SOFTWARE REFERENCE MANUAL

HDOS SYSTEM

Chapter 1

Operating Instructions

HEATH COMPANY
BENTON HARBOR, MICHIGAN 49022

595-2335
Printed in the United States of America

TABLE OF CONTENTS

GENERAL OPERATION

At this point you should have completed the "System Set-Up Procedure". If this is your first time through this Manual, you should have also performed "Initialization," "Diagnostic," "Initialization" (again), "System Generation," and "ONECOPY", in that order.

Up until now, it was assumed that you were working exclusively with the distribution diskette and two spare diskettes. One of those diskettes became a duplicate of your distribution diskette, and the other became your BASIC disk. The purpose was to familiarize you with a minimum of HDOS and your system. Now that you have gained "hands-on" experience with the system, you are ready to further explore the capabilities of HDOS.

You will need to go back to the "System Set-Up Procedure" at a later time. All new diskettes must be initialized. You should also run the M option in TEST on all diskettes. This option tells you if there are any bad sectors on the diskette. If there are, you can (through INIT) instruct HDOS not to write any data there.

You might also wish to create several system volumes, each with a different complement of resources such as BASIC (Extended BASIC), DBUG, EDIT, and ASM. You could also include INIT, TEST, ONECOPY, and SYSGEN on some of the system volumes. It would be best to accomplish this by first creating a "master" system volume by copying the entire contents of the distribution diskette. You can do this by means of ONECOPY. You should then store the distribution diskette in a safe place and use the master system volume instead; substituting it in any procedure which calls for the distribution diskette.

This section will tell you how to use HDOS to create and manipulate files. Like "Theory of Operation," its purpose is tutorial. You cannot successfully manage a system such as HDOS without a basic understanding of its functions and capabilities. Perhaps you had only intended to use your diskette as a fast storage device for the BASIC interpreter and a collection of programs. You can do this with a minimal understanding of "General Operation," but not very conveniently or efficiently. The HDOS commands in this section will allow you to examine and manipulate your files directly. They give you power and flexibility similar to that of the system manager in a computation center, as well as the responsibility. The computer and disk drive are the main resources in your personal computation center; you are the system manager.

NOTE: You can use two disk drives with HDOS. Some of the commands and capabilities described in this chapter will not be very useful if you only have one drive. However, HDOS will run on either configuration.

For details on how to use BASIC or other system resources such as DBUG or EDIT, refer to the appropriate sections in the Software Manuals. These programs will use some of the techniques explained here to allow you to write and run your own programs, using the diskette as a mass storage device. You may wish to go back and review the "Theory of Operation" before proceeding with "General Operation."

File Names

Each file must have a unique name so HDOS can store it and find it on the diskette(s). HDOS will not allow two files with identical names at the same time on the same diskette. The general format for file names is:

```
DEV:FNAME.EXT
```

Where DEV: is the device on which the file is stored, usually SY∅: or SY1:; FNAME is the general name of the file; and .EXT is an "extension" which allows HDOS to distinguish between files of the same general category. Remember you can only refer to SY1: if you have two drives on your system.

For example, you might write two programs to search through a list of telephone numbers for the one belonging to John Doe. Such a program written in BASIC might be stored in a file called "PHONE.BAS", while the same program written in assembly language could be stored in a file called "PHONE.ASM".

A "fully qualified" file name includes a device specification, as in the following examples:

```
SY1:PHONE.ASM
SY0:PHONE.BAS
```

However, the device specification may be omitted if you refer to a file that resides on the system volume (SY∅:). The following two examples are equivalent:

```
SY0:PHONE.BAS
PHONE.BAS
```

The FNAME field is comparable to a person's last name, while the .EXT field is similar to a first name. Personal names such as Jim Jones, Nancy Jones, and Jim Miller are all different, although they may have something in common. In the same way, file names such as TEST.BAS, TEST.ASM, and TEMP.ASM are all different, even though they have the same general name or extension.

The DEV: portion of the file name is a special case. Normally, one thinks of files being stored somewhere, in this case on a diskette. A device specification of SYØ: or SY1: indicates that the file is stored on one of the two allowable drive units. The same file will have a different device specification, depending on whether it is mounted on SYØ: or on SY1:

When no device is specified, HDOS normally assumes that SYØ: is the intended device. Allowing the same file to be associated with either SYØ: or SY1:, depending on which it is mounted on, is permitted by the "device-independent" characteristics of HDOS, as discussed in "Theory of Operation." Another feature of device independence allows you to specify only a device, in some cases, in exactly the same manner as you would specify a file name. For example, one of the commands discussed below shows you how to copy data from one file to another. The general format for the command is:

```
>COPY△DEV:FNAME.EXT=DEV:FNAME.EXT ↵
```

If you wished to print the contents of a file on the system console, you would simply type:

```
>COPY△TT:=DEV:FNAME.EXT ↵
```
Where "△" indicates a required space.

Of course, you must substitute a real file name in this example. The device specification TT: refers to the system console. The valid device specifications under HDOS are:

SY0:	System drive zero.
SY1:	System drive one.
TT:	Console terminal.
AT:	Alternate terminal.
LP:	Line printer.
ND:	Null device, (used for test purposes).

The two "directory" devices in the system, SYØ: and SY1:, are the storage devices. The diskettes mounted on them contain directories that allow many files to be referenced. The "non-directory" devices, TT:, AT:, and ND:, have no storage capability; thus, no files are directly associated with them. You may specify an FNAME and EXT with them, such as TT:OUTPUT.DOC, but the system will ignore everything except the name of the non-directly device.

The "null device" is not a "real" peripheral. Any data that is output to ND: is simply discarded. Any input attempt from ND: results in an end-of-file condition. The null device is provided as a program debugging aid. Programs can

write to ND: without using the extra storage space and access time an output file would require.

It is usually best to follow certain conventions when you are naming files. The two previous examples were: "PHONE.BAS" and "PHONE.ASM". The general name of each file is the same (PHONE), indicating they have similar uses or at least have something in common. If you use the assembler provided with this system (ASM), you might assemble the contents of PHONE.ASM to produce an executable machine-code program. If you follow the recommended conventions, this machine-code program would be stored in a file called "PHONE.ABS". (Since no device was specified, HDOS assumes SY∅:). Conventions such as this allow you to keep track of the various files without keeping a separate list.

The FNAME portion of the file name is limited to eight characters, but need not consist of more than one character. The ".EXT" portion is limited to three characters, but can be omitted entirely or shortened to one or two characters. The characters themselves must be either letters or numbers, and the first character of a FNAME must be an alphabetical letter.

The following is a list of valid file names:

SY0:PRIME.BAS	
SY1:LETTER.DOC	
INCOMTAX.ASM	HDOS assume SY∅: when no device is specified.
TT:	The system console.

The following is a list of invalid file names:

SY0:TELEPHONE.LST	FNAME is too long.
COMPUTER.PROG	EXT is too long.
SY0:8LETTERS.DOC	FNAME cannot begin with a number.
SY1:PROG,NEW.BAS	Only letters and numbers are allowed.
SY3:MYPROG.ASM	SY3: is an illegal specification.

In order to perform any manipulations on files, you must specify the name of the file in the proper format. HDOS, however, obeys a convention which makes it simpler to run programs that are stored on files. Programs which are written in assembly language and then translated into executable machine-code should be stored in files with the .ABS extension. An example is INIT. The full name of INIT on your distribution diskette is: "SY∅:INIT.ABS"; yet you ran the program by simply typing INIT while in the command mode. HDOS recognizes the .ABS extension as an identifier for an executable program.

In general, there are several conventions for the extension, and you should adhere to them as much as possible. These are:

ABS	Absolute binary machine code.
ACM	Assembler-common subroutines.
ASM	Assembly language source programs.
BAS	BASIC programs.
DOC	Documentation, such as instructions for using programs.
DVD	Device driver subroutines.
SYS	Operating system programs.

Command Mode

There are a number of commands that will cause HDOS to list the contents of files, copy files, rename files, delete files, run programs, configure your system, and give status reports. In general, each command is typed on the system console in a strict format. If you do not use the correct format or "syntax", then HDOS will respond with at least one error message. Error messages are nothing to be alarmed about. They simply indicate that you should attempt to perform the command again, using the correct method. Refer to "Appendix B" for more information about error messages.

Most of the commands follow the same general format, but this can be very flexible, depending upon what you wish to accomplish. The most general command type is:

```
>COMMAND△DEV:FNAME.EXT ↵
```

In other words, most commands deal with files, although there are exceptions and several variations. If you omit the DEV: portion of the file name, then HDOS will assume you refer to SY0:. To reference SY1:, you must type SY1: specifically. It is a legal reference ony if you have a second drive on your system. Other variations are possible, and are explained below in "Wildcards and Multiple File Designations" (Page 1-22).

In order to test these commands, HDOS should be booted up and in the command mode, with a prompt printed at the left margin of the system console. The instructions for using these commands are printed below. You should perform the numbered instructions in the sequence given, and refer to the paragraphs printed below the instructions for an explanation of the function of the various commands.

File Listings

One of the most basic system commands allows you to type the contents of a file on the console. Some files contain written text in ASCII, which is meaningful when listed. Such files have a .BAS or .DOC extension. Other files are written in binary code and have no meaning when listed on the console; these files have .ABS, .SYS or .DVD extensions.

One file that contains meaningful and helpful information is called "SYØ:SYSHELP.DOC". Take the following step to examine the contents of this file:

1. Type TYPE△SYSHELP.DOC and a carriage return.

   Note that a device was not specified. In this case, HDOS assumes you refer to SYØ:. The result of this system command is a printed list, as follows:

```
Valid System Commands:

HELP             Print This List
BYE              Dismount all Mounted Disks, and Reboot
CAT [DEV:]       List Files on Disk
COPY TO=FROM     Copy FROM file to TO
DATE [NEWDATE]   Display or Set Date
DELETE FNAME     Delete File(s)
DISMOUNT DEV:    Dismount Volume from Drive
PIP              Execute PIP
MOUNT DEV:       Mount Volume on Device
RENAME TO=FROM   Rename File FROM to TO
RUN FNAME        Run Program
SET dev:.opt     Select Option for Device
SET HELP         Documentation for SET Command
STATUS           Display Disk Statistics
TYPE FNAME       Type file contents on terminal
VER              Display the Current Version of HDOS

FNAME            Same as RUN SY0:FNAME.ABS
```

This is a list of the commands that are permissible under HDOS. Note that you can reproduce this listing from the command mode at any time, by simply typing HELP and a carriage return. HDOS actually translates the HELP command to TYPE△SY0:SYSHELP.DOC.

Cataloging Files

The CAT command produces another useful listing of information about a file or files. You may use it with or without a device specification, either SYØ: or SY1:, but not both. This command may or may not be used with a /S modifier, depending upon whether you wish to list information about all the files in the system or just those which are not essential system files, such as BASIC.ABS. You may also specify a file name, as in the following example, if you wish only to examine a single file. If no file name is specified, HDOS assumes you wish to catalog multiple files.

1. Type CAT△HDOS.SYS/S and a carriage return. The following listing will be printed:

```
>CAT△HDOS.SYS/S ↵

NAME    .EXT    SIZE    DATE          FLAGS    21-JUN-79

HDOS    .SYS    26      25-MAY-79     SLW

    1 FILES, USING 26 SECTORS (nnn FREE)
```

   SYØ:HDOS.SYS is a file that contains a major portion of the operating system. It consists of 26 sectors and has FLAGS which indicate that it is a system file, that the file flags are locked, and that the file is write-protected. Refer to the FLAGS section for more information about setting and resetting the flags, and the results of such manipulations.

   In the "DATE" column, you will find the date when the file was first created. The last line of the listing tells how many files were listed, the total number of sectors they occupy, and the number of sectors which are free for other purposes, such as program storage.

2. Type CAT/S and a carriage return.

   The result of this action is a list of all the files on SYØ:. Most of them are of no particular interest to you at this time because they are system files and cannot be manipulated in any useful fashion. If you try to list them, you will probably receive a listing of what is called "garbage" in computer jargon.

Running Programs

When you initialized your spare diskette(s) in the "System Set-Up Procedure," you were required to type "INIT". Had you desired, you could have typed:

```
>RUN△SY0:INIT.ABS ↵
```

HDOS recognizes the contents of any file with the .ABS extension as an executable machine-code program. If you type only the FNAME portion of the file name, while in the command mode, HDOS assumes you mean RUN SYØ:FNAME.ABS. To run BASIC, assuming it has been transferred to your system volume, simply type:

```
>BASIC ↵
```

Refer to the appropriate section of your Software Manual for more information about system resources such as BASIC.

Examining and Changing the Current Date

When you booted up the system, you were required to enter the current date. If you wish to examine the date, type DATE while HDOS is in the command mode. If you wish to change the current date, simply type DATE△ DD-MMM-YY, using the same format you were required to use when HDOS was booted up.

```
>DATE△17-JAN-78 ↵
```

Duplicating Files

You may wish to have an extra copy of a file for the purposes of modification or safekeeping. You may use the COPY command to accomplish this from the command mode. In general, all commands are a form of COPY command. When you list the contents of a file, you are actually "copying" the file to the system console. When you RUN a program, you are actually "copying" the contents of a file into the memory of your computer. This concept will be clarified in the section entitled "Peripheral Interchange".

For now, you may copy one of the HDOS files by typing COPY△TEMP.ABS=ONECOPY.ABS and a carriage return. The result will be a console output, as follows:

```
>COPY△TEMP.ABS=ONECOPY.ABS ®

1 FILES COPIED

>
```

You have created an exact duplicate on the system volume of the file containing the program ONECOPY. This file is also executable; you can run the program it contains by typing any of the following commands:

```
>TEMP ®
>RUN△TEMP.ABS ®
>RUN△SY0:TEMP.ABS ®
```

A copy of a system file, such as SY∅:HDOS.SYS, is not particularly useful. System files are a special case and may be copied in a usable form only by means of the program SYSGEN, as explained in "System Generation."

