 0003H</td>
<td>LIST</td>
</tr>
<tr>
<td></td>
<td>PUNCH</td>
</tr>
<tr>
<td></td>
<td>READER</td>
</tr>
<tr>
<td></td>
<td>CONSOLE</td>
</tr>
<tr>
<td>bits 6,7</td>
<td>bits 4,5</td>
</tr>
<tr>
<td>bits 2,3</td>
<td>bits 0,1</td>
</tr>
</tbody>
</table>

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 SELDISK 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 responsibilties 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+0 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+0)
- **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+0)

(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 Siegler 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.

**SELDISK**
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.

**SEITRK**

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, the READ subroutine attempts one read based upon these parameters, and returns the following error codes in register A:

<table>
<thead>
<tr>
<th>Code</th>
<th>Error Description</th>
<th>Bit Positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no errors occurred (bit 0 thru 7 = 0)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Hardware malfunction</td>
<td>(bit 0 = 1)</td>
</tr>
<tr>
<td>2</td>
<td>Unit not ready</td>
<td>(bit 1 = 1)</td>
</tr>
<tr>
<td>4</td>
<td>Command sequence error</td>
<td>(bit 2 = 1)</td>
</tr>
<tr>
<td>8</td>
<td>CRC error</td>
<td>(bit 3 = 1)</td>
</tr>
<tr>
<td>16</td>
<td>Seek error</td>
<td>(bit 4 = 1)</td>
</tr>
</tbody>
</table>

Currently, CP/M responds only to a zero or non-zero value as the return code. That is, if the value in register A is 0 then CP/M assumes that the disk operation completed properly. If the return code is non-zero, then CP/M retries the operation to see if the error is recoverable. There is a maximum to 10 retries by CP/M before the "PERM ERR DISK d" message is printed at the console. Future versions of CP/M will, however, perform more sophisticated error recovery and thus
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
...a areas are given below for reference purposes.

<table>
<thead>
<tr>
<th>Locations</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000H - 0002H</td>
<td>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.</td>
</tr>
<tr>
<td>0003H - 0003H</td>
<td>Contains the Intel standard IOBYTE, which is optionally included in the user’s CBIOS, as described in Section 6.</td>
</tr>
<tr>
<td>0004H - 0004H</td>
<td>(not currently used - reserved)</td>
</tr>
<tr>
<td>0005H - 0007H</td>
<td>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 &quot;CP/M Interface Guide,&quot; and LHLH 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.</td>
</tr>
<tr>
<td>0008H - 0027H</td>
<td>(interrupt locations 1 through 5 not used)</td>
</tr>
<tr>
<td>0030H - 0037H</td>
<td>(interrupt location 6, not currently used - reserved)</td>
</tr>
<tr>
<td>0038H - 003AH</td>
<td>Contains a jump instruction into the DDT program when running in debug mode for programmed breakpoints, but is not otherwise used by CP/M.</td>
</tr>
<tr>
<td>003BH - 003FH</td>
<td>(not currently used - reserved)</td>
</tr>
<tr>
<td>0040H - 004FH</td>
<td>16 byte area reserved for scratch by CBIOS, but is not used for any purpose in the distribution version of CP/M</td>
</tr>
<tr>
<td>0050H - 005BH</td>
<td>(not currently used - reserved)</td>
</tr>
<tr>
<td>005CH - 007CH</td>
<td>default file control block produced for a transient program by the Console Command Processor.</td>
</tr>
<tr>
<td>007DH - 007FH</td>
<td>(not currently used - reserved)</td>
</tr>
<tr>
<td>0080H - 00FFH</td>
<td>default 128 byte disk buffer (also filled with the command line when a transient is loaded under the CCP).</td>
</tr>
</tbody>
</table>

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 8000H.

3000 ORG 3000H ; WE ARE LOADED HERE ON COLD START
0080 = BOOTIB EQU 80H ; START OF COLD BOOT PROGRAM
0080 = BOOTIL 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,BOOTIL ; LENGTH OF BOOT
3010 118000 LXI D,BOOTIB ; 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 BOOTIB ; 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

0010 = MSIZE EQU 16 ;MEMORY SIZE IN KILOBYTES
2000 = CBASE EQU (MSIZE-8)*1024 ;CPM BASE ADDRESS BIAS BEYOND 3K
2900 = BDOSB EQU CBASE+900H ;BASE OF DOS LOAD
3206 = BDOS EQU CBASE+1206H ;ENTRY TO DOS FOR CALLS
4000 = BDOS EQU MSIZE*1024 ;END OF DOS LOAD
3E00 = BOOT EQU BDOS-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 BDOS-BDOSB
0002 = NTRKS EQU 2 ;NUMBER OF TRACKS TO READ
002E = BDOSL EQU BDOSL/128 ;NUMBER OF SECTORS IN DOS
0019 = BDOS  EQU 25 ;NUMBER OF BDOS SECTORS ON TRACK 0
0015 = BDOS1 EQU BDOS-BDOS0 ;NUMBER OF SECTORS ON TRACK 1

F800 = MON80 EQU 0F800H ;INTEL MONITOR BASE
F0F = RMON80 EQU 0FF0FH ;RESTART LOCATION FOR MON80
0078 = BASE EQU 078H ;'BASE' USED BY CONTROLLER
0079 = KTYPE 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

0080 310001 LXI SP,STACK;IN CASE OF CALL TO MON80

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
008B D379 OUT LOW
008D 7C MOV A,H
008E D37A OUT HIGH
0090 DB78 WAIT0: IN DSTAT
0092 E604 ANI 4

B-1
`0097 DB79 IN RTYPE
0099 E603 ANI 11B
009B FE02 CPI 2
009D D40FF CQC RMON80 ;GO TO MONITOR IF 11 OR 10
00A0 DB7B IN BYTE ;I/O COMPLETE, CHECK STATUS
; IF NOT READY, THEN GO TO MON80
00A2 17 RAL
00A3 DC0FF CC RMON80 ;NOT READY BIT SET
00A6 1F RAR ;RESTORE
00A7 E61E ANI 11110B ;OVERRUN/ADDR ERR/SEEK/CRC/XXXX
00A9 C40FF CNZ RMON80 ;TRY ALL OVER AGAIN
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 JUMPS
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
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 1.0  SEPT, 1976
;
; COPYRIGHT (C) 1976
; DIGITAL RESEARCH
; BOX 579, PACIFIC GROVE CA.
;
0010 =
00A0 =
3E00 =
3E00 =
2000 =
2900 =
3206 =
1500 =
0020 =
0002 =
0008 =
0000 =

; PERFORM FOLLOWING FUNCTIONS
; BOOT  COLD START
; WBOUT WARM START (SAVE I/O BYTE)
; (BOOT AND WBOUT 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
;

; MSIZE EQU 16  ;MEMORY SIZE IN KILOBYTES
; VERS EQU 10  ;CPM VERSION NUMBER
; PATCH EQU MSIZE*1024-2*256 ;BASE OF THIS MODULE (ABOVE DOS)
;
; ORG PATCH
; CBASE EQU (MSIZE-8)*1024 ;BIAS FOR SYSTEMS LARGER THAN 8K
; CPMB EQU CBASE+900H  ;BASE OF CPM (CONSOLE PROCESSOR ENTRY)
; BDOS EQU CPMB+1206H ;BASIC DOS (RESIDENT PORTION)
; CPML EQU $-CPMB ;LENGTH (IN BYTES) OF CPM SYSTEM
; NSECTS EQU CPML/128 ;NUMBER OF SECTORS TO LOAD
; LBIAS EQU 980H-CPMB ;LOADER BIAS VALUE USED IN SYSGEN
; OFFSET EQU 2  ;NUMBER OF DISK TRACKS USED BY CP/M
; BUFF EQU 80H  ;DEFAULT BUFFER ADDRESS

C-1
UMP VECTOR FOR INDIVIDUAL 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 HWEM
3E1B C3413F JMP SEDSK
3E1E C35A3F JMP SETTRK
3E21 C35F3F JMP SETSEC
3E24 C3643F JMP SETDMA
3E27 C36A3F JMP READ
3E2A C3733F JMP WRITE

; END OF CONTROLLER - INDEPENDENT CODE, THE REMAINING SUBROUTINES
; ARE TAILORED TO THE PARTICULAR OPERATING ENVIRONMENT, AND MUST
; BE ALTERED FOR ANY SYSTEM WHICH DIFFERS FROM THE INTEL MCS.
;
; THE FOLLOWING CODE ASSUMES THE MCS MONITOR EXISTS AT 0F800H
; AND USES THE I/O SUBROUTINES WITHIN THE MONITOR
;
; WE ALSO ASSUME THE MCS SYSTEM HAS TWO DISK DRIVES AVAILABLE

