

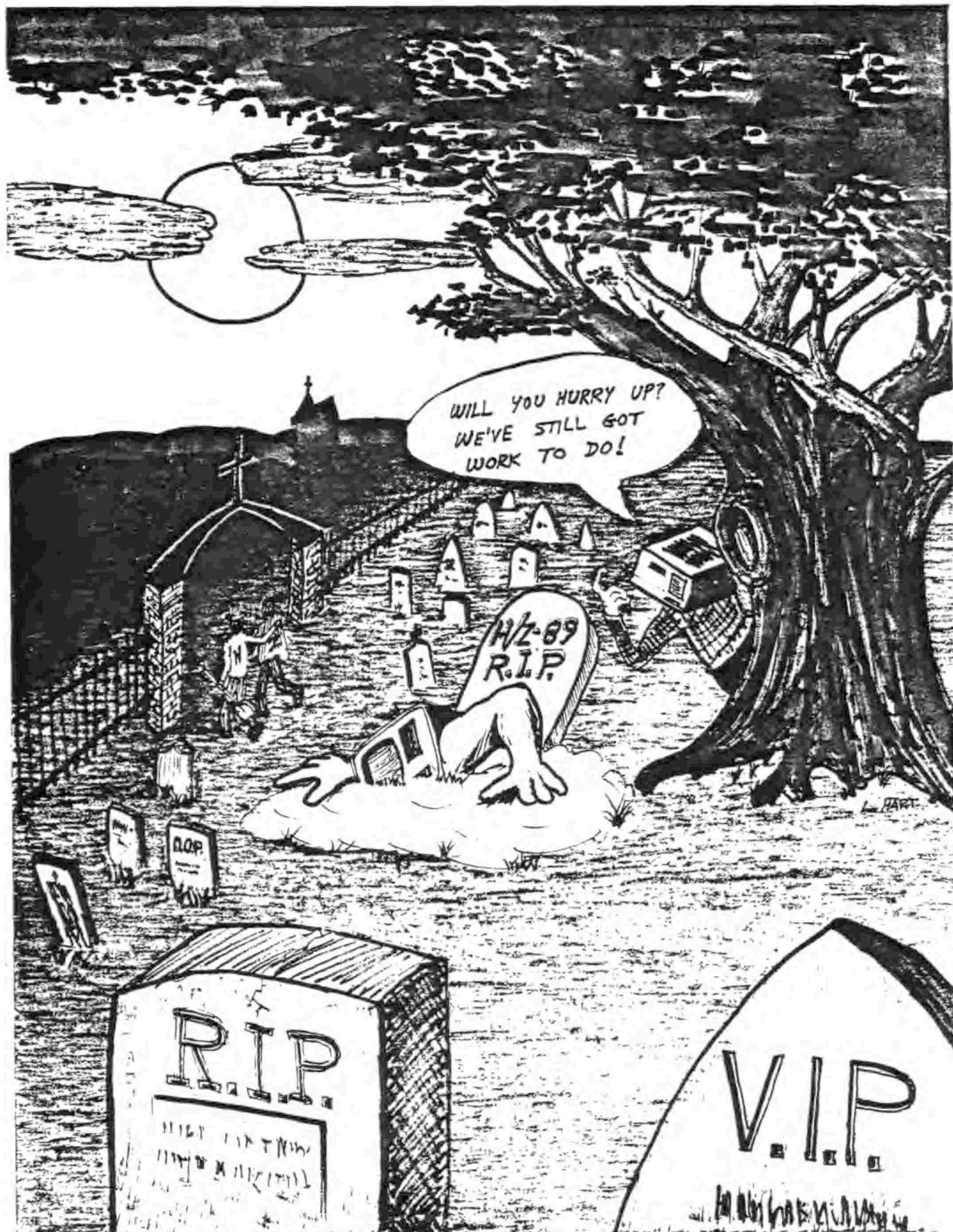
Saving Our HEATH Eight-Bit Machines!

# SETHO JOURNAL

Volume 1, Number 3

\$2.50 a copy, \$12.50 a year

October, 1986



\* Editorial Policy \*

We carry advertisements free of charge for any vendor who continues supporting all H/Z 8-bit machines. Also we solicit compliments and complaints from our subscribers about vendor products and services --or lack thereof. Your comments must be explicit and detailed, no vague statements such as "I don't like it". Please tell us EXACTLY WHY you're unhappy! Vendor response will be printed in due course.

\* We also carry subscriber want ads free of charge. Please send ad between first and 10th of a month to meet our publishing deadline.

NOTE: If you submit an article to us which we subsequently publish we will either extend your present subscription by a year, or will give you a free one if you are not yet a subscriber. Also you automatically will become a full Society member for that same year.

## EDITOR'S COMMENTS

We're receiving many more quite interesting letters from subscribers which back up our sincere belief that we're on the right course, serving the best interests of all Eight-Bit H/Z users. Unlike other computer publications, we shall continue keeping readers well-informed about the Eight-Bit H/Z not-yet-dead-machines (see cartoon on Page one).

Many correspondents have written us, saying the company that swallowed Heath will never re-introduce H-89 kits. But they have indicated that we should continue urging SEBHC members to write Zenith's top brass to get them to resume supplying peripherals and software for the H-8s and H-89s. After all, Zenith is in business to make money, and if we keep after them, they even may make some friends too--a nice combination! We therefore urge you to send at least one letter or postcard a month and let them know how we feel about our 8-bit machines.

\* This month you'll enjoy the three informative articles which should help upgrade our member skills and knowledge. One helps the novice in checking out a used H/Z computer, another demystifies CP/M's PIP utility and the last one is an excellent--almost tutorial--review of TMSI's CP/M Write-Hand Man utility. Enjoy!

-----0-----

*Next month some HDOS articles to enlighten, edify and entertain you!*

---YEAH!---

## Add Parallel I/O to Your H/Z-89 -- CHEAPLY!

Finally! A Use For Those Left-Hand CPU Card Slots!

Our new Parallel Multi-Port I/O board lets you put those skimpy left-hand CPU-board slots (where only the memory expansion board now fits) to good use!

With the Parallel Multiport I/O board you can run CP/M, HDOS or ZCPR on your H/Z-89-90 and use accessories such as graphics tablets, joysticks, or parallel (Centronics) input printers, or multiple-pen plotters. This is difficult to do with the standard 3-port serial board. And the Multiport I/O board draws only an average 400mA at 5V, about the same as a 3-port serial card.

Re-discover the joy of computing! Devise your own A/D-D/A I/O circuits for special laboratory projects. Connect your office or household security system to "old reliable" and be assured you're protected from break-ins, fire or other nasty events. Transfer data between your computer and another at dazzling speeds (80k-baud or higher). Develop NC Machining circuits and software your extra H/Z-89 can use to send TTL orders to a milling machine or other production machinery. . . The possibilities are endless!

Order your Model 170 Parallel Multiport I/O board today! Comes complete with warranty, installation instructions, and operator's manual, for the special SEBHC JOURNAL subscriber's price of \$149.95 (regular list price--\$195.00). Send check, or money order (CODs also accepted) to:

**TECHNICAL ADVISORS, Inc.**  
861 Washington Avenue  
Westwood, NJ 07675  
phone: 201-666-0504

**Tell 'em you saw this ad in the SEBHC JOURNAL!**

# A PIP of a Program

or

## "Your Obedient Servant"

by Lee A. Hart

Anyone who uses CP/M must have met PIP, the "Peripheral Interchange Program". You've undoubtedly called him many times to copy files between disks. But many people think of PIP like a snooty butler; supposedly your servant, but unwilling to help you unless things are done HIS way.