Copying Files to and From Peripherals

The console terminal is a device with the name TT:, just as the system drive is a device with the name SY∅:. It is possible to copy files to TT:, and this is exactly the effect of the TYPE command. It is also possible to copy files from TT:, as demonstrated in the following example:

```
>COPY△THISFILE.DOC=TT: ®
THIS△IS△A△TEST. ®
^D(CTRL-D typed)
1 FILES COPIED
>
```

It is possible that you would wish to test the COPY command without adding any files to your system volume. For this purpose, a "null" device is provided by HDOS. The null device driver resides on a file called SY∅:ND.DVD. Copying data to the null device has no effect except to exercise HDOS and use up a little time and electricity. You may test this feature by typing:

```
>COPY△ND:=SY0:THISFILE.DOC ®
```

Two general-purpose device drivers reside on the distribution disk. ATH84.DVD is used to address port 300Q and ATH85.DVD is configured for port 374Q. You should rename the appropriate file as "AT.DVD", and then re-boot the system so HDOS can recognize it. The "AT" portion of the file name stands for "Alternate Terminal". You may copy files to or from this device in much the same way as you would copy files to or from TT:, the system console. Of course, to do this, you must have a serial terminal device connected to your system. Refer to the "AT: Option" on Page 1-28 for more information.

A COPY command to AT: will type the contents of the file on the alternate terminal.

```
>COPY△AT:=SYSHELP.DOC ↵
VALID SYSTEM COMMANDS:

HELP             PRINT THIS LIST
BYE              DISMOUNT ALL MOUNTED
                 DISKS AND REBOOT
CAT [DEV:]       LIST FILES ON DISK
COPY TO=FROM     COPY FROM FILE TO TO
DATE [NEWDATE]   DISPLAY OR SET DATE
DELETE FNAME     DELETE FILE(S)
DISMOUNT DEV:    DISMOUNT VOLUME FROM DRIVE
PIP              EXECUTE PIP
MOUNT DEV:       MOUNT VOLUME ON DEVICE
RENAME TO=FROM   RENAME FILE FROM TO TO
RUN FNAME        RUN PROGRAM
SET DEV: OPT     SELECT OPTION FOR DEVICE
SET HELP         OOCUMENTATION FOR SET COMMAND
STATUS           DISPLAY DISK STATISTICS
TYPE FNAME       TYPE FILE CONTENTS ON TERMINAL
VER              DISPLAY THE CURRENT VERSION OF HDOS
FNAME            SAME AS RUN SYO:FNAME.ABS
>                (Prompt appears on original terminal.)
```

This list was printed on the alternate terminal.

The command >COPY△LP:=SYSHELP.DOC↵ will print the contents of this file on an H14 Line Printer, if you have one connected to your system.

Renaming Files

By using the RENAME command, you can change the name of any file except the essential system files. System files cannot be renamed because they are write-protected and locked. For an explanation of how write-protection is accomplished, refer to "FLAGS" (Page 1-20). As an example of the use of the RENAME command, you may rename the file which was created in the previous section by typing the following:

```
>RENAME△NEW.ABS=TEMP.ABS ↵
```

Deleting Files

From time to time you may decide you have too many files in your system. You can get ride of extraneous files by using the DELETE command. This is the most "dangerous" command in HDOS because there is no way to recover a file that has been deleted except by copying it from a "back-up" diskette such as the distribution diskette. This command is the reason for write-protection of your distribution diskette and the system files. Write-protection insures that essential system files will not be inadvertently destroyed. As a "safe" example of this command, you may delete the files that you copied in the previous section by typing:

```
>DELETE△NEW.ABS,THISFILE.DOC ↵
```

Mounting and Dismounting Diskettes

The diskette drive units are known as directory devices. HDOS maintains a directory on the diskettes which are "mounted" on the drives. The operating system also uses a table which "maps" the location of every file on a diskette. For efficiency, parts of the directory and map are kept in memory while HDOS is running. When a diskette is removed from the system, or "dismounted", these tables must be written back to the diskette to reflect the most recent changes in the status of the various files. You must use the MOUNT and DISMOUNT commands to insert and remove diskettes and to prevent data from being lost.

Use the MOUNT command when you install a diskette. The correct syntax is:

```
>MOUNT△SY1: ↵
```
(△ signifies a necessary blank)

This command informs the operating system that there is a diskette installed in SY1:.HDOS reads the table and directory segments from the diskette into the computer memory in preparation for your file commands. HDOS will not recognize any commands dealing with SY1: until the diskette is properly mounted.

When you are finished using the diskette mounted in SY1:, you must use the DISMOUNT command to inform HDOS. This will allow you to mount another diskette in SY1:. The correct command syntax is:

```
>DISMOUNT△SY1:↵
```

This command causes the updated table and directory segments to be written back onto the diskette so that any changes to its files will be preserved. DO NOT remove the diskette before it is dismounted, or files may be lost.

The mounting procedure for SYØ: is automatically accomplished during the "Bootstrap." You cannot use the MOUNT command with SYØ:, but you can use the DISMOUNT command. Since SYØ: is the system volume, you lose HDOS when it is dismounted. Because of this, dismounting SYØ: will force an entry to the "Bootstrap" procedure. As an example:

```
>DISMOUNT SY0: ↵
VOLUME 001, DISMOUNTED FROM SY0:
LABEL: MY SYSTEM VOLUME

INSTALL A BOOTABLE DISK IN SY0:. HIT RETURN TO REBOOT: ↵

TYPE SPACES TO DETERMINE BAUD RATE △△△△△

ACTION?<BOOT>
```

VER

The VER command displays the current version of HDOS. The correct format is:

```
>VER ↵

HDOS              VERSION    1.5
>
```

This number is used by HDOS to ensure that you have the correct revisions on your system volume. If the version number of a program is incompatible, HDOS will issue a SYSERR.

Bye

The BYE command automatically dismounts all mounted devices and returns to the "Bootstrap". The correct procedure is:

```
>BYE ↵
VOLUME 001, DISMOUNTED FROM SY0:
LABEL: HDOS DISK
INSTALL A BOOTABLE DISK IN SY0: HIT RETURN TO REBOOT: ↵

TYPE SPACES TO DETERMINE BAUD RATE

ACTION? <BOOT>
```

Peripheral Interchange

To execute the commands explained above, HDOS uses a system program called "PIP", an acronym for "Peripheral Interchange Program". When you type a command, the system monitor first decodes it, using a program which resides on the file SYØ:SYSCMD.SYS. The monitor then either prints an error message, which it finds on SYØ:ERRORMSG.SYS, or passes the command to PIP for execution. PIP normally resides on a file called "SYØ:PIP.ABS". In order to execute the command, HDOS reads SYØ:PIP.ABS and reads PIP into your system's memory. The command is passed to PIP, which uses other system resources such as the device drivers to execute the command.

Since the file on which PIP resides has the .ABS extension, by convention you may assume that it contains an executable machine-code program. You can use PIP directly, by simply typing PIP from the command mode. The result should be a printed prompt, as follows:

```
>PIP ↵
:P:
```

To exit from PIP back to the command mode, type CTRL-D on the system console.

A prompt (:P:) will be printed at the left margin of the system console when the program is awaiting input. The legal commands in PIP are somewhat different from "normal" system commands. The general form is the copy command, where a "destination" is followed by an "=", which is then followed by one or more "source" specifications. For example:

```
:P:SY0:TEMP.ABS=SY0:ONECOPY.ABS ↵

1 FILES COPIED
:P:
```

This example has the same effect as the COPY command, which was explained in "Duplicating Files." In this case, the "destination" is SY∅:TEMP.ABS and the "source" is SY∅:ONECOPY.ABS.

If you do not specify a destination, PIP assumes that you refer to the system console, and will copy the contents of the file onto it. Since the console is a device, like SY∅: or SY1:, it has a name (TT:), as explained above. Each of the following commands has exactly the same result:

```
:P:ONECOPY.ABS ↵
:P:TT:=ONECOPY.ABS ↵
:P:TT:=SY0:ONECOPY.ABS ↵
>TYPE△ONECOPY.ABS ↵
>COPY△TT:=SY0:ONECOPY.ABS ↵
```

If you attempt any of these examples, the result will be a listing of binary "garbage". Hit CTRL-C to terminate the output. SY∅:ONECOPY.ABS contains a machine-code program rather than written text in ASCII.

It is also possible to catalog, rename, and delete files under PIP. These functions are accomplished by means of a "switch", which is typed after the general copy command, or by itself in response to the prompt. The legal switches are:

/L	Gives a catalog listing of non-system files.
/L/S	Gives a catalog listing of both system and non-system files.
/B	Gives a "brief" catalog listing of non-system files.
/B/S	Gives a "brief" catalog listing of all files.
/R	Used for renaming files.
/DEL	Used for deletion of files.
/MOU	Used to mount the specified device.
/DIS	Used to dismount the specified device.
/RES	Used to reset the specified device.
/VER	Gives the current version of PIP.

The first four switches allow you to obtain a directory listing of the various files in the system. The usage is similar to the "CAT" command, which was explained in "Cataloging Files."

The /L switch produces a directory listing, or catalog, of those files that are not an essential part of HDOS but are listed in the directory. If you performed the "copy" example above, at least one file should be listed in exactly the same format demonstrated in "Cataloging Files." The /L/S switch allows you to list all the files in the system, exactly like the CAT/S command explained in "Cataloging Files." The /S switch is actually a modifier for other switches or commands, which causes the system files to be included with those files which you have created or copied. It is used in PIP exactly as it is used with the CAT command.

If you only wish to know whether a file is listed in the directory, you can type its name, followed by a /L or /L/S switch, depending upon whether it is a system file or one of the "extra" files. An example is as follows:

```
:P:AT:=HDOS.SYS /L/S ↵

    NAME     .EXT    SIZE    DATE            FLAGS       12-MAY-78

    HDOS     .SYS    24      09-MAY-78       SLW

    1 FILES, USING 24 SECTORS (nnn FREE)
:P:
```

Note that this PIP command had the same effect as the CAT HDOS.SYS/S command, used in "Cataloging Files." You can obtain a more abbreviated listing by using the /B and /B/S switches. They are used in exactly the same way as /L and /L/S; however, they omit details such as the creation date of the file when the list is printed. In addition, note that a different destination, AT: instead of the default TT: was specified. This list was printed on the alternate terminal, instead of the console.

The /R switch is used in the same manner as the RENAME command. For example:

```
:P:NEWFILE.ABS=TEMP.ABS/R ↵
:P:
```

Note that this switch does not allow you to rename essential system files, and that it fails if the source file does not exist in the first place or if the destination file already exists.

The /DEL switch, like the DELETE command, can be dangerous if it is misused. You can never recover a file that is deleted with this command, except by making a new copy from a back-up file. It is used in a way similar to the /R switch, as follows:

```
:P:NEWFILE.ABS/DEL ↵
:P:
```

The /DIS and /MOU switches are used in the same manner as the DISMOUNT and MOUNT command. They allow you to change diskettes in the drives. For example:

```
:P: SY1:/DIS ↵
VOLUME 090 DISMOUNTED FROM SY1:
LABEL: BASIC DATA FILES
:P: SY1:/MOU ↵
VOLUME 082 MOUNTED ON SY1:
LABEL: ASSEMBLY PROGRAMS
:P:
```

There is an important difference between the HDOS commands and the PIP switches, /DIS and /MOU. In PIP, you dismount the diskette from SYØ: and proceed. In effect, you are forcing PIP to be a "stand-alone utility". This feature is useful if you have two drives and wish to make copies of diskettes.

However, if you do dismount the system volume, you will enter the boot routine when you exit PIP.

The /RES switch does both, dismount and mount operations. For example, if you want to change the diskette in SY1:,

```
:P: SY1:/RES ↵

 PLEASE REPLACE THE DISKETTE IN SY1:
 VOLUME 020 DISMOUNTED FROM SY1:
 LABEL: F&G
 VOLUME 040 MOUNTED ON SY1:
 LABEL: RECORDS

:P:
```

When the message "Please replace the diskette in SYn:" is displayed, remove the diskette in the drive and install the diskette you want mounted. The /RES switch will automatically continue the operations when you close the drive door.

You can also reset SYØ:. If you do this, you are using PIP as a stand-alone program, and making HDOS inactive. **When you exit PIP, you will enter the BOOT routine.**

The /VER switch will display the current version of PIP. The version number is used by the software to ensure that you have the correct revisions on your system volume. Generally, the version numbers of HDOS, PIP, and utility programs will correspond. If you get a SYS ERR #40, it is because your software has incompatible version numbers.

When you become accustomed to PIP, you will probably find its "shorthand" notation more convenient than the command mode. Both PIP and the command mode have special features which make them more useful than it may seem at this point. These features are explained in "Wildcards and Multiple File Designations."

The various file functions of copying, renaming, etc., are not actually duplicated between PIP and the system command mode, as it may seem. The various system commands, such as COPY, DELETE, etc., actually cause PIP to be run to perform the function. If you have only one or two file operations to perform, you will probably find it more convenient to use the command mode forms. If you plan more extensive file manipulation, it will be faster to run PIP and command PIP directly.

Remember that you can exit from PIP back to the command mode by typing CTRL-D on the system console. You can also obtain a console listing of PIP commands by typing HELP from the PIP command mode. This causes the file SYØ:HELP. to be listed on the console. For example:

```
:P:HELP

PIP COMMAND FORMAT:

        DEST=SOURCE1,...,SOURCEN/SWITCH1.../SWITCHN

Switch  Meaning

/LIST           List Device Directory
/BRIEF          Short Listing of Device Directory
/DELETE         Delete Source Files
/DISMOUNT       Dismount Volume from Device
/MOUNT          Mount Volume on Device
/RENAME         Rename Source File(s) to Destination Name
/RESET          Dismount/Mount Volume on Device
/SHOW           Display Files with 'S' Flag Set
/VERSION        Display the Current Version of PIP

1 FILES COPIED
:P:
```

Flags

You may decide to write-protect your own files to prevent them from being inadvertently deleted or modified. You can do this with the FLAGS program. In order to test FLAGS, you will need a file to manipulate. You may execute the following steps, from the command mode, to create a file and examine its contents:

1. Type COPY△NEWFILE. DOC=TT: and a carriage return.
2. Type THIS△IS△A△TEST. and a carriage return.
3. Type CTRL-D on the system console.
4. Type TYPE△NEWFILE.DOC and a carriage return.

The result of these steps should be printed output on the system console, as follows:

```
>COPY△NEWFILE.DOC=TT: ®
THIS△IS△A△TEST. ®
<CTRL-D>
1 FILES COPIED
>TYPE△NEWFILE.DOC ®
THIS IS A TEST.
>
```

You now have a file on SYØ: that can be used to test the FLAGS program, as follows:

5. Type FLAGS and a carriage return. HDOS will turn control over to the FLAGS program, which will identify itself and ask if you would like instructions.