0002 = NDISK$ EQU 2 ;NUMBER OF DRIVES AVAILABLE
00FD = REVERT EQU 0FDH ;INTERRUPT REVERT PORT
00FC = INTC EQU 0FCH ;INTERRUPT MASK PORT
00F3 = ICON EQU 0F3H ;INTERRUPT CONTROL PORT
007E = INTE EQU 78H$111B ;ENABLE RST 0 (WARM BOOT), RST 7 (MONITOR)

; MCS MONITOR EQUATES
F800 = MON90 EQU 0F800H ;MCS MONITOR
FF0F = RMON90 EQU 0FF0FH ;RESTART MON90 (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 CHARM FROM C TO CONSOLE OUT
F80C = PO EQU 0F80CH ;PUNCH CHARM 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)
007A = HIGH EQU BASE+2 ;IOPB HIGH ADDRESS (OUTPUT)
; READF EQU 4H ; READ FUNCTION
; WRITF EQU 6H ; WRITE FUNCTION
; RECAL EQU 3H ; RECALIBRATE DRIVE
; IORDY EQU 4H ; I/O FINISHED MASK
; CR EQU 0DH ; CARRIAGE RETURN
; LF EQU 0AH ; LINE FEED

; SIGNON: ; SIGNON MESSAGE: XXX 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 312E30 DB VERS/10+'0','.',VERS MOD 10+'0'
3E41 0D0A00 DB CR,LF,0

; BOOT: ; PRINT SIGNON MESSAGE AND GO TO DOS
3E44 318000 LXI SP,BUFF
3E47 212D3E LXI H,SIGNON
3E4A CD7C3F CALL PRMSG ; PRINT MESSAGE
3E4D AF XRA A ; CLEAR ACCUMULATOR
3E4E 32ED3F STA DISKT ; SELECT DISK 0 ON ENTRY
3E51 C3A63E JMP GOCM ; 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 PF AVAILABLE FOR STACK
3E57 3AE3CF LDA DISKN ; CURRENTLY LOGGED DISK, RETURN TO DISKN IF NOT 0
3E5A 32ED3F STA DISKT ; STORE INTO DISK TEMP SINCE WE BOOT OFF OF 0

3E5D 0E0A MVI C,10 ; MAX 10 RETRIES
3E5F C5 PUSH B

; WBOOT: ; ENTER HERE ON ERROR RETRIES
3E60 010029 LXI B,CPMB ; SET DMA ADDRESS TO START OF DISK SYSTEM
3E63 CD643F CALL SETDMA
3E66 0E02 MVI C,2 ; START READING SECTOR 2
3E68 CD5F3F CALL SETSEC
3E6B 0E00 MVI C,0 ; START READING TRACK 0
3E6D CD5A3F CALL SETTRK
3E70 0E00 MVI C,0 ; START WITH DISK 0
3E72 CD413F CALL SELDISK ; CHANGES DISKN TO 0
;
; READ SECTORS, COUNT NSECTS TO ZERO
3E75 C1 POP B ; 10-ERROR COUNT
3E76 062A MVI B,NSECTS
3E78 C5 PUSH B ; SAVE SECTOR COUNT
3E79 CD6A3F CALL READ

C-3
E7C  C2E03E  JNZ  BOOTERR ;RETRY IF ERRORS OCCUR
3E7F  2AF33F  LHLD  IOD ;INCREMENT DMA ADDRESS
3EB2  118000  LXI  D,128 ;SECTOR SIZE
3EB5  19  DAD  D ;INCREMENTED DMA ADDRESS IN HL
3EB6  44  MOV  B,H
3EB7  4D  MOV  C,L ;READY FOR CALL TO SET DMA
3EB8  CD643F  CALL  SETDMA
3EBB  3AF23F  LDA  IGS ;SECTOR NUMBER JUST READ
3EBE  FE1A  CPI  26 ;READ LAST SECTOR?
3EB9  DA9C3E  JC  RD1
3EBB  3AF13F  LDA  IOT ;GET TRACK TO REGISTER A
3EB9  3C  INR  A
3EB7  4F  MOV  C,A ;READY FOR CALL
3EB8  CD5A3F  CALL  SETTRK
3EB9  AF  XRA  A ;CLEAR SECTOR NUMBER
3EBC  3C  RDI:  INR  A ;TO NEXT SECTOR
3EBF  4F  MOV  C,A ;READY FOR CALL
3EBF  CD5F3F  CALL  SETSEC
3ECB  32  POP  B ;RECALL SECTOR COUNT
3EC2  05  DCR  B ;DONE?
3EC3  C2783E  JNZ  RDSEC
3EC6  F3  DI
3EC7  3E12  MVI  A,12H ;INITIALIZE COMMAND
3EC9  D3FD  OUT  REVT
3ECB  AF  XRA  A
3ECC  D3FC  OUT  INTC ;CLEARED
3ED0  3E7E  MVI  A,INTE ;RST0 AND RST7 BITS ON
3ED3  D3FC  OUT  INTC
3ED5  AF  XRA  A
3ED7  D3F3  OUT  ICON ;INTERRUPT CONTROL
3ED9  018000  LXI  B,BUFF
3EDC  CD643F  CALL  SETDMA
3EE2  3EC3  MVI  A,JMP
3EE5  320000  STA  0
3EE8  21033E  LXI  H,WBOOTE
3EEB  220100  SHLD  1 ;JMP WBOOT AT LOCATION 00
3EEC  320500  STA  5
3ED0  210632  LXI  H,BDOS
3ED3  220600  SHLD  6 ;JMP BDOS AT LOCATION 5
3ED7  323800  STA  7*8 ;JMP TO MON80 (MAY HAVE BEEN CHANGED BY DDT)
3EDD  2100F8  LXI  H,MON80

; MUST BE SECTOR 26, ZERO AND GO TO NEXT TRACK
; DONE WITH THE LOAD, RESET DEFAULT BUFFER ADDRESS
GOCME: ;(ENTER HERE FROM COLD START BOOT)
; ENABLE RST0 AND RST7
; SET DEFAULT BUFFER ADDRESS TO 80H
; RESET MONITOR ENTRY POINTS
3ED5 223900  SHLD 7*8+1
; LEAVE IOBYTE SET
; PREVIOUSLY SELECTED DISK WAS B, SEND PARAMETER TO CPM
3ED8 3AED3F  LDA  DISKT
3ED8 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

BOOTTERR:
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 CD893F CALL  ERROR
3EEF C3543E JMP  WBOOT  ;FOR ANOTHER TRY
;
BOOTMSG:
3EF2 2A4314E4E 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 MILLISECONDS)
3F19 0632 MVI  B,50    ;NUMBER OF MILLISECONDS TO WAIT
3F1B 0EB6 T1:  MVI  C,182    ;COUNTER TO CONTROL 1 MICROSEC LOOP
3F1D 0D T2:  DCR  C    ;1 CYCLE = .5 USEC
JNZ T2 ; 10 CYCLES = 5.5 USEC

= 5.5 USEC PER LOOP * 182 = 1001 USEC

DCR B
JNZ T1 ; FOR ANOTHER LOOP
RET

JMP CO

LIST: ; LIST DEVICE OUT
; (EXACTLY THE SAME AS MDS CALL)
JMP LO

PUNCH: ; PUNCH DEVICE OUT
; (EXACTLY THE SAME AS MDS CALL)
JMP PO

READER: ; READER CHARACTER IN TO REG-A
; (EXACTLY THE SAME AS MDS CALL)
JMP RI

HOME: ; MOVE TO HOME POSITION
; USE RECALIBRATION IN CASE SEEK ERRORS HAVE OCCURRED
MVI C, RECAL ; SET TO RECALIBRATE
CALL SETFUNC ; SET IO FUNCTION
CALL WAITIO ; RECALIBRATE THE CURRENT DRIVE
LXI H, 0 ; SET TRACK TO 00 FOR SUBSEQUENT OPERATIONS
SHLD IOT ; SELECT TRACK 00
RET ; MAY HAVE ERROR SET UPON RETURN

SELDISK: ; 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
MOV A, C
CPI NDISKS ; TOO LARGE?
CNC RMON80 ; GIVES #ADDR MESSAGE AT CONSOLE
STA DISKN ; SELECT DISK N

