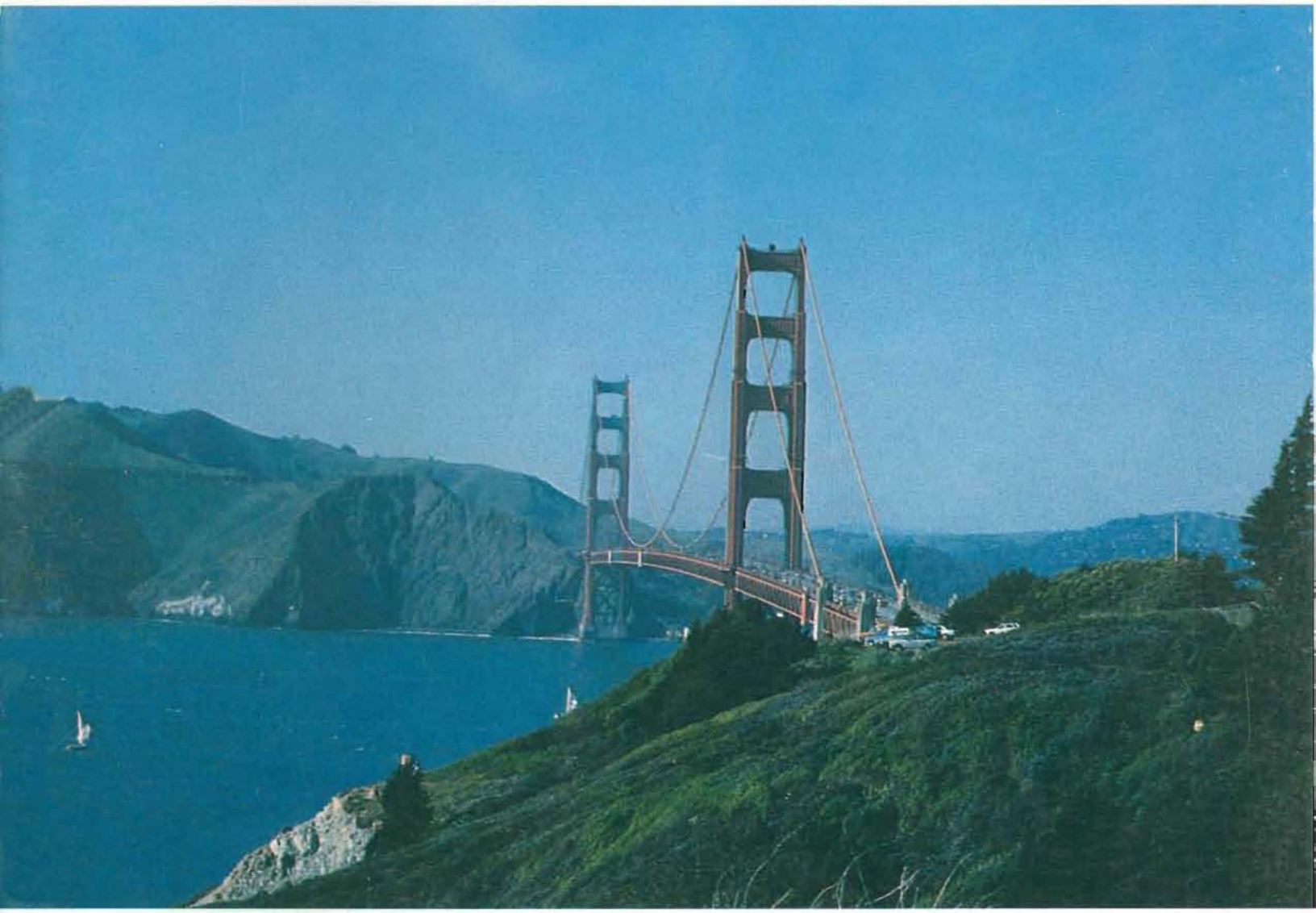
REMark

Issue 27 • April 1982



Official magazine for users of Heath/Zenith computer equipment.

on the cover . . . .

Springtime in San Francisco, California.

West Coast Computer Faire

Photo by: Gerry Kabelman

on the stack

>CAT

"REMark" is a HUG membership magazine published 12 times yearly. A subscription cannot be purchased separately without membership. The following rates apply.

	U.S. Domestic	Canada & Mexico	International
Initial	$18	$20	US FUNDS $28
Renewal	$15	$17	US FUNDS $22

Membership in England, France, Germany, Belgium, Holland, Sweden and Switzerland is acquired through the local distributor at the prevailing rate.

Back issues are availble at $2.50 plus 10% handling and shipping. Requests for magazines mailed to foreign countries should specify mailing method and add the appropriate cost.