Actually PIP is a very helpful, willing servant, with many hidden talents. He just doesn't speak english very well (and neither do the CP/M manuals). So this article will re-introduce you to this valuable servant. Once you speak his language, he'll do all sorts of things for you that would be very difficult to do yourself!

### Calling PIP

Using PIP is simplicity itself. At the A) prompt just type "PIP", followed by the task he is to perform. PIP will hop into memory, perform his job, and return you back to the A) prompt when done. Here's the simplest way to copy a file:

A)PIP destination=source (return)

PIP is big on moving things around, so all his commands have this form. "Source" and "destination" can be any file name or I/O device. Standard CP/M naming conventions apply so all of the following are valid PIP commands:

A)PIP A:TEXT=B:TEXT	(copy the file TEXT to the A: disk from the B: disk)
A)PIP A:=C:LETTER.TXT	(copy LETTER.TXT from disk C: to disk A:)
A)PIP A:LETTER.TXT=C:	(same, but the destination supplies the name to copy)
A)PIP NEW=B:TEXT	(copy TEXT from B: to A:, and rename the copy NEW)

In the first example, B:TEXT is the source file name, and A:TEXT is the destination name. You don't have to name both the source and destination. If you type only one, PIP will use the same name for both the source and destination (examples 2 and 3). The fourth example shows how to rename a file after you copy it.

There's a shortcut when you have several commands to give PIP. Start by just typing "PIP (return)" at the A) prompt. PIP responds with a "\*" prompt, which is his way of saying "yes sir, I'm ready". You can then type the command lines themselves, without retyping the word PIP each time. This is faster since you don't have to load and exit from PIP for each command. The following does the same thing as the above examples, but much quicker:

A)PIP	(load PIP)
*A:=B:TEXT	(copy the file TEXT)
*A:=C:LETTER.TXT	(LETTER.TXT from C: to A:)
*A:LETTER.TXT=C:	(same as above)
*NEW=B:TEXT	(copy TEXT and rename NEW)
* (return)	(done; exit PIP)
A)	(and return to CP/M)

You can copy multiple files by using CP/M's "wildcard" characters "\*" and "?" in the source name. A "?" automatically matches any character at that position in a filename. A "\*" matches that character, and every remaining character in the name. When "?" or "\*" is included, PIP displays the word "COPYING". It then displays the name of each matching filename on the source disk, and copies it to the destination.

A)PIP B:=A:*.COM	(copy all files ending in .COM)
A)PIP A:=C:TEXT?	(copy all 5-character filenames beginning with TEXT (TEXT1, TEXT2, TEXT+, etc.)
A)PIP A:=B:*.*	(copy all files from B: to A:)

If PIP doesn't understand, he'll say "INVALID FORMAT" followed by the unintelligible command. That's just computerese; if his programmers were human, he'd say "pardon", or "huh"?

The syntax and punctuation used by PIP would drive an english teacher to drink, but it makes sense once you understand it. "Source" and "destination" use the standard CP/M file naming conventions, so they consist of 3 optional parts. First comes the device name, which ends in a colon (:). The device name tells where the file is coming from or going to. The most familiar devices are the disk drives, named A: B: C: etc. But your console, printer, and serial input and output ports are also valid devices (named CON: LST: RDR: and PUN: respectively).



Next comes an 8-character filename. You don't have to use all 8; CP/M will fill in the rest with blanks if needed. All letters, numbers, and most punctuation symbols can be used to name a file. Lowercase letters are usually converted to uppercase. You should avoid using comma, colon, semicolon, square brackets, asterisk, dollarsign, and question marks because they all have other uses in CP/M. Programs (and programmers) that stick these characters in filenames will get all sorts of new nicknames from users.

Third is a 3-character suffix called a filetype. It works just like the filename, and is used primarily as an identifier to describe what type of file you have (text=.TXT, basic=.BAS, computer programs=.COM, etc.). But it actually can be used any way you like. A period (.) separates the filename and filetype, and the same limitations apply to the characters used.

Now I've told you what 99% of the CP/M users already know about PIP. I see a man in the front row has fallen asleep, and an old lady in the back is shouting "Where's the beef". So it's about time I tell you something you DON'T know!

## Gripes

There are two big complaints I hear about PIP. First, PIP is a stupid name; it should have been COPY or something like that. Second, some people think PIP has its source and destination backwards; the command should have been PIP source=destination, like MS-DOS.

The name problem is easy to fix; rename it! One I like is LET, because it reminds us that PIP is like BASIC's LET statement; LET A=B+1. The destination (A) comes first, and the part after the equal sign describes the source operands and the operations to be performed on them.

```
A) RENAME LET.COM=PIP.COM
A) LET NEWFILE=OLDFILE
```

But why must the destination be first? Partly, because that's how it was done on a number of minicomputers and mainframes. It's also consistent with the rest of CP/M. Every CP/M command puts the filename that is being created, edited, erased, renamed, etc. immediately following the command itself (ERA file, STAT file, REN file, etc.). Good programmers know that having consistent rules AND KEEPING THEM makes learning a lot easier than if you make up rules as you go along. If you don't believe it, describe the rule for making a word plural in English.

Next you'll see that PIP can have multiple source files, and do all sorts of manipulations on them before copying. With such complex commands, readability is definitely improved if the destination is first.

## Combine Several Files into One

Suppose you want to print your own letterhead. You carefully create a file called `HEADING` to print on your dot-matrix printer, just like you want it. Now how do you get this heading at the top of every letter you write?

You could include the file `HEADING` in every letter, but that would waste time, memory, and disk space. A better way is to write the letter normally, with no special heading. Then have `PIP` combine the heading and the letter into one file before printing, like this:

A) PIP BOTH=HEADING, LETTER

Here PIP creates the file BOTH that contains first the file HEADING, then the file LETTER. The comma (,) separates the individual source names. You can have as many sources as will fit on a line. Here are some other examples.

```
A) PIP PRICELST=HEADING, PRICES, ORDBLANK
A) PIP FINISHED=FINISHED, 20, 24, 32, 48, 60, 90
A) PIP 10X=X, X, X, X, X, X, X, X, X, X
A) PIP PICTURE=GRAPH, DISPLAY, TEXT
```

The first example could create a monthly price list, consisting of a standard heading, the current month's prices, and a standard order blank at the end. Edit the PRICES file each month, which is smaller and easier to handle. Then use PIP to make the PRICELST file, and print it. Once printed, the PRICELST can be discarded.

Suppose we have an order entry system that has a disk file for each open customer order (numbered 1,2,3 etc.). Once the order is filled, we can put all the completed orders into a FINISHED file as shown. The new FINISHED file contains the old FINISHED file, plus all of today's finished orders.

The third example shows a way to create a file 10X which is 10 pages of identical forms. We first create a blank form X, and then have PIP make 10 copies of it. We could repeat the process a 2nd time with 10X to make 100 copies.

The files copied by PIP can be of any size. Create a file GRAPH which contains just an "ESC F" (this will put the terminal in graphics mode), and a file TEXT which contains "ESC G" (to return to text mode). The fourth example then creates a PICTURE file which when typed will enable graphics, display a picture, and then return to text mode.

