H R U N

An HDOS Emulator for CP/M Systems
by Patrick Swayne, HUG

616 982-3463

Heath Users' Group Part Number **885-1223[-37]**

HRUN is a CP/M program that emulates the Heath Disk Operating System (HDOS). Virtually any non-hardware dependent HDOS program can be run under it, including HDOS Microsoft(TM) BASIC, Benton Harbor BASIC, the HDOS Editor and Assembler, and sophisticated debugging programs such as HUG's ALDT and RDT. Not only does HRUN support nearly every HDOS system call and H17 ROM calls, but it maintains many of the memory tables that "real" HDOS uses, such as the Channel table, the Device table, the Active I/O table and the System Date.

Heath Users' Group
Hilltop Road
St. Joseph, Michigan 49085

TABLE OF CONTENTS

INTRODUCTION

The Heath Disk Operating System is the most popular operating system among members of the Heath Users' Group, but many of the newer members are familiar only with CP/M. For those members, we present the following brief discussion.

How HDOS Works (for CP/M Users)

HDOS is a file oriented operating system. That means it is broken up into several little programs, each of which does a particular job. (CP/M in its original form, by contrast, used a reserved area of the system disk for the entire system, which was invisible to the user.) The central part of HDOS is a program called HDOS.SYS, which manages the use of the other programs. It loads them into memory from the disk as required, and frees up memory space when segments are not required. The programs that "talk" to the outside world (that is, to disks, printers, etc.) are called "device drivers". The device drivers and other temporary parts of HDOS (called "overlays") can be permanently loaded into memory if you wish for faster operation at the expense of memory space. Tables are maintained in memory to indicate to HDOS (and to user programs that wish to use them) which device drivers are in use at the time, their status in memory (temporary or permanent), and how they are being used (read, write, etc.), along with other information.

The Device Table is the table that indicates which device drivers are available, where they are in memory (if they are in memory), and their capabilities, among other things. The Channel Table is used to show when a device is actually in use, and shows how it is being used. For disk file operations, the channel table will contain a copy of the file's directory entry. The Active I/O Table contains a copy of part of the Channel table, indicating the current I/O operation. (Note: You do not need to know all of this to use HDOS. It is presented for the benefit of those familiar with operating systems in general.)

Probably the most important thing to remember about HDOS is that it treats all communications with the outside world in basically the same way. Here are two HDOS Microsoft BASIC programs to illustrate this.

```
10 OPEN "O",1,"SY1:DATA.DAT"        10 OPEN "O",1,"LP:"
20 PRINT #1,"THIS IS A TEST"        10 PRINT #1,"THIS IS A TEST"
30 CLOSE #1                         30 CLOSE #1
```

The first of these programs writes a line of text to a disk file, and the second one writes it to a printer. The only difference is in line 10, in the argument to the OPEN command. We can analyze the "SY1:DATA.DAT" and the "LP:" as follows: The "SY" and the "LP" are the names of the device drivers used in each case. Every HDOS device driver has a two letter name. The number 1 following the "SY" is the unit number within that device. Since "SY" is a disk driver (or, more properly, a "directory device"), the unit number corresponds to the particular drive in use. Unit 1 corresponds to CP/M drive B:. If a device driver only has one unit (or if the unit wanted is zero), the unit number may be left off, as in the case with "LP". Following the device driver name and unit number (if any) is a colon, which completes the device driver designation. Following the colon in the first program is the file name, which is required if the device is a directory device. HDOS is very protective, and will spit error messages at you if any of this stuff is not "just so".

How HRUN Works

The HRUN program emulates the parts of HDOS discussed above. It combines the functions of HDOS.SYS, the HDOS overlays, and 5 device drivers into one program. I decided to emulate device drivers in HRUN rather than trying to make it use "real" ones for the following reasons. First, none of the existing device drivers will work on H/Z100 computers, which, for example, use different UART chips to communicate with printers, etc. Second, it would have used more memory to have real device drivers, and I wanted to allow the user as much memory as possible. As it turned out, HRUN gives the user more memory in some applications than real HDOS. Finally, it would have taken longer to implement the use of real device drivers, and I wanted to make HRUN available as soon as possible.

The following chart lists the 5 pseudo device drivers in HRUN.

DEVICE	USE
SY:	The directory (disk) device. Unit numbers are mapped to corresponding CP/M drive letters (0 = A, 1 = B, etc.)
TT:	The console device. Output is the CRT screen, and input is the keyboard.
LP:	The printer device. It uses the CP/M LST: device. Provision is made to expand tabs to spaces (within HRUN) for printers that cannot handle tabs.
AT:	The alternate terminal device. It uses the CP/M TTY: device (or the CP/M CRT: device if your terminal is TTY:). AT: can be used to drive a second printer in some applications.
ND:	The null device. This device can be used to test the read-ability of a disk file by copying it to ND:, or to create an empty directory entry by copying ND: to a disk.

HDOS users will notice that there is only one disk device driver. Real HDOS uses a separate device driver for each controller in the computer. For example, if you have an H89 with a hard sector controller and a soft sector controller, you would have two disk device drivers. Since HRUN works through CP/M, only one disk device driver is needed, which will "talk" to any drive that CP/M can access. That means that the drive you access as DK0: under real HDOS might be SY3: under HRUN.

GETTING STARTED

The first thing you should do is make a back up of at least the first HRUN disk (disk A). You can do it with DUP.COM or by FORMATting a blank disk and copying all of the files on HRUN disk A to it.