Send payment to:

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

HUG Manager ..... Bob Ellerton
HUG Secretary ..... Margaret Bacon
Software Coordinator and Developer ..... Jim Blake
REMark Editor and
Software Developer ..... Pat Swayne
Assistant Editor and Layout ..... Nancy Strunk
HUG Bulletin Board and
Software Developer ..... Terry Jensen

REMark

HUG National Conference ... Update

Hmmmmmmm? Seems when a body needs a rest, strange things happen in your absence such as the announcement of the National HUG Conference. Well folks, all I can say is that JB: has done an excellent job of putting the First National HUG Conference on wheels. Let's take another look at the schedule that has been put together (semi-firm) so far.

Friday August 6, 1982

8:00 PM

Registration begins at the O'Hare Hyatt Regency with the National HUG staff (Margaret and Nancy) handing out a package of necessities that will help you get through the grueling conference schedule.

Saturday August 7, 1982

8:00 AM

Registration continues with a wake-up snack provided by the National HUG Conference sponsors: Heath, Zenith, and the Local Heathkit Electronic Centers.

9:00 AM

Introduce the National HUG staff and the Official Press staff for the National HUG Conference (you will be able to corner these individuals later for some "real words" if you care to).

9:30 AM

Bill Johnson, the President of Heath Company, will give us an inside track to the direction of Heath philosophy with regard to computer products and a walk-through of the twelve acre Heath facilities located in St. Joseph Michigan.

10:30 AM

Jerry Pearlman, Senior Vice President of Finance for Zenith Radio Corporation and the man responsible for Zenith's purchase of Heath Company, will address the Heath Users' Group.

11:00 AM

Tom Dornback, Vice President of Software Development, will give us a talk on the direction of Heath/Zenith software and the growing pains associated with the development of solid software products. Tom has reserved a block of his time to answer questions you may have regarding his discussion.

12:30 PM

Lunch for all HUGGIES in attendance is provided by our sponsors.

2:00 PM

Mike Brenner, Product Line Manager for Terminals and Applications Software, will be giving us some insight on products that fall under his able leadership.

3:00 PM

Barry Watzman, Product Line Manager for Computer Hardware and Operating Systems, will give us a look at computer futures for this rapidly growing field. And, yes folks, Barry will be there to handle your questions and abuse!

5:00 PM

Free time for exploring the vendor exhibit area and a close look at some goodies that we aren't supposed to know about yet.

6:00 PM

Cash bar preceding dinner. A little chance to pick on those people that you will have seen during the previous hours of discussion or a chance to relax for the BIG evening ahead.

7:00 PM

Dinner! You guessed it! The dinner will be provided by our friendly neigbhorhood sponsors again.

8:30 PM

Don Moffet, President of Zenith Data Systems, will deliver The Keynote Address. Following Don, the HUG staff, with the assistance of various vendors, will conduct a drawing for door prizes (don't miss this event ... many neat things!).

Are you tired yet? This is just the first day!

Sunday August 9, 1982

8:00 AM

The vendor exhibit area will be available so that you may do a little more exploring of the neat goodies that are available for Heath/Zenith computer products.

10:00 AM

Our Conference will start off with a little information from one of the larger HUG groups in the country. Hopefully, they can tell us the importance of being an active HUG member at the local level.

10:30 AM

A little discussion from one of our good friends on what is now a classified subject. We will talk about this one a little more in coming issues of REMark.

12:00 PM

Break for an hour for lunch or a last look at the vendor exhibit area.

1:00 PM

The name in this catagory is yet to be announced. However, the discussion will be centered around HDOS. I'm sure that you won't want to miss this one.

2:30 PM

Doug Bonham, the Director of Educational Products, will give us an idea what the computer user can expect in the way of educational "tools" for computers as well as some of the more interesting educational products to look forward to in the future.

4:00 PM

Closing remarks.

There it beez! As of this writing, only a few minor details for participation are left to the imagination. All of this looks good, but we do have some bad news. The capacity for the First National HUG Conference is set at 1000 attendees by the amount of seating and the individuals that the Hyatt can handle for the meals. For this reason the "Interest Form" published in REMark 26 is now an official "Registration Form". For $15.00 sent to the HUG National Conference address, you will receive an Official HUG Conference Name Tag. This tag will be considered your "ticket" to the events outlined previously. One "ticket" admits one HUGGIE. We will be checking the name tags at the door of the Conference during registration. You will find the Registration Form on the inside back cover of this issue. Since we will be pressed for time and materials, the registration fee will go to $20.00 on June 1, 1982.