---

## Extracting a Piece of a File

---

Now that you can combine files, it would be nice if you could get them back apart again. PIP can do this with the "square bracket" commands [S] and [Q]. The square bracket commands immediately follow the source file they apply to, like this:

```
A) PIP HEADING=PRICELST[QCompany^Z]
```

Suppose our letterhead file ends with the phrase "Company". Then we could extract it from our PRICELST file as shown above. This command tells PIP to read from the PRICELST file until you find "Company", then quit copying. The string "Company" itself is included in the destination file. The string to search for is ended by typing a control-Z (shown as ^Z above).

We can cut something off the end of a file, too. Again using our PRICELST, let's extract only the order blank off the end with the [S] command:

```
A) PIP ORDBLANK=PRICELST[SOrder Blank^Z]
```

PIP starts to copy only when and if it finds the phrase "Order Blank" in the PRICELST file. Again, the ^Z marks the end of the phrase to find, and "Order Blank" is itself included in the destination.

Both [Q] and [S] can be used together to extract any desired piece of a file, as long as you tell it where to start and end. Let's say you have a large BASIC program (BIG.BAS), and want to extract the following subroutine from it:

```
1000 REM PRINT
1010 PRINT A,B,C
1020 RETURN
```

PIP can do it with the following command:

```
A) PIP extract.bas=big.bas[S1000 REM PRINT^Z QRETURN^Z]
```

This says copy part of BIG.BAS into EXTRACT.BAS, starting at line 1000, and ending at the RETURN. Be sure you are explicit enough in describing the phrase to find; if I had used [S1000^Z], it might start copying at "LET A=1000".

The [S] and [Q] commands are most useful if you set up the file with them in mind. Remember our example of a customer order system? Each completed order was merged together into one big FINISHED file. Suppose we goofed, and want to extract one of those finished orders?

Let each order start with a page eject (control-L), followed by the order number itself. Then we can extract order 12 like this:

```
A) PIP 123=FINISHED[S^L123^Z Q^L^Z]
```

Notice you can search for control codes just like any other ASCII key. PIP can delete 123 from the FINISHED file, too.

```
A) PIP FINISHED=FINISHED[Q^L123^Z],FINISHED[S^L124^Z]
```

We did it by copying the old FINISHED file, quitting when we reached 123. Then we copied FINISHED again, but didn't start until just after 123 (assumed order 124). The two pieces are then combined to form the new FINISHED file, with 123 removed.

The Start and Quit commands are sensitive to upper and lower case, so searching for "END" will not find "end". This causes one "gotcha" if you type the whole PIP command on one line. CP/M translates command lines into upper case before passing them to a program like PIP. So by the time PIP sees the command, it is all uppercase. To search for lowercase strings, type PIP all by itself, then enter the command at PIP's ">" prompt.

---

## Copying to the Screen and Printer

---

PIP can obviously copy files between disks. But your console, printer, and serial ports are just as valid a source and destination. For instance, PIP knows your printer as the LST: device, and your keyboard and screen are the CON: device. Thus:

```
A) PIP LST:=B:LETTER.TXT (print the file LETTER.TXT
                           from the B: disk)
A) PIP CON:=B:LETTER.TXT (display B:LETTER.TXT on
                           the screen)
```

This is similar to typing a control-P (to enable printing) and then using the command TYPE B:LETTER.TXT. But PIP has advantages. First, since it does its reading and writing separately, PIP can send and receive data faster. Thus it's a good way to test a console or printer to see if it misses characters at high baud rates. Second, PIP can combine and even alter the files it transfers.

```
A) PIP LST:=HEADING,LETTER,CLOSING
A) PIP CON:=GRAPH,PICTURE,TEXT
```

These work just like the multi-file examples, except that no intermediate file is needed. The first example prints the HEADING, then your LETTER, and finally the CLOSING all with one command. The second example enables character graphics, displays the PICTURE, and switches back to text mode.

PIP can also add line numbers, expand tabs, add or remove page ejects, and otherwise "pretty-up" the way a file is displayed or printed. These options are selected by "square bracket" commands, like this:

```
A) PIP LST:=B:LETTER[T8]      (expand tabs to 8 spaces)
A) PIP LST:=PROGRAM.ASM[N]    (add line numbers)
A) PIP LST:=CHAPTER1[P50]    (new page every 50 lines)
```

The first line tells PIP to print our LETTER, with every TAB character (ASCII 9) expanded to the next multiple of 8 spaces. Most text editors save disk space by using TABs to replace multiple spaces. If your printer can't handle a TAB (and most can't), the file prints with everything scrunched over to the left. PIP can fix this by replacing the offensive TABs with the proper number of spaces.

The second line asks PIP to add line numbers. Line numbers start at 1, and increment by 1. Leading zeros are not printed, and a colon (:) follows each line number. If you use [N2], leading 0's are kept, and a TAB replaces the colon. Line numbering is handy for assembly programs, and some high-level languages like BASIC.

Line 3 tells PIP to start printing at the top of a page, and begin a new page (via an ASCII Form Feed) every 50 lines. A page is normally 60 lines long, so this is the default value if you use [P] with no number.

```
A) PIP LST:=CHAPTER3[T&NP    (expand tabs, number lines,
                               and paginate)
A) PIP PRN:=CHAPTER3         (same as above; a shortcut)
```

Several square bracket options can be used at once, in any order. Also, the closing bracket (]) is not actually needed. If there are multiple sources, each source can have its own square bracket commands. The above shows us printing the file CHAPTER3, with tabs expanded to 8 spaces, line numbers added, and a new page started every 60 lines. Actually, this defines the PRN: device, so the second line does the same thing.

### Mysteries of I/O Devices Revealed

This brings us to I/O devices, a subject that is well obscured by Heath's and CP/M's obfuscatory pontifications. It's not as screwy as they make it look. Simply put, the I/O devices (CON: LST: etc.) work like your disk devices (A: B: C: etc.). Each has a name, ending in a colon. Each device can be read/write, read-only, or write-only. Once you know the names, you can copy data back and forth between them just like disks.

CP/M itself has four "generic" names for such devices: CON: is the console, LST: is the printer, RDR: is an external "read" device (nowdays usually a modem), and PUN: is an external output device or "punch" (modem again). These are present in all CP/M systems, and will handle up to 3 serial ports (PUN: and RDR: are the transmit and receive halves of the same device).

But most H8s and H89s have four serial ports. So Heath added more devices, 12 in all. Some are just alternates for CP/M's names, and can be used interchangeably. Others have extra features, like hardware handshaking etc. And still others are so weird that they have little use except at midnight on Halloween.