; RAL
RAL
RAL
RAL

ANI 10000B ; UNIT NUMBER IN POSITION
MOV C, A ; SAVE IT
LXI H, IOF ; IO FUNCTION
MOV A, M
ANI 11001111B ; MASK OUT DISK NUMBER
ORA C ; MASK IN NEW DISK NUMBER
3F58 77 MOV M,A ;SAVE IT IN IOPB
3F59 C9 RET
;
;
SETTRK: ;SET TRACK ADDRESS GIVEN BY C
3F5A 21F13F LXI H,1OT
3F5D 71 MOV M,C
3F5E C9 RET
;
;
SETSEC: ;SET SECTOR NUMBER GIVEN BY C
3F5F 21F23F LXI H,IOS
3F62 71 MOV M,C
3F63 C9 RET
;
;
SETDMA: ;SET DMA ADDRESS GIVEN BY REGS B,C
3F64 69 MOV L,C
3F65 60 MOV H,B
3F66 22F33F SHLD IOD
3F69 C9 RET
;
;
READ: ;READ NEXT DISK RECORD (ASSUMING DISK/TRK/SEC/DMA SET)
3F6A 0E04 MVI C,READF ;SET TO READ FUNCTION
3F6C CD9A3F CALL SETFUNC
3F6F CDA33F CALL WAITIO ;PERFORM READ FUNCTION
3F72 C9 RET ;MAY HAVE ERROR SET IN REG-A
;
;
WRITE: ;DISK WRITE FUNCTION
3F73 0E06 MVI C,WRITF
3F75 CD9A3F CALL SETFUNC ;SET TO WRITE FUNCTION
3F78 CDA33F CALL WAITIO 
3F7B C9 RET ;MAY HAVE ERROR SET
;
;
UTILITY SUBROUTINES
PRMSG: ;PRINT MESSAGE AT H,L TO 0
3F7C 7E MOV A,M
3F7D B7 ORA A ;ZERO?
3F7E C8 RZ
; MORE TO PRINT
3F7F E5 PUSH H
3F80 4F MOV C,A
3F81 CD09F8 CALL CO
3F84 E1 POP H
3F85 23 INX H
3F86 C37C3F JMP PRMSG
;
ERROR: ;ERROR MESSAGE ADDRESSES BY H,L
3F89 CD7C3F CALL PRMSG
; ERROR MESSAGE WRITTEN, WAIT FOR RESPONSE FROM CONSOLE
CALL CONIN ; CARRIAGE RETURN
MVI C,CR
CALL CONOUT
MVI C,LF ; LINE FEED
CALL CONOUT ; MAY BE RETURNING FOR ANOTHER RETRY

SEIFUNC:
; SET FUNCTION FOR NEXT I/O (COMMAND IN REG-C)
LXI H, IOF ; IO FUNCTION ADDRESS
MOV A,M ; GET IT TO ACCUMULATOR FOR MASKING
ANI 1111100B ; REMOVE PREVIOUS COMMAND
ORA C ; SET TO NEW COMMAND
MOV M,A ; REPLACED IN IOPB
RET

WAITIO:
; START THE I/O FUNCTION AND WAIT FOR COMPLETION
IN RTYPE
IN RBYTE ; CLEARS THE CONTROLLER

MVI A, IOPB AND 0FFH ; LOW ADDRESS FOR IOPB
OUT LOW ; TO THE CONTROLLER
MVI A, IOPB SHR 8 ; HIGH ADDRESS FOR IOPB
OUT HIGH ; TO THE CONTROLLER, STARTS OPERATION

WAIT0: IN DSTAT ; WAIT FOR COMPLETION
ANI IORDY ; READY?
JZ WAIT0

; CHECK I/O COMPLETION OK
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)
CPI 10B ; READY STATUS CHANGE?
JZ WREADY

; MUST BE 00 IN THE ACCUMULATOR
ORA A
JNZ WERROR ; SOME OTHER CONDITION, RETRY

; CHECK I/O ERROR BITS
IN RBYTE
RAL
JC WREADY ; UNIT NOT READY
RAR
ANI 11111110B ; ANY OTHER ERRORS? (DELETED DATA OK)
JNZ WERROR

; READ OR WRITE IS OK, RETURN ZERO FLAG
3FD2 F602 ORI 10B
3FD4 C9 RET

; WERROR: ;RETURN HARDWARE MALFUNCTION (CRC, TRACK, SEEK, ETC.)
; CP/M EXPECTS THE FOLLOWING RETURN CODES
; 0 - HARDWARE MALFUNCTION
; 1 - UNIT NOT READY (RETURNED ABOVE)
; 2 - COMMAND SEQUENCE ERROR
; 3 - CRC ERROR
; 4 - SEEK ERROR
; 5 - ADDRESS ERROR (HARDWARE MALFUNCTION)
; 6 - DATA OVER/UNDER FLOW (HARDWARE MALFUNCTION)
; 7 - WRITE ERROR (HARDWARE MALFUNCTION)
; 8 - NOT READY

3FD5 47 MOV B,A ;SAVE CONDITION
3FD6 E620 ANI 00100000B ;WRITE PROTECTED?
3FD8 C2D13F JNZ WREAD0 ;TREATED AS NOT READY STATUS
3FDB 78 MOV A,B
3FDC E606 ANI 00000110B ;CRC OR SEEK?
3FDE C2E63F JNZ WERR0
; NO, RETURN HARDWARE MALFUNCTION
3FE1 3E01 MVI A,l
3FE3 C3E3F JMP WERR1
; CRC OR SEEK ERROR
3FE6 17 WERR0: RAL
3FE7 17 RAL
3FE8 E618 ANI 11000B ;INTO PROPER POSITION
3FEA B7 WERR1: ORA A ;SET FLAGS
3FEB C9 RET
; DATA AREAS (MUST BE IN RAM)
3FEC 00 DISKN: DB 0 ;CURRENT DISK
3FED 00  DISKT:  DB  0  ; TEMP FOR CURRENT DISK DURING WARM START
       IOPB:  ; IO PARAMETER BLOCK
3FEF 80  DB  80H  ; NORMAL I/O OPERATION
3EFF 04  IOF:  DB  READF  ; IO FUNCTION, INITIAL READ
3FF0 01  ION:  DB  1  ; NUMBER OF SECTORS TO READ
3FF1 02  IOT:  DB  OFFSET  ; TRACK NUMBER
3FF2 01  IOS:  DB  1  ; SECTOR NUMBER
3FF3 8000  IOD:  DW  BUFF  ; IO ADDRESS

3FF5  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 PART OF THE 10 BYTE SCRATCH AREA AHEAD
; OF THE CBIOs 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)

3DF6 =
SCRAT  EQU PATCH-10  ;START OF 10 BYTE SCRATCH AREA
3DF6 =
TRACK  EQU SCRAT    ;CURRENTLY SELECTED TRACK
3DF7 =
SECTOR EQU SCRAT+1  ;CURRENTLY SELECTED SECTOR
3DF8 =
DMAAD  EQU SCRAT+2  ;CURRENT DMA ADDRESS
3DFC =
DISKNO EQU DMAAD+4  ;CURRENT DISK NUMBER
;
3E00 =
ORG PATCH  ;ORIGIN 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

WBOOT:
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 SELDISK;SELECT DISK
3E1E C3F53E  JMP SETTRK ;SET TRACK NUMBER
3E21 C30A3E  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

D-1
BOOT: ;SIMPLEST CASE IS TO JUST PERFORM PARAMETER INITIALIZATION
JMP GOCPM ;INITIALIZE AND GO TO CP/M
;
WB: ;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 SELDISK
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 F000 CPI 00H ;ANY ERRORS?
3E55 C2303E JNZ WB ;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

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 OFFH IF CHARACTER READY, 00H IF NOT
3E99 DS  16H ;SPACE FOR STATUS SUBROUTINE
3EA9 3E00 MVI  A,00H
3EAB C9 RET

; CONIN: ;CONSOLE CHARACTER INTO REGISTER A
3EAC DS  16H ;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  16H ;SPACE FOR OUTPUT ROUTINE
3ED0 C9 RET

D-3
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 B67F 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
;
SELDISK: ; SELECT DISK GIVEN BY REGISTER C
3EE0 79 MOV A,C
3EE1 32FC3D STA DISKNO
3EE4 DS 10H ; SPACE FOR DISK SELECTION ROUTINE
3EF4 C9 RET
;
SEITRKR: ; SET TRACK GIVEN BY REGISTER C
3EF5 79 MOV A,C
3EF6 32F63D STA TRACK
3EF9 DS 10H ; SPACE FOR TRACK SELECT
3F09 C9 RET
;
SEITSEC: ; SET SECTOR GIVEN BY REGISTER C
3F0A 79 MOV A,C
3F0B 32F73D 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 22F83D Sbild 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  C3F83F  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)
;
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 =