BE:

Getting Started with CP/M Part 4

William N. Campbell, M.D.
855 Smithbridge Road
Glen Mills, PA 19342

(Note: Any members of HUG and/or users of Heath/Zenith equipment may copy this material, if desired, for their own use. The reason for the "Copyright" is that I may use this material along with my previous articles in REMark (after converting all to CP/M) in a CP/M and MBASIC vers 5.x "course" in the future. Thank you.)

Terry Jensen has eloquently covered the basics of getting Heath's implementation of CP/M booted and configured and has correctly pointed out that it is extremely important that the user should read the documentation. I should like to point out that (in my opinion) the new user should ONLY read Heath's documentation (which is in the front of the supplied manual) and should totally disregard any and all currently supplied documentation written by Digital Research which makes up the bulk of the manual. The foregoing statement applies only to the newcomer to computers and to the "first time user" of CP/M. It has been pointed out that Digital Research's documentation is intended only for the programmer, and not the end user. It is written in a fashion such that the end user may only become horribly confused. IF you understand and are conversant with CP/M, THEN you may read their documentation. So you ask, "How am I going to learn CP/M?". Heath offers a "CP/M Course" (EC-1120 $99.95). I have no personal experience with this course, although Heath's educational courses are usually excellent. On the other hand, there are currently available four reasonably priced "paperback" guides to CP/M. I have read them all. I strongly suggest that you purchase one of them, and I will indicate my personal preferences.

Osborne CP/M User Guide by Thom Hogan
Publisher: OSBORNE/McGraw-Hill
630 Bancroft Way
Berkeley, CA 94710
Cost: $12.99 plus postage

This book is the most comprehensive "guide" available at this time; my number one recommendation, excellent in all respects. Chapter 8 should be required reading for every actual or potential computer user. This guide is worth twice the cost! Summing it up, Hogan's User Guide is #1!

CP/M Primer by Murtha and Waite
Publisher: Howard W. Sams & Co., Inc.
4300 W. 62nd Street
Indianapolis, IN 46268
Cost: $14.95 plus postage

The above Primer is well written, a good text, but lacks discussion of some important details and has no discussion at all of the "SUBMIT" program usually distributed with CP/M.

Using CP/M by Fernandez and Ashley
Publisher: John Wiley & Sons, Inc.
605 Third Avenue
New York, NY 10158
Cost: $8.95 plus postage

This is a "self-teaching" guide and is very well done. Some may prefer this self-teaching method of presentation.

CP/M Handbook with MP/M by Rodnay Zaks
Publisher: Sybex
2344 Sixth Street
Berkeley, CA 94710
Cost: $13.95 plus postage

I do not recommend Zaks' text as it contains too many errors.

After purchasing one or more of the above, you can sit down at your computer and explore the use of CP/M. There is still one remaining problem, however. CP/M has certain poorly documented flaws. Once you are aware of them, you can usually work around them without much difficulty. In many instances, you won't even notice them because you will give up trying to make a certain CP/M supplied program work as you think it should, and will erroneously have concluded that YOU were doing something incorrectly.

With the above in mind, I will try to help you get started with CP/M and give some illustrations that hopefully will aid you in exploring CP/M. I will go into some detail of certain aspects that are not fully explained elsewhere. Also, I will try to explain things that most confused me when I began using CP/M.

THE MOST IMPORTANT THING I CAN TELL YOU IS TO LEARN THE CP/M OPERATING SYSTEM **BEFORE** YOU INVESTIGATE MBASIC, ASSEMBLY LANGUAGE PROGRAMING, OR SIMPLY "PLAY GAMES". OTHERWISE YOU **WILL** WASTE MUCH TIME!!

ANOTHER VERY IMPORTANT ITEM IS TO ALWAYS USE A DUPLICATE DISK WHEN YOU ARE LEARNING **ANY** OPERATING SYSTEM. IF SOMETHING UN-

FORTUNATE HAPPENS, THEN SIMPLY COPY A NEW DISK FROM THE ORIGINAL AND START OVER.

NOTE THAT THE BEGINNER CAN ONLY LEARN THE CP/M SYSTEM (or any other operating system) BY ACTUALLY SITTING DOWN AT A COMPUTER AND TYPING IN THE VARIOUS COMMANDS, AND TRYING OUT THE VARIOUS PROGRAMS!

SINCE YOU COMMUNICATE WITH THE COMPUTER THROUGH A TYPEWRITER-LIKE KEYBOARD, YOU SHOULD BE A FAIRLY GOOD TOUCH TYPIST. THIS SAVES YOU MUCH TIME. If you need to brush up on your "typing", I recommend "Touch Typist", for CP/M systems, from Newline Software, P.O. Box 402, Littleton MA 01460. Cost $29.95. This is a program that uses your computer to teach you how to "touch type" (or sharpen your skills).