6. Type YES and a carriage return.

FLAGS IS USED TO SET AND/OR CLEAR THE FILE FLAGS. WHEN PROMPTED FOR THE NEW FLAGS, SPECIFY ALL THE FLAGS THAT ARE TO BE SET. NOTE THAT IF YOU SET THE "L" FLAG, YOU WILL NOT BE ABLE TO CLEAR IT AGAIN. THE LEGAL FLAGS ARE:

W WRITE PROTECT FILE. MAY NOT BE REMAMED, REPLACED, OR DELETED.
S SUPPRESS NORMAL LISTING OR COPYING OF FILE.
L LOCK THE FILE FROM FURTHER FLAG CHANGES.

As the instructions printed on your console explained, the W flag, when set, prevents a file from being written on. You cannot change the contents of a write-protected file unless you clear the W flag. The S flag prevents the directory listing of files when the CAT command is used, or when the /L or /B switches are used in PIP. You can override this flag by using the CAT/S command, or /L/S and /B/S, respectively. The L flag locks the other flags. Once the L flag is set, you will not be able to change the flags for a given file. If you also set the "W" flag, you will not be able to delete the file or rename the file.

The FLAGS program will prompt you with "FILE NAME?" whenever it is ready for a new file. Type the name of the file and a carriage return to examine the flags. The CURRENT FLAGS will be identified and you will be prompted to enter new flags. If you simply type a carriage return, the flags will be cleared, provided they are not locked. The new flags will be whatever you choose to enter from the console, as demonstrated in the following examples:

```
FILE NAME?  NEWFILE.DOC ↵
CURRENT FLAGS =
 NEW FLAGS:  W ↵

FILE NAME?  NEWFILE.DOC ↵
CURRENT FLAGS = W
 NEW FLAGS:  ↵

FILE NAME?  NEWFILE.DOC ↵
CURRENT FLAGS =
 NEW FLAGS:  SL ↵
```

```
FILE NAME?  NEWFILE.DOC ↵
CURRENT FLAGS = SL
THIS FILE IS LOCKED; ITS FLAGS CANNOT BE CHANGED.

FILE NAME? <CTRL-D>

>
```

You cannot change the flags further, since you set the "L" flag. However, since the "W" flag is not set, you can delete the file. If you had also set the "W" flag along with the "L" flag, the file would be permanently recorded on your system volume until you reinitialize the diskette and create a new system volume by means of SYSGEN.

Wildcards and Multiple File Designation

There are basically two ways to manipulate more than one file with the same command. The simplest way to use more than one file name in the source or destination fields of the commands. For example:

```
>TYPE△NEWFILE.DOC,ERRORMSG.SYS/S ↵
THIS IS A TEST.
128CTRL-C STRUCK
129CTRL-B STRUCK
130DATA EXHAUSTED  <CTRL-C>
```

The contents of both files are listed on the console. Typing CTRL-C terminates the print-out.

When multiple file designations are used with the COPY command, the result will be a file that is a concatenation of the files which are specified on the right-hand side of the = symbol. An example which uses the files listed above is as follows:

```
>COPY△BIGFILE.DOC=NEWFILE.DOC,ERRORMSG.SYS/S ↵
```

The result of this command is a file on SYØ: called BIGFILE.DOC, which is a combination of NEWFILE.DOC and ERRORMSG.SYS.

You can also specify multiple file names in PIP, and with the DELETE and CAT commands.

The "wildcard" is another way of accessing multiple files. By typing an * in place of either the FNAME or .EXT portions of the file name, or both, the command will affect all files that have the unmodified portion of the specified file name in common. The following example demonstrates the use of a wildcard.

```
>PIP ↵
:P:*.SYS/B/S ↵

HDOS      .SYS   SYSCMD    .SYS   HDOSOVLO .SYS   ERRORMSG.SYS
RGT       .SYS   GRT       .SYS   DIRECT   .SYS   HDOSOVLI.SYS
:P:
```

This is a brief listing of the essential system files. Note that all the file names have the .SYS extension, and that the FNAME portions are all unique. The wildcard allows either portion to be specified ambiguously.

You can use another type of wildcard as a substitute for a single letter in a portion of a file name. This wildcard is the "?". Since the FNAME portion of a file name may be up to eight characters in length, and the .EXT portion may be up to three characters, the wildcard "????????.???" is exactly the same as "*.*". The following example demonstrates the use of the "?" wildcard.

```
>CAT△ONECOPY.A?S/S ↵

NAME     .EXT    SIZE      DATE        FLAGS   11-MAY-78

ONECOPY .ABS     19        09-MAY-78     SW
   1 FILES, USING 19 SECTORS (20 FREE)
>
```

Multiple file specifications and wildcards can both be used with either the command mode or PIP. If you have a dual-drive system, you can transfer many files between the two devices in only one operation. First mount an initialized diskette on the second drive, and then specify SY1:*.* = SYØ:*.* by means of a COPY command or PIP. All files, except system files, will be transferred from SYØ: to SY1:, assuming that some of the files on SYØ: lacked the S flag.

You may use this technique to copy files from your distribution diskette to your system volume. It will copy files such as BASIC.ABS and INIT.ABS. You can have the distribution diskette and the system volume mounted in either SYØ: and SY1:, or SY1: and SYØ:, respectively. Of course, the distribution diskette must be the source and the system volume must be the destination in order to transfer the "extra" files. The example shown assumes that SY1: is the destination.

Note that you can use the CAT command and the /L and /S switches with multiple file designations and wildcards, but they may not refer to more than one device in the same command. The next example is invalid and will cause an error message:

```
>CAT△SYSHELP.DOC,SY1:FNAME.EXT ↵
```

This example is invalid because both SYØ: and SY1: are used in the same CAT command.

Onecopy

This program allows you to copy files from one diskette to another using only a single-drive system. ONECOPY will copy multiple files in a single operation, or you can specify individual files. (See "System Set-Up" in the Introduction Manual for more information.)

ONECOPY will allow you to copy multiple files in a single operation, if you so desire. The following example will copy all the files listed above onto your system volume. If you decide to do this, you can delete those files which you don't need by use of the HDOS command mode or PIP.

To copy all the "extra" files, simply type *.* instead of a specific file name at the :OC: prompt. ONECOPY will notify you when this has been accomplished and issue another prompt.

ONECOPY uses the /V,/L,/L/S,/B, and /B/S switches in exactly the same way as PIP. Refer to "Peripheral Interchange" for an explanation of the function of these switches.

ONECOPY has one switch which is different than PIP. This is the /MOU switch. It makes it possible to switch to a different diskette whenever a prompt is printed at the left margin of the system console. The following example shows how this may be accomplished:

```
:OC:/MOU ↵

INSERT NEW DISK:          {Install an initialized diskette and type a carriage return.}
:OC:
```

The new diskette is now the source. You may specify one or more files to be copied from it, and you may use the other switches to determine its contents. If you wish to switch to yet another diskette, simply type /MOU again. Note that this is NOT the same function as the PIP switch "/MOU" performs. ONECOPY's /MOU actually does a "dismount" and "mount" function.

SYSTEM OPTIMIZATION

The SET Command

You can use the SET command to configure your system to the particular input and output devices that you have. For example, you can use the SET command to specify how many characters your console terminal can handle on a line. If you have an H36, or some other terminal that can handle 132 characters on a line, you would use:

```
>SET△TT:△WIDTH△132 ↵
```

Once you have SET one of these options, HDOS saves the information so you do not have to re-SET each time you re-boot HDOS. However, if you change your hardware, you can re-use the SET command to change these options. Again, the change will become permanent until you re-use SET.

The HDOS system contains a SET VER that prints the version of the SET program. You would enter:

```
>SET△VER ↵
```

The HDOS system contains a SET HELP command that will give you information about the SET command. You would enter:

```
>SET△HELP ↵
```

There is also a help entry for each device. To obtain information about the SET options for your console terminal, you would enter:

```
>SET△TT:△HELP ↵
```

The tables that follow summarize the devices that have SET options. Also, the SET options for each device are summarized and the default settings are shown. To make the most use of this information, you will have to refer to the section on device drivers. You may have to rename a device driver.

With the exception of the general SET HELP or SET VER commands, the SET commands have the form:

```
>SETΔdd:Δ00000 ⏎
```

In this, "dd:" represents a device name. Table A, on the next page, lists all HDOS devices. Tables B, C, D, and E list the options for each device.

NOTE: In the following tables, the options that are SET on your distribution diskette are marked by an asterisk.

HELP	Prints the help message.
VER	Prints the version number of SET.

TABLE A

HDOS DEVICES

Device Name	Description
SY:	System disk drives (all drives installed on system)
TT:	Console terminal, input and output
LP:	Line printed (H14 only)
AT:	Alternate terminal

TABLE B

SETΔSY:ΔOPTION

STEPΔn	Sets the step time between tracks on the diskette. Step time for both drives is set by the command. Use TEST to determine the value of "n"; it should be between 8 and 30. The seek time of the slower drive will be the fastest time you can use on your system.

TABLE C

SET△TT:△OPTION

OPTION	DESCRIPTION
HELP	Prints the help message.
* NOBKS	Uses the back-slash character (\) for errors.
BKS	Allows backspacing to correct typing errors.
* BKM	Causes backspace (CTRL/H) to be treated as a DELETE.
NOBKM	Lets HDOS receive the backspace character.
* MLI	Maps lower case input to upper case.
NOMLI	Allows lower case input into HDOS.
* MLO	Maps lower case output to upper case.
NOMLO	Allows lower case output from HDOS.
* NOTAB	HDOS expands TAB (CTRL/I).
TAB	Lets the terminal process TABS. (Faster)
* 2SB	Uses two stop bits. (Universal)
1SB	Uses one stop bit. (Normal)
WIDTH△n	Sets the console width to n characters, 80 is the default value. HDOS starts a new line if more than n characters are sent. n must be between 20 and 255.
FILL△c△n	Sets "c" as a character that needs fill. This is needed on some slow hard-copy terminals. 15Q<CR> and 4 are the default values.

With most terminals, you should change the following SET options.

```
SET TT: 1SB
SET TT: FILL 15Q 0
```

* — Denotes the option that has been set on the distribution diskette.

TABLE D

SET△LP:△OPTION

OPTION	DESCRIPTION
HELP	Prints the help message.
* 6LPI	Sets the H14 for 6 lines per inch.
8LPI	Sets the H14 for 8 lines per inch.
PAGE△n	Sets the number of lines per page to "n". If "n" is zero, lines are printed continuously. Default value is 60.
PORT△n	Sets the port address for LP: to "n". Default value for "n" is 340Q.
WIDTH△m,n	Sets the width control switch position. "n" is the "narrow" position, and "m" is "wide". The only legal values are 80, 96, and 132. Default is 80,132.
BAUD△n	Sets the baud rate for LP:. You should only set standard rates (4800, 2400, 1200, ...) Default is 4800.

TABLE E

SET△AT:△OPTION

OPTION	DESCRIPTION
HELP	Prints the help message.
* 1SB	Sets for one stop bit. Note default is one.
2SB	Sets for two stop bits.
* MLC	Sets to map lower case to upper for both input and output.
NOMLC	Allows lower case input and output.
WIDTH△n	Sets the terminal width to "n" characters. "n" must be greater than 20. Default is 80.
PAD△n	Sets to send "n" "pad" characters after a ⓒ>. Needed with some terminals. Default is zero.
PORT△n	Sets the port address for AT: to "n". If AT: is renamed from ATH84, then the default PORT is 300Q. If AT: is renamed from ATH85, the default is 374Q. See "Device Drivers." An H89 will use ATH84.DVD, renamed as AT.DVD. However, the H88/H89 hardware port for AT: is 320Q. Therefore, do a SET△AT:△PORT△320Q command.
BAUD△n	Sets the AT: baud rate to "n". Default is 300. You must use standard rates (4800, 2400, 1200, ...).

* — Denotes the option that has been set on the distribution diskette.

Device Drivers

To facilitate expansion and maintenance, HDOS was designed, in a modular fashion, as a number of sub-programs that communicate with one another. Each of these sub-programs is responsible for a logically distinct task. For example, the sub-program which processes commands is separate from that which handles I/O. When the command processor needs input, it asks the I/O handler for data. Likewise, when the command processor generates output, it passes the data along to the I/O handler. In comparing HDOS to a person, one might liken the command processor to the brain, the sense organs to the external peripherals, and the device drivers, or device handlers, to the nerves which translate and transmit the data from the sense organs to the brain.

HDOS communicates only with device drivers. Any device that is to be interfaced to HDOS must then be interfaced by such a driver. When HDOS writes to device xx:, HDOS merely supplies the bytes to be written, calls the driver, and then relies on the driver to convert the data into the format required by the specific device. Thus, HDOS itself does not have to understand each possible device type, only the device driver "knows how to talk to" that device.

For this reason, HDOS I/O is quite flexible. Heath will support new device configurations by writing the appropriate device driver and supplying it on the distribution diskette. To manage the various devices, HDOS maintains a table which supports up to seven device drivers. Of these seven devices, two are permanently reserved for SY: and TT:, the disk and system console, respectively. You would normally select and use two other HDOS device drivers, AT: and LP:, the alternate terminal and line printer. This leaves room for three additional device drivers that will accommodate your specific system requirements.

The system distribution diskette is supplied with a number of general-purpose device drivers designed to accommodate Heath Products, and general peripherals:

ATH84.DVD	An alternate terminal configured at address 300Q via an H8-4 Card.

(Can be reconfigured for address 320Q via an H88-4 Card.)

ATH85.DVD	An alternate terminal configured at address 374Q via an H8-5 Card.
LPHRD.DVD	An H14 Line Printer interfaced via a Multiport Serial I/O Card at address 340Q.
ND.DVD	A null device driver (used primarily for test purposes).

HDOS builds a table of device drivers when the system is booted up. HDOS scans the disk directory for entries of the form: xx.DVD. Therefore, device driver program names must be two characters long, and the file name extension must be "DVD". The two-character file name is then used to define the device name. (Note: no SY.DVD or TT.DVD files are present because, as these drivers are permanently resident, they are incorporated into HDOS and HDOSOVLn files.) Once HDOS has entered the devices in the device table during boot-up, they remain usable as long as the current system disk is mounted on SY0:, and the device driver files are present.

To incorporate any of the optional Heath-supplied device drivers:

1. Decide which device drivers you need to use, remembering that the maximum number of optional user-selectable device drivers is currently five. For instance; assume that an H14 interfaced with a Multiport Serial I/O Card is to be included in your system. In this case, you would have to select LPHRD.DVD as your device driver.

2. Configure your hardward to match the specifications of the selected device driver. (See "Systems Configuration" in your Manual.)