To use HRUN, you must first decide which of the pre-assembled versions (HRUN0.COM, HRUNT.COM, HRUN100.COM, or HRUN100T.COM) you need. If your computer is one of the H/Z100 series, use HRUN100 or HRUN100T. If your printer cannot process TAB characters, use a version with "T" at the end of the name (HRUNT or HRUN100T). Among printers that cannot process TAB characters are Diablo "daisy wheel" printers and some Epson printers (Epson has released so many different versions of their printer ROMs that it is impossible to say for certain which printers will handle TAB characters correctly). If your computer is a non-Heath CP/M compatible computer, use HRUN100 or HRUN100T if it does not allow level two software interrupts (RST 2). Otherwise, use HRUN0 or HRUNT. In any case, your computer must allow level 7 software interrupts in order for

HRUN to work.

Rename the version you have selected to HRUN.COM (on the duplicate disk you have made). Then prepare a new bootable CP/M system disk that is empty (no visible files), copy HRUN.COM to it, and boot up on it. Now, enter

```
A>HRUN
```

and hit RETURN. When HRUN signs on, it will show the amount of memory in your computer less 8k. That is because HDOS counts memory starting at 2000 hex instead of at 0 as in CP/M. When HRUN prompts for input (with the character ">"), enter

```
>RESET SY0:
```

When HRUN prompts you to replace the disk, take out the new system disk and insert the duplicate HRUN disk A, and hit RETURN. If you have two drives, place the new system disk in drive B. (NOTE: Operation with only one drive is not recommended unless you are using Heath/Zenith CP/M.) Enter the command

```
>SUBMIT MAKESYS
```

and hit RETURN. HRUN will copy several files to your system disk, creating an HRUN system disk. If you have only one drive (and you are using Heath/Zenith CP/M), you will be prompted to make several disk swaps while the process proceeds. When it is finished, enter

```
>RESET SY0:
```

again and remove the HRUN disk A from drive A:, and replace it with the newly created HRUN system disk. Now, enter

```
>MAKEDIR
>MAKEDIR
```

MAKEDIR creates an HDOS-type directory (a file called DIRECT.SYS) on the CP/M disk. It should be run twice the first time it is used on a disk if you want the file DIRECT.SYS itself in the directory.

Your HRUN system disk is now ready to use. If you would like HRUN to start up automatically when you boot the disk, you can run CONFIGUR and enter HRUN as a cold boot command line. Just boot up on a regular CP/M system disk, copy CONFIGUR to your HRUN disk, boot up on it, and run it to set up the command. Later, you may want to copy BASIC, EDIT, ASM, XREF, PATCH, or XFORM to the disk. You may also want to put FORMAT, SYSGEN, and/or DUP (CP/M programs) on it to make generating new system disks easier.

Any HDOS programs that you want to run with HRUN must be copied to CP/M format disks. The program HTOC is provided with HRUN to do the copying. Its use will be explained later (see Other HRUN Support Programs).

THE HRUN COMMANDS

HRUN has several built-in commands for file handling on your disks, etc. Many of the commands make use of external programs. You can make more disk space on a system disk by deleting one or more of those programs, as long as you realize that the commands associated with the programs will not be available. A summary of the commands follows.

HELP	Print a list of HRUN commands on the user's terminal. The program HELP.ABS is used to make the list, and can be deleted when you are familiar with the commands.
BYE	Return to CP/M. You should not attempt to return to CP/M by any other method (such as JMP 0), because interrupt vectors will be left in memory that may cause the system to crash.
CAT [DEV:]	List files on disk. This command uses PIP.ABS to list the files in the HDOS directory, DIRECT.SYS. The files on SY0: will be listed unless another device is specified. Files that have the S flag set (the CP/M SYS attribute) will not be listed unless /S is typed after the command. If /B is typed after the command, the files will be listed in a brief format, four across the screen.
COPY TO=FROM	Copy FROM file to TO. The command COPY is simply mapped to PIP, and files may be copied with the command PIP TO=FROM. See the discussion of PIP.ABS for more information on copying files.
DATE [NEWDATE]	Display or set date. The date should be entered in the format DD-MMM-YY, for example, 12-Jan-83. A leading zero is not needed if the day is less than 10. The date is stored in ASCII format in locations 8383 through 8391 (decimal), and may be used in your programs.
DELETE FNAME	Delete File(s). You may also use DEL. PIP.ABS is used to perform the deletion.
DIR [DEV:]	List files on disk. This command uses DIR.ABS, and lists files directly from the CP/M directory, so a DIRECT.SYS file is not needed on the disk. Both DIR and CAT show space used on the disk after the files are listed, in HDOS sectors (256 bytes each -- DIR also shows the space in K's). The readings may not agree because DIR lists the actual disk space used while CAT shows the sum of the size of the files. A file will use disk space in CP/M Group increments which are always at least 1K, so the disk space used may be more than the actual size of the file. Both DIR and CAT will accept a unique or ambiguous file name argument to determine the presence of a single file or group of files on the disk.
MAKEDIR [DEV:]	Make a DIRECT.SYS file [on DEV:]. This command uses MAKEDIR.ABS to construct the DIRECT.SYS file. For those familiar with HDOS internal workings, the DIRECT.SYS file is altered some from one on real HDOS. The First Group Number and Last Group Number bytes in each entry are replaced with a 16-bit number containing the actual size of the file in HDOS sectors. PIP.ABS as supplied with HRUN is patched to use that information in reporting file sizes. Note: MAKEDIR must be run on a disk each time files are added or deleted if you want an up-to-date DIRECT.SYS file.
PIP	Execute PIP. See the discussion of PIP.ABS.