PRELIMINARY CONSIDERATIONS

The features discussed in this article apply to CP/M version 2.0 and above.

You will communicate with your computer by entering certain information and/or commands at the keyboard of your terminal, AND many times you will execute the commands by hitting the Return key on your keyboard. Throughout this article <cr> means hit the Return key. I will repeat this for emphasis. IF YOU SEE "<cr>" IN THIS ARTICLE, IT MEANS YOU HIT THE RETURN KEY ON YOUR KEYBOARD!

CP/M uses A: as name of 1st drive, B: as name of 2nd drive, C: as name of 3rd drive, etc..

Throughout, I assume you are "logged on" drive A: (the drive you booted from), and have the A> prompt showing. If you have more than one drive it is simple to log on to another drive. For example if the A> prompt is displayed, and you wish to move to a second drive (B:) simply type B:<cr>, and there you are with the B> prompt showing. To get back to A:, just type A:<cr> and you are back with the A> showing. (Note that drive B: must contain a disk, and drive A: must always contain a SYSGENed disk.)

If you try any of the examples in this article, you must have on your disk the files required for it. For example, if you are using the PIP command, you MUST have PIP.COM on your disk; if you are using SUBMIT and XSUB programs, then you MUST have SUBMIT.COM and XSUB.COM on your disk.

Many operations require the use of Control characters. They are all performed by holding down the CTRL key, and then ALSO depressing the appropriate desired "character". To do a CTRL C, for example, hold down the CTRL key, and while it is held down, also depress the C key. You may use upper or lower case "characters", incidentally.

WHENEVER YOU ARE RUNNING CP/M, AND FOR SOME REASON REMOVE A DISK FROM A DRIVE AND INSERT ANOTHER DISK, **ALWAYS** DO A "CTRL-C". THIS IS KNOWN AS A WARM BOOT, AND UNLESS YOU DO THIS AFTER SWITCHING A DISK, CP/M WILL NOT KNOW YOU HAVE CHANGED DISKS! Always do this! (Note that if you are in MBASIC always do (or have your program do) a RESET - this will accomplish the same thing.)

FILE NAME AND EXTENSION CONVENTIONS

A complete file name including extension consists of one to eight alphanumeric characters, followed by a period (.), followed by a 1 to 3 character alphanumeric extension (Example: FILENO1.TXT). No spaces are allowed. You do NOT have to include the period (.) or extension (ext.) UNLESS the file is of a special type, or you wish to distinguish it from a file of the same name. Here are some of the special file types used:

.COM	Any machine language program.
.ASM	Any assembly language program you may create using an editor.
.LIB	Any text file you create with an editor for use with ED's "R" command.
.PRN	A "listing" which the assembler creates for you if you want it.
.SUB	A file which you create with an editor for use with SUBMIT (see SUBMIT).
.BAS	This is an extension which the MBASIC interpreter puts on for you if you "SAVE" a program from MBASIC.)

Other than the "special types", most of the files that you create yourself may have, or may not have a period (.) and a 1 to 3 character extension AS YOU DESIRE. For example, here are some valid file names that you might create: THISFILE.TXT, THISFILE.DAT, THISFILE.PGM, THISFILE, FILE1, FILE2, FILE.123, FILE24. All of these files could reside on one disk at the same time, since they are ALL different, one from the other! You will usually create text files using an editor, and programs and/or files using a BASIC language interpreter.

COMMANDS AND COMMONLY USED CP/M UTILITY PROGRAMS

All of these commands and programs are executed from the monitor prompt (A>). Note that our default drive is the A: drive, and you do NOT have to explicitly type A: if the drive involved is the A: drive. If you are logged on to drive B:, then B: is the default that you need not explicitly type, but you must specify all others, including A:. There are times, however, when you do need to type the default drive name. (See below under PIP examples.)

Note also that when we refer to a "file" we usually are referring to "contents of the file", not just the name we (or someone else) has named any given file.

The CTRL-P Toggle

Although this is not a "command" it is a most useful feature of CP/M, if you have a printer! If you do a CTRL-P (hold down the Control key and then, while the Control key is held down, depress the P key) then everything carried out on the screen of your terminal is simultaneously (practically) printed on your printer (echoed)! Doing a second CTRL-P turns off the feature. This is an especially nice feature when you are learning the various CP/M commands since it gives you hard copy of what you did, especially if you made an error. You can see precisely what you typed, and have a permanent record of it. This feature also works nicely in CP/M's editor, ED (although it will not work with many editors). Just be sure and toggle CTRL P on BEFORE you invoke ED (see ED). (This "toggle" does NOT work with SUBMIT or MBASIC, but it does work with BASIC-E (type CTRL-P before you RUN a program). In HUG's editor (885-1210), it works while you are IN the editor.)