Here is a list of the most useful I/O device names, along with their usual configuration. Incidentally, the command STAT VAL: will list all the device names, and STAT DEV: will list the current CP/M-vs-Heath equivalents. The CONFIGUR program can change the port addresses, baud rate, etc.

device name	Port	addr	normally	as a source	destination
CP/M	Heath	hex	octal	used for	(reads from) (writes to)
CON:	CRT:	EBh	350o	console	keyboard screen
LST:	LPT:	E0h	340o	printer	DTE 340-347
RDR:	UR1:	D8h	330o	modem	DCE 330-337
PUN:	UP1:	D8h	330o	modem	DCE 330-337
	TTY:	D0h	320o	spare	DTE 320-327 DTE 320-327

Here's how it works. CON: and CRT: are equivalent names for the same device. It is the console, physically at I/O port address 350 octal (EB hex); using it as a source gets input from the keyboard, and using it as a destination writes data to the screen. Let's use PIP to create a file, using the keyboard as the source:

```
A) PIP TEST=CRT:              (create a file called TEST)
This is a test 12345          (input from the keyboard)
^Z                             (control-Z marks the end)
A) TYPE TEST                  (now type our TEST file)
This is a test 12345          (see?)
A)
```

This is a quick way to write "one-liners"; short little files for special purposes. After the (return) at the end of the PIP command, absolutely anything we type on the keyboard will go into the TEST file; this includes control codes and escape sequences as well. The "file" from the keyboard ends with a control-Z (which also goes into the TEST file).

Take our previous example of a file called GRAPH that puts the terminal in the graphics mode, and its mirror image TEXT. PIP can quickly create them like this:

```
A) PIP GRAPH=CON:
(ESC)F^Z          (ESC key, F, then control-Z)
A) PIP TEXT=CON:
(ESC)G^Z          (ESC key, G, then control-Z)
A)
```

Suppose we just bought a new printer, and it has all sorts of trick ESC sequences to put it in various modes. Let's use PIP to hook our keyboard to the printer, so we can play with it like a typewriter:

```
A) PIP LST:=CON:      (source=keyboard,
Print this!          destination=printer)
(ESC)I and ESC sequence!
^Z                  (all done playing)
A)
```

So now we have a straightforward technique to a) play with the printer to see what its ESC commands do, b) create disk files with the command sequences to select various printer modes, and c) a way to send these files to the printer.

You might have noticed that not all I/O devices can be used as both a source and a destination. For example, printers don't have anything to say, so it makes no sense to use them as the source for a file. If you try this, PIP will give you an error message.

But it's possible to "hang up" the computer by selecting a device that exists but isn't ready to read or write. If you read from the TTY: port and nothing is plugged into the DTE 320-327 connector, PIP will wait forever for something to come in. You probably have less patience and will hit RESET.

### Who Needs a Modem to Transfer Files?

Since you've got spare serial ports doing nothing, you can plug them into a modem, or even into another computer. Then PIP can send and receive files from the remote computer. For simplicity, let's assume we connect our H89s together with an RS-232 cable as shown:

MY END:		YOUR END:
chassis ground - pin 1	_____	pin 1 - chassis ground
transmit data - pin 2	_____	pin 2 - transmit data
receive data - pin 3	_____	pin 3 - receive data
signal ground - pin 7	_____	pin 7 - signal ground

Note that pins 2 and 3 are crossed; the transmit data from one computer goes to the receive data of the other, and vice versa. We'll plug this cable into the connector labelled "DCE 330-337" on the back of each H89. We should also run CONFIGUR and be sure the UL1: and UP1: devices are set to the same baud rate and both talk to port 330 octal (08 hex).

Now the fun begins. Let's send the file LETTER from your machine to mine. I begin by telling my PIP to read a file from the UR1: device (modem input):

```
A) PIP LETTER=UR1:
```

Now you tell your PIP to transmit the file to your UP1: device (modem output):

```
A) PIP UP1:=LETTER
```

The two PIPs will cheerfully transfer the file from your computer to mine, and return to the A) when finished.

If it's so easy, why did so many people write modem programs to do the same job? Well, PIP has some limitations. First, it does no error checking to see if the file was received correctly. This could result in scrambled files if you send them a long distance in a noisy environment. Second, PIP can only transfer a file that can all fit in memory at once. The problem is the PIP on the receiving end; it must pause when its memory gets full to write to disk, but it has no way to tell the transmitting end to wait. The result is that characters are lost while the disk access is done.

Note that you could have used any serial port just as well; the TTY:, CON:, or even the LST: device. For instance, I could send you a file and have it printed on your printer:

```
A) PIP LST:=UR1:      (type this on your machine)
A) PIP UP1:=LETTER    (..and I type this on mine)
```

In other words, I can use your machine as a print spooler! I can send the file at a high baud rate, so my machine is free for my next project. Your computer receives the file, and then leisurely prints at whatever rate your printer allows.

### The [Square Bracket] ABC's

We've discussed a few of the [square bracket] commands, but there are actually a whole alphabet full of them. Some are rather obscure, but others are quite useful.

Each source filename in a PIP command can be followed by a pair of square brackets, with one or more letters within it. You can include blanks between the letters for readability. Many of the letter commands can be followed by an optional number, telling how many times to do it, etc.



- B - Block transfer mode. PIP reads data until an X-OFF (control-S) is encountered in the source file. PIP then writes whatever it has so far to the destination device, and goes back to read more from the source.
- Dn - Delete any characters on the line past column n. Good for shortening long lines for printing or editing by a device that can't handle long lines.
- E - Echo all transfers to the console, too. Lets you see what is getting copied to the destination.
- F - Filter out form feeds (control-L) from the source. Eliminates page ejects, usually so you can add your own back in with the [P] command.
- Gn - Get file from USER n. Lets you read a file from a different USER number.
- H - Hex file transfer. xxx.HEX files are produced by Assemblers and some compilers. They are an ASCII representation of a .COM file, and have built-in checksums for error checking. The [H] command tests to see that the source is a valid HEX file, and strips out any extraneous non-hex data.
- I - Ignore any ":00" records when transferring Intel format .HEX files. Pretty useless nowadays.
- L - Lowercase conversion. Translate all uppercase letters into lowercase.
- Nn - Number lines, starting at 1 and counting by 1s. Leading zeros are not printed, and the number is followed by a colon (:). [N2] prints leading zeros, and replaces the (:) with TAB so the next column lines up (and [Tn] will expand the TAB into spaces). Useful for assembly language programming, and for writing BASIC programs with a text editor.
- O - Object file transfer; ignore control-Z's, and copy to the physical end-of-file. ASCII files normally end in a control-Z, so PIP stops copying when it finds one. PIP automatically ignores control-Z's in .COM files. Use [O] to copy ASCII files that contain control-Z's, or program files not named .COM.
- Pn - Page eject every "n" lines, with an initial page eject at the beginning. The page eject is an ASCII Form-Feed (control-L). The default is 60 lines per page if "n"=1 or not given.

Qstring^Z - Quit copying if "string" is found, terminated by a control-Z. The "string" itself is copied, too. Useful to divide a large file into smaller pieces.

Sstring^Z - Start copying only when "string" is found, terminated by a control-Z. The "string" itself is copied, too. [S] and [Q] can be used simultaneously to extract a portion of a file.

Tn - Tabs expanded to every "n"th column. Tabs are replaced with the number of spaces needed to reach the nth column.