3. Copy the device drivers with the appropriate two-letter file name and the "DVD" extension. For example: To use LPHRD:

```
>COPY LP.DVD=LPHRD.DVD
```

   Note: SYSGEN will copy all device driver files onto the new diskette. That is, if you SYSGENed from your distribution diskette or a diskette that had LPHRD.DVD, then the new diskette will have LPHRD.DVD on it.

4. Enter a "BYE" command, and then re-BOOT HDOS. From this point on, whenever the configured system disk is booted, HDOS will recognize the valid device drivers.

NOTE: The HDOS device table is only built on boot-up. That is why you have to re-BOOT before you can use the copied device drivers.

By way of example, an illustration implementing LP: is given below:

```
ACTION? <BOOT>BOOT

SYSTEM HAS 32K OF RAM
```

```
VOLUME 100, MOUNTED ON SY0:
LABEL: SYSTEM VOLUME

HDOS VERSION 1.0
 ISSUE   50.01.00
DATE (17-OCT-78)? <CR>
>COPY LP.DVD=LPHRD.DVD <CR>
>BYE <CR>
VOLUME 100, DISMOUNTED FROM SY0:
LABEL: SYSTEM VOLUME

INSTALL A BOOTABLE DISK IN SY0: AND HIT RETURN TO REBOOT:

ACTION? <BOOT>
```

It has been assumed that LPHRD.DVD has been copied to this disk from the distribution disk.

From this point on, each time that this disk is booted, HDOS will know about device LP:.

From now on, the new device will appear as part of the operating system. In general, you may now use device LP: throughout HDOS. For example, to list a file on device LP: from PIP, type:

```
>PIP <CR>
:P:LP:=HDOS.ACM <CR>

1 Files Copied
:P:
```

Similarly, BASIC programs may now access device LP:. For example,

```
10 OPEN 'LP:' FOR WRITE AS  FILE #1
20 FOR  I = 1  TO  10
30    PRINT  #1,I
40    NEXT  I
50 CLOSE  #1
60 END
```

This program will simply print the numbers 1,2,..., 10 on device LP:.

In summary, device driver I/O lends a significant degree of flexibility to HDOS. As the system expands and new peripherals are added, new device drivers will be available.

NOTE: When you finish copying the device driver files that you need onto your working diskette, you can delete any other .DVD files. If you get a new device later, you may copy the appropriate device driver file from your distribution diskette.

System Status

You may wish to check the performance of your system now and then. To do so, type:

```
>STAT ↵
```

HDOS will identify itself by printing the version number and the date on the system console. It will then print the number of reads and writes performed since the system was last booted up.

The more important consideration, from the standpoint of system performance, is the number of read/write errors. HDOS first prints the number of "hard" errors. If any hard errors have occurred, the chances are good that you have either a diskette with a bad sector or something is wrong with your drive(s) or controller board. The recoverable, or "soft" errors are less serious. They can result from many causes, as explained in "Appendix A." Among these causes are dust on the diskette, electrical interference, and momentary losses of synchronization between the electrical and mechanical portions of the system. If HDOS experiences an error ten times in succession on the same disk operation, the error is classified as a hard error. If the number of recoverable errors is larger than about 25 per hour of operation, the most probable cure is to slow the track seek time by two milliseconds, using the SET△SY:△STEP△nn option. Thus, if your step time was set at 20 milliseconds, change it to 22. Other possible solutions include reductions in room temperature and airborne dust.

Your system should now be configured for your console terminal, and the drive seek time should be optimized for your drive(s). The examples you have used throughout this procedure are only a small sampling of the possible commands and functions. By varying the file names, switches, wildcards, modes, and commands, you can explore a nearly endless range of possibilities.

Some of the system files are not absolutely necessary for everyday operations. For instance, after you have optimized your system using the SET options, you may delete the file SYØ:SET.ABS from your system volume. First clear the SW flags. Then use the DELETE command, or the /DEL switch to remove the file. Of course you will no longer be able to use SYSGEN with this volume because SYSGEN expects the entire system to be intact; so you should delete

SYØ:SYSGEN.ABS as well. It is not necessary that you delete these files and it may even be inconvenient in some applications. However, the removal of nonessential files will provide more free space for program or data storage. Other files that you may delete, are:

```
FLAGS.ABS {After clearing the SW flags on the others.}
SYSGEN.ABS
SYSHELP.DOC
HELP.
ONECOPY.ABS
ATH85.DVD
ND.DVD
ATH84.DVD
LPHRD.DVD
```

Of course the functions provided by the programs residing on these files will no longer be available on that particular diskette. We recommend that you keep at least one master system volume, in addition to the distribution diskette, which provides all the available functions.

Summary

Your system is now up and running. Your goal should be to familiarize yourself with the system so that you will be able to use it efficiently. You can do this by going back through "General Operation," varying the examples as necessary to get "hands-on" experience. This will be very instructive, and can cause no damage, due to the error-detection and write-protection facilities of HDOS. If you should delete or damage any files, you can always re-SYSGEN from the distribution disk.

Backups

Remember that, while a powerful system like HDOS keeps your files conveniently at your fingertips, accidental destruction of your files is only a slip of the finger away. For this reason, you should always keep copies, called "backup copies," of all your important files. Heath has already provided such protection for the HDOS files in the form of the write-protected distribution disk and individually write-protected files. If you should somehow manage to damage an HDOS file from your SYSGENed volumes, you can restore it from the distribution disk.

APPENDIX A

Glossary

NOTE: ALL commands are listed in full capital letters.

Access: The act of finding a storage location in memory or on a mass-storage medium in order to read data from it or write data into it. See "direct-access," "random-access," and "sequential-access."

Allocation: The act of setting aside a certain amount of memory or an area of a mass-storage device, to be used for running programs or storing data.

Block: See "cluster."

Bootstrap: The method by which communication is established between hardware and software. In order for a computer to "run," it must contain a program. In order to load programs into a computer, it must be running. In other words, the computer system must "lift itself by its bootstraps" before it can operate. Early computer systems werearted, or "booted up", by manually switching in a series of binary instructions from the front panel. Nowadays, most computers have bootstrap programs already loaded into read-only memories, or ROMs. The bootstrap program allows them to start whenever power is turned on. Bootstrap, or "boot" can also mean a transfer from the basic start-up program to a more sophisticated program, such as an operating system.

Buffer: An area of user or system RAM set aside for communications with peripherals, including the disk. A buffer typically could consist of 256 bytes of memory, which is the same size as a diskette sector. When it is accessing a file, the operating system would read a sector into the buffer so a program could gain access to the data. Buffers vary in size, depending on the peripheral with which they're associated, considerations of efficiency, and the amount of available memory. A buffer for a terminal might consist of only one byte; while a disk buffer should be at least as large as the minimum unit of storage on the disk, 256 bytes for the diskette and HDOS.

CATALOG: A command that instructs the operating system to output a summary of useful information about a set of files. Examples are name, size, and date created.

CLOSE: A command that indicates to the operating system that a process no longer requires access to the data in a file. If the file was changed during the execution of the program, the disk storage area utilized for the file may be updated. The directory will also be updated to reflect the changes to the file, such as its size and location on the disk media.

Cluster: A contiguous portion of storage area on a disk medium. In the case of HDOS, the minimum cluster size is two sectors of 256 bytes each.

Command: Information communicated to the operating system by a process or from the console, instructing the operating system to perform some action, such as opening a file.

Console: Another word for the peripheral from which a computer system is controlled. An "operator" or "user" communicates with the operating system by means of a console.

Contiguous: Describes bjects or storage areas that are located next to each other. Similar to "continuous."

Copy: The act of placing data contained on one file into another file. The contents of the two files are then identical; however, their names and physical locations are different.

Create: The act of setting-up a new file. This involves giving it a name for future reference. The operating system will find space for the file on the disk if sufficient space is available. It will also update the directory to indicate the presence of the new file, unique among the files on this particular disk.

Default: A condition that exists when no action is taken to override it. For instance, a device driver may print lines which are 80 characters in length unless it is instructed to make them shorter or longer. The default line length would then be 80 characters.

DELETE: A command that instructs the operating system to remove a file from the directory, and to free the area on the disk that it occupies for other purposes.

Destination: A file into which data is to be written.

Device: A peripheral into which data is to be written, or from which data is to be read, by means of input/output commands or instructions.

Device Driver: An operating system program that controls a peripheral. See "device independence."

Device Independence: A concept that allows a user program to refer to a peripheral by a symbolic name, like a file, instead of requiring a section of the program to be written specifically for the purpose of controlling the peripheral. The process commands the operating system to input data from the named device or output data to it. The operating system, in turn, uses a device driver that is associated with the device name to accomplish the I/O.

Diagnostic: A program that is used to troubleshoot computer systems, or the various components of a computer system. The most common "diagnostics" are programs that are used to find possible read/write errors in memory devices.

Direct-Access: A concept used with some disk systems to describe the ability to access a given block of data by using the directory to find its physical position on the disk. This eliminates the need to read all the data that precedes the desired block as a means of finding it. The term "random access" is sometimes used to describe this capability.

Directory: A data area used by the operating system that holds the location and size of each disk file, referenced by its name. In some ways, it is similar to a city telephone directory, but with files instead of people.

Driver: See "device driver."

Executive: See "monitor".

Extension: The portion of a file name that distinguishes it from another file of the same general category. For instance, an assembly language program that is used to compute poker odds could be called "POKER.ASM", while the assembled machine-instructions for the program could be stored on a file called "POKER.ABS". The extension is the portion of the name that is located to the right of the period; it may consist of zero to three characters under HDOS.

File: A data structure that is generally associated with a disk or other direct-access device. The disk is analogous to an office filing cabinet, with the files corresponding to the folders of information on the magnetic recording medium of the disk. Data is read from files and written onto files by means of operating system commands which reference each file by a unique name. The system handles the problems of finding the data and making it available to a process. Files must be "open" to be accessed, and must be "closed" when no longer needed.

Free: The act of making an area of memory available for other putposes; as when a file is closed, its buffer is "freed".

Handler: See "device driver."

Hard Error: A disk read/write error caused by a malfunction in the electronic or electromechanical hardware. A hard error is usually the result of an error in writing caused by dust, static electricity, a scratched disk, or by various kinds of electronic interference or noise from electric motors, radio transmitters, and so on.

Initialize: A command to the operating system that instructs it to prepare a floppy-disk for new data storage. A new floppy-disk must be initialized before you can use it. If the floppy-disk already contains data, the data will be destroyed if that volume is initialized.

Interrupt: A hardware signal to the computer, used extensively by operating systems, that causes the current process to cease and another to take its place. This facility speeds up the operation and handling of peripherals. The interrupt routine is similar to a subroutine in that it eventually returns control to the original process. The difference is that an interrupt may occur at almost any time and is controlled by external events, such as a keystroke at a terminal.

Library: A collection of programs that may be stored on the same disk and used in conjunction with each other. For example, an operating system can be a library of separate programs that are capable of calling one another.

Load: The process of transferring data from a peripheral into RAM.

Loader: A program that transfers data from a peripheral into RAM.

Map: A picture of how data and programs are distributed in memory, or a table which shows where files are located on a mass storage device.

Medium: Generally a magnetic substance, such as a floppy-disk surface, upon which data can be permanently recorded. Media can usually be removed and replaced by other physically similar media.

Monitor: The most basic portion of an operating system. A monitor resides in memory and may call other programs to accomplish the various functions of an operating system. For instance, a complete operating system may consist of 28K of machine-executable instructions, with the monitor comprising only 4K of the total. When commands are issued to the operating system that are beyond the capabilities of the monitor, it may bring in another program from a disk file to carry out the command. When the subprogram has completed its task, it returns control to the monitor, which calls other subprograms as required.

OPEN: A command to the operating system that makes the contents of a specific file available to a process.

Operating System: A rather complicated set of programs that is generally associated with disks and other mass storage devices. Its function is analogous to that of a policeman directing traffic at a busy intersection. Specifically, it may keep track of large amounts of data on disk files, control peripherals, control the distribution of memory among various programs, regulate the execution of programs, keep track of the amount of time and memory used for various purposes, and even improve its own speed and efficiency. The degree of sophistication is generally directly related to the size and cost of the computer system.

Overhead: That portion of the computer system's time and memory required to implement the functions of the system. Generally, the greater the overhead, the greater the versatility and functionality of the system (and the higher the cost).

Overlay: A fixed-size area of memory that is shared, in turn, by more than one process. For example, HDOS may require extra memory for the purpose of opening and closing files; the available memory may also be required by a user program. When this occurs, HDOS will save the contents of that memory area in a file, "swap-in" the appropriate subprogram, open or close the appropriate file(s), and then "swap-in" the original contents of the area. The process is called "overlaying."

Primary Memory: The high-speed RAM in which programs are executed and data is stored for immediate availability.

Protection: The means by which a process is prevented from destroying or overwriting an important area of memory or disk space.

RAM: An acronym for "Random Access Memory": A RAM allows a given location to be read from or written to in the same amount of time as any other equivalent location, regardless of physical position. This term is sometimes used interchangeably with "direct access".

Read: The act of examining the contents of a memory location, or the process of transferring the contents of a file into a buffer or area of RAM.

Real-Time Clock: An electronic counter that interrupts the processor at given time intervals to allow a process to keep track of the passage of time by counting the "interrupts." The computer has a real-time clock which initiates interrupts at intervals of two milliseconds.

RENAME: A command that changes the name of a file without affecting its contents or physical location.

Resource: A valuable portion of a computer system, such as a peripheral, a memory, or a program. Resources can be shared by several processes in advanced system; in any case, they are reusable and relatively permanent.

ROM: An acronym for "Read-Only Memory." A ROM is a memory whose contents cannot be changed; however, it can be read like any other memory.

Secondary Memory: Generally, a large-volume, low-cost, and relatively slow memory device. It can be a peripheral such as a diskette. In the case of large computer systems, it could be a cheaper version of the primary memory, where programs and data are stored when they are not being used frequently.

Sector: The minimum accessible unit of storage on a disk. The size may be determined by physical or logical parameters, or both. In the case of the diskette and HDOS, the sector size is 256 bytes; while the minimum file size is one cluster, or two sectors.

Seek: The action taken by the head of a disk drive in finding the correct track when data is read from a file or written into a file. "Seek time" partially determines the speed of a disk access. The other main factor, called "rotational latency", is the time required for the desired sector to rotate under the head.

Sequential-Access: A type of I/O in which a unit of storage can be made available for reading or writing only by reading every unit of storage which precedes it on the recording medium. This may result from the physical characteristics of the storage device, such as a magnetic tape, or it may be a limitation imposed by the operating system.