RENAME TO=FROM	Rename file FROM to TO. You may also use REN. This command uses PIP.ABS.
RESET DEV:	Change disk in drive DEV:. In real HDOS, a disk must be "mounted" before you can access it, using either a MOUNT command or a system call to mount disks. When the disk is to be removed, it must be "dismounted" if another disk is to be used in the drive. If HDOS is running in what is called the "stand alone" mode, a third command, RESET, is available to perform dismount and mount in one command. Since HRUN emulates HDOS in the stand alone mode, and since disks do not have to be "mounted" in CP/M, the MOUNT and DISMOUNT commands were left out of HRUN. The RESET command simply performs a CP/M disk system reset (and prompts you to change the disk in the drive specified), which, in effect, dismounts all of the disks in the system so that any disk can be changed. The HDOS dismount system call also performs a CP/M disk system reset, but without a prompt to change disks. The mount system call does nothing.
RUN FNAME	Run a program. You may also simply type the program name to run it. The name must be preceded by a drive designation if it is not on the system drive (SY0:).
SET DEV: opt	Select an option for a device. In HRUN, SET only works on the TT: device, and changes are temporary (not recorded on the disk). The program SET.ABS is used for this command. See the discussion on SET.ABS for more information.
TYPE FNAME	Display file contents on terminal. This command uses PIP.ABS.
VER	Display the current version of HDOS. HRUN emulates version 2.0, but there is an EQUate at the beginning of HRUN.ASM that allows you to change the version number.

Line Input

While you enter commands to HRUN, the following control characters are available for line editing.

DELETE	Delete the last character entered and backspace one position (if BKS is set -- see SET.ABS), or echo the deleted character.
BACK SPACE	Identical in function to DELETE.
CONTROL-C	Abort the current entry and start over.
CONTROL-D	The same as Control-C.
CONTROL-U	Abort the current line and start over. This is not really the same as Control-C and Control-D, because Control-U always works the same way as long as HDOS is in the Line Input mode. The responses to Control-A through Control-D, however, are under the control of the currently running program. Control-A through Control-C work like interrupts, and a program may set up a vector to a particular address for each character. Control-A and Control-B are "cleared" while you are in the HRUN command mode. Control-D terminates line input the same way as RETURN. It is up to the program to test if the last character was RETURN or Control-D.
CONTROL-O	Toggle console output. If you type Control-O while a program is listing text to the terminal, listing will cease but the program

	will continue to send characters. Another Control-O will restore normal operation.
CONTROL-P	Toggle printer output. If you type Control-P, any text sent to your terminal also goes to the LP: device. Another Control-P will cancel the first one. Control-P is always available in HRUN, not just during line input, as in CP/M.
CONTROL-S	Suspend character output. When you type Control-S, output is suspended to the TT:, AT:, or LP: device, and execution of the current program ceases. If you suspend output to a printer, it may not stop immediately if it has an input buffer.
CONTROL-Q	Cancel Control-S. You cannot cancel Control-S with anything except Control-Q. However, Control-A through Control-C interrupts are still active, if vectors have been set up. The interrupts take place immediately, not after Control-Q as in real HDOS.

HDOS Error Messages

In HDOS, there is a system call that most programs and HDOS itself make use of to print error messages when errors occur. This program searches the system disk for a file called ERRORMSG.SYS, extracts the appropriate message in it, and prints it on the terminal. If ERRORMSG.SYS is not on the disk, a number representing the message is printed instead, in the form

```
?02 SYSTEM ERROR #nnn
```

where nnn is the number of the error. (The "?02" in the message means "run phase error". There are also "boot phase errors" and "build phase errors" in real HDOS that do not apply to HRUN.) In HRUN, only the numerical error message is printed, in the form shown above. The file ERRORS provided with HRUN is a list of the error messages in numerical order, and you can TYPE that file whenever you get an error. If you use HRUN a lot, you will soon become familiar with the most common error numbers.

HDOS PROGRAMS SUPPLIED WITH HRUN

Some of the programs supplied with real HDOS are also supplied with HRUN. These programs will be described briefly here, with Heath part numbers for documentation on the programs supplied for those who want more information.

PIP.ABS

PIP.ABS is the HDOS Peripheral Interchange Program. Operation is similar to the CP/M PIP program, except that it is a bit more flexible in some areas. It may be used in the command line mode:

```
>PIP [argument]
```

or in the stand alone mode:

```
>PIP
:P:[argument 1]
:P:[argument 2]
:P:[argument n]
:P:^D              (Control-D)
>
```

The argument usually takes the form:

```
DESTINATION=SOURCE[/SWITCH 1 ... /SWITCH n]
```

The destination and source can be any device that HRUN supports. For example,

```
>PIP LP:=SY1:TEXT.FIL
```

Will copy the contents of TEXT.FIL from SY1: (a disk) to LP: (a printer). If the device is not capable of what you ask, you will get an error message. You cannot, for example, use LP: as a source device, because it is capable of write only. If either the source or destination device is a disk, you must supply a file name, which may be either ambiguous (wild cards) or unique. If the disk device is SY0:, you may leave off the device designation and enter the file name only. If the destination is left off entirely, it is assumed to be TT: (your terminal).

You can use multiple sources, separated by commas. If the destination is unique (unambiguous), the sources will be concatinated into one output (one file if the destination is a disk). If the desitnation is ambiguous (this applies only to disk files), the multiple sources will be copied to multiple destinations. Consider the following examples

```
>PIP TT:=SY1:FILE1.TXT,FILE2,FILE3
```

This command would copy FILE1.TXT, FILE2.TXT, and FILE3.TXT from disk SY1: to the terminal, one after the other. Note that the device and extension designations are left off of the second and third sources. PIP will use the device and extension of the first source for any others unless you specify different ones.

```
>PIP *.*=SY1:FILE1.TXT,FILE2,FILE3
```

This would copy the three files to disk SY0: as three files with the same names as on SY1:

```
>PIP *.DOC=SY1:FILE1.TXT,FILE2,FILE3
```

This would copy the three files as above, except that the new ones on SY0: will have .DOC as their extensions.

```
>PIP ALL.DOC=SY1:FILE1.TXT,FILE2,FILE3
```

This would concatinate (combine) the three files on SY1: into one file called ALL.DOC on SY0:.