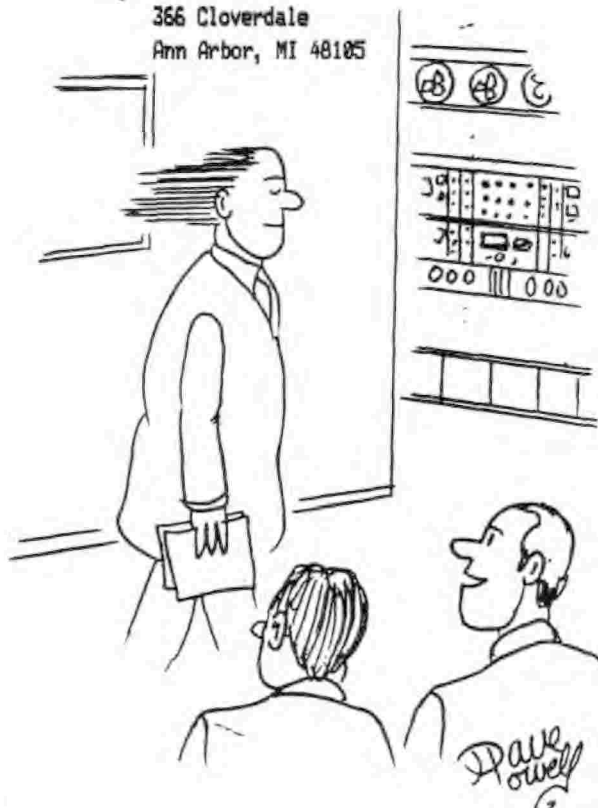
U - Uppercase conversion. Translate all lowercase letters into uppercase.

V - Verify that the destination does indeed match the source file (read after write). Useful if you don't trust you disk copying, or "just in case".

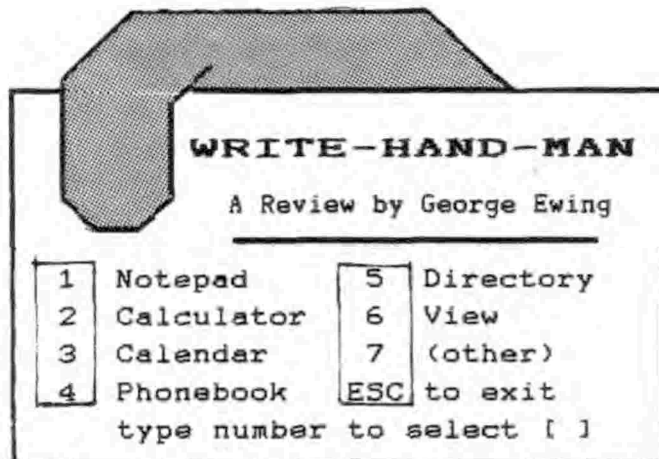
Z - Zero the parity, or most significant bit of each character. Useful to copy Wordstar files, or other text files that have had the most significant bit set somehow.

That all, folks. I'll bet even the most hardened CP/M veteran picked up a few new tricks. And I still probably left out your favorite technique. If so, write and let me know. Pretty good for a 18-year old program, eh?

by: Lee A. Hart  
366 Cloverdale  
Ann Arbor, MI 48105



"I HEAR HE'S ONE OF THE FASTEST COMPUTER PROGRAMMERS AROUND!"



One reason that many people give for changing from CP/M to an MS-DOS system is that CP/M's 64K memory limit is too small to allow memory-resident programs like Sidekick (tm). These "pop-up" programs are a great convenience. Once loaded, they hide somewhere in memory, invisible until you need them. Then while you are in the middle of another program, such as a word processor, spreadsheet, or database, you can "pop" them up with a single trigger character, use the calculator, view a file, etc. and then return to your original program, undisturbed.

In 1985 Alan Bomberger of Poor Person Software, Inc. wrote Write-Hand-Man (tm), a pop-up that did the same things as Borland's Sidekick, but on a CP/M system. Compared to Sidekick which takes over 64K, Write-Hand-Man used just 3K of memory space at any given time. This program has been very successful, and widely reviewed.

Now Technical Micro Systems, Inc. has released an enhanced version of Write-Hand-Man. Called WHMT for short, it is specially tailored to take advantage of the capabilities of the Heath/Zenith H-19 terminals and H-89 series computers. WHMT runs under CP/M on computers like the H/2-89 and H-8, and with other non-Heath/Zenith computers using an H-19 terminal or H-19 terminal emulation.

The TMSI enhancements are in five areas:

1. Improved screen-restore to clear the screen of the text and graphics left by the pop-up session.
2. Excellent use of H-19 graphics; the Phonebook looks like an actual Rolodex file, the calculator looks real, etc.
3. Added features, like string search, printing, etc.
4. Uses H-19 keypad and function keys for commands and cursor positioning.
5. More info to customize and write your own applications.

The enhancements do come at a price; they add 2-3k to the memory required, depending on how you have your system and applications configured. Practically speaking, when WHMT is used with a word processor, it will reduce the maximum file size you can edit in memory by approximately one or two pages of text. Most users should find this a small price to pay for the added convenience.

TMSI sells several different packages of the software, ranging from the cheapest at \$29.95 which includes only the stock version of Write-Hand-Man itself, to a \$99.95 version with complete source code for all applications and a great deal of additional help for programmers who want to write their own applications. The version I am reviewing here is the 5/11/86 release, which included both the original Write-Hand-Man and the latest TMSI WHMT enhancements in a package deal for \$49.95.

The software is not copy-protected, and there is no fine-print "shrinkwrap warranty". The purchaser can legally use the program on more than one machine, a big bonus if you have both an H-89 and a Kaypro for example. You could use the enhanced version for your Heath system at home, and the regular version on the Kaypro when travelling.

The distribution disk came with 17 files. They are:

WHM.COM	6K	Original Write-Hand-Man, unmodified
WHMCONF.COM	3K	Configuration program for WHM.COM
WHMT.COM	8K	Write-Hand-Man with screen restore
WHMTCONF.COM	4K	Configuration program for WHMT.COM
NOTEPAD.REL	2K	Notepad application
NOTEPAD.DAT	2K	Notepad data file
PHONEBOO.REL	2K	Phonebook application
PHONEBOO.DAT	3K	Phonebook data file
CALENDAR.REL	2K	Calendar application
CALENDAR.DAT	6K	Calendar data file
DIRECTOR.REL	2K	Disk Directory
VIEW.REL	2K	File-viewing window
CALCULAT.REL	2K	A 4-function Calculator
HEX.REL	2K	A Hex/Decimal Calculator
ASCII.	3K	ASCII look-up table
OPCODES	5K	8080 opcode look-up table

Note: WHM occupies 6K on disk, but only uses 3K of program space once enabled; likewise, WHMT occupies 8K on disk, but only uses 6K in memory.

Write-Hand-Man has been reviewed elsewhere in detail by more experienced programmers. Suffice to say that both WHM and WHMT provide a miniature version of the CP/M environment. When an application is called by WHM or WHMT, it loads into memory from disk and does its job without disturbing the original user program, which is untouched in its regular location. When the relocated application is finished, it exits with a single JMP instruction. This returns control to the main program, which continues undisturbed.

WHMT provides the operating system hooks for people who want to write their own pop-up applications. As you might guess from the ".REL" extents, the application files are written in machine language using a relocating assembler. Source code for the NOTEPAD application is included as an example of how it's done. Applications are written like any other CP/M program, assembled by the CP/M assembler, and tested and debugged with DDT. The final application is assembled with a relocating assembler such as Microsoft's M80 in order to be recognized and loaded by WHMT.

The regular Write-Hand-Man applications have reviewed elsewhere, so I will just give the salient points on the TMSI enhancements here. On the favorable side, the Notepad, Phonebook, and file View functions all allow you to print the output from the screen. This gives the user hard copy without having to exit the program to bring up a printer or having to copy or type the information from the screen manually, very handy indeed.