Soft Error: An error in reading a disk that may be caused by dust, noise, or an interrupt at the incorrect time. It is similar to the "hard error" except that a soft error may be corrected by an attempt to repeat the failed process. If several retries do not correct the problem, the error is reclassified as a hard error.

Source: The original file in the case of operating system commands, or the one which is to be renamed or copied. In the case of programs, it means the highest level code that is converted by the compiler, interpreter, or assembler into machine-executable instructions, or "object code".

Supervisor: See "monitor."

Swap: The act of removing the contents of a memory area temporarily while the memory is used for other purposes. See "overlay."

Switch: A symbolic code that is used to issue a command to the operating system. Also a variable that is interpreted by a process in order to influence its flow-of-control.

Syntax: The formal or "rigid" order in which commands or instructions must be written to enable an operating system or other software process to recognize them.

Target: See "destination."

Track: A circular area on a disk that consists of a given number of sectors. HDOS allows 40 tracks on each disk, with 10 sectors on each track.

Utility: A general-purpose program that may be shared by a variety of processes. An example would be a program to sort a list of names into alphabetical order. The use of a utility program would save considerable inconvenience and wasted space, which would be required if each program that needed such a capability had to be written to accomplish it independently. See "library."

Volume: An interchangeable storage unit, such as a cassette tape or a floppy-disk. The volume contains data and is placed in a "drive" so the data may be "accessed."

Write: The act of transferring data into a memory location or register, or outputting it to a peripheral, including a disk. The head on a disk writes binary information onto the magnetic medium, which is the physical location of a given file.

APPENDIX B

System Error Messages

This section describes the error messages generated by the HDOS operating system. Error message falls into two general categories: those which start with ?nn,where nn = two digits, and those which don't. Error messages with no ?nn are produced by the program you are currently running. For example, if you are using the Text Editor, EDIT, and get a message with no ?nn in it, look in the Text Editor Manual for an explanation. Messages with the ?nn in them are produced by some component of the HDOS operating system, and are discussed here. The messages are grouped according to their ?nn number.

?00 — Bootstrap Errors

Error messages which start with ?00 are generated by the system while it is being booted up.

```
?00 DISK READ ERROR DURING BOOT
```

An unrecoverable (hard) disk error occurred during the bootstrap process. Try booting again. If the problem persists, either your drive or your volume is bad. Try booting a different system disk.

```
?00 *  ERROR *
```

An unrecoverable (hard) disk error occured while checksuming the disk. The sector number printed immediately after this message is the one containing the error.

```
?00 REQUIRED FILE HDOS.SYS MISSING
```

The file HDOS.SYS is not on the volume in SY0:. The disk has not been SYSGENed, or has been SYSGENed incorrectly. Reinitialize it and then SYSGEN it correctly.

```
?00 THIS DISK HAS NOT BEEN PROPERLY SYSGENED
```

Some error in the format of the HDOS system files was detected. The disk cannot be booted. The disk must be reinitialized, and then SYSGENed.

```
?00 THIS DISK MUST BE INITIALIZED AND THEN SYSGENED BEFORE IT  CAN BE USED
```

This disk must be initialized before you can use it. This message normally appears when you try to boot up a disk that has been destroyed by TEST.

```
?00 THIS DISK MUST BE SYSGENED BEFORE IT CAN BE BOOTED.
```

This disk has not been SYSGENed, and thus can not be booted as a system disk. Use SYSGEN to make it a system disk.

?01 — Build Phase Errors

These error messages appear during the second half of the boot process when the HDOS operating system is being built into memory from the system disk. Most of these messages indicate damage to the files on the disk. First, try rebooting the system. If the problem persists, then this disk cannot be booted as a system disk. If you own two drives, mount the disk in SY1: and copy the files you want to keep onto a different disk. If you own only one drive, use ONECOPY (run by booting up on some other disk) to copy off your important files. Then, reinitialize the disk, and reSYSGEN it.

```
?01 DISK I/O ERROR DURING BOOT
```

An unrecoverable (hard) disk error occurred on the system disk, and the boot operation cannot proceed. The disk volume may be bad, or you may have a bad drive. Retry the boot.

```
?01 DISK STRUCTURE IS CORRUPT
```

The directly and/or the free space table on this disk are damaged, and HDOS cannot restore the damaged files. Contact the Heath Technical Correspondents for advice.

```
?01 FORMAT ERROR IN DRIVER FILE
```

The file does not contain a valid device driver program. Currently only Heath-written device drivers are supported. You should not attempt to write your own.

```
?01 HDOS  REQUIRES AT LEAST 12K!
```

Your H8 system does not contain enough RAM to run HDOS, or the RAM is faulty, or it is not addressed correctly. Use a memory diagnostic to make sure that the RAM is working properly, and is jumpered to the correct addresses.

```
?01 MISSING FILE SY0:HDOSOVL.SYS
```

The file SYØ;hdosovlsys is necessary to run HDOS, and is not present. This normally indicats an incorrect SYSGEN.

```
?01 SYSTEM NOT SYSGENED PROPERLY,  OR FILES DAMAGED
```

A system file is damaged. This can be the result of a software or hardware error.

?02 — Error Messages

These messages are generated by the operating system and may appear at any time. Usually they are in response to some request from the program you are running which, in turn, is usually caused by some command from you. Normally, HDOS looks up these error messages in the file SYØ:ERRORMSG.SYS to give an understandable message. If the file SYØ:ERRORMSG.SYS is missing, or if the system disk has been dismounted, HDOS will simply type the error message number. The numbers are listed first, followed by the message they represent. Look up the message in the second group for a discussion of its meaning.

Most of these error messages will be meaningless to you; they are generated by HDOS when a program makes a mistake when issuing a request to HDOS. The Heath products supplied with HDOS will not make these mistakes. Normally, only users debugging assembly language programs will see most of these error messages. The ones that the average user will see are self explanatory.

```
?02 SYS ERR # 001
```

End of file.

?02 SYS ERR # 002

No free space on media.

?02 SYS ERR # 003

Illegal "SYSCALL" function code.

?02 SYS ERR # 004

Channel is already in use.

?02 SYS ERR # 005

Device is not capable of this operation.

?02 SYS ERR # 006

Illegal device name.

?02 SYS ERR # 007

Illegal format for file name.

?02 SYS ERR # 008

Not enough memory for the device driver.

?02 SYS ERR # 009

Channel is not open.

?02 SYS ERR # 010

Illegal function request.

?02 SYS ERR # 011

File usage conflicts.

?02 SYS ERR # 012

File cannot be located.

?02 SYS ERR # 013

Unknown device name.

?02 SYS ERR # 014

Illegal channel number.

?02 SYS ERR # 015

The device directory is full.

?02 SYS ERR # 016

The file's contents are not correct for this operation.

?02 SYS ERR # 017

Not enough RAM for this program.

?02 SYS ERR # 018

Read failure on the device.

?02 SYS ERR # 019

Write failure on the device.

?02 SYS ERR # 020

Attempted write-protection violation.

?02 SYS ERR # 021

Disk is write-protected.

?02 SYS ERR # 022

The file is already present.

?02 SYS ERR # 023

Aborted by Device Driver.

?02 SYS ERR # 024

File is locked against flag change.

?02 SYS ERR # 025

A file is already open.

```
?02 SYS ERR # 026
```

Illegal or unknown switch specified.

```
?02 SYS ERR # 027
```

Unknown unit for this device.

```
?02 SYS ERR # 028
```

Non-null file name is required.

```
?02 SYS ERR # 029
```

Device is incapable of write operations (or is write locked).

```
?02 SYS ERR # 030
```

Unit not available.

```
?02 SYS ERR # 031
```

Illegal value.

```
?02 SYS ERR # 032
```

Illegal option.

```
?02 SYS ERR # 033
```

Volume presently mounted on the device.

```
?02 SYS ERR # 034
```

No volume presently mounted on the device.

```
?02 SYS ERR # 035
```

File open on the device.

```
?02 SYS ERR # 036
```

No provisions made for remounting more disks.

```
?02 SYS ERR # 037
```

This disk must be initialized before it can be mounted.

```
?02 SYS ERR # 038
```

Unable to read this disk, it probably has not been properly initialized.

```
?02 SYS ERR # 039
```

Disk structure is corrupt. Contact Technical Correspondence for help.

```
?02 SYS ERR # 040
```

Not the correct version of HDOS for this program.

```
?02 SYS ERR # 041
```

No operating system mounted. Required for this operation.

```
?02 SYS ERR # 042
```

Illegal overlay index.

```
?02 SYS ERR # 043
```

Overlay too large.

```
?02 A FILE IS ALREADY OPEN
```

The specified channel is already open.

```
?02 ATTEMPTED WRITE PROTECTION VIOLATION
```

You requested a write-type operation on a write-protected file. These write-type operations are WRITE, DELETE, RENAME, and REPLACE.

```
?02 CHANNEL IS ALREADY IN USE
```

The I/O channel specified in the HDOS call is already in use.

```
?02 CHANNEL IS NOT OPEN
```

The I/O channel must be opened before you issue this request.

```
?02 DEVICE IS INCAPABLE OF WRITE OPERATION (OR IS WRITE LOCKED)
```

The device is write-disabled, or is incapable of accepting write operations.

```
?02 DEVICE IS NOT CAPABLE OF THIS OPERATION
```

The device specified is not capable of the operation. For example: a read request from a write-only device such as a line printer, or a directory operation such as LIST on a non disk device.

The most common cause of this message is a write operation on a disk that has the write-protect tab installed.

```
?02 DISK IS WRITE PROTECTED
```

The write operation was refused because the disk is write protected.

```
?02 DISK STRUCTURE IS CORRUPT. CONTACT TECHNICAL CORRESPONDENCE FOR HELP.
```

The internal map of the disk has changed in HDOS. The most common cause is incorrectly dismounting or mounting diskettes, especially when they have the same volume number.

```
?02 END OF FILE
```

An End-of-File was read on the file. There are no more sectors to read.

```
?02 FATAL SYSTEM ERROR
```

A read or write error occurred on a distribution or a system diskette. The distribution diskette must always be inserted in SYØ:.

```
?02 FILE CANNOT BE LOCATED
```

The specified file name is not on the specified device.

```
?02 FILE IS LOCKED AGAINST FLAG CHANGE
```

The file cannot have its flags changed because the L (locked) flag is set.

```
?02 FILE OPEN ON THE DEVICE
```

A dismount or reset was issued to a device, and a channel was not previously closed.

```
?02 FILE  USING CONFLICTS
```

Conflicting requests have been made for this file. A file may not be deleted, replaced, written to, or renamed while it is open for read. Also note that a program may not delete, replace, write to, or rename the file it was loaded from.

```
?02 ILLEGAL 'SYSCALL' FUNCTION CODE
```

An illegal request code was given in an assembly language SCALL statement. This will only occur with user-written programs.

```
?02 ILLEGAL CHANNEL NUMBER
```

A request specified a nonexistent channel number.

```
?02 ILLEGAL  FORMAT FOR DEVICE NAME
```

The device specification part of the file name is not correctly specified.

```
?02 ILLEGAL FORMAT FOR FILE NAME
```

The file specification was in an illegal format.

```
?02  ILLEGAL FUNCTION REQUEST
```

A request was made to HDOS to perform an illegal function. For example: requesting a write to a channel opened for read.

```
?02 ILLEGAL OR UNKNOWN SWITCH SPECIFIED
```

An illegal or unknown option switch (/xxx) was specified in the command line.

```
?02 ILLEGAL OPTION
```

A specified SET option was not recognized by HDOS.

```
?02 ILLEGAL OVERLAY INDEX
```

A call was made to an invalid index number.

```
?02 ILLEGAL VALUE
```

The value entered was out of bounds. Most commonly, this error occurs in a SET command.

?02 NO FREE SPACE ON MEDIA

All sectors on the volume are in use, so the write request cannot be honored.

?02 NO OPERATING SYSTEM MOUNTED, REQUIRED FOR THIS OPERATION

HDOS and the required overlays are not present. Check the directory of SYØ:.

?02 NO VOLUME PRESENTLY MOUNTED ON THE DEVICE

A dismount was issued to a device that had nothing previously mounted.

?02 NO PROVISIONS MADE FOR REMOUNTING MORE DISKS

The overlays of HDOS required to process a mount command are not present.

?02 NON-NULL FILE NAME IS REQUIRED

Diskfiles require that the name field in the file specification must contain at least one character. The extension may be empty (null). Non-disk devices allow empty (null) file names.

?02 NOT THE CORRECT VERSION OF HDOS FOR THIS PROGRAM

The version numbers of HDOS and the requested program do not agree. You should make sure that all files have the most recent version numbers.

?02 NOT ENOUGH MEMORY FOR THE DEVICE DRIVER

Not enough free RAM exists to load the necessary device driver.

?02 NOT ENOUGH RAM FOR THIS PROGRAM

There is not enough free RAM to load this program.

?02 OVERLAY TOO LARGE

The assemble overlay exceeds maximum size. It cannot be larger than HDOSOVLØ.SYS:

?02 READ FAILURE ON THE DEVICE

An unrecoverable (hard) error occurred on the last attempted read operation from this device.

?02 THE FILE IS ALREADY PRESENT

You attempted to rename a file to a new name that already exists in that volume's directory.

?02 THE FILE'S CONTENTS ARE NOT CORRECT FOR THIS OPERATION

An attempt to RUN a file which is not an absolute binary program. Only absolute binary (assembly language) files may be run (the programs supplied by Heath are all in this format). Note that the extension .ABS, by convention, represents absolute binary files. But an absolute binary file does not have to have the .ABS extension, and a file may have the .ABS extension and yet not be in absolute binary format.

?02 THE VOLUME DIRECTORY IS FULL

The volume's directory is full; no more file names can be added until some are deleted. A volume directory for the diskette holds about 198 file names.

?02 UNABLE TO READ THIS DISK, IT PROBABLY HAS NOT BEEN PROPERLY INITIALIZED

An attempt was made to mount a diskette that appears to be uninitialized. Be sure that the diskette was re-initialized after "Test".

?02 UNIT NOT AVAILABLE

The device unit requested is not installed or not operable.

?02 UNKNOWN DEVICE NAME

An unknown device was specified in the file name. Note that a device driver for each device you wish to use must be on the system disk when it is booted (except for SY: and TT:, which are built into HDOS).