You can use wild cards in the destination to copy groups of files, or all of the files on a disk. For example,

```
>PIP SY1:*.*=SY2:*.BAS
```

would copy all of the files with the .BAS extension from SY1: to SY2:.

```
>PIP SY1:ALL.BAS=SY2:*.BAS
```

This would concatinate all of the .BAS files on SY2: into one file on SY1:.

PIP Switches

Switches are parameters that modify actions that PIP takes. Switches must immediately follow the source, and be preceded with a slash (/). The valid switches are as follows.

/L	Send a list of the files on the source device to the destination. The source must be a disk. If a unique file name is supplied in the source, that file is listed if found. If wild cards are specified, the files meeting the specifications are listed.
/S	Include files with the S flag set in the source designation (system files). If you are copying files with wild cards in the source, files with the S flag set will not be copied unless the /S switch is used. When used with /L, files with the S flag are included in the file listing.
/B	This is the same as /L, except files are listed in a brief format to take less screen space.
/R	This switch causes the destination to be renamed to the source given. Both the destination and source must be disk files, and must be unique names.
/DEL	Delete the file(s) specified. When this switch is used, the destination and the equal sign should be left off.
/RES	Change the disk in the drive specified. Only a drive designation should precede this switch.
/VER	Show the current version of PIP.
/SUP	Suppress the last line printed by PIP. If you are listing the file directory (/L switch), the line stating the number and total size of the files is not listed. If you are copying files, the line stating the number of files copied is not listed.

Here are some more examples of the use of PIP.

```
>PIP SY1:FILE1.TXT,FILE2,FILE3/DEL
```

Delete FILE1.TXT, FILE2.TXT, and FILE3.TXT from disk SY1:. If PIP.ABS is on SY0:, you can use the command DELETE instead of PIP and the /DEL switch.

```
>PIP README.DOC
```

List the contents of README.DOC (on SY0:) on the terminal. If PIP.ABS is on SY0:, you can use the command TYPE instead of PIP.

```
>PIP LP:=SY1:/L/S
```

Send a list of all files on SY1: to LP: (a printer).

```
>PIP FILES=SY1:*.BAS/L
```

Put a list of all files on SY1: having the .BAS extension (and without the S flag) into a file on SY0: called FILES.

```
>PIP SY1:FILE.BAK=SY1:FILE.DOC/R
```

Rename FILE.DOC to FILE.BAK on SY1:. If PIP.ABS is on SY0:, you can use the command RENAME instead of PIP and the /R switch.

```
>PIP LP:=TT:
```

Copy input at the terminal to a printer (you can also copy input to a disk file). Line editing functions (DELETE, BACK SPACE, and Control-U) are available for correcting mistakes while you are typing. When you are finished, type Control-D. At this point, all of the text you have typed will be sent to the printer (it is stored in memory until then). In all of these examples, you can use the command COPY instead of PIP if PIP.ABS is on SY0:. If PIP.ABS is on another drive, the command PIP would be preceded by the appropriate drive designation.

SET.ABS

This program is used to change certain parameters in device drivers. In HRUN, it only works with TT: (the terminal driver). Commands to SET take the form

```
>SET TT: [option]
```

Here are the options that can be changed in the TT: driver with SET.

HELP	This causes a list of the options available to be printed. The ones that are not listed in this document are not set-able in HRUN.
BKS	Set this parameter if your terminal can backspace. If it is set, pressing BACK SPACE or DELETE will cause the previous character typed on the line to be deleted from the screen, and the cursor will back up. If it is not set, the previous character is echoed.
MLI	Map lower case input to upper case.
MLO	Map lower case output to upper case.
TAB	The terminal can process TAB characters. If TAB is not set, tabs are expanded to the appropriate amount of spaces.

The above options can be preceded by "NO" to negate their effect. For example, if your terminal cannot process TAB characters, you would SET TT: NOTAB.

WIDTH nn	Set console width to NN characters. The cursor will be returned to a new line if the number of characters set is exceeded on a line, except when the setting is 255.

The default settings in HRUN are BKS, NOMLO, NOMLI, TAB, and WIDTH 255. These settings affect output to TT:, AT:, and to the printer if Control-P is in use.

FLAGS.ABS

This program is used to change file flags, which are the equivalent of CP/M file attributes. To use the program, enter its name as a command. It will ask if you need help, and if you answer YES, brief instructions will be printed. It will then prompt you for a file name. After you enter the name of a file, it will print the current flags on the file, and prompt you for new ones. Enter any flags you wish to set, and press RETURN. To clear flags, press RETURN only. The flags that can be set in HRUN are the S (system) flag, which corresponds to the CP/M SYS attribute, and the W (write protect) flag, which corresponds to the CP/M R/O attribute. In real HDOS, there is also the L flag, which locks a file against further flag changes, and the C flag, which indicates that a file is recorded contiguously (sectors in sequential order) on the disk. The C flag is used only by the operating system. FLAGS will let you enter the L flag, but it will have no effect, since there is no CP/M equivalent.

BASIC.ABS

This is Heath's Benton Harbor BASIC. This discussion of it will be brief. If you need more information, order 595-2479 from the Heath Parts. dept. The following lists some Microsoft BASIC commands that are different in Benton Harbor BASIC, along with their B H BASIC equivalents.