0010 =

0000 =

0000 =

; 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: ;READ NEXT SECTOR

010A CD0003 CALL READSEC ;CHANGE LOAD ADDRESS TO NEXT 1/2 PAGE
010D 118000 LXI D,128
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 FE01 CPI 27
0115 DA8A01 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 DA8001 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 ($+100H) AND 0FF00H
; REGISTER USE
; A (SCRATCH REGISTER)
; B
; C
; D,E
; H,L
; SP
; SET TO STACK ADDRESS
;
PSTART: ;START OF THE PUTSYS PROGRAM
0200 318028 LXI SP,2880H+BIAS ;SET STACK POINTER TO SCRATCH AREA
0203 218028 LXI H,2880H+BIAS ;SET BASE DUMP ADDRESS
0206 0600 MVI B,0 ;START WITH TRACK 0
0208 0E01 WRTRK: MVI C,1 ;WRITE FIRST (NEXT) TRACK
020E 118000 WRSEC: MVI C,1 ;WRITE FIRST (NEXT) SECTOR
0210 19 CALL WRSEC ;PERFORM THE WRITE
0213 79 CALL WRSEC ;WRITE FIRST (NEXT) SECTOR
0215 DA0A02 JC WRTRK ;WRITE FIRST (NEXT) SECTOR
0218 118000 LXI D,128 ;MOVE DUMP ADDRESS TO NEXT 1/2 PAGE
021B 19 DAD D ;HL=HL+128
021E 0C INR C ;SECTOR=SECTOR+1
0221 79 MOV A,C ;CHECK FOR END OF TRACK
0224 FE02 CPI 27 ;SECTOR=27?
0227 DA0A02 JC WRTRK ;CARRY GENERATED IF SECTOR < 27
;
;ARRIVE HERE AT END OF TRACK, MOVE TO NEXT TRACK
022A 04 INR B ;TRACK=TRACK+1
022D 79 MOV A,B ;TEST FOR LAST TRACK
0230 FE02 CPI 2 ;TRACK=2?
0233 DA0A02 JC WRTRK ;CARRY GENERATED IF TRACK < 2
;
;ARRIVE HERE AT END OF DUMP, HALT FOR NOW
0236 FB EI
0239 76 HLT
;
;USER-SUPPLIED SUBROUTINES FOR SECTOR READ AND SECTOR WRITE
;
;MOVE TO NEXT PAGE FOR READSEC AND WRITSEC
0300 ORG ($+100H) AND $FF00H
;
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 WRITSEC SUBROUTINE
ORG $(S \text{ 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 ORIGIN 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
0110 = 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

; SECT: ;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:

F-1
; 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 F18B CPI  27 ;END OF CURRENT TRACK?
0077 DA0000 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
APPENDIX 1

IMSAI BOOT PROGRAM LISTING
; BOO'T.ASM VER 1.31 REV 0  JRB 12/13/76
;
; IMSAI CP/M BOOTSTRAP ROUTINE
;
; THIS PROGRAM RESIDES ON TRACK 0, SECTOR 1 AND 2 OF
; ALL CP/M SYSTEM DISKS AND IS READ INTO RAM AT
; LOCATION 0 AND EXECUTED FOR BOTH COLD AND WARM
; START BOOTSTRAPS.

; COPYRIGHT (C) 1976,
; IMSAI MANUFACTURING CORP.
; 14860 WICKS BLVD, SAN LEANDRO, CA  94577, USA

; SYSTEM RAM EQUATES

0000 = BBASE  EQU 0 ;WHERE THIS PROGRAM RUNS
0080 = BBASE2 EQU 80H ;WHERE SECTOR 2 OF THIS PROGRAM RUNS
4000 = MEMT  EQU 4000H ;TOP OF MEMORY, BEFORE RELOCATION BY CPM PROGRAM
2800 = SYSBOTTOM EQU MEMT-1800H ;FIRST LOCATION OCCUPIED BY SYSTEM
3106 = ENTRYPOINT EQU MEMT-4*1024+256+6 ;WHERE SYSTEM CALLS ENTER SYSTEM
3D00 = BIOS  EQU MEMT-300H ;WHERE BASIC I/O SYSTEM ENTRIES ARE
3D00 = BOOTR EQU BIOS ;WHERE COLD BOOT EXITS TO
3D03 = WBOOT EQU BIOS+3 ;WHERE TO GO TO INITIATE WARM BOOT
3D0C = CONOUT EQU BIOS+0CH ;(IE TO GET THIS PROGRAM READ IN AND ENTERED)
3D2D = NXM   EQU BIOS+02DH ;ROUTINE TO OUTPUT CHARACTER TO CONSOLE
3D30 = WBOOTR EQU BIOS+030H ;WHERE RESTART 7 SHOULD GO
3D33 = FIFSTRING EQU 40H ;WHERE WARM BOOT RETURNS TO
0040 = MESSAGE EQU BIOS+33H ;WHERE TO PRESET FIF STRING PTR 0 TO FOR BIOS
3D33 = MESSAGE EQU BIOS+33H ;LOCATION OF SIGN-ON MSG TEXT IN BIOS
;N. B. ABOVE MUST MATCH VALUES USED IN BIOS !
0005 = ENTRYJMP EQU 5 ;WHERE TO PUT JMP ENTRYPOINT
0000 = WBOOTJMP EQU 0 ;WHERE TO PUT JMP REBOOT
0003 = IOBYTE EQU 3 ;LOCATION OF IO STATUS BYTE
; ; SYSTEM DISK LAYOUT EQUATES
;
0000 = FTRK EQU 0 ; FIRST TRACK TO READ
0001 = LTRK EQU 1 ; LAST TRACK TO READ
0001 = NSECTB4SYSTEM EQU 1 ; NUMBER OF SECTORS TO SKIP BEFORE READING SYSTEM:
; THE ONE SECTOR CONTAINING THIS PROGRAM
; SECOND SECTOR OF THIS PROGRAM IS TREATED
; AS PART OF SYSTEM EXCEPT IT IS REDIRECTED TO 80H
; AFTER SECTOR 2 SECTORS ON DISK ARE READ INTO MEMORY IN ORDER,
; STOPPING AT MEMT.
;
; ; I/O DEVICE CONFIGURATION EQUATES
;
00FD = DISK EQU OFDH ; FLOPPY DISK PORT
00F7 = PIC8 EQU 0F7H ; PRI INT CNTRL BOARD. INITIALIZED, NOT USED.
0000 = SIOBD EQU 0 ; BASE PORT # OF SIO BOARD
0003 = SIOS1 EQU SIOBD+3 ; STATUS PORT OF IMSAI SERIAL I/O BOARD TO INITIALIZE
0005 = SIOS2 EQU SIOBD+5 ; STATUS PORT OF TOOTHER...
0008 = SIOC EQU SIOBD+8 ; CONTROL PORT FOR SIO BOARD (BOTH CHANNELS)
;
00F6 = PRINTER EQU 0F6H ; IMSAI PTR-300@LINE@PRINTER PORT
0080 = PINIT EQU 80H ; COMMAND TO INIT LINE PRINTER
0082 = POFF EQU 82H ; COMMAND TO TURN MOTOR OFF AND FORM FEED
;
; ; MISCELLANEOUS
;
0000 = SPTR EQU 0 ; FIF STRING POINTER USED BY BIOS AND BOOT
000C = FF EQU 0CH ; ASCII FORM FEED CHARACTER
THE BOOTSTRAP PORTION OF THIS PROGRAM READS
EVERY NTH SECTOR ON ONE DISK REVOLUTION FOR SPEED,
INTO APPROPRIATE MEMORY LOCATIONS. DISC CONTAINS
DIRECT MEMORY IMAGE.
TO MAKE ITS ADDRESS INCREMENTING SCHEME WORK,
THE PROGRAM MUST LOOP OVER AN INTEGRAL NUMBER OF TRACKS.
HOWEVER, THE PROGRAM DOES NO IO FOR SECTORS
AT BEGINNING OF FIRST TRACK OR END OF LAST TRACK WHICH
DO NOT CONTAIN INFORMATION TO BE READ.
FLASH! ANOTHER KLUDGE 1; THE SYSTEM INITIALIZER TURNS OUT TO BE
TOO BIG FOR ONE SECTOR AND THE DISK READER IS A VERY TIGHT FIT.
THE MINIMUM MEMORY MODIFICATION TO READ TRACK 0, SECTOR 2 INTO
LOCATION 80H AS THE REST OF THE INITIALIZER TURNS OUT TO BE
TO ALTER THE RAM ADDRESS FOR THE LOWEST
SECTOR READ. HENCE THE FOLLOWING 2 EQUATES ARE AS THOUGH
THE SYSTEM STARTS 128 BYTES LOWER IN RAM THAN IT DOES.

2700 =
FSTDMA EQU SYSBOTTOM-NSECTB4SYSTEM*128-128
;WHERE 1ST SECTOR WOULD BE READ
;IF IT WERE TO BE READ
2780 =
FSTRDMA EQU SYSBOTTOM-128;FIRST LOCATION TO REALLY READ INTO
;EXCEPT SEE FLASH NOTE ABOVE
4000 =
LSTDMA EQU MEMT;FIRST LOCATION BEYOND END OF PROGRAM

;PARAMETERS FOR SKewing SCHEME
0005 =
SKED EQU 5;READ EVERY 5FH SECTOR. CHANGE WITH
;CARE! 7 ALSO WORKS.
001B =
ETSCT EQU 27;ENDTEST SECTOR NUMBER. 27 WORKS
;FOR 5 AND 7, BUT OTHER SKEWS MAY
;CHANGE THIS!