?02 UNKNOWN UNIT FOR THIS DEVICE

This device type does not have the specified unit number.

```
?02 WRITE FAILURE ON THE DEVICE
```

An unrecoverable (hard) error occurred on the last attempted write operation on this device.

```
?02 VOLUME PRESENTLY MOUNTED ON THE DEVICE
```

A mount was issued to a device that has not been dismounted.

APPENDIX C

HDOS Cookbook

This Appendix provides you with a number of examples of possible HDOS functions and commands. The command mode, PIP, ONECOPY, and SET will be covered. The exact command syntax, including the prompts, will be printed, along with an explanation of the results of the command.

These examples are not intended as explanations of how HDOS accomplishes various functions. Rather, they are intended as a survey of some of the more useful commands and options. For a more detailed explanation of HDOS, refer to the "General Operation" section. In these examples, DEV: refers to a device, FNAME refers to the general name of a file, and .EXT refers to the extension which distinguishes the file from others of the same general name.

Listing Files

The following commands allow you to type the contents of various files on your system console. It is necessary to specify SY1: if you wish to type files from that device. You may type files on SYØ: without an actual specification of SY0:. Wildcards and multiple file designations are allowed and cause the files to be typed one after the other. Note that you may use CTRL-C to stop the listing, CTRL-S to halt it temporarily, and CTRL-Q to get it restarted. It is not recommended that you type files with the .ABS, .DVD, or .SYS extensions because of their generally non-ASCII contents. If you wish to type the contents of a system file, you must specify the /S modifier.

```
>TYPE␣FNAME.EXT
>TYPE␣*.EXT
>TYPE␣FNAME.*
>TYPE␣*.*
>TYPE␣SY0:FNAME.EXT
>TYPE␣FNAME.EXT/S
>TYPE␣SY0:FNAME.EXT/S
>TYPE␣SY1:FNAME.EXT
>TYPE␣SY1:FNAME.EXT/S
>TYPE␣FNAME.EXT, . . . ,FNAME.EXT
>TYPE␣SY1:*.EXT, . . . ,SY0:FNAME.*/S
```

Cataloging Files

The following commands produce a catalog listing of a specific file or files on either SYØ: or SY1:, depending upon which is specified. HDOS defaults to SYØ: if neither is specified. You must use the /S modifier if a system file is to be catalogued. You may catalog other files with or without the /S modifier. If the * wildcard is used in FNAME or .EXT, multiple files may be cataloged if two or more have either part in common. Note that there are a great many possible combinations of commands that are not shown.

```
>CAT
>CAT△SYØ:
>CAT△SYØ:*.*
>CAT/S
>CAT△SYØ:/S
>CAT△SYØ:*.*/S
>CAT△SY1:
>CAT△SY1:*.*
>CAT△SY1:/S
>CAT△SY1:*.*/S
>CAT△FNAME.EXT
>CAT△SYØ:FNAME.EXT
>CAT△SYØ:FNAME.*
>CAT△SYØ:*.EXT
>CAT△FNAME.EXT/S
>CAT△SYØ:FNAME.EXT/S
>CAT△SY1:FNAME.EXT
>CAT△SY1:FNAME.EXT/S
>CAT△FNAME.EXT, . . . ,FNAME.EXT
```

Mounting and Dismounting Diskettes

If you wish to use SYØ: or SY1:, it is necessary to "mount" them on the H17 drive units. The system volume is automatically mounted during the "Bootstrap" procedure. When you are finished using a volume, it is necessary to "dismount" it in order to power-down the system or mount a new volume. If you do not dismount the volumes, data which is held in memory may be lost because HDOS is given no change to update the volume(s). The correct procedures are listed as follows:

```
>DISMOUNT△SY1: ↵
VOLUME 002, DISMOUNTED FROM SY1:
LABEL: SPARE SYSTEM VOLUME
>DISMOUNT△SY0: ↵
VOLUME 001, DISMOUNTED FROM SY0:
LABEL: MAIN SYSTEM VOLUME
TYPE SPACES TO DETERMINE BAUD RATE
>MOUNT SY1: ↵
VOLUME 002, MOUNTED ON SY1:
LABEL: SPARE SYSTEM VOLUME
```

Running Programs

Executable binary programs have the .ABS extension under the HDOS conventions. This extension allows you to type the FNAME portion of a file name from the command mode as an abbreviated command to run the program contained in the file SYØ:FNAME.ABS. In order to run a program contained in a file on SY1:, you must specify SY1:. For example:

```
>FNAME
>RUN△FNAME
>RUN△FNAME.ABS
>RUN△SY0:FNAME.ABS
>RUN△SY1:FNAME.ABS
```

Duplicating Files

It is possible to copy the exact contents of one or more files with a single command. The general command syntax is:

```
>COPY△DEV:FNAME.EXT=DEV:FNAME.EXT ↵
```

Some examples of this command are:

```
>COPY△FNAME.EXT=FNAME.EXT
>COPY△FNAME.EXT=SY1:FNAME.EXT
>COPY△SY1:FNAME.EXT=AT:
>COPY△SY1:*.*=SY0:*.EXT
>COPY△AT:=*.*
```

Note that when you write more than one file specification on line, HDOS adapts a slightly different scheme for default devices and extensions. The default device for the first file is "SYØ", and the default extension is NULL. For the following file specifications, the default device and extension become the fields from the previous file specification. For example:

```
>COPY△TT:=SY1:TEMP.TXT.TEMP2 ↵
```

is equal to

```
>COPY△TT:=SY1:TEMP.TXT,SY1:TEMP2.TXT ↵
```

Deleting Files

It is possible to delete one or more files from your system using only one DELETE command. You must be careful with this command because valuable information could be lost if you delete the wrong file(s). System files are protected by flags and cannot be deleted until the flags are changed. Those files that are locked and write protected (see Page 1-48 to 1-50) can never be deleted unless the diskette is reinitialized. Here are a few examples of the DELETE command:

```
>DELETE△FNAME.EXT
>DELETE△SY0:FNAME.EXT
>DELETE△SY1:FNAME.EXT, . . . ,FNAMEn.EXT
>DELETE△*.EXT
>DELETE△FNAME.*
>DELETE△*.*
```

Peripheral Interchange

In general, you can use PIP to accomplish the same functions and commands that are supported by the command mode. This does not include running program, or such general-purpose functions as SET or DATE. However, PIP is the basis for all command mode file manipulations. The following list includes some useful PIP commands:

```
:P:/V
:P:/L
:P:/L/S
:P:/B
:P:/B/S
:P:DEV:FNAME.EXT/L
:P:DEV:FNAME.EXT
:P:FNAME.EXT
:P:FNAME.EXT, . . . ,FNAME.EXT/L/S
:P:DEV:FNAME.EXT=DEV:FNAME.EXT/S
:P:*.*/B/S
:P:FNAME.*/L
:P:SY1:*.*=*.*
:P:/DIS
:P:HELP
:P:/RES
:P:/MOU
:P:DEV:FNAME.EXT, . . . ,DEV:FNAME.EXT/DEL
```

(NOTE: You may not refer to more than one device in the same command.)

```
:P:DEV:=/SWITCH
:P:NEWNAME.EXT=OLDNAME.EXT/R
:P:LP:=/L
```

Onecopy

This program is a stand-alone utility that allows owners of single-drive systems to copy files from one diskette to another. To use this program, load information from the source diskette into the memory, and then swap-in the destination diskette. The data will be subsequently dumped from memory into a file on the destination. The destination file(s) automatically has (have) the same name as the source file(s). Multiple files can be copied in a single operation, depending upon the specification of the source(s). The number of swaps will depend upon the amount of installed memory. The following list demonstrates some of the possible commands under ONECOPY.

```
:OC:FNAME.EXT
:OC:FNAME.*
:OC:*.EXT
:OC:*.*
:OC:*.*/L/S
:OC:/B
:OC:/B/S
:OC:/MOU
```

System Optimization

There are several commands which allow you to optimize your disk drive system to function at top performance. You can adjust the seek time of the drive(s) to operate at the highest reliable speed. You can also configure the operating system to utilize a terminal that can backspace and support lower-case ASCII. Only valid HDOS device drivers may be SET. Use the following command to obtain HELP with the SET command.

```
>SET△HELP ↵
GENERAL COMMAND FORMAT
   SET△XX:△OPT
     XX: -- DEVICE NAME
     OPT -- DESIRED OPTION
FOR HELP WITH A SPECIFIC DEVICE.TYPE:
   SET XX: HELP
```

To determine the version of Set, type:

```
SET VER
```

APPENDIX D

Transferring Cassette Data to Files (BASCON)

The purpose of this Appendix is to demonstrate how you can transfer BASIC programs and TED-8 text from cassette tapes onto diskette files. The transfer capability applies only to Heath standard cassettes, and does not apply to binary data such as machine-language programs. Tape data files created with Extended BASIC 10.02.XX or later are not converted. This includes TYPE 5 cassette files and the data portions of TYPE 6 files. All program text is converted to diskette files. HDOS BASIC has a different method for handling data, and these tape data files would be unusable.

No provision is made for transferring in the opposite direction; you cannot transfer disk files onto cassette tapes. Also, after the BASIC programs have been transported, they may have to be modified slightly to use HDOS input/output commands. Refer to the appropriate Heath Software Manuals for more information.

Two program files are included with your distribution diskette to provide transportability. The program which transfers cassette BASIC programs to HDOS files is called BASCON, and resides on the file called BASCON.ABS. The TED-8 text transfer program is called TXTCON, and resides on TXTCON.ABS.

Before attempting to use these programs, you should perform the entire "System Set-Up Procedure." Transfer the data to a system volume that has been created with SYSGEN, or to a data volume that has been initialized and mounted on SY1:. DO NOT attempt to transfer data to the system distribution volume; HDOS does not support this option and you may accidentally destroy your system distribution volume.

Conversion of BASIC Programs (BASCON)

BASCON operates with a memory overhead of approximately 6K bytes. This means that you can expect a minimum system with 12K bytes of memory to convert BASIC programs which run in 6K or less. A 24K system will convert 18K BASIC programs. The amount of storage space remaining on a diskette also limits the size of programs that can be converted. The number of free sectors on a diskette is obtained by the CAT command or the /L PIP switch. Each free sector consists of 256 bytes. Use the following procedure to transfer cassette BASIC programs to a diskette:

1. Set-up your cassette deck just as you would to load tapes from BASIC. The tape should be installed and the PLAY button should be depressed.

2. Type BASCON and a carriage return from the HDOS command mode. The conversion program will identify itself and ask if you want instructions.

3. Type YES and a carriage return if you want instructions, or hit a carriage return if you do not.

   BASCON will ask for an output file name. This refers to the name of the file that will be stored on your diskette. You may reply with a file name in any of the three following formats:

   ```
   DEV:FNAME.EXT
   FNAME.EXT
   FNAME
   ```

   If you specify only the FNAME, BASCON will supply the extension .BAS to signify that the file is a BASIC program and it will copy the file onto SYØ:. You must specify SY1: as the device if you want the file copied there. You must also specify the extension if you want it to be other than .BAS.

4. Type the output file name and a carriage return.

   You will be asked for an input file name. This refers to the name of the BASIC program on your cassette tape. If you want a specific program transferred, you must supply its name, without quotation marks. If you want to transfer the next program on the tape, simply hit a carriage return.

5. Respond as explained above.

   The cassette deck will begin to run as BASCON searches for the desired BASIC program. You will be notified when it is found, as it is being read into the memory buffer and when it has been successfully converted to a disk file. You will also be notified if BASCON finds TED-8 text files or Extended BASIC TYPE 5 data files. These will be skipped and BASCON will continue searching for the desired BASIC program.

When the program has been converted to a diskette file, BASCON will ask for a new output file name. You may then restart from step 4 of this procedure and repeat the cycle with a new BASIC program, or you may type CTRL-D to escape to the HDOS command mode.

Conversion of TED-8 Cassette Text Files (TXTCON)

The TXTCON program operates in a minimum system with no apparent overhead. This means that you can convert any amount of text with the minimum 12K system, unless you run out of storage space on a diskette. Remember that each diskette sector consists of 256 bytes, and that the number of free sectors is printed at the end of a CAT command. Use the following procedure to convert TED-8 cassette tape records to diskette files:

1. Prepare your cassette deck just as you would to load cassette tapes from the TED-8 Text Editor.

2. Type TXTCON and a carriage return from the HDOS command mode. The TXTCON program will identify itself and ask if you want instructions.

3. Type YES and a carriage return if you want instructions, but type only a carriage return if you do not want instructions.

   TXTCON will print simple instructions if you asked for them; then it will ask if you want tabs inserted into the text. The ASCII tab code is used in exactly the same way as the TAB key on a standard typewriter. They allow several blanks to be printed each time a tab character is encountered. Tab codes are used to provide regular spacing between columns of text, as in an assembly language listing. They also save space because one tab character can replace as many as many eight blanks. If you answer YES, the TED-8 compressed text tab codes will

be transferred to the diskette as ASCII tab codes. If you answer NO, the compressed text tab codes will be converted to the appropriate number of ASCII space characters before being written onto the output file. HDOS allows you to handle ASCII tab codes by means of the SET options, thereby conserving space on your diskettes.

4. Type YES and a carriage return if you want tabs inserted. Otherwise hit a carriage return.

   You will be asked for an output file name. This refers to the name of the file that will be stored on the diskette. The file name may be formatted in either of the three following ways:

```
DEV:FNAME.EXT
FNAME.EXT
FNAME
```

If you do not specify a device, the file will automatically reside on SYØ:. You must specify SY1: if you want the file to reside there. If you do not specify an extension, TXTCON will assume you want a null extension, meaning a nonexistent extension.

5. Type the desired file name and a carriage return.

   TXTCON will request an input file name. This refers to the name of the text file on the cassette. If you want to convert a specific text file, you must enter its name without quotation marks and then a carriage return. If you want the next available text file, simply hit a carriage return.

6. Specify the desired text file as explained above.

   The cassette deck will begin to run. You will be informed when the desired text file has been found and when it has been successfully converted to an HDOS file. You will also be notified if TXTCON finds anything other than TED-8 text. The program will ignore other files such as BASIC programs and continue to search for the specified text file.

When this is completed, TXTCON will request another input file name. At this point you may restart the cycle at step 4 of this procedure, or type a CTRL-D to escape to the HDOS command mode.