MBASIC	BH BASIC	DISCUSSION
SYSTEM	BYE	BH BASIC will ask "Sure?", and you must answer "Y" to exit to the system.
AUTO	BUILD	You must supply a starting point and step size in the form "BUILD n,n". There are no defaults.
INSTR	MATCH	The argument to MATCH takes the form (X$,Y$,I), while INSTR has the form (I,X$,Y$).
LOAD	OLD	You must have a closing quote after the file name.
INP	PIN	The function is identical.
SAVE	REPLACE	You can use SAVE if there is not already a file with the same name as the one you are saving on the disk.
NEW	SCRATCH	BH BASIC will ask "Sure?", as with BYE.
SPACE$	SPC	SPACE$ returns a string of spaces for any purpose, but SPC is strictly for positioning the cursor.
KILL	UNSAVE	Unsave has a default extension of .BAS, so be careful.
FRE	FREE	FREE is a command, not a function. By itself, it is equivalent to PRINT FRE(0) in MBASIC.

The following commands are unique to BH BASIC, and have no counterpart in other versions of BASIC from Heath or HUG.

CIN	Single character input function. With this function, you can input from a file opened for Read operations one character at a time. The function is used as CIN(n), where n is the channel number of the file.
CNTRL	This command changes certain parameters in BASIC, and is used in the form CNTRL n1,n2 , where n1 is the parameter number, and n2 is the argument. CNTRL 0 sets a GOSUB to line n2 when a Control-B is typed. CNTRL 1 sets n2 digits before exponential format is used. CNTRL 2 controls the H8 front panel, and has no effect under HRUN. CNTRL 3 sets the print zone width (the number of spaces printed when commas are used between items in PRINT statements) to n2. CNTRL 4 controls the HDOS overlays, and has no effect under HRUN.
FREEZE	Save everything, as is, on a disk. The FREEZE command saves the BASIC interpreter, your program, and all program variables in a single executable file that can be run directly from the HDOS prompt. When you later run the program, you can continue where you left off by using CONTINUE instead of RUN. A patch is given in REMark issue #29 that modifies BASIC so that FREEZE saves just the program (and variables if it has been RUN), and not the interpreter, to save disk space.
LNO	Line number function. LNO allows you to use an expression instead of an actual number as the argument to GOTO or GOSUB, as in GOTO LNO(X+10).
LOCK	Protect a program from being run or modified.
MAX	The maximum function, used in the form MAX(n1,...,nn). It returns the maximum value of the expressions n1 through nn.
MIN	The minimum function. It works like MAX, but returns the minimum value.
PAUSE (n)	When used with no argument, BASIC waits for the user to hit RETURN. When used with a numerical argument n, it waits for n*2 mS. PAUSE n will not work on the H/Z100.
UNLOCK	Remove the effect of LOCK.
UNFREEZE	Link to any machine language program, including one saved with FREEZE. If FREEZE has been modified according to REMark #29, UNFREEZE can only be used to load a program saved with FREEZE.

BH BASIC supports only sequential disk I/O. The syntax for opening a file is OPEN FILENAME.EXT FOR READ (or WRITE) AS FILE #n, where n is the cnannel number desired. There is no EOF function, so you should place some kind of delimiter or indicator at the end of a file that you can test for when reading it.

Note: EDBASIC and BC.DVD from HUG disk 885-1119 will not work under HRUN.

EDIT.ABS

EDIT is the text editor supplied with HDOS. It has a very unusual command structure, and no attempt will be made to explain it completely here. The manual for EDIT is Heath part no. 595-2477.

EDIT uses command completion. That means that it completes a command for you when you type the first two or three characters. The command for loading a text file for editing is NEWIN/FILENAME/ followed by READ. After you have edited the file, you can save it with NEWOUT/FILENAME/ and BYE.

The command structure while editing is:

[range][verb][qualifier][option][parameters]

The range can be a null, (nothing typed) to use the current line, a space to specify the entire buffer, an equal sign (=) to use the previous range, or an expression composed of combinations of the following symbols.

^	The first line in the buffer.
$	The last line in the buffer.
NULL	The first line in the previous range.
COMMA	Separates items in range expression.
+n	Move forward n lines.
-n	Move backward n lines.
+'STRING'	Move forward until STRING found.
-'STRING'	Move backward until STRING found.

The verbs (commands) in EDIT are as follows:

BLITZ	Discard all text.
BYE	Exit to operating system after writing buffer and closing file.
DELETE	Delete all lines in range specified.
EDIT	Replace old string with new string. The syntax of the parameter field is: <delimiter>old string<delimiter>new string<count>. The delimiter can be any character not used in the strings. The count determines the number of replacements.
FLUSH	Write buffer, and balance of input file to output file. Text is deleted from the buffer when complete.
INSERT	Add to text from keyboard. Use Control-C to end insertion. Use a space for the range to insert before the first line.
NEWIN	Open a file for input.
NEWOUT	Open a file for output.
NEXT	Write the working buffer to the output file, then fill it from the input file. This command is for editing files larger than memory.
PRINT	Print lines on console. Use a space for the range to print the entire buffer.
READ	Read text into memory from the input file.
REPLACE	Replace lines in the command range from the keyboard. This is the

	same as DELETE followed by INSERT.
SEARCH	Search the buffer and then the input file for the string specified as the parameter to SEARCH. The range is set to the line containing the string.
USE	Display the number of lines in the range, bytes used, and bytes free. Use a space for the range to show a count of all lines in the buffer.
WRITE	Write text from the buffer to the output file. Writes from the start of the buffer to the first line of the range specification. Lines are not deleted after writing.

The options in a command line are B, to print the line before the operation, and A, to print the line after the operation. You can use BA to do both, or leave off the option to do neither.