The Notepad is multi-page "scratchpad" where you can create, edit, and print notes to yourself. Like all the enhanced applications, editing can be done with either the keypad arrows and function keys, or with "Wordstar" type control keys (control E/X/S/D for cursor up/down/left/right etc.). There is a "help" function (shown in fig. 1), and you can even edit and add notes to it!

```

*****
NOTEPAD help                                page 0
-----
Move cursor with the keypad,
Home, Backspace, Delete; or
'Wordstar' keys ^S, ^X, ^D, ^E.
f1=^Forwd page    ^C=Cut/Paste
f2=^Top page      ^Y=ERASE page
f3=^Jump page     ^K=BLUE=find
f4=^Last page     RED=^Question
f5=^Back page     WHITE=^Print

```

fig. 1 - Write-Hand Man's NOTEPAD

The Phonebook application (fig. 2) works like a "rolodex" card file, with one card per name. It is really a miniature database program, allowing you to create your own forms, setup field names, etc. The "FIND" command will locate a given name, phone number, or ZIP code for you. Phonebook can print addresses, and even dial phone numbers for you with an autodial modem. It is very nice, though I haven't used it much, as I already have a dedicated address and phone number program, called FAMDEX.

```

----- PHONEBOOK HELP -----
^I=TAB to next field (after :)
^K=BLUE find "text" this field
^Y=ERASE page to match page 1
^P=WHITE=print ^Q=RED=help
-----
f1=^Forward f3=^J dial phone#
f5=^Backward f2=^Top f4=^Last
ESC=exit Page 000

```

```

NAME: George Ewing
ADDRESS: 1524 Brightwater NE
CITY: St. Petersburg
STATE: Florida ZIP: 33704

PHONE: ATD (813) 895-3567
NOTES: This is what a sample
       data page looks like.
10/4/86 Page 002

```

fig. 2a - The PHONEBOOK Help Page  
2b - A Typical Data Page

The Directory application (fig. 3) is not as elaborate as those supplied with ZCPR3, but it does share some very nice features with them. It allows you to examine the directory of ANY disk in any user level, even one that is not currently logged, without losing your place in your regular program. It lists file sizes in K, and their R/W and SYS attributes as well.

The View program (fig. 4) is especially nice. I used it extensively for footnoting and indexing a book. You can load a table of footnotes or an index into your word processor, and then use the View window to check the actual items in the regular text as you write the footnotes. Though not a true multi-tasking 'window' (View only shows you the 2nd file; you can't edit it), it is still very useful. The global Search feature is VERY useful for this kind of work, though it does equate upper and lower case characters. For example, it doesn't distinguish between "Ford" and "ford".

The Calendar application is not a real-time clock calendar, nor is it a file with a 365-day data equivalent of a paper calendar. What it really is, is a 14 page appointment book, set up to record two weeks of business meetings, etc. I understand this is the last of the original unenhanced applications, and due to be replaced by a true calendar application soon.

```

A>DIR B:*.DAT

filename.ext  size  r/o  sys
NOTEPAD .DAT   2K
PHONEBOO.DAT  2K  R/O
CALENDAR.DAT  6K   SYS
  3 files, totalling 10K

<ESC> to exit, <RET> for more

```

fig. 3 - Sample DIRECTORY Display

Unfortunately, the Calculator (fig. 5) did not allow printing. I used the calculator a great deal, which is a full 14-digit floating point. Being able to prepare paper 'tape' of your financial calculations at tax time, etc. would be handy. Simple trig functions, or at least a square root key would also have been nice, but probably would have made the calculator too complicated and bulky for 2K of memory. I don't do Hex/Decimal calculations much, but I can see how handy it is for those who do.

#### Compatibility:

Even in the memory-fat world of MS-DOS, compatibility is always a concern with memory-resident programs. Running a keyboard-mapping macro program, a print spooler, and a pop-up all from different vendors can generate real headaches. You could go to an integrated system like Microsoft Windows, but that really amounts to changing to a whole new operating system. In the more confined 64K world of CP/M, special care must be taken.

Configuring Write-Hand-Man needs only two special keys, the trigger that calls it up (typically BREAK or control-@) and

the "home-cursor" sequence that it finishes with. By making those codes re-assignable with the configure program, WHM will run on just about any CP/M system. Keeping the memory requirement small helps a lot, too. The TMSI enhancements, besides adding a small additional memory requirement, affect compatibility in a couple of ways.

The manual is very nicely laid out, and quite useful. It has 42 pages, in a 8-1/2" x 5-1/2" digest size, and clearly explains the functions. The new screen-restore function supports the following terminals:

Televideo 910, 920, 925, 950	SOROC
Heath/Zenith H-19, H-29	Osborne
DEC VT52	Kaypro
Freedom 50/100	QX-10
Wyse 50/100	Hazeltine 1500
ADM 3A/31	

It ought to work with terminals that emulate one of the above, providing the emulation is adequate. VT100-style ANSI terminals are not supported, as this would make the program far too large and complex. Please note that we are talking about the screen-restore function only, here, not the H-19 character graphics that are used in displays like the Phonebook or Calculator. For these, you must have an H-19 terminal, or a terminal that can emulate the Heath/Zenith H-19 character graphics.

Low cost kit-built terminal boards like the ZRT-80 and the Linger 65/9028VT OUGHT to work, if they in fact do a complete H-19 emulation. But if you build up one of these with a surplus ASCII keyboard, you may have to make allowances for differences from the standard H-19 keyboard. For example, you may have to type 'ESC-R' when the WHMT instructions call for a "WHITE" function key, etc. I also ought to point out that the screen restore doesn't always handle H-19 graphics perfectly; it sometimes leaves the terminal toggled so that it ignores reverse-video characters in the original main program display, a minor nuisance.

View B:NOTEPAD.MAC		FIND:	<ESC> to exit
org	100h		
call	home	; home cursor	
mvi	b,16		
topline:			
lxi	h,top	; display top line of notepad	
call	pstring		
dcr	b		
jnz	topline		
f1=Ahead f2=Start f3=Jump f4=End f5=Back ERA=Wrap ON BLU=Find RED=Help WHITE=Print OFF			

fig. 4 - VIEW Window Appears at Bottom of Screen



Software compatibility is another can of worms entirely. Write-Hand-Man WILL work with other memory-resident programs such as print spoolers, RAM disks, etc. providing there is adequate memory available. If the other program uses a trigger character, both that program and Write-Hand-Man must be configured so their trigger characters do not conflict. The manual warns you to load the spooler or other program first, and then engage WHMT. The 5/86 release of the manual also warns of incompatibility with ED-a-Sketch, (because of the way it uses graphics), Textpro, (no available trigger character) and MDM730, a modem program.

I can add another incompatible program to this list, the Software Toolworks screen editor PIE 1.5. Early releases of PIE didn't work at all; the trigger character doesn't get through. A later release of PIE 1.5(d) did work, but there was still an annoying and potentially serious bug: everything worked and appeared normal, but at some random time from a few minutes to an hour or more after loading PIE to edit a file, the whole system crashed! I am just guessing, but it may be that this is because PIE was originally written as an HDOS program, and then converted over to the CP/M operating system.