APPENDIX E

H17 ROM Code Listing

```
                                    50  *       COPYRIGHT (C) HEATH CO., 1977
                                    53
                                    54
030.000                             55          ORG     30000A
                                    56
030.000  303 014 037               57          JMP     BOOT            BOOT CODE


                                    59  **      MEMORY DIAGNOSTIC.
                                    60  *
                                    61
030.003  041 300 377               62          LXI     H,-64
030.006  071                       63          DAD     SP              (HL) = END
030.007  353                       64          XCHG                    (DE) = END+1
030.010  041 100 040               65          LXI     H,40100A        (HL) = START
030.013  166                       66          HLT                     PAUSE FOR ADJUSTMENT
                                    67
                                    68
                                    69  *       (HL) = START
                                    70  *       (DE) = END
                                    71
                                    72  *       ZERO TEST AREA
                                    73
030.014  042 076 040               74          SHLD    40100A-2
030.017  066 000                   75  MEM1    MVI     M,0
030.021  043                       76          INX     H
030.022  315 216 030               77          CALL    $CDEHL
030.025  302 017 030               78          JNE     MEM1
                                    79
                                    80  *       START TESTING MEMORY, INCREMENT EACH BYTE IN TURN, AND COMPARE
                                    81  *       THAT RESULT TO THE EXPECTED VALUE
                                    82
030.030  006 000                   83          MVI     B,0             (B) = EXPECTED VALUE
030.032  052 076 040               84  MEM2    LHLD    40100A-2
030.035  004                       85          INR     B
                                    86
030.036  064                       87  MEM3    INR     M
030.037  176                       88          MOV     A,M             (A) = VALUE
030.040  270                       89          CMP     B
030.041  312 046 030               90          JE      MEM4            IS OK
                                    91
                                    92  *       HAVE ERROR, (HL) = ADDRESS OF BYTE IN ERROR
                                    93
030.044  166                       94          HLT
030.045  000                       95          NOP
                                    96
030.046  043                       97  MEM4    INX     H
030.047  315 216 030               98          CALL    $CDEHL
030.052  302 036 030               99          JNE     MEM3            NOT AT END OF PASS
030.055  303 032 030              100          JMP     MEM2            AT END OF PASS
```

```
                                105X **      $COMP - COMPARE TWO CHARACTER STRINGS.
                                106X *
                                107X *       $COMP COMPARES TWO BYTE STRINGS.
                                108X *
                                109X *       ENTRY   (C) = COMPARE COUNT
                                110X *               (DE) = FWA OF STRING #1
                                111X *               (HL) = FWA OF STRING #2
                                112X *       EXIT    'Z' CLEAR, IS MIS-MATCH
                                113X *                (C) = LENGTH REMAINING
                                114X *                (DE) = ADDRESS OF MISMATCH IN STRING#1
                                115X *                (HL) = ADDRESS OF MISMATCH IN STRING #2
                                116X *               'C' SET, HAVE MATCH
                                117X *                (C) = 0
                                118X *                (DE) = (DE) + (OC)
                                119X *                (HL) = (HL) + (OC)
                                120X *       USES    A,F,C,D,E,H,L
                                121X
                                122X
030.060  032                    123X $COMP   LDAX    D
030.061  276                    124X         CMP     M                  COMPARE
030.062  300                    125X         RNE                        NO MATCH
030.063  023                    126X         INX     D
030.064  043                    127X         INX     H
030.065  015                    128X         DCR     C
030.066  302 060 030            129X         JNZ     $COMP              TRY SOME MORE
030.071  311                    130X         RET                        HAVE MATCH

                                133X **      $DADA - PERFORM (H,L) = (H,L) + (0,A)
                                134X *
                                135X *       ENTRY   (H,L) = BEFORE VALUE
                                136X *               (A) = BEFORE VALUE
                                137X *       EXIT    (H,L) = (H,L) + (0,A)
                                138X *               'C' SET IF OVERFLOW
                                139X *       USES    F,H,L
                                140X
                                141X
030.072  325                    142X $DADA   PUSH    D
030.073  137                    143X         MOV     E,A
030.074  026 000                144X         MVI     D,0
030.076  031                    145X         DAD     D
030.077  321                    146X         POP     D
030.100  311                    147X         RET                        EXIT
```

```
                                150X **      $DADA. - ADD (0,A) TO (H,L)
                                151X *
                                152X *       ENTRY   NONE
                                153X *       EXIT    (HL) = (HL) + (0A)
                                154X *       USES    A,F,H,L
                                155X
                                156X
030.101  205                    157X $DADA.  ADD     L
030.102  157                    158X         MOV     L,A
030.103  320                    159X         RNC
030.104  044                    160X         INR     H
030.105  311                    161X         RET


                                164X **      $DU66 - UNSIGNED 16 / 16 DIVIDE.
                                165X *
                                166X *       (HL) = (BC)/(DE)
                                167X *
                                168X *       ENTRY   (BC), (DE) PRESET
                                169X *       EXIT    (HL) = RESULT
                                170X *               (DE) = REMAINDER
                                171X *       USES    ALL
                                172X
                                173X
030.106  172                    174X $DU66   MOV     A,D             TWOS COMPLEMENT (DE)
030.107  057                    175X         CMA
030.110  127                    176X         MOV     D,A
030.111  173                    177X         MOV     A,E
030.112  057                    178X         CMA
030.113  137                    179X         MOV     E,A
030.114  023                    180X         INX     D
030.115  172                    181X         MOV     A,D
030.116  263                    182X         ORA     E
030.117  312 205 030            183X         JZ      DU665           IF DIVIDE BY 0
030.122  257                    184X         XRA     A
                                185X
                                186X *       SHIFT (DE) LEFT UNTIL:
                                187X *
                                188X *       1) DE > BL
                                189X *       2) OVERFLOW
                                190X
030.123  142                    191X DU661   MOV     H,D
030.124  153                    192X         MOV     L,E
030.125  011                    193X         DAD     B
030.126  322 143 030            194X         JNC     DU662           IS TOO LARGE
030.131  074                    195X         INR     A               COUNT SHIFT
030.132  142                    196X         MOV     H,D
030.133  153                    197X         MOV     L,E
030.134  051                    198X         DAD     H
030.135  353                    199X         XCHG                    (DE) = (DE)*2
030.136  332 123 030            200X         JC      DU661           IF NOT OVERFLOW
                                201X
                                202X *       (DE) OVERFLOWED. PUT IT BACK.
```

```
                                    203X
030.141  353                       204X          XCHG
030.142  075                       205X          DCR     A               REMOVE EXTRA COUNT
                                    206X
                                    207X *        READY TO START SUBTRACTING. (A) = LOOP COUNT
                                    208X
030.143  140                       209X DU662    MOV     H,B             (H,L) = WORKING VALU
030.144  151                       210X          MOV     L,C
030.145  001 000 000               211X          LXI     B,0             (BC) = RESULT
030.150  365                       212X DU663    PUSH    PSW             SAVE (A)
030.151  031                       213X          DAD     D
030.152  332 163 030               214X          JC      DU664           IF SUBTRACT OK
030.155  175                       215X          MOV     A,L             ADD BACK IN
030.156  223                       216X          SUB     E
030.157  157                       217X          MOV     L,A
030.160  174                       218X          MOV     A,H
030.161  232                       219X          SBB     D
030.162  147                       220X          MOV     H,A
030.163  171                       221X DU664    MOV     A,C
030.164  027                       222X          RAL
030.165  117                       223X          MOV     C,A
030.166  170                       224X          MOV     A,B
030.167  027                       225X          RAL
030.170  107                       226X          MOV     B,A
                                    227X
                                    228X *        RIGHT SHFT (DE)
                                    229X
030.171  067                       230X          STC
030.172  172                       231X          MOV     A,D
030.173  037                       232X          RAR
030.174  127                       233X          MOV     D,A
030.175  173                       234X          MOV     A,E
030.176  037                       235X          RAR
030.177  137                       236X          MOV     E,A
030.200  361                       237X          POP     PSW
030.201  075                       238X          DCR     A
030.202  362 150 030               239X          JP      DU663           IF NOT DONE
030.205  353                       240X DU665    XCHG                    (D,E) = REMAINDER
030.206  140                       241X          MOV     H,B             (HL) = RESULT
030.207  151                       242X          MOV     L,C
030.210  311                       243X          RET


                                    246X **       $HLIHL - LOAD HL INDIRECT THROUGH HL.
                                    247X *
                                    248X *        (HL) = ((HL))
                                    249X *
                                    250X *        ENTRY   NONE
                                    251X *        EXIT    NONE
                                    252X *        USES    A,H,L
                                    253X
030.211  176                       254X $HLIHL   MOV     A,M
030.212  043                       255X          INX     H
```

```
030.213  146          256X          MOV     H,M
030.214  157          257X          MOV     L,A
030.215  311          258X          RET


                      261X **       $CDEHL - COMPARE (DE) TO (HL)
                      262X *
                      263X *        $CDEHL COMPARES (DE) TO (HL) FOR EQUALITY.
                      264X *
                      265X *        ENTRY   NONE
                      266X *        EXIT    'Z' SET IF (DE) = (HL)
                      267X *        USES    A,F
                      268X
                      269X
030.216  173          270X $CDEHL   MOV     A,E
030.217  255          271X          XRA     L
030.220  300          272X          RNZ                     IF DIFFERENT
030.221  172          273X          MOV     A,D
030.222  254          274X          XRA     H
030.223  311          275X          RET


                      278X **       $CHL - COMPLEMENT (HL).
                      279X *
                      280X *        (HL) = -(HL)            TWO'S COMPLEMENT
                      281X *
                      282X *        ENTRY   NONE
                      283X *        EXIT    NONE
                      284X *        USES    A,F,H,L
                      285X
                      286X
030.224  174          287X $CHL     MOV     A,H
030.225  057          288X          CMA
030.226  147          289X          MOV     H,A
030.227  175          290X          MOV     A,L
030.230  057          291X          CMA
030.231  157          292X          MOV     L,A
030.232  043          293X          INX     H
030.233  311          294X          RET
```

```
                        297X **     $INDL - INDEXED LOAD.
                        298X *
                        299X *      $INDL LOADS DE WITH THE TWO BYTES AT (HL)+DISPLACMENT
                        300X *
                        301X *      THIS ACTS AS AN INDEXED FULL WORD LOAD.
                        302X *
                        303X *      (DE) = ( (HL) + DSPLACEMENT )
                        304X *
                        305X *      ENTRY   ((RET)) = DISPLACMENT (FULL WORD)
                        306X *              (HL) = TABLE ADDRESS
                        307X *      EXIT    TO (RET+2)
                        308X *      USES    A,F,D,E
                        309X
                        310X
030.234  343           311X $INDL   XTHL                    (HL) = RET, ((SP)) = TBL ADDRESS
030.235  136           312X         MOV     E,M
030.236  043           313X         INX     H
030.237  126           314X         MOV     D,M             (DE) = DISPLACEMENT
                        315X
030.240  043           316X         INX     H
030.241  343           317X         XTHL                    ((SP)) = RET, (HL) = TBL ADDRESS
030.242  353           318X         XCHG                    (DE) = TBL ADDRESS, (HL) = DISPLACEMENT
030.243  031           319X         DAD     D               (HL) = TARGET ADDRESS
030.244  176           320X         MOV     A,M
030.245  043           321X         INX     H
030.246  146           322X         MOV     H,M
030.247  157           323X         MOV     L,A             (HL) = ((HL))
030.250  353           324X         XCHG                    (DE) = VALUE, (HL) = TABLE ADDRESS
030.251  311           325X         RET


                        328X **     $MOVE - MOVE DATA
                        329X *
                        330X *      $MOVE MOVES A BLOCK OF BYTES TO A NEW MEMORY ADDRESS.
                        331X *      IF THE MOVE IS TO A LOWER ADDRESS, THE BYTES ARE MOVED FROM
                        332X *      FIRST TO LAST.
                        333X *
                        334X *      IF THE MOVE IS TO A HIGHER ADDRESS, THE BYTES ARE MOVED FROM
                        335X *      LAST TO FIRST.
                        336X *
                        337X *      THIS IS DONE SO THAT AN OVERLAPED MOVE WILL NOT 'RIPPLE'.
                        338X *
                        339X *      ENTRY   (BC) = COUNT (MUST BE < 32768)
                        340X *              (DE) = FROM
                        341X *              (HL) = TO
                        342X *      EXIT    MOVED
                        343X *              (DE) = ADDRESS OF NEXT FROM BYTE
                        344X *              (HL) = ADDRESS OF NEXT *TO* BYTE
                        345X *              'C' CLEAR
                        346X *      USES    ALL
                        347X
                        348X
030.252                 349X $MOVE   EQU     *
```

```
030.252  170            350X          MOV     A,B
030.253  261            351X          ORA     C
030.254  310            352X          RZ                      NONE TO MOVE
030.255  175            353X          MOV     A,L             COMPARE *FROM* TO *TO*
030.256  223            354X          SUB     E
030.257  174            355X          MOV     A,H
030.260  232            356X          SBB     D
030.261  332 311 030     357X          JC      MOV2            IS MOVE DOWN (TO LOWER ADDRESSES)
                        358X
                        359X *        IS MOVE UP (TO HIGHER ADDRESSES)
                        360X
030.264  013            361X          DCX     B
030.265  011            362X          DAD     B               (HL) = *TO* LWA
030.266  345            363X          PUSH    H               SAVE *TO* LIMIT
030.267  353            364X          XCHG
030.270  011            365X          DAD     B               (HL) = *FROM* LWA
030.271  345            366X          PUSH    H               SAVE *FROM* LIMIT
                        367X
030.272  176            368X MOV1     MOV     A,M             MOVE BYTE
030.273  022            369X          STAX    D
030.274  033            370X          DCX     D               INCREMENT *TO* ADDRESS
030.275  053            371X          DCX     H               INCREMENT *FROM* ADDRESS
030.276  013            372X          DCX     B               DECREMENT COUNT
030.277  170            373X          MOV     A,B
030.300  247            374X          ANA     A
030.301  362 272 030     375X          JP      MOV1            MORE TO GO
030.304  321            376X          POP     D               (DE) = *FROM* LIMIT
030.305  341            377X          POP     H               (HL) = *TO* LIMIT
030.306  023            378X          INX     D
030.307  043            379X          INX     H
030.310  311            380X          RET                     DONE
                        381X
                        382X *        IS MOVE DOWN (TO LOWER ADDRESSES)
                        383X
030.311  032            384X MOV2     LDAX    D               MOVE BYTE
030.312  167            385X          MOV     M,A