ASM.ABS and XREF.ABS

ASM is the HDOS 8080 assembler. The Heath documentation for it is 595-2478. You may be able to learn to use it by examining the source files provided with HRUN (except HRUN.ASM and HTOC.ASM). The syntax for using the assembler is

```
ASM FILE.ABS,FILE.LST,FILE.TMP=FILE.ASM,DEV:/switches
```

where FILE.ABS is the executable output file, FILE.LST is the listing file, FILE.TMP is a temporary file used by XREF, FILE.ASM is the input file, and DEV: is the device containing any XTEXT (include) files. If any of the file names are left off, the associated function will be skipped. For example, if you leave off the LST file name, no listing file will be produced, and if you leave off the TMP file name, no cross reference will be generated. XREF.ABS must be on the same disk as ASM.ABS if you want a cross reference.

ASM has an include facility that allows the source of a program to be placed in separate files. The pseudo opcode XTEXT followed by a file name FNAME will cause FNAME.ACM to be included as part of the source at that point. If the .ACM file is not on the same disk as the .ASM file or the system disk, you must specify the device containing it in the command line.

The assembler supplied with HRUN has been patched so that it can produce either hexadecimal or octal listing files (octal is standard), and it will ask you which you want when you run it. Just hit RETURN if you are not making a LST file.

Switches in ASM are specified in the form /SWITCH:parameter. Valid switches are:

/LARGE	Use maximum memory for assembly (take out HDOS overlays). This switch has no effect under HRUN.
/ERR	Write program lines with errors on the console as well as in the LST file. If there is no LST file, error lines go to the console by default.
/PAGE:nn	Put nn lines on each page in the LST file. The default is 60.
/WIDE	Inform XREF to use 132 columns for the cross reference instead of 80.
/FORM:nn	Issue nn spaces after each page instead of a form feed character (for printers that cannot use form feeds).
/LON:ccc	Override any LON or LOF pseudo instructions that may be in the program. LON arguments are as follows.

L	List all program lines (default mode). If off, only error lines are listed.
I	List skipped IF lines.
G	List all generated bytes. Normally, only 3 bytes per line (octal) or 4 per line (hex) are listed.
C	List XTEXT included lines. These are normally not listed.

	R	List lines which reference labels in cross reference table. This is normal unless turned off.
/LOF:ccc	Override any LON or LOF instructions in the program. Takes the same arguments as LON, but turns the option off.	

Here are some sample ASM command lines

```
ASM DEMO,DEMO=DEMO
```

This would assemble SY0:DEMO.ASM, producing SY0:DEMO.ABS (the executable file), and SY0:DEMO.LST (the listing file).

```
ASM DEMO,DEMO,TEMP=DEMO,SY2:
```

Assemble SY0:DEMO.ASM to get SY0:DEMO.ABS and SY0:DEMO.LST. Generate a cross reference in the LST file, using SY0:TEMP.TMP as the temporary file. Use .ACM files from SY2:.

```
ASM ,LP:=DEMO/LOF:L
```

Assemble SY0:DEMO.ASM and do not produce an executable file. Send the LST file to the printer, and list only lines with errors.

```
ASM =DEMO
```

Assemble SY0:DEMO.ASM but do not produce any output files (syntax check only).

PATCH.ABS

This program is used to modify other machine code programs. When you run it, it asks you for a file name. Enter the name of the file to be patched, such as BASIC.ABS. If the file has a special patch protection placed by Heath Co., the program will ask you for a patch ID. Any patches requiring ID codes that are printed in REMark will be listed with the codes.

After you enter the name and code, if any, you are asked for an address. Enter the address to be patched. PATCH will display that address and the value there, followed by a backslash. You can enter a new value to replace the old one, a RETURN to leave the old value unchanged, or Control-D to enter a new address. If you enter a new value or just RETURN, PATCH displays the next address in sequence, and the value there so you can continue patching.

When you are finished, enter Control-D in response to the address prompt. PATCH will insert the patches or ask you for more ID codes if the file is protected, and then insert the patches if the codes are correct. A patch to remove the ID code requirement is published in REMark #28.

OTHER HRUN SUPPORT PROGRAMS

HTOC.COM

HTOC is a CP/M (not HDOS) program that copies files from HDOS disks to CP/M disks. To use it, you must be in CP/M (enter BYE to get out of HRUN, if necessary). When you run HTOC, it will first ask you which drives you wish to use for HDOS and CP/M disks. Then it will ask you to place the CP/M disk in the drive you have selected, and a CP/M disk of equal size and density to the HDOS disk you wish to read into the drive selected for HDOS. The purpose of this is to allow CP/M to "select" the disk and update its tables as to the size and density before HTOC attempts to read the HDOS disk. If you are copying a HUG program, you can use one of your HRUN disks for this, since they are in the same format as all disks released by HUG. If the HDOS disk is hard sector (and your system is hard sector), or 8-inch (and you have an H47), you can put it in right away instead of a CP/M disk. In all other cases, you must insert a CP/M disk first. After HTOC sets up things, it will ask you to insert the HDOS disk. Just hit RETURN at this point if the HDOS disk is already in place.

HTOC will attempt to read the directory on the HDOS disk, and if it is successful, it will display a menu of options you can select. The options are:

C -- Display the CP/M disk directory.
H -- Display the HDOS disk directory.
T -- Transfer a file (or files) from HDOS to CP/M
I -- Insert new HDOS and CP/M disks.
X -- Exit and return to CP/M.

If you select T, HTOC will ask you for a file name. You can either enter a specific file name, or wild cards (for example, *.BAS to copy all of the BASIC files from the HDOS disk). HTOC will reprint the name of each file it finds that matches your entry and ask if the file is for HRUN. If it is, type Y or just hit RETURN. Type Control-C if you decide not to copy the file. If HTOC cannot find the file you request, or it has copied all of the files matching a wild card specification and cannot find another match, it will inform you and return to the menu.

If you answer N to "Is this file for HRUN?", HTOC will ask you if you want ASCII translation. Answer Y if the file is a text file, and you want it to be readable under CP/M (rather than HRUN). If you answer Y, it will ask you if the file is an MBASIC program. If you answer Y, HTOC will convert any occurences of "@" in the program (used to extend logical lines) to the format used in CP/M for that purpose.