001A =
NSCTPT EQU 26;NUMBER OF SECTORS PER TRACK
;N.B. IF THIS CHANGES, REVIEW SKED AND ETSCT!
0001 =
FSCT EQU 1;NUMBER OF FIRST SECTOR
0080 =
SCSIZ EQU 128;SIZE OF SECTORS

0100 =
STACK EQU 100H;WHERE TO PUT STACK
ORG BBASE

; WARNING TO ANYONE ALTERING FOLLOWING CODE:
; * THERE IS SOMETHING ORG'D AT 40H A FEW LINES
; * DOWN. IF YOU INSERT INSTRUCTIONS BEFORE
; * IT, YOU MUST MOVE SOME INSTRUCTIONS
; * FROM ABOVE TO BELOW IT

ENTRY VECTOR

0000 16FF  MVI D,OFFH  ;COLD START, ENTERED BY HARDWARE BOOT
0002 21    DB 21H     ;LXI H, OP CODE: SKIP OVER MVI D,0
0003 ORG BBASE+3
0003 1600  WBOOTE: MVI D,0  ;WARM START. BIOS JMPS HERE AFTER READING SECTOR
                   ;1, TRACK 0. B AND C CONTAIN VALUES TO
                   ;PRESERVE II
                   ;FLAG IN D IS PRESERVED THRU DISK READ:
                   ;NON-O FOR COLD START, O FOR WARM START.

* ROUTINE TO READ SYSTEM OFF DISK

REGISTER USE

A    SCRATCH
BC   WARM START PARAMETERS TO SAVE
D    WARM/COLD FLAG
E    SECTOR NUMBER
HL   RAM ADDRESS/SCRATCH
SP   SCRATCH/RAM ADDRESS

STACK IS NOT AVAILABLE UNTIL READ IS COMPLETE:
A SECTOR IS READ IN OVER IT.
N. B. DO NOT CLOBBER B, C, OR D!
; INITIALIZE TO READ OFF DISK

0005 3E10  MVI A,10H+SPTR ;"SET STRING PTR"
0007 D3FD  OUT DISK
0009 3E40  MVI A,FIFSTRING AND OFFH
000B D3FD  OUT DISK. ;SET LO STRING ADDRESS
000D AF   XRA A ;MVI A,FIFSTRING SHR 8 AND OFFH
000E D3FD  OUT DISK ;SET HI STRING ADDRESS

; INIT RAM ADDRESS IN HL

0010 210027  LXI H,FSTDMA

; TOP OF READ TRACKS LOOP

0013 3EFC  R0: MVI A,(FSCT-SKEW) AND OFFH ;FIRST SECTOR

; TOP OF READ SECTORS LOOP

0015 C605  R1: ADI SKEW ;COMPUTE NEXT SECTOR
0017 5F   MOV E,A ;SAVE SECTOR NUMBER FOR PROGRAM
0018 324400 STA BSECT ;STORE SECTOR FOR FLOPPY INTERFACE
001B 224500 SHLD BBUFAD ;SET MEMORY ADDRESS FOR DISC

; DON'T READ IF OUT OF ADDR RANGE, BUT CONTINUE LOOPING
001E F9   SPHL ;RAM ADDRESS TO SP UNTIL INCREMENTED BELOW
001F 218027 LXI H,FSTRDMA ;THIS MUST BE POSITIVE FOR RELOCATION

; NEGATE HL
0022 AF   XRA A
0023 95   SUB L
0024 6F   MOV L,A
0025 3E00  MVI A,0
0027 9C   SBB H
0028 67   MOV H,A
0029 39   DAD SP
002A D26A00 JNC OK ;ADDR TOO LOW: PROVISION FOR THIS PROGRAM TO BE ON 1ST SCTR

;FLASH KLUDGE1; IF ADDRESS IS EXACTLY LOWEST, READ

; TO 80H INSTEAD. THIS GETS 2ND HALF OF THIS PROGRAM.

002D 7D   MOV A,L
002E B4   ORA H
002F C23800 JNZ R1A
0032 218000 LXI H,BBASE2
0035 1500  SHLD BBUFAD
LXI H,LSTDMA
AGAIN, NEGATE HL
XRA A
SUB L
JMP BOOT1

PRE-INITIALIZED FIF COMMAND STRING
STUCK IN THIS REDICULOUS PLACE CAUSE IT'S WHERE BIOS USES IT, AND IT SAVES THE FEW BYTES OF CODE IT TAKES TO SET STRING POINTER AGAIN IN BOOT.

ORG FISTRING
BCMD: DB 21H ;READ SECTOR, UNIT 0 COMMAND
BSTAT: DB 0 ;STATUS BYTE
BTRK: DB 0,FTRK ;TRACK
BSECT: DB 1 ;SECTOR
BBUFAD: DW FSTDMA ;BUFFER ADDRESS

BOOT1: ;NOW, BACK TO WHAT WE WERE DOING
MOV L,A ;FINISH NEGATING HL
MVI A,0
SBB H
MOV H,A
DAD SP
JC OK

;READ THIS SECTOR
XRA A ;HERE TO RETRY AFTER ERROR
LXI H,BSTAT ;POINT STATUS BYTE
MOV M,A ;ZERO STATUS BYTE
;N. B. A=0 IS ALSO COMMAND FOR DISK
OUT DISK ;DO IT!
ADD M ;TOP OF WAIT LOOP
JZ WAIT ;WAIT FOR FIF TO STORE NON-O STATUS
CPI 1 ;TEST FOR EXACT GOOD RETURN
JZ OK ;GO INCREMENT TO NEXT SECTOR
; DISPLAY ERROR CODE IN LIGHTS
; LIGHTS DISPLAY COMPLEMENT
0060 2F CMA ; LIGHTS DISPLAY COMPLEMENT
0061 D3FF OUT OFFH ; RESTORE THE DRIVE - IT HELPS
0063 3E21 MVI A,21H
0065 D3FD OUT DISK
0067 C35000 OUT

; TRY AGAIN, AND AGAIN

OK:

; INCREMENT ADDRESSES WITH SKEW

; NEXT SECTOR, MOVE UP "SKEW" SECTORS, AND ADJUST
; MEMORY ADDRESS CORRESPONDINGLY
006A 7B MOV A,E ; SECTOR NUMBER TO A
006B 218002 ;("SKEW" IS ADDED TO THIS AT TOP OF LOOP)
006C 218002 LXI H,SKEW*SCTSIZ ; MEM ADDR INCREMENT

; END OF PASS THRU THIS TRACK TEST
006E FE16 CPI NSCTPT+FSCT-SKEW
0070 FA7B00 JM DADSP ; GO UPDATE HL THEN READ. SECTOR IS IN A

; END OF TRACK TEST
; CPI ETSCT-SKEW ; TEST ON MAGIC NUMBER
; N.B. MAGIC NUMBER IS SAME AS NUMBER JUST TESTED ON!

0073 CA7F00 JZ NXTTRK ; IF DONE THIS TRACK, GO INCREMENT TRACK

; RECYCLE THRU THIS TRACK, GETTING A DIFFERENT GROUP
; OF SPACED SECTORS
0076 DE1A SBI NSCTPT ; UNCREMENT SECTOR #
0078 2180F5 LXI H,(SKEW-NSCTPT)*SCTSIZ ; MEM ADDR DECREMENT

; CODE ABOVE HERE MUST FIT BELOW 80H.
; BYTES 80H-EFH ARE READ IT BY THE TIME FIRST
; TRACK READ IS COMPLETE.

; NEXT TRACK
007B 39 DADSP: DAD SP ; COMPUTE NEXT MEM ADDR, LEAVE IN HL
007C C31500 JMP R1 ; NOW GO READ. SECTOR IS IN A

; NEXT TRACK
007E 39 NXTTRK: DAD SP ; INCREMENT MEM ADDR
0080 3A4300 LDA BTRK+1 ; TRACK
0083 3C INR A
0084 324300 STA BTRK+1
0087 FE02 CPI LTRK+1 ; HAVE WE DONE LAST TRACK?
0089 FA1300 JM RO ; NO, GO READ TRACK

; DONE READING FROM DISC!

; AT THIS POINT STACK CAN BE USED
008F 10001 LXI SP, STACK
; INITIALIZE BOTH CHANNELS OF IMSAI SERIAL INTERFACE BOARD

; IF SIO HAS JUST BEEN RESET, IT EXPECTS A "MODE" THEN A "COMMAND".
; BUT IF IT HASN'T BEEN RESET (WARM START), IT IS NOT EXPECTING A "MODE".
; SO WE SEND IT A DUMMY THAT LEAVES IT EXPECTING A COMMAND REGARDLESS,
; THEN A RESET COMMAND (40H), THEN DESIRED MODE AND COMMAND.