030.313  043            386X          INX     H               INCREMENT *FROM*
030.314  023            387X          INX     D               INCREMENT *TO*
030.315  013            388X          DCX     B               DECREMENT COUNT
030.316  170            389X          MOV     A,B
030.317  261            390X          ORA     C
030.320  302 311 030     391X          JNZ     MOV2            IF NOT DONE
030.323  311            392X          RET                     DONE


                        395X **       $MU10 - MULTIPLY UNSIGNED 16 BIT QUANTITY BY 10.
                        396X *
                        397X *        (HL) = (DE)*10
                        398X *
                        399X *        ENTRY   (DE) = MULTIPLIER
                        400X *        EXIT    'C' CLEAR IF OK
                        401X *                (HL) = PRODUCT
                        402X *                'C' SET IF ERROR
```

```
                                403X *          USES    D,E,H,L,F
                                404X
                                405X
030.324  353                    406X $MU10      XCHG                    (HL) = MULTIPLIER
030.325  051                    407X            DAD     H               (HL) = X*2
030.326  330                    408X            RC
030.327  124                    409X            MOV     D,H
030.330  135                    410X            MOV     E,L
030.331  051                    411X            DAD     H               (HL) = X*4
030.332  330                    412X            RC
030.333  051                    413X            DAD     H               (HL) = X*8
030.334  330                    414X            RC
030.335  031                    415X            DAD     D               (HL) = X*10
030.336  311                    416X            RET


                                419X **         $MU66 - UNSIGNED 16X16 MULTIPLY.'
                                420X *
                                421X *          ENTRY (BC) = MULTIPLICAND
                                422X *                (DE) = MULTIPLIER
                                423X *          EXIT    (HL) = RESULT
                                424X *                'Z' SET IF NOT OVERFLOW
                                425X *          USES    ALL
                                426X
                                427X
030.337  257                    428X $MU66      XRA     A
030.340  365                    429X            PUSH    PSW             SAVE OVERFLOW STATUS
030.341  041 000 000            430X            LXI     H,0             (HL) = RESULT ACCUMULATOR
                                431X
030.344  170                    432X MU661      MOV     A,B
030.345  037                    433X            RAR
030.346  107                    434X            MOV     B,A
030.347  171                    435X            MOV     A,C
030.350  037                    436X            RAR
030.351  117                    437X            MOV     C,A
030.352  322 364 030            438X            JNC     MU662           IF BIT CLEAR
030.355  031                    439X            DAD     D
030.356  322 364 030            440X            JNC     MU662           IF NOT OVERFLOW
030.361  361                    441X            POP     PSW
030.362  074                    442X            INR     A
030.363  365                    443X            PUSH    PSW
030.364  170                    444X MU662      MOV     A,B
030.365  261                    445X            ORA     C               SEE IF MULTIPLIER 0
030.366  312 005 031            446X            JZ      MU663           IS ZERO!  AM DONE
030.371  353                    447X            XCHG
030.372  051                    448X            DAD     H               (D,E) = (DE)*2
030.373  353                    449X            XCHG
030.374  322 344 030            450X            JNC     MU661           IF NOT OVERFLOW
030.377  361                    451X            POP     PSW
031.000  074                    452X            INR     A
031.001  365                    453X            PUSH    PSW             FLAG OVERFLOW
031.002  303 344 030            454X            JMP     MU661           PROCESS NEXT BIT
                                455X
```

```
031.005  361            456X MU663   POP     PSW             (A,F) = OVERFLOW STATUS
031.006  311            457X         RET
                        458X
                        459X
                        460X **      $MU86 - MULTIPLY 8X16 UNSIGNED.
                        461X *
                        462X *       $MU86 MULTIPLIES A 16 BIT VALUE BY A 8
                        463X *       BIT VALUE.
                        464X *
                        465X *       ENTRY   (A) = MULTIPLIER
                        466X *               (DE) = MULTIPLICAND
                        467X *       EXIT    (HL) = RESULT
                        468X *               'Z' SET IF NOT OVERFLOW
                        469X *       USES    A,F,H,L
                        470X
                        471X
031.007  041 000 000    472X $MU86   LXI     H,0             (HL) = RESULT ACCUMULATOR
031.012  305            473X         PUSH    B
031.013  104            474X         MOV     B,H             (B) = OVERFLOW FLAG
031.014  267            475X MU860   ORA     A               CLEAR CARRY
                        476X
031.015  037            477X MU861   RAR
031.016  322 026 031    478X         JNC     MU862           IF NOT TO ADD
031.021  031            479X         DAD     D
031.022  322 026 031    480X         JNC     MU862           NOT OVERFLOW
031.025  004            481X         INR     B
031.026  267            482X MU862   ORA     A
031.027  312 044 031    483X         JZ      MU863           IF DONE
031.032  353            484X         XCHG
031.033  051            485X         DAD     H
031.034  353            486X         XCHG
031.035  322 015 031    487X         JNC     MU861           LOOP IF NOT OVERFLOW
031.040  004            488X         INR     B
031.041  303 014 031    489X         JMP     MU860
                        490X
031.044  260            491X MU863   ORA     B               SET *Z* FLAG IF NOT OVERFLOW
031.045  301            492X         POP     B               RESTORE (BC)
031.046  311            493X         RET
                        494X
                        495X
                        496X **      $RSTALL - RESTORE ALL REGISTERS.
                        497X *
                        498X *       $RSTALL RESTORES ALL THE REGISTERS OFF THE STACK, AND
                        499X *       RETURNS TO THE PREVIOUS CALLER.
                        500X *
                        501X *       ENTRY   (SP) = PSW
                        502X *               (SP+2) = BC
                        503X *               (SP+4) = DE
                        504X *               (SP+6) = HL
                        505X *               (SP+8) = RET
                        506X *       EXIT    TO *RET*, REGISTERS RESTORED
```

```
                                507X *       USES    ALL
                                508X
                                509X
031.047  361                   510X $RSTALL POP     PSW
031.050  301                   511X         POP     B
031.051  321                   512X         POP     D
031.052  341                   513X         POP     H
031.053  311                   514X         RET


                                516X **      $SAVALL - SAVE ALL REGISTERS ON STACK.
                                517X *
                                518X *       $SAVALL SAVES ALL THE REGISTERS ON THE STACK.
                                519X *
                                520X *       ENTRY   NONE
                                521X *       EXIT    (SP) = PSW
                                522X *               (SP+2) = BC
                                523X *               (SP+4) = DE
                                524X *               (SP+6) = HL
                                525X *       USES    H,L
                                526X
                                527X
031.054  343                   528X $SAVALL XTHL                    PUSH H, (HL) = RETURN ADDRESS
031.055  325                   529X         PUSH    D
031.056  305                   530X         PUSH    B
031.057  365                   531X         PUSH    PSW
031.060  351                   532X         PCHL                    RETURN TO CALLER


                                535X **      $TJMP - TABLE JUMP.
                                536X *
                                537X *       USAGE
                                538X *
                                539X *       CALL    $TJMP           (A) = INDEX
                                540X *       DW      ADDR1           INDEX = 0
                                541X *       .         .
                                542X *       .         .
                                543X *       .         .
                                544X *       DW      ADDRN           INDEX = N-1
                                545X *
                                546X *       ENTRY   (A) = INDEX
                                547X *       EXIT    TO PROCESSOR
                                548X *               (A) = INDEX*2
                                549X *       USES    A,F
                                550X
                                551X
031.061  007                   552X $TJMP   RLC                     (A) = INDEX*2
                                553X
031.062                        554X $TJMP.  EQU     *
031.062  343                   555X         XTHL                    (HL) = TABLE ADDRESS
031.063  365                   556X         PUSH    PSW             SAVE INDEX*2
031.064  315 101 030           557X         CALL    $DADA.
```

```
031.067  176          558X          MOV     A,M
031.070  043          559X          INX     H
031.071  146          560X          MOV     H,M
031.072  157          561X          MOV     L,A
031.073  361          562X          POP     PSW             (A) = INDEX*2
031.074  343          563X          XTHL                    ADDRESS ON STACK
031.075  311          564X          RET                     JUMP TO PROCESSOR


                      567X **       $TBRA - BRANCH RELATIVE THOUGH TABLE.
                      568X *
                      569X *        $TBRA USES THE SUPPLIED INDEX TO SELECT A BYTE FROM THE
                      570X *        JUMP TABLE. THE CONTENTS OF THIS BYTE ARE ADDED TO THE
                      571X *        ADDRESS OF THE BYTE, YEILDING THE PROCESSOR ADDRESS.
                      572X *
                      573X *        CALL    $TBRA
                      574X *        DB      LAB1-*          INDEX = 0 FOR LAB1
                      575X *        DB      LAB2-*          INDEX = 1 FOR LAB2
                      576X *        DB      LABN-*          INDEX = N-1 FOR LABN
                      577X *
                      578X *        ENTRY   (A) = INDEX
                      579X *                (RET) = TABLE FWA
                      580X *        EXIT    TO COMPUTED ADDRESS
                      581X *        USES    F,H,L
                      582X
                      583X
031.076               584X $TBRA    EQU     *
031.076  343          585X          XTHL                    (HL) = TABLE ADDRESS
031.077  325          586X          PUSH    D
031.100  137          587X          MOV     E,A
031.101  026 000       588X          MVI     D,0
031.103  031          589X          DAD     D               (HL) = ADDRESS OF ELEMENT
031.104  136          590X          MOV     E,M
031.105  031          591X          DAD     D               (HL) = PROCESSOR ADDRESS
031.106  321          592X          POP     D
031.107  343          593X          XTHL
031.110  311          594X          RET


                      597X **       $TBLS - TABLE SEARCH
                      598X *
                      599X *        TABLE FORMAT
                      600X *
                      601X *        DB      KEY1,VAL1,
                      602X *        .       .
                      603X *        .       .
                      604X *        DB      KEYN,VALN
                      605X *        DB      0
                      606X *
                      607X *        ENTRY   (A) = PATTERN
```

```
                                    608X *              (H,L) = TABLE FWA
                                    609X *      EXIT    (A) = PATTERN IF FOUND
                                    610X *              'Z' SET IF FOUND
                                    611X *      USES    A,F,H,L
                                    612X
                                    613X
031.111  305                       614X $TBLS  PUSH    B
031.112  107                       615X        MOV     B,A
031.113  176                       616X $TBL1  MOV     A,M             (A) = CHARACTER
031.114  270                       617X        CMP     B
031.115  312 133 031               618X        JZ      $TBL2           IF MATC
031.120  247                       619X        ANA     A
031.121  043                       620X        INX     H
031.122  043                       621X        INX     H               SKIP PAST
031.123  302 113 031               622X        JNZ     $TBL1           IF NOT END OF TABLE
031.126  053                       623X        DCX     H
031.127  053                       624X        DCX     H
031.130  264                       625X        ORA     H               CLEAR 'Z'
031.131  076 000                   626X        MVI     A,0             SET (A) = 0 FOR OLD USERS
                                    627X
                                    628X *      DONE
                                    629X
031.133  301                       630X $TBL2  POP     B
031.134  043                       631X        INX     H
031.135  311                       632X        RET



                                    635X **     $TYPTX - TYPE TEXT.
                                    636X *
                                    637X *      $TYPTX IS CALLED TO TYPE A BLOCK OF TEXT ON THE SYSTEM CONSOLE.
                                    638X *
                                    639X *      IMBEDDED ZERO BYTES INDICATE A CARRIAGE RETURN LINE FEED.
                                    640X *      A BYTE WITH THE 200Q BIT SET IS THE LAST BYTE IN THE MESSAGE.
                                    641X *
                                    642X *      ENTRY   (RET) = TEXT
                                    643X *      EXIT    TO (RET+LENGTH)
                                    644X *      USES    A,F
                                    645X
                                    646X
031.136  343                       647X $TYPTX XTHL                    (HL) = TEXT ADDRESS
031.137  315 144 031               648X        CALL    $TYPTX.         TYPE IT
031.142  343                       649X        XTHL
031.143  311                       650X        RET
                                    651X
031.144  176                       652X $TYPTX. MOV     A,M
031.145  346 177                   653X        ANI     177Q
031.147  377 002                   654X        DB      SYSCALL!.SCOUT
031.151  276                       655X        CMP     M
031.152  043                       656X        INX     H
031.153  312 144 031               657X        JE      $TYPTX.         MORE TO GO
031.156  311                       658X        RET
```

```
                                661X **      $UDD - UNPACK DECIMAL DIGITS.
                                662X *
                                663X *       UDD CONVERTS A 16 BIT VALUE INTO A SPECIFIED NUMBER OF
                                664X *       DECIMAL DIGITS. THE RESULT IS ZERO FILLED.
                                665X *
                                666X *       ENTRY   (B,C) = ADDRESS VALUE
                                667X *               (A) = DIGIT COUNT
                                668X *               (H,L) = MEMORY ADDRESS
                                669X *       EXIT    (HL) = (HL) + (A)
                                670X *       USES    ALL
                                671X
                                672X
031.157                         673X $UDD    EQU     *
031.157  315 072 030            674X         CALL    $DADA
031.162  345                    675X         PUSH    H               SAVE FINAL (H,L) VALUE
                                676X
031.163  365                    677X UDD1    PUSH    PSW
031.164  345                    678X         PUSH    H
031.165  021 012 000            679X         LXI     D,10
031.170  315 106 030            680X         CALL    $DU66           (H,L) = VALUE/10
031.173  345                    681X         PUSH    H
031.174  301                    682X         POP     B               (B,C) = REMAINDER
031.175  341                    683X         POP     H
031.176  076 060                684X         MVI     A,'0'
031.200  203                    685X         ADD     E               ADD REMAINDER
031.201  053                    686X         DCX     H
031.202  167                    687X         MOV     M,A             STORE DIGIT
031.203  361                    688X         POP     PSW
031.204  075                    689X         DCR     A
031.205  302 163 031            690X         JNZ     UDD1            IF MORE TO GO
031.210  341                    691X         POP     H               RESTORE H
031.211  311                    692X         RET                        RETURN


                                695X **      $ZERO - ZERO MEMORY
                                696X *
                                697X *       $ZERO ZEROS A BLOCK OF MEMORY.
                                698X *
                                699X *       ENTRY   (HL) = ADDRESS
                                700X *               (B) = COUNT
                                701X *       EXIT    (A) = 0
                                702X *       USES    A,B,F,H,L
                                703X
                                704X
031.212  257                    705X $ZERO   XRA     A
031.213  167                    706X ZRO1    MOV     M,A
031.214  043                    707X         INX     H
031.215  005                    708X         DCR     B
031.216  302 213 031            709X         JNZ     ZRO1            IF MORE
031.221  311                    710X         RET
                                711
```

INDEX