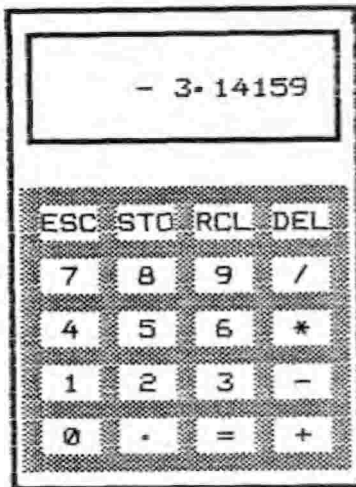


fig. 5 - Decimal Calculator

I have used WHMT successfully with many other programs, including Magic Wand, VEDIT, Scribe, and CP/M's ED with no problems. But the bug with PIE is particularly annoying for me, as I use it for a lot of routine typing. For the time being, I would recommend that you not use WHMT with PIE for any vitally important files, unless you save them frequently. I DID once run for several hours without a crash, but it's living dangerously. Even disengaging WHMT before loading a PIE program is no protection; WHMT 'poisons' the operating system for PIE. You have to reset and cold boot CP/M again to prevent eventual crashes.

In conclusion, the TMSI version of Write-Hand-Man is a worthwhile piece of software, especially for use in a busy office or professional environment. Serious programmers get a beautifully pre-engineered shortcut for writing their own pop-up applications. A couple that come to mind are an automatic save program that would automatically save your word processing file every so many minutes or so many hundred keystrokes, or an automatic reminder to the console for the operator to do the same thing.

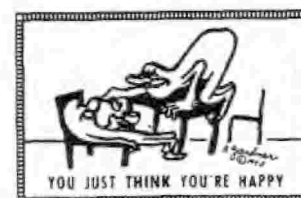
It is also a heartening sign for an old Heathkit H-89 user that companies like TMSI are still bringing out new products supporting a popular machine long after the parent company, or rather the company that swallowed the parent company, has abandoned it and joined the lemmings swarming mindlessly onto the IBM-PC bandwagon.

At a Glance: TMSI's enhanced Write-Hand-Man  
 Type of Program: Memory resident pop-up utilities package  
 Operating system: CP/M (64 K system recommended)  
 Memory: 3K for WHM, 6K for WHMT (see text). Each application takes 2K on disk; each data file uses 1K+, depending on the amount of information currently stored. Typically, all the applications, drivers, and data files take 35-40K on a system disk, but only a few K of memory at a given time.

Price: \$49.95 for a package including both the original Write-Hand-Man and all the TMSI WHMT enhancements. Other packages are available, including special support for custom applications.

Supplier: Technical Micro Systems, Inc.  
 P.O. Box 7227  
 Ann Arbor, MI 48107  
 (313) 994-0784

reviewed by: George M. Ewing  
 1524 Brightwater NE  
 St. Petersburg, FL 33704  
 (813) 895-3567



## STARTUP ADVICE FOR NOVICES So You Bought a Used H/Z-8-bit Computer?!

by A. Stapher

Helped my brother Charlie locate and purchase a computer and printer setup from a couple HUGgers a few days back. Charlie is an architect down in Florida, and had played with my venerable H-8/H-19 during his annual visits to our place here in Michigan. Last year he had played with Heather, the H-89 kit I'd put together a year or so earlier. He had liked the way she runs--almost the same as Nachiban-san (my H-8) except she's smaller, "all in one box, without the external disc drives, those LEDs, and that darned keypad to confuse me!" Also he's interested in the possibility of using the computer to store common drafting graphics symbols on disc for later recall. . .

Unfortunately, I had to go on an analogue computer service call before I could explain to Charlie how to check out and fire up the Z-90 he'd bought from the California HUGger. And Charlie'd left for Florida only minutes before I got back. So I'm writing this for him now, hoping it'll get to him about the same time his computer arrives. Maybe it will prevent him from seeing his acquisition go up in flames and billowing clouds of evil-smelling black smoke. . .

Unpacking, inspecting, and checking the critter --

1. Check the package your computer came in for signs of having been subject to Shipper's Soccer or other mistreatment. Note all signs of damage so you can later submit a claim to the transport company.

2. Open package carefully. Again look for and note any obvious damage as above. Check all packings and wrappings for anything else which might have been included or that might have shaken loose during shipping. Set aside anything that appears to be a computer part; you may need it later. Also save any instruction sheets, nuts, bolts, books, cables, or discs you find.

3. If the computer seems to be in one piece, set it on a piece of corrugated cardboard to keep screws in the computer bottom from gouging nasty holes in your desk top. Get a small screwdriver, then locate the two latches hiding in the gap between top and base on either side of the cabinet. Insert screwdriver behind one latch and slide it towards front of computer. If cover can't spring up, lift on it as you release latch and hold until you've withdrawn screwdriver. Release other latch the same way.

4. Carefully swing the cover up and back, noting there's a power cord attached to the cooling fan mounted inside the cover's right rear corner. Hold, or prop up cover and disconnect cord, then remove cover and set it aside until later called for. If operation manual came with computer, refer to it throughout the next steps.

5. Inspect computer interior for anything which may have shaken loose, such as connectors, printed-circuit cards or cables. Rock computer from side to side, listening for parts rattling around. Fish out loose items with a small magnet or wire hook. If one or more circuit boards placed vertically behind cathode ray tube (CRT) are loose, remove board mounting screws at top corners, reposition boards in their plastic clips and replace screws. A flat cable running across the cabinet bottom hides under a printed-circuit card and connects keyboard to the vertically-mounted card at computer's rear. This cable's connector may be loose at keyboard end. If so, remove six 10-32 mounting screws from beneath keyboard, slide it forward and reseat connector. Be sure to align connector accurately with keyboard pins before reassembly or your computer won't work! Open disc drive door(s), use a flashlight and see if anything has shaken loose inside. Remove all drive packing material before proceeding.

6. If satisfied computer is OK inside, locate power switch on computer back panel, press it OFF, and then plug power cord into computer socket and a 120-volt A-C power receptacle.

7. At left rear of computer as you face it there's a brightness control knob. Rotate the knob so it points up. Press power switch ON. If fuse is good and no boards inside computer were damaged you'll hear two beeps; the computer is saying "I'm OK boss, and ready to go!"

8. Look for an orange glow inside the CRT neck near its socket. The glow means you'll soon be seeing 'H' appearing on the CRT screen. If no H: shows, press OFF LINE key so that it pops UP, then press SHIFT and RESET keys; H: should now appear. Adjust brightness to suit.

9. Press OFF LINE key down so it locks. Hold down any letter key together with REPEAT key. The screen will soon fill with the letter you're holding down. Release both keys when screen is full and study display; it should appear reasonably uniform, level and approximately rectangular. If display is off center, tilted, or both, examine CRT voke assembly--copper wire bunched in a strange way on a plastic form on CRT against bulge. Find a clamp and screw at back of voke, then gingerly

(there's lots of high voltage around there!) press yoke against CRT bulge and rotate until display is reasonably plumb. Don't bother recentering display now. Correct that in the next step. Tighten yoke clamp just enough that yoke won't shift with gentle pressure on it. Note: Tool may affect display, but only so long as it's near yoke.

10. At yoke assembly rear there are two flat tabs which rotate like clock hands. Experiment with tabs until display is as nearly centered as possible. Note: A distorted display which can't be centered means one or more little flat magnets at either side of the yoke assembly have been dislodged, maybe even lost! In this event call Heath Company Computer Technical Assistance line--616-982-3309, 9am - 4:30pm, Mon-Fri EST--for help in ordering parts and correcting this or other problems. They're very nice people!