008F 21E800  LXI H,SIOSTRING
0092 7E      MOV A,M
0093 D303    SIOLUP: OUT SIOS1
0095 D305    OUT SIOS2
0097 23      INX H
0098 7E      MOV A,M
0099 B7      ORA A
009A C29300  JNZ SIOLUP ;TURN OFF INTERRUPTS AND
009D AF      XRA A ;...CARRIER DETECT, BOTH CHANNELS
009E D308    OUT SIOC

; INITIALIZE LINE PRINTER

00A0 3E80    MVI A,PINIT ;PRINTER INITIALIZE COMMAND
00A2 D3F6    OUT PRINTER

; PUT THE VARIOUS JUMPS IN LOWER RAM

00A4 3EC3    MVI A,OC3H ;"JMP" OP CODE
;"JMP REBOOT" AT 0
00A6 21033D  LXI H,WBOOT
00A9 320000  STA WBOOTJMP
00AC 220100  SHLD WBOOTJMP+1
;"JMP ENTRYPOINT" AT 5 FOR SYSTEM CALLS
00AF 210631  LXI H,ENTRYPOINT
00B2 320500  STA ENTRYJMP
00B5 220600  SHLD ENTRYJMP+1
;"JMP NXM" FOR RESTART 7 (DDT WILL CHANGE IF USED). THIS IS ACCESSED AFTER
; A JMP INTO NON-EXISTENT MEMORY, ALSO BY WRITE PROTECT VIOLATION ON RAM-4A
; IF WIRED DIRECT TO "INT" LINE BY USER. AN IMSAI EXTENSION OF BIOS FUNCTIONS
00B8 212D3D  LXI H,NXM
00BB 323800  STA 038H
00BE 223900  SHLD 038H+1

; INITIALIZE IOBYTE FROM SWITCHES

00C1 DBFF    IN OFFH
00C3 320300  STA IOBYTE
COLD START ONLY:

; SIGN-ON MESSAGE
00CB 21333D LXI H,MESSAGE ;MESSAGE TEXT IS IN BIOS
00CE 4E MSLOOP: MOV C,M
00CF CD0C3D CALL CONOUT ;USE CHAR TYPING ROUTINE IN BIOS
00D2 23 INX H
00D3 7E MOV A,M
00D4 B7 ORA A
00D5 C2CE00 JNZ MSLOOP

; INITIALIZE IMSAI PRIORITY INTERRUPT CONTROL BOARD
; THIS CODE SETS IT AS THO A LEVEL 6 INT IS RUNNING,
; TO DISABLE CHANNELS AND MINIMIZE CHANCE OF PROBLEM FROM
; SPURIOUS INTERRUPT.
; BUT CHANNEL 7 IS LEFT ACTIVE FOR USER TO USE FOR MEMORY
; PROTECT VIOLATION, SINCE IMSAI CP/M DOES INTERCEPT RST-7'S
; USERS USING OTHER INTERRUPTS MAY CHANGE THIS.
00D8 3E09 MVI A,00001001B
00DA D3F7 OUT PIC8

; EXIT TO BIOS WITH DISK TO SELECT IN C
00DC 0000 MVI C,0 ;SAY SELECT DISK A
00DE C3003D JMP BOOTR ;EXIT TO BIOS

; WARM RESTART ONLY:
; RESTORE I/O BYTE SAVED IN B REGISTER
; SAVED LOGGED DISK NUMBER IN C IS USED BY CCP
00E1 78 WBOOT9: MOV A,B
00E2 320300 STA IOBYTE
00E5 C3303D JMP WBOOTR ;GO TO BIOS WITH DISK # IN C

;BYTES TO SEND TO SIO STATUS PORTS
00E8 AE40AE3700 SIOSTRING: DB 0AEH,40H,0AEH,37H,0
00ED 2843292031 DB '(C) 1976',

;NOTE THAT STACK WRITES OVER EN D OF THIS SECTOR
00F5 END
APPENDIX 2

IMSAI BIOS PROGRAM LISTING
; BIOS.ASM VERSION 1.31 REV 0 12/13/76
; BASIC I/O SYSTEM FOR IMSAI CP/M
; VERSION 0.0 GE 8/76
; LATER VERSIONS BY JRB
;
; COPYRIGHT (C) 1976,
; IMSAI MANUFACTURING CORP, 14860 WICKS BLVD, SAN LEANDO, CA 94577 USA

;NUMBER OF DISK DRIVES IN SYSTEM
0002 =
NDISKS EQU 2

;MEMORY SIZE
4000 =
MEMT EQU 4000H ;BEFORE RELOCATION WITH CPM PROGRAM

;WHERE TO ENTER CP/M AFTER WARM OR COLD BOOT:
2800 =
CPMB EQU MEMT-1800H

3D00 =
BIOS EQU MEMT-3*256 ;LOCATION OF BASIC I/O SYSTEM
3D33 =
MESSAGE EQU BIOS+33H ;LOCATION OF SIGN-ON MSG TEXT

0004 =
LOGDISK EQU 4 ;WHERE CCP PUTS LOGGED DISK #

0040 =
BIOSTOR EQU 40H ;WHERE BIOS STORAGE IS IN PAGE 0 RAM
0040 =
FIFSTRING EQU BISTOR ;FLOPPY INTERFACE COMMAND STRING

; WARM BOOT ENTRY INTO BOOTSTRAP ROUTINE ON TRACK 0, SECTOR 1
0003 =
WBOOTE: EQU 3
; I/O ASSIGNMENT BYTE IN LOWER RAM
0003 =
    IOBYT EQU 3
;
; I/O PORT EQUATES
;
00FD =
    DISK EQU 0FDH
0000 =
    SPTR EQU 0 ;FIF STRING POINTER USED BY BIOS
0002 =
    TTY  EQU 02H
0003 =
    TTYS EQU 03H
0004 =
    CRT  EQU 04H
0005 =
    CRTS EQU 05H
;
; LINE PRINTER
00F6 =
    PRINTER EQU 0F6H ;OUTPUT PORT
0080 =
    PINIT EQU 80H ;INITIALIZE COMMAND
0082 =
    POFF  EQU 82H ;TURN OFF MOTOR, FORM FEED
00F6 =
    PRINTERS EQU PRINTER ;STATUS PORT IS SAME AS OUTPUT PORT
    PREADY EQU XXX ;MASK FOR PRINTER READY STATUS
    ;NO STATUS CHECK NEEDED AFTER SENDING COMMANDS OR SINGLE CHAR
    ;ANYTHING SENT TO PRINTER WITH B7=0 IS TAKEN AS ASCII CHAR TO PRINT
;
; EQUATES FOR ASCII CHARACTERS
;
0003 =
    CTRLz EQU 3
0009 =
    TAB  EQU 9
000A =
    LF   EQU 0AH
000C =
    FF   EQU 0CH
000D =
    CR   EQU 0DH
001A =
    CTRLZ EQU 1AH
005F =
    UNDERLINE EQU 5FH
007F =
    RUBOUT  EQU 7FH
;
ORG BIOS ; ORIGIN EQUATED ABOVE

ENTRY POINT TABLE

ORG BIOS

ENTAB: JMP CPMB ; COLD START. BOOT HAS DONE ALL INIT, GO DIRECT TO CP/M
JMP WBOOT ; COME HERE TO INITIATE REBOOT (VIA LOCATION 0)
JMP CONSTAT
JMP CONIN
JMP CONOUT
JMP LIST
JMP PUNCH
JMP READER
JMP HOME
JMP SELDSK
JMP SETTRK
JMP SETSEC
JMP SETDMA
JMP READ
JMP WRITE
JMP NXM ; FOR RESTART 7: GIVE ERROR MESSAGE
JMP CPMB ; WARM BOOT RETURNS HERE. GO DIRECT TO CP/M.