Note: HTOC can only read HDOS disks that have a "cluster" size of 2 or 4 sectors. That means that any 5.25 inch disk you try to read must be in one of the following formats: 40 track, single side, single or double (16 sector) density; 40 track, double side, single density; 80 track, single side, single density. If the disk is 8 inches, the format must be single side, single density. You should prepare a "transfer disk" that meets the above specifications if you are transferring HDOS files, and your disks are in other formats. You will also need a CP/M disk in the same format.

SUBMIT.ABS

SUBMIT allows you to execute several HRUN commands automatically. Its use is virtually identical to the CP/M SUBMIT program with a few extensions. A text file of the commands to be executed must be created using a text editor. Parameters can be placed in the command file in the form $n, where n is the number of the parameter. To execute a SUBMIT command file, enter

```
>SUBMIT FILENAME parm1 ... parmn
```

The default extension for the command file is .SUB. The parameters you specify in the command line will replace the $n parameters in the command file in order. A temporary file called SUB.SUB is created on disk SY0: containing the text of the original file with parameters expanded. For example, if you have a file DIR.SUB containing

```
DIR $1
DIR $2
```

and you enter

```
>SUBMIT DIR SY0: SY1:
```

the temporary file will contain

```
DIR SY0:
DIR SY1:
```

and those commands will be executed. The temporary file is deleted after all commands have been processed. You can also place more than one command on a line, separated by semicolons. The semicolons will be changed to RETURNs in the temporary file. Our original example file could have been written

```
DIR $1;DIR $2
```

If you wish to pass commands on to a program, they must be on the same line that envokes the program. For example, if you want to run BASIC, load a program, and run it, you could say

```
BASIC;OLD "GAME";RUN
```

The commands OLD "GAME" and RUN would be passed to BASIC. The characters on any one line must not exceed 100 after expansion of parameters.

You can put comments in a command file by placing a single quote (') in the first column:

```
'This is a comment
BASIC;OLD "GAME";RUN
```

If you wish to put control characters in a command file, enter "^" plus the printable form of the character (such as ^C for Control-C).

XFORM.ABS

XFORM.ABS is a program for converting text files from HDOS format (a line feed character only at the end of each text line) to CP/M format (a carriage return and line feed after each line), and from CP/M to HDOS. To use the program, enter

```
>XFORM FILENAME.EXT [DEV:]
```

XFORM will read in the file and output one in the opposite format with .XFM as its extension. You can place the output file on a different drive from the input file by specifying the output device in the command line after the file name (with a space in between). Using XFORM, you can develop source files for use with HRUN using CP/M editors or vice versa.

HRUN OPERATING HINTS

The HRUN System Disk

The CP/M system on the disk that you use as your HRUN system disk should be configured to all of the memory in your computer, to give HDOS programs you run the most room available. For example, if you have 64k of memory, your CP/M should sign on with 64K when you boot. If it does not, you will have to run MOVCPM to allow CP/M to use all of your memory. (Obviously, if you have a special program that requires memory above CP/M, such as Style Writer (see REMark issue #35), you may want to allow for it.)

Running Prologues

Real HDOS allows a program to be executed at cold boot by naming the program to be run PROLOGUE.SYS. In HRUN, this ability can be simulated by creating a SUBMIT command file, and configuring CP/M to execute HRUN SUBMIT FILE at cold boot, where FILE is the name of the command file. An existing PROLOGUE.SYS file may also work if it does not use the HDOS type-ahead buffer. Prologue files created with HUG's SUBMIT programs or the SYSMOD DOCOM command use the type-ahead buffer, and will not work. If yours will work, you can configure CP/M to run HRUN PROLOGUE.SYS at cold boot.

Using the Type-Ahead Buffer

Some HDOS programs make use of the HDOS type-ahead buffer, either to insert commands into it for execution later, or to test it for keyboard input. Those programs will have to be modified to run under HRUN.

HRUN maintains a simulated type-ahead buffer that can be used to simulate most of the features of the real one under HDOS. To insert commands into it, you simply insert the characters into memory starting at 104 hexadecimal (260 decimal), and a count of the characters at 103 hex (259 decimal). Just be sure to include HDOS new line characters (ASCII value = 10) after each command. Consider the following BH BASIC program.

```
10 C$="FREE"+CHR$(10)+"LIST"+CHR$(10)
20 POKE 259,LEN(C$)
30 FOR I=1 TO LEN(C$)
40 POKE 259+I,ASC(MID$(C$,I,1))
50 NEXT I
```

When this program is run, it will execute the commands FREE and LIST just as if they had been typed in.

Some programs test the type-ahead buffer to accomplish what is sometimes called "keyboard polling" or "real time input". This is done usually to make up for the fact that neither HDOS MBASIC nor BH BASIC have an INKEY$ function. Three methods have been presented in REMark magazine for simulating INKEY$. The "machine code method" presented in REMark issue #18, page 24 for MBASIC and issue #29, page 5 for BH BASIC should work without modification under HRUN. Reading the console port directly may work, but is not recommended. The last method, testing the type-ahead buffer, is the best method, but must be done differently in HRUN. REMark issue #33, page 10 showed the right way to test the type-ahead buffer under real HDOS, and issue #35, page 29 showed the WRONG way.