DIR

Entering DIR<cr> at the monitor prompt displays the directory of files currently on the diskette, UNLESS a particular file has been set to "SYS" using the STAT command (see STAT), in which case that file will NOT be displayed. Examples:

DIR<cr> displays files on disk in drive A: (default).

DIR B:<cr> displays files on disk in drive B:. If you have only one drive and your Heath/Zenith CP/M is configured for one drive, you will be asked to place disk B: in drive A: and hit RETURN. If you access the system again (type another command), you will be prompted to replace disk A:.

The display invoked with the DIR command simply lists the files and does NOT show the period (.) that actually exists between the file name and its extension (if there is an extension). For example, the file ED.COM is displayed as ED COM.

Note that SOME files may NOT be listed with the DIR command. (ALL files WILL be listed if you execute the STAT *.*<cr> command - see STAT below.)

STAT

This is used in several different ways and displays certain information about your disk files. STAT also allows you to change certain "file attributes". Examples:

STAT<cr> shows the amount of unused disk space on your disk. STAT B:<cr> shows the amount of unused disk space on disk in drive B:.

STAT *.*<cr> shows an alphabetical listing of ALL files on disk in drive A: (if the default is A:). Any file set to "SYS" is enclosed in parentheses. Such a file (if set to SYS) would look like (ED.COM) in the listing, assuming we had set ED.COM to SYS. Any R/O (read only) files are so noted, as are the R/W (read/write) files. STAT B:*.*<cr> does the same thing for drive B:.

STAT filename.ext $R/O<cr> will make filename.ext a "read only" file. This means that you will not be able to "write" that file. For example, if you were editing that file, you would NOT be able to write the edited version to disk using the same file name. In my illustration here "filename.ext" means the exact file name that you are setting to read only. For example, STAT B:THISFILE.TXT $R/O<cr> would set file by the name of THISFILE.TXT which is on disk in drive B: to a read only status.

STAT filename.ext $R/W<cr> is exactly the same as above, except it sets the file to a "read-write" status. Most of your files usually are of this type to begin with. So, the main use of this form of STAT command is to change a preexisting "read-only" file to a "read-write" file.

STAT filename.ext $SYS<cr> will fix the file name so that it does NOT show up when you request a listing of all files with the DIR command. Note that a file set this way can NOT be copied by PIP unless you use PIP's "R" switch which you tack on the end of PIP's command line. The R is enclosed in square brackets, like this: [R].

STAT filename.ext $DIR<cr> will restore the file so that it DOES show up in the listing of the the DIR command.

(Editor's Note: This discussion does not cover all of the uses of STAT, such as displaying disk parameters and displaying and setting I/O device assignments.)

PIP

This is the copy command. PIP means Peripheral Interchange Program.

Note that, in all uses of the PIP command, the original file (the file to be copied) is left unchanged, unless you are using PIP to concatenate several files, and the new "combined" file has same name as one of the "subfiles".

NOTE THAT THE DIRECTION OF THE COPYING IS ALWAYS FROM THE FILE INDICATED TO THE RIGHT OF THE "=", TO THE FILE NAMED ON THE LEFT OF THE "=". In other words, the com-

mand always contains an "=". The file to the left of the "=" is the file that is being CREATED, while the file to the right of the "=" is the file that is being copied. Another way of remembering this: TO = FROM! Of course, you can copy from disk in drive A: to disk in drive B: OR you can copy from disk in drive B: to disk in drive A:. Here are some examples that may clarify this for you.

PIP ABC.TXT=B:XYZ<cr> will copy the contents of file named XYZ FROM disk in drive B: TO the disk in default drive (A:) and give it the name ABC.TXT. Note again that the file on the right side of the "=" in this illustration is copied to the file named on the left side of the "=", e.g. from RIGHT TO LEFT (TO = FROM)! If you have only one drive, you will be prompted to swap disks (if you have Heath/Zenith CP/M) at the appropriate times (see DIR).

PIP B:ABC.TXT=XYZ<cr> copies the contents of a file named XYZ FROM disk in drive A: TO disk in drive B: and gives file the name ABC.TXT.

(You don't have to say "A:" in the above 2 examples, as PIP interprets the ABSENCE of any drive specification as a default, and assumes you mean drive A: which is normally the default drive.)