SIGN-ON MESSAGE, TYPED BY BOOT ROUTINE

MESSAGE: DB CR,LF,'IMSAI 16K CP/M VERS 1.31',0
MESSAGE: DB '(C) 1976',0
; DISK ROUTINES

; READ FROM SELECTED DRIVE/TRACK/SECTOR

3D57 3A4000 READ: LDA CMD
3D5A E60F ANI OFH ; STRIP OLD CMD
3D5C F620 ORI 20H ; CMD=READ
3D5E C3683D JMP W1 ; GO DO IT

; WRITE TO SELECTED DRIVE/TRACK/DISK

3D61 3A4000 WRITE: LDA CMD
3D64 E60F ANI OFH ; STRIP OLD CMD
3D66 F610 ORI 10H ; CMD=WRITE
3D68 324000 W1: STA CMD
EXECUTE COMMAND STRING

3D6B C5  \hspace{1em} PUSH B ; WARM BOOT REQUIRES BC PRESERVED
3D6C 0E0F  \hspace{1em} MVI C,15 ; RETRY COUNT: KEEP AT IT!
3D6E 214100 \hspace{1em} EX0: LXI H,STAT ; POINT AT STATUS
3D71 AF  \hspace{1em} XRA A
3D72 77  \hspace{1em} MOV M,A ; ZERO STAT.BYTE
3D73 D3FD  \hspace{1em} OUT DISK ; EXEC CMD STRING
3D75 86 \hspace{1em} EX1: ADD M ; GET STATUS
3D76 CA753D  \hspace{1em} JZ EX1 ; LOOP UNTIL STAT<>0
3D79 FE01  \hspace{1em} CPI 1 ; TEST FOR EXACT GOOD RETURN
3D7B CA8E3D  \hspace{1em} JZ EX2 ; GO EXIT IF GOOD
3D7E FEAl  \hspace{1em} CPI OA1H ; TEST FOR NOT READY
3D80 CA6E3D  \hspace{1em} JZ EX2 ; WAIT FOREVER FOR DOOR TO BE CLOSED
3D83 2F  \hspace{1em} CMA ; CAUSE LIGHTS DISPLAY COMPLEMENT
3D84 D3FF  \hspace{1em} OUT OFFH ; TO LIGHTS
3D86 3E2F  \hspace{1em} MVI A,2FH ; HOMES ALL DRIVES
3D88 D3FD  \hspace{1em} OUT DISK
3D8A 0D  \hspace{1em} DCR C
3D8B C26E3D  \hspace{1em} JNZ EX0
3D8E E6F0 \hspace{1em} \hspace{1em} EX2:
3D90 1F  \hspace{1em} ANI OF0H ; ISOLATE ERROR CLASS
3D91 1F  \hspace{1em} RAR ; PUT IN LOWER HALF BYTE
3D92 1F  \hspace{1em} RAR
3D93 1F  \hspace{1em} RAR
3D94 C1  \hspace{1em} POP B
3D95 C9  \hspace{1em} RET
<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D96</td>
<td>MOV A, #02FH</td>
<td>Home: set register A to 02FH</td>
</tr>
<tr>
<td>3D98</td>
<td>OUT DISK</td>
<td></td>
</tr>
<tr>
<td>3D9A</td>
<td>MVI C, #0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JMP SETTRK</td>
<td></td>
</tr>
<tr>
<td>3D9C</td>
<td>MOV A, C</td>
<td>Settrack: set register A to C</td>
</tr>
<tr>
<td>3D9D</td>
<td>STA TRK+1</td>
<td></td>
</tr>
<tr>
<td>3DA0</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>3DA1</td>
<td>MOV A, C</td>
<td>Setsector: set register A to C</td>
</tr>
<tr>
<td>3DA2</td>
<td>STA SECT</td>
<td></td>
</tr>
<tr>
<td>3DA5</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>3DA6</td>
<td>MOV A, C</td>
<td>SetDMA: set register A to C</td>
</tr>
<tr>
<td>3DA7</td>
<td>STA BUFADR</td>
<td></td>
</tr>
<tr>
<td>3DAA</td>
<td>MOV A, B</td>
<td></td>
</tr>
<tr>
<td>3DAB</td>
<td>STA BUFADR+1</td>
<td></td>
</tr>
<tr>
<td>3DAE</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>3DAF</td>
<td>MOV A, C</td>
<td>SELECT DISK: set register A to C</td>
</tr>
<tr>
<td>3DB0</td>
<td>CPI NDISKS</td>
<td></td>
</tr>
<tr>
<td>3DB2</td>
<td>MOV A, #80H</td>
<td></td>
</tr>
<tr>
<td>3DB4</td>
<td>MVI #07</td>
<td></td>
</tr>
<tr>
<td>3DB5</td>
<td>DCR C</td>
<td></td>
</tr>
<tr>
<td>3DB6</td>
<td>JP SD1</td>
<td></td>
</tr>
<tr>
<td>3DB9</td>
<td>STA CMD</td>
<td></td>
</tr>
<tr>
<td>3DBC</td>
<td>RET</td>
<td></td>
</tr>
</tbody>
</table>
LOGICAL DEVICE ROUTINES

THESE ROUTINES USE VARIOUS PHYSICAL DEVICES
DEPENDING ON CONTENTS OF IOBYT

CONSOLE STATUS

; GETS STATUS OF SPECIFIC DEVICE
3DBD CDC53D
CONSTAT: CALL CONS
3DC0 B7 ORA A
3DC1 C8 RZ
3DC2 3EFF MVI A,OFFH
3DC4 C9 RET
3DC5 3A0300 CONS: LDA IOBYT
3DC8 CD213E CALL RLCDISPATCH
3DCB 3C3E DW TTYSTAT
3DCD 613E DW CRISTAT
3DCF D33D DW READERSTAT ; 2: BATCH MODE, USE READER DEVICE
3DD1 883E DW NULLSTAT ; 3: UNASSIGNED CHANNEL

; READER STATUS FOR BATCH MODE: NEVER A CHARACTER READY.
; THIS IS CAUSE PRESENCE OF A CHARACTER FREQUENTLY MEANS
; "ABORT WHAT YOU'RE DOING".
3DD3 AF READERSTAT: XRA A
3DD4 C9 RET

; CONSOLE IN

3DD5 3A0300 CONIN: LDA IOBYT
3DD8 CD213E CALL RLCDISPATCH
3DDB 313E DW TTYIN ; 0: TTY
3DDD 663E DW LSCRTIN ; 1: CRT
3DDF 123E DW READER ; 2: BATCH MODE: READER INPUT
3DE1 883E DW NULLI ; 3: UNASSIGNED CHANNEL
BIOS:PRN: IMSAI CP/M BASIC INPUT/OUTPUT SYSTEM VERSION 1.31 REV 0

CONSOLE OUT

; MUST PRESERVE HL FOR NKM AND BOOTSTRAP

3DE3 3A0300 CONOUT: LDA IOBYT
3DE6 CD213E CALL RLCDISPATCH ; GO TO ONE OF FOLLOWING ADDRESSES
3DE9 413E DW TTYOUT ; BITS=0: USE TTY AS CONSOLE
3DEB 723E DW CRTOUT ; 1: CRT
3DEF F13D DW LIST ; 2: BATCH MODE: OUTPUT TO LIST DEVICE
3DE9 723E DW NULLO ; 3: UNASSIGNED

LIST OUT

3DF1 3A0300 LIST: LDA IOBYT ; BITS 7-6 TO 2-1
3DF4 07 RLC
3DF5 07 RLC
3DF6 CD213E CALL RLCDISPATCH
3DF9 413E DW TTYOUT ; 0: TTY
3DFB 723E DW CRTOUT ; 1: CRT
3DFD 843E DW LPTOUT ; 2: LINE PRINTER
3DEF 723E DW NULLO ; 3: UNASSIGNED

PUNCH OUT

3E01 3A0300 PUNCH: LDA IOBYT ; BITS 4-5 TO 1-2
3E04 0F RRC
3E05 0F RRC
3E06 0F RRC
3E07 CD223E CALL DISPATCH
3E0A 413E DW TTYOUT ; 0: TTY
3E0C 723E DW PUNO ; 1: HIGH SPEED PUNCH
3E0E 723E DW NULLO ; 2: UNASSIGNED
3E10 723E DW NULLO ; 3: UNASSIGNED

READER IN

3E12 3A0300 READER: LDA IOBYT ; BITS 3-2 TO 2-1
3E15 0F RRC
3E16 CD223E CALL DISPATCH
3E19 313E DW TTYIN ; 0: TTY
3E1B 883E DW RDRIN ; 1: HIGH SPEED
3E1D 883E DW NULLI ; 2: UNASSIGNED
3E1F 883E DW NULLI ; 3: UNASSIGNED
; SUBROUTINE TO DISPATCH TO ONE OF 4 FOLLOWING ADDRESSES
; DEPENDING ON IOBYTE BITS CALLER HAS POSITIONED IN
; BITS 2 AND 1 OF A.
; RETURNS TO SUBROUTINE CALL PRIOR TO CALL TO DISPATCH.

;RLCDISPATCH: RLC
;DISPATCH: ANI 06H
;MASK BITS
;SAVE CALLER'S H, GET TABLE ADDRESS

XTXL

;SET UP FOR DAD

MVI D,0

;INDEX INTO TABLE

DAD D

;TABLE WORD TO HL

MOV A,M

MOV H,M

;GO TO ROUTINE!