Under HRUN, you can test the type-ahead buffer by simply examining the character count at 103 hex (259 decimal). The following BASIC program illustrates this.

```
10 REM   THIS PROGRAM PRINTS A MESSAGE OVER AND OVER
20 REM   UNTIL ANY KEY IS PRESSED
30 REM   THEN THE CHARACTER AT THE KEY IS PRINTED
40 PRINT "HELLO, THERE!"
50 IF PEEK(259)=0 THEN GOTO 40
60 A$=CHR$(PEEK(260)):REM  GET THE CHARACTER
70 POKE 259,0:REM  CLEAR THE BUFFER
80 PRINT "YOU TYPED THE ";A$;" KEY."
```

Programs that poll the keyboard can be modified using this technique. Lines 50, 60, and 70 can be made into a subroutine as follows:

```
5000 REM   SUBROUTINE TO POLL THE KEYBOARD
5010 IF PEEK(259)=0 THEN A$=CHR$(0):RETURN
5020 A$=CHR$(PEEK(260))
5030 POKE 259,0:RETURN
```

This routine returns a null in A$ unless a key has been pressed. Note: The HRUN type-ahead buffer is not truly interrupt driven, as in real HDOS, but HRUN checks for input during console output. Therefore, a game program (games are usually the kind of program requiring keyboard polling) that is maintaining some kind of screen activity will work normally, but a program that tests the keyboard using this method without any screen activity may not work. The "machine code" method mentioned before will work without screen activity because of Heath/Zenith CP/M's own type-ahead capability.

The Real Time Clock

In H89, Z89, Z90, and H8 computers, an interrupt occurs every 2 mS, which is called the "clock interrupt". The operating system (HDOS or CP/M) uses that interrupt to increment a 16-bit (2 bytes) location in memory called the TIC counter. Several programs in the HUG library, mostly games, use that counter to time their operations. HRUN maintains the HDOS TIC counter when it is run on H89, Z89, Z90, or H8 computers so that those programs run normally. On H/Z100 computers, however, there is no clock interrupt on the 8-bit side, so the HDOS TIC counter cannot be maintained normally. There is a CP/M TIC counter, but its location in memory is different from the HDOS TIC counter. HRUN makes an attempt to maintain the HDOS counter by examining the CP/M counter during calls to the system, and updating the HDOS counter if the CP/M counter changes. However, a program that has a delay loop with no system calls will not work.

The solution is to use the CP/M TIC counter on the H/Z100. BASIC games that use the TIC counter will have PEEKS and/or POKES to locations 8819 or 8820. Those locations should be changed to 11 and 12, respectively. Similarly, assembly programs that reference 201B or 201C hex (40033 and 40034 split octal) should be modified to reference 0B and 0C hex. This is not a complete solution to the problem, however, because the H/Z100 counter increments every 10 mS instead of every 2 mS. However, the designers were thoughtful enough to make the clock rate programmable. It can be switched to 2 mS in assembly language as follows:

```
        MVI     A,500 AND 0FFH
        OUT     0E4H
        MVI     A,500 SHR 8
        OUT     0E4H
```

And, you can switch back to 10 mS with:

```
          MVI     A,2500 AND 0FFH
          OUT     0E4H
          MVI     A,2500 SHR 8
          OUT     0E4H
```

If we evaluate the expressions (such as 500 AND 0FFH) and the port number to decimal, we can make it easy to implement the clock change in BASIC.

```
10 OUT 228,244:OUT 228,1:REM  SWITCH TO 2 MS
20 REM:  THE PROGRAM STARTS HERE
  - - - - -
100 REM:  AFTER THE PROGRAM IS RUN, WE GO HERE
110 OUT 228,196:OUT 228,9:REM  SWITCH TO 10 MS
```

It is important that you return the clock rate to 10 mS before exiting a program because certain disk operations on the H/Z100 require that rate.

Disk Access Under HRUN

HDOS programs that access the disk by track and sector (for making patches) will not work under HRUN. If you want to patch your HRUN disks by track and sector, you will have to do it in CP/M, using an appropriate utility such as HUG's SDUMP (885-1213). A utility that patches FILES on a sector by sector basis will work under HRUN as long as it works through HDOS system calls (not directly through the disk device driver). The UDUMP program sold by HUG uses the device driver directly, even in the file mode, and will not work under HRUN. However, a similar program called SuperZAP (sold by The Software Toolworks, 14478 Glorietta Drive, Sherman Oaks, CA 91423) works fine with HRUN in the file mode (not in the track-sector mode). The FDUMP program sold by HUG (885-1060) will also work.

Writing Assembly Programs for HRUN

If you want to write assembly language programs that run under HRUN, you will need the "HDOS System Programmer's Guide" which is available as Heath part no. 595-2553-02. It explains the HDOS system calls. There are also several useful routines in the H17 ROM (the code from which is incorporated in HRUN) that you may want to use. A list of those routines can be found in the source code for HRUN (HRUN.ASM) under the comment "Useful H17 ROM Subroutines".

ASSEMBLING HRUN

If you make changes to HRUN and re-assemble the source, the resulting HEX file must be combined with H17ROM.HEX (on HRUN disk B) using DDT to make a working program. This can be done as follows: (what you type is in **bold print**)

DDT HRUN.HEX
DDT VERS. 2.2
NEXT PC
1678 0100
-IH17ROM.HEX
-R
NEXT PC
2000 1800
-G0   (Letter G, number 0)

```
A>SAVE 31 HRUN.COM
```

If the file H17ROM.HEX is not on drive A:, insert these lines after "IH17ROM.HEX".

```
-S5C
 005C 00 1
 005D 48 .
```

Change the data at location 5C to 1 if you want to read drive B:, 2 for drive C:, etc.

The file HRUN.ASM is quite large, and if you have 5.25 inch single density single sided 40 track disks, you will have to direct the assembler to place the output file on a different disk from the source, and will have to do without a disk PRN file. A sample command line for envoking the assembler would be:

```
A>ASM HRUN.BAZ
```

This tells the assembler that the source is on drive B:, the HEX file should go to drive A:, and no PRN file is wanted.