PIP C:DRIVE3.ABC=B:HOHOHO.XYZ<cr> will copy file HOHOHO.XYZ from disk in drive B: and place it on disk in drive C: and give the copied file the name of DRIVE3.ABC.

PIP NUNAME=OLDFILE<cr> copies file OLDFILE on disk A: (default) to a file called NUNAME on disk A:. The contents of the files are identical. Both files are on the disk after the copy command is carried out.

PIP B:=THISFILE<cr> copies a file called THISFILE from disk in drive A: (or the default drive) to disk in drive B: AND gives it same name as the original file on disk in A:, in this case THISFILE.

PIP A:=B:*.*<cr> copies ALL files (file names and the contents of the named files) from disk in drive B: to disk in drive A:.

PIP BIGFILE=FILE1,FILE2,FILE3<cr> combines (concatenates) FILE1 and FILE2 and FILE3 into ONE file named BIGFILE. In this example, all files are on disk in A:.

PIP LST:=THISFILE<cr> will print out on your printer the contents of a file named THISFILE (FROM disk in drive A: TO your "LST:" device - your printer.)

PIP also has some very useful "switches". "Switches" means that in addition to the fundamental copying that is done, certain other things can also be simultaneously carried out. They are all implemented by typing the "switch", enclosed in square brackets ([]), immediately after the desired PIP command (NO space typed), just before the <cr> which executes the PIP command. Examples:

PIP B:=THISFILE.TXT[V]<cr> will copy THISFILE.TXT from disk in drive A: (default) to disk in drive B: and give it same name, and THEN will compare the two files (V=Verify) and make sure they are identical. The switch used here is the "V" switch.

PIP THISFILE.TXT=B:THISFILE.TXT[L]<cr> copies the file named THISFILE.TXT which resides on disk in B: to a file named THISFILE.TXT on disk in A: (default) AND simultaneously changes any upper case letters in original file to lower case (The new file on drive A: will be entirely lower case). Note that this has nothing to do with the NAMES of the file being in upper or lower case. You can always enter the file names in EITHER upper or lower case when you are typing from the keyboard. Note that nothing is changed in the original file, just in the new file created.

PIP B:UPPER.TXT=LOWER.TXT[U]<cr> copies contents of file named LOWER.TXT on disk in A: to disk in B: and names it UPPER.TXT, AND also changes any lower case letters to upper case in the contents of the new file, while doing so. The contents of the new file UPPER.TXT will be entirely upper case!

PIP LST:=THISFILE[N]<cr> prints the contents of file named THISFILE on your printer and puts a line number in front of each line. Also, try the "N2" switch.

PIP LST:=THISFILE[T8]<cr> prints the contents of the file THISFILE on your printer and expands tabs to spaces. The 8 means that tabs are assumed to be at every 8 columns, which is normal. This switch MUST be used if you PIP files to a printer that cannot handle tabs.

PIP LST:=THISFILE[NT8]<cr> prints THISFILE with both the N and T switches set. Any number of switches can be combined in this way.

There are many more nice switches, or "parameters" that you may wish to investigate. (Editor's Note: I suggest that you read pages 18 through 25 in An Introduction to CP/M Features and Facilities, one of the manuals provided with CP/M, then read pages 8 and 9 in CP/M 2.0 User's Guide for CP/M 1.4 Owners. This is the only way to fully learn the power of PIP, and will introduce you to the "dreaded" Digital Research manuals. -- PS:)

ERA

This command deletes files. Note that you CAN delete a file with the SYS attribute set (see STAT), but that you can NOT erase a file with the R/O (read only) attribute set (see STAT). Examples:

ERA FINISHED.TXT<cr> will erase from disk in drive A:, a file named "FINISHED.TXT".

ERA B:*.*<cr> erases all files on disk in drive B:. This is a MOST DANGEROUS command! Because of this, you are asked if you really want to do it, and must answer Y (for Yes) to complete the command.

The "ERA" command actually removes the file name from the directory, rather than erasing the contents of the file. However, the net result is the same! The contents of the file with its file name removed from the directory is no longer accessible to the user. Note that there are ways to recover from this if you have the necessary knowledge and special programs (such as SDUMP on HUG disk 885-1213).

TYPE

Allows you to display on your terminal screen, the contents of files which contain ASCII characters (alphanumeric characters such as are on your keyboard -- files created with ED, for example). Note that files with the extension .COM are in machine language and only display "gibberish" if you try to TYPE them. (Try it and see.) Example:

TYPE EDITED.TXT<cr> will display on your screen the contents of a file named EDITED.TXT.