INX H

MOV L,A

POP D

XTHL

;PUT ADDRESS OF ROUTINE, GET CALLER'S H

RET
PHYSICAL DEVICE ROUTINES

; ADDRESSED BY LOGICAL DEVICE ROUTINES ABOVE,
; ALSO TTY AND CRT MAY HAVE EXTERNAL ENTRY POINTS

; TELETYPETE INPUT

TTYIN:    CALL TTYSTAT
          JZ  TTYIN       ; WAIT FOR A CHAR TO BE AVAILABLE
          IN  TTY        ; INPUT IT
          ANI 7FH        ; REMOVE PARITY
          RET

TTYSTAT:  ; USED HERE AND IN CONSTAT ABOVE
          IN  TTYS       ; GET STATUS
          ANI 02H        ; MASK BIT
          RET           ; A IS NON-0 IF CHAR AVAILABLE

; TELETYPETE OUTPUT

TTYOUT:   IN  TTYS       ; STATUS
          RRC           ; TEST BIT 0
          JNC TTYOUT    ; WAIT TILL READY TO ACCEPT CHARACTER
          MOV A,C       ; OUTPUT THE CHARACTER
          CPI CR        ; DONE EXCEPT CR
          RNZ
          JP TTYWT1     ; LOOP TAKES 9.5 USEC PER COUNT
          RET

; DELAY 100 MSEC FOR CR, FOR SLOW-RETURNING TERMINALS

TTYWT1:   DCX D         ; DEPENDS ON A7=0 AT ENTRY TO ROUTINE
          ORA D
          JP TTYWT1
          RET
CRT INPUT

CRTIN:  CALL CRTSTAT
JZ    CRTIN
IN    CRT
ANI   7FH
RET

CRTSTAT:  IN CRTS
ANI   02H
RET

MORE CONVENIENT CRT INPUT FOR LEAR-SIEGLER ADM-3

LSCRTIN:
CALL CRTIN ;GET CHAR FROM REGULAR ROUTINE
JZ    LSCRTIN
ANI   02H
RET

;NOTE: IF TYPEING "Z TO THE EDITOR ERASES THE SCREEN ON YOUR ADM-3,
;OPEN IT UP AND SET THE 'CLEAR SCREEN' SWITCH TO 'DISABLE'.

CRT OUTPUT

;MUST PRESERVE HL FOR BOOT, NXM
;CLOBBERS DE
CRTOUT:  IN CRTS
RRC
JNC CRTOUT
MOV A,C
OUT CRT
CPI CR
RNZ

;HOOK FOR USER TO PATCH IN CR WAIT IF DESIRED ON THIS CHANNEL
LXI    D,1
JMP TTYWT1
; LINE PRINTER OUT

LPTOUT:
; INSERT A STATUS CHECK HERE TO BE SAFE?

3E84 79 MOV A,C ; THE CHARACTER
3E85 D3F6 OUT PRINTER
3E87 C9 RET

; NULL DEVICE, FOR UNDEFINED DEVICES.

; FOR UNASSIGNED AND AND UNIMPLEMENTED INPUT DEVICES,
; HERE IS AN INFINITE SOURCE OF EOF'S:

3E88 3E1A NULLI: MVI A,CTRLZ
3E8A C9 RET
3E88 = NULLSTAT EQU NULLI ; CHARACTER ALWAYS READY

; DON'T USE CRT FOR UNASS INPUT DEVICES CAUSE IF THERE
; IS NO CRT ON SYSTEM BUT INTERFACE BOARD IS PRESENT,
; SYSTEM WILL HANG.

; FOR UNASS AND UNIMP OUTPUT DEVICES, USE CRT.
; IF NO CRT IS PRESENT, THIS IS AN INFINITE DATA SINK.

3E72 = Nullo EQU Crtout

; HERE IS WHERE TO PUT HIGH SPEED READER DRIVER

3E88 = Rdrin EQU NULLI ; MEANWHILE, USE NULL DEVICE
3E88 = Rdrstat EQU NULLSTAT

; HERE IS WHERE TO PUT HIGH SPEED PUNCH DRIVER

3E72 = Puno EQU Nullo ; MEANWHILE, USE NULL DEVICE
STARTUP & RESTART STUFF

; RESTART 7 ROUTINE. PRESUMABLY MEANS JMP TO NON-EXISTENT MEMORY
; TYPES "CRASH" AND TOP OF STACK (PRESUMED TO BE PC)
; AND BYTE TOP OF STACK POINTS TO

NXM: POP B ;GET PC OF CRASH (OR MAYBE GARBAGE)
LXI SP,100H ;SET UP STACK BELOW 100H
PUSH B ;SAVE THAT PC
LXI H,NXMMSG
CALL CONOMSG ;TYPE "CRASH"
POP H ;GET WHAT WAS ON STACK AT ENTRY TO NXM
MOV A,H ;HI ORDER BYTE
CALL HOUT ;HEX OUTPUT A
MOV A,L ;LO ORDER BYTE
CALL HOUT ;TYPE BYTE TOP OF STACK-1 POINTS TO: THIS MIGHT BE THE INSTRUCTION
; THAT CAUSED CRASH (RST-7, ETC)
MVI C,''
CALL CONOUT ;TYPE A SPACE
DCX H ;POINT ONE LESS
MOV A,M ;GET BYTE
CALL HOUT ;OUTPUT IT
;REBOOT THE SYSTEM, SAME AS ANY WARM RESTART
;JMP WBOOT
ROUTINE TO INITIATE WARM RESTART

;SET UP TO
;READ UNIT A, TRACK 0, SECTOR 1 TO LOCATION 0

WBOOT: MVI A, 1
STA CMD ;UNIT 0
STA SECT ;SECTOR 1
LXI H, 0
SHLD TRK ;TRACK 0
SHLD BUFADR ;RAM LOCATION 0

;PRESERVE IOBYTE IN B, SELECTED DISK IN C
;(BOOT DOES NOT ALTER THESE REGISTERS)
LDA LOGDISK ;CCP SETS THIS.
MOV C, A
LDA IOBYT
MOV B, A

;NOW DO READ - CLOBBERS IOBYTE, DISKN
CALL READ ;PRESERVES BC

;GO TO ROUTINE READ FROM SECTOR 1
JMP WBOOTR

;ROUTINE READ FROM SECTOR 1 RETURNS TO WBOOTR ENTRY TO THIS PACKAGE.
;ENTRY CURRENTLY JMXS DIRECTLY TO CONSOLE COMMAND PROCESSOR.
; OUT OF LINE STUFF FOR NXM
;
; TYPE MESSAGE HL POINTS TO ON CONSOLE. TERMINATED BY 0 BYTE

CONOMSG:

3EC8 7E MOV A,M ; GET A CHAR OF MESSAGE
3EC9 B7 ORA A ; SET FLAGS
3ECA 87 RZ ; DONE IF 0 BYTE
3ECB 4F MOV C,A ; TO C-REG FOR CONOUT
3ECC CDE33D CALL CONOUT ; OUTPUT IT ON CONSOLE
3ECF 23 INX H ; POINT NEXT CHARACTER
3ED0 C3C83E JMP CONOMSG ; KEEP OUTPUTTING TO END

NXMMSG: DB 'CRASH ',0 ; TEXT USED BY "NXM" ROUTINE

; HEX OUTPUT (A) TO CONSOLE

OUT: PUSH PSW
3EDB 0F RRC
3EDC 0F RRC
3EDD 0F RRC
3EDE 0F RRC
3EDF CDE33E CALL HOUTNIBL
3EE2 F1 POP PSW

HOUTNIBL:

3EE3 E60F ANI OFH ; MASK 4 BITS
3EE5 FE0A CPI 10 ; IS IT A OR BIGGER
3EE7 FAEC3E JM HNBL1 ; IF NO
3EEA C607 ADI 'A'-'0'-10 ; YES, ADD DIFFERENCE BETWEEN ASCII A AND 9+1
3EEC C630 HNBL1: ADI 'O' ; CONVERIT IT TO ASCII CHARACTER
3EEF 4F MOV C,A ; TO C REGISTER FOR CONOUT
3EEF C3E33D JMP CONOUT ; PRINT IT AND RETURN
ENDBIOS:
;
; I/O VARIABLES
;
; IN PAGE 0 RAM

0040  ORG  BIOSTOR
;
; DISC INTERFACE COMMAND STRING
FIFSTRING:

0040  CMD:  DS 1
0041  STAT: DS 1
0042  TRK:  DS 2
0044  SECT: DS 1
0045  BUFADR: DS 2

; OLD ENTRY POINT FOR SYS
; OFFICIAL ENTRY IS NOW VIA 5, BUT SOME PROGRAMS MAY STILL USE THIS

3FFD  ORG  MEMT-3
3FFD C38B3E  JMP  NXM  ; GO TO NXM ROUTINE WHICH WILL PRINT LOC OF "CALL 3FFD"

3EF2  ORG  ENDBIOS  ; MAKES ASSEMBLER TYPE OUT END OF VARIABLE CODE
3EF2  END