11. Assuming display is OK, locate a disc with "boot" or "system" written on its label. Check disc for a sticker wrapped around one edge over a square notch. If there's no sticker and notch is open, temporarily use a bit of masking tape to close off notch and prevent disc from being erased while you're learning how to run the computer. Press tape down so it won't catch inside drive. Insert disc in drive (if two drives, put it into one closest to CRT screen) and close door. Check keyboard; make sure OFF LINE key is UP. Next hold down SHIFT key and press RESET key. The screen should blank out and then the H: prompt and blinking underline cursor will reappear. If it's a two drive computer and nothing happens as described above, put disc into other drive and redo these steps. It should work this time.

12. Tap B key, see letter B appear on screen, then RETURN (henceforth <CR>) key and look for the red drive indicator to light up. If it does, you're in luck! You'll next see a "sign-on" message appear on-screen, followed either by A) and cursor or just the > sign and cursor. Note: Sometimes HDOS waits for you to enter a <CR> after everything you enter.

13. Type DIR and <CR>. A listing of what's on the disc should come on-screen. If you've the Heath Disc Operating System (HDOS), part of the directory may scroll up and off screen. Enter DIR/B <CR> and it will again appear but in a short, 4-column form. If you have CP/M and get a "DIR?" message, enter CAT. This should give you a 4-column directory display. In either case, if the disc catalogue or directory has appeared, your computer has passed the preliminary check and should run programs correctly.

From now on, it's up to you, Charlie. Good Luck!

## WANT ADS

### FOR SALE

M-89 with 2 new internal half-height hard-sector drives. TMSI Flicker-Free kit, Magnolia RAM expansion, Analytical Products 4Mhz mod, auto-repeat kits, plus ULTRA-RQM installed. CP/M, MBASIC, PIE included. Price--\$600. Includes LLL single-sided 8-inch controller w/software but no installation instructions. Douglas W. Tibbles, home 804-588-0962, office 804-464-7840 xtn 1223.

-----0-----

### LAFFLINES

--> How come people who haven't made up their minds show up so often in polls and so seldom in discussions?

--> A worker recently celebrated his 100th birthday and his employer offered him a handsome retirement package. "No, sir!" countered the centenarian. "When I took this job in 1910, the boss said it was permanent!"

--> Some football teams will go to any lengths to get a wide receiver. The end justifies the means?

--> Nostalgia: Turns past tense into past perfect!

--> The first thing one learns during a hospital stay is that you're not fully covered by your insurance or your hospital gown.

--> A pedestrian was so busy ogling an attractive young woman that he walked into the side of a moving car. Unhurt, he was helped up by a policeman who commented, "That was close -- your eyes were almost on their last legs!"

--> It takes a lot of brass to speculate in gold.

--> November is when you can't find the Christmas cards you bought on sale last January...

--> A hypochondriac was positive he had a particular fatal liver condition. "Nonsense," snorted his doctor. "Besides, you wouldn't know if you had it or not: it causes no discomfort of any kind."



"Good heavens!" gasped the patient. "My symptoms exactly!"

--> Second-hand-store sign: "We buy old junk. We sell antiques."

--> College journalism student: "Dad, when I graduate, I'm going to write for lots of adney."

Father: "So, what's new?"

# Checkbook Manager Data Base

 An Inovative Program 

**Simplifies Work -- Eliminates Tedium!**

**Here's the program you've been searching for!**

**AND at a price you can't afford to refuse.**

**With Checkbook Manager you can file cheques  
by payee or categories you set up yourself.**

**You can easily correct errors, verify missing items,  
and keep tight rein on personal finances. Never be  
left in the dark again; the program never forgets  
a cheque--even when YOU forget you wrote it!**

***ORDER YOUR COPY, TODAY!***

**Send \$39.95, cheque or money order to:**

**Russell M. Spencer  
54 Bateswood Road  
Waterbury, CT 06706**

**CP/M, ss40trk 5.25-inch hard-sector only**

**Z-80 H-8 or H/Z-89 with 64k RAM**

**Soft-sector distribution discs available soon.**

**Read Program Review of Checkmanager in November** **SEBHC JOURNAL**



# SEBHC JOURNAL

*Composed, Edited and Published by*

Leonard E. Geisler

895 Starwick Drive, Ann Arbor, Michigan 48105

## N O T I C E

Single copies are \$2.50 and 1-year subscriptions are \$12.50.

**\*\* Free \*\*** SEBHC membership with subscription! **\*\* Free \*\***

## Subscription Blank

Name \_\_\_\_\_

Street & Number \_\_\_\_\_ Apt # \_\_\_\_\_

City, State, Zip \_\_\_\_\_

Phone number(s) \_\_\_\_\_

In order to provide better service for our advertisers, subscribers and readers, please fill in the appropriate blanks below:

Computer type: H-8 \_\_\_ H-89 \_\_\_ H-1000 \_\_\_ DG Super-89 \_\_\_ Other \_\_\_\_\_

Data-storage: H-17 \_\_\_ H-37 \_\_\_ H-47 \_\_\_ H-67 \_\_\_ Hard disc \_\_\_\_\_

Accessories: \_\_\_\_\_

\_\_\_\_\_

Programming languages: \_\_\_\_\_

\_\_\_\_\_

Computer used mainly for: \_\_\_\_\_

I'm going to submit my Heath 8-bit article for publication in a future issue. Here is a brief outline of what it's about:

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

Please make out subscription cheque to L.E. Geisler Thank you!

Please show your Journal to your H/Z Eight-Bit friends!

\* WHEN RESPONDING TO ADS, PLEASE SAY YOU SAW IT IN THE JOURNAL \*

## Thank you for your interest

## The SEBHC JOURNAL's Back Page

### Society and Journal Policies

\* The SEBHC JOURNAL is published twelve times a year and is mailed on or about the 22nd of each month. Editorial deadline is the 20th.

\* All advertising is printed free of charge. Vendors must submit seven inches wide by 9 inches high b&w "camera-ready" copy (one page/issue) no later than the 15th of the month in which it is scheduled to appear. SEBHC members are entitled to a free want ad (no more than 250 words, please) in each issue.

\* Subscriptions are \$12.50/year in Canada, Mexico, the U.S.A. and its possessions, and start the month following receipt of application. (Make cheques or money orders payable to L.E. Geisler until further notice.) Single back-issue copies now available on special order only--allow 6 weeks for processing.

\* Subscribers automatically become members in the Society of Eight-Bit Heath Computerists. Member's ID number appears after their name on the JOURNAL mailing label. Any REGULAR member can vote and hold any Society office.

\* There are three classes of membership: REGULAR (H/Z 8-bit user), ADVERTISING (one vote only for each vendor), and ASSOCIATE (non-8-bit computerist, library, etc.). ASSOCIATE members cannot hold offices or vote in Society elections.

\* The SEBHC JOURNAL is composed, edited and printed by Plain English Services (PES), and L.E. Geisler at 895 Starwick Drive, Ann Arbor, MI 48105. Phone -- 313-662-0750, 9am - 6pm EST daily except weekends. A recorder is on-line af - 3pm seven days a week -- maximum message time is about 50 seconds.

## *SEBHC Journal*

895 Starwick Drive  
Ann Arbor, MI 48105