Note that here, as elsewhere in CP/M, you may temporarily halt the scrolling of the file by typing a CTRL S (hold down the control key and simultaneously hit the "S" key). This is actually a "toggle". Do it once and scrolling is stopped. Do it again, and scrolling starts again, etc.

REN

This command will rename a file on disk. Note that the file on the right side of the = is renamed to that on the left side of the =. Examples:

REN B:THATFILE=B:THISFILE<cr> The file named THISFILE on disk in drive B: is renamed to THATFILE. The second B: is not required (you could say REN B:THATFILE=THISFILE).

REN NEWNAME=OLDNAME<cr>. File named OLDNAME on disk in drive A: is renamed to NEWNAME.

SAVE

Most of the time the average user will not use this command. He WILL use it if he intends to write and use assembly language code. The use of this command therefore will not be covered here. It is nicely covered in Hogan's book and pages 9 and 10 of An Introduction to CP/M Features and Facilities.

ED

ED is the text editor supplied with CP/M. It is very similar to two editors offered by HUG, one for use with HDOS, and one for use with CP/M. ED is a "character" editor and uses an invisible pointer. "Screen" oriented editors are easier for beginners to use. Nevertheless, I advise getting acquainted with this CP/M supplied editor as there are some things you just can not do with many other editors. If you are familiar with HUG's ED.ABS, then you will have no trouble learning how to use this ED.COM, supplied with CP/M. If you are not familiar with HUG's ED, then I refer you to an article "Getting Started with a Text Editor" which I wrote and which is in REMark issue 11. I used HUG's ED as an example. There are some differences, however, and the following lists what you should do using CP/M's ED.COM. (I assume you have "Getting Started with a Text Editor" handy.) You invoke the editor in exactly the same way. An example:

ED THISFILE.TXT B:<cr>. This command loads ED.COM into memory, and tells ED that we are going to be editing THISFILE.TXT, and that we wish the edited file (when we are through editing) to be placed under same name on drive B:. (ED will do all this for us and when we terminate our editing session, ED will rename the original file to THISFILE.BAK and leave it for us unchanged on disk in drive A:.) Now, the B: in this example is entirely optional and if B: is left off the command line that we used to invoke ED, then ED puts the edited file back on disk in drive A:, but still renames the original file to THISFILE.BAK. This is one difference between ED and the HUG editors, which always put the output file on the same disk as the input file, unless you specify differently.

After you invoke the editor, ED responds with its prompt (*). (ONCE YOU ARE IN ED, YOU WILL SOON LEARN THAT THERE ARE 2 POSSIBLE "MODES" THAT YOU MAY BE IN -- IF YOUR CURSOR IS PRESENT JUST AFTER AN ASTERISK PROMPT (*), THIS MEANS YOU ARE IN ED'S COMMAND MODE. YOU ENTER ALL OF ED'S SINGLE LETTER COMMANDS IN THIS COMMAND MODE. THE OTHER MODE IS THE "INSERT" MODE, AND IT IS THIS MODE -- WITHOUT ED'S PROMPT (*) -- IN WHICH YOU ENTER YOUR TEXT. THIS WILL BECOME SELF EVIDENT AS YOU PROCEED!)

Our file contents (contents of "THISFILE.TXT") are NOT yet in ED's buffer memory. If there was no file named THISFILE.TXT, then ED informs us of this fact -- NEW FILE is displayed -- and we are

ready to start inserting text with the Insert command.

All ED commands should be entered in lower case. (I will explain why later.) Thus, the "insert" command should be "i" rather than "I".

To get preexisting contents of file into buffer memory you type #a<cr>. This appends (puts all of preexisting file into buffer memory) ALL into memory, if it will fit. Otherwise, it NEARLY fills available memory, then stops. (See below.)

The invisible pointer is at the beginning of the file. (Note that if you are working with an extremely LARGE file, and buffer memory is not large enough to hold the entire file, then the invisible pointer is at the END of the partially appended file, and you must put it at the beginning using the command b<cr>.) When you are finished editing this first "append", then type #w<cr> (writes buffer memory to disk), then type #a<cr> to append the next portion of the large file so you can edit it.)

The Insert command (i) in ED.COM is used by simply typing i<cr> and then you start typing the text you want to insert. You finish the insertion by typing a CTRL-Z (hold down the CTRL key and depress the z key).

A CTRL-Z is also used as a delimiter in the "search and replace" command feature.

Now, lets summarize the differences between HUG's ED (one referred to in Getting Started with a Text Editor) and CP/M's ED:

With HUG's ED you hit the ESC key one time as a delimiter, and you hit the ESC key 2 times to terminate an insertion, and hit the ESC key 2 times to execute a command (or series of commands). On the other hand, with CP/M's ED, you type a CTRL z as a delimiter, and a CTRL z to terminate an insertion, and hit the RETURN key to execute a command or series of commands. And, when you use CP/M's ED, always use lower case when you type the commands.

Other than the above differences the two Editors are used in much the same way.

Here is an example of how to create a short file using CP/M's ED - (Note that the text within square brackets [text] tells you what YOU do.)

From the monitor prompt (A>) you type:

```
ED newfile.txt<cr>      [this is how you invoke ED; material to
                        right of the space after ED is called a
                        "command line"]
                        [ED will be loaded into memory and will
                        respond NEW FILE and then its prompt (*)]
i<cr>                   [you type i and hit RETURN]
This is first line of file.<cr>
                        [you type this line and hit RETURN]
This is last line of newfile.txt.<cr>
                        [you type RETURN and a CTRL z. This terminates the insertion. Now type following:]
b#t<cr>                 [you type b#t and hit Return. b=go to beginning, #t=type all. ED responds by printing out the 2 lines we inserted. The pointer is still at the beginning.]
```

Now, type t<cr> and ED will type the first line for you. Now, just hit the RETURN key and ED will type the second line. If you had more lines in your file, just hitting the RETURN key would display each succeeding line. (The invisible POINTER also moves down one line each time you hit Return key.)

```
e<cr>                   [Now, assume we are finished with file and desire to exit. We type e for End.]
```

Congratulations, you have just written a file using ED.COM. Use DIR<cr> to note that it is indeed on disk. Use TYPE newfile.txt<cr> to see the disk contents of that file.

With the above in mind, and referring to "Getting Started with a Text Editor", I hope you are able to master ED.COM. Just learn the essential basic commands that I covered in that article, and this should give you a good start in using CP/M's ED.COM.

Why we use lower case for ED's commands

CP/M's ED as supplied has the "u" switch set to minus (-u). This means that you can insert material into a file using lower case, and also, using the search and replace command, you can insert lower case material. What is not yet well documented is the fact that if the respective commands are entered in upper case, then (even though the printed material is displayed on your terminal in lower case) the inserted material that goes into ED's memory buffer is translated into UPPER CASE...even though the "u" switch is set for lower case! So, ALWAYS USE LOWER CASE WHEN ENTERING ED'S COMMANDS!

ED has many powerful features. Here is one. You will have noticed by now that ED has an automatic line numbering feature.

(If you do NOT want this feature simply type, at ED's asterisk prompt, -v<cr>. This turns this feature off.) The automatic line numbering is a very nice feature. If you have some text in memory that you are editing, and you desire to move the invisible pointer to any given line, simply type the line number, a colon (:), and <cr>. Done!

(Note: these line numbers are NOT part of YOUR text. They are simply a feature provided by ED. If you want hard copy (copy provided by your printer) of any given text file, just do a CTRL-P BEFORE you invoke ED! Then, get your text into memory with #a<cr>, then type it all out to your screen using the #t<cr> command. You will get hard copy containing line numbers at the same time!)

Here is an example of how to move the pointer to a line number. Let's pick line number 123. From ED's prompt (*) you simply type

123:<cr>

and there you are. Just type t<cr> to see line 123. (You can, of course, combine these 2 commands as 123:t<cr> and accomplish the same thing.)

Another remarkable feature of ED, and one that is often overlooked, is its ability to include pre-existing files (pre-existing on disk) into a file that you are presently editing. The file on disk MUST have an extension of ".LIB", and it MUST reside on the same disk as the file to be edited (can NOT be on disk B: if file to be edited is on A:). The ED command that carries this out is:

rfilename<cr>

What happens is that whatever text "filename.lib" contains is inserted into whatever text you are editing, just before the current location of the invisible pointer. Here is how to try this out:

Using ED, create a file called XYZ.LIB. Just insert (i<cr>) 2 or 3 lines into the file, then CTRL Z to stop the insertion, type e<cr> to exit ED. You now have on disk a file called XYZ.LIB.

Now, go into ED with another filename given when you invoke ED, insert several lines, terminate insertion with CTRL z, go to beginning of file with b<cr>. Next, hit return key 2 or 3 times (assuming you put a half dozen lines in with the insertion), type rxyz<cr>. You have just inserted the contents of file xyz, just before the invisible pointer (which was at the beginning of the last line displayed before you typed the rxyz<cr>.) To prove this, simply type b#t<cr> and the present contents of your current file are displayed. Note that it includes the contents of xyz.lib file! Type e<cr> to exit ED.

The "h" command. This is a very nice command that can save the user many headaches! If you type h<cr> (from ED's asterisk prompt), ED writes your total file to disk, and THEN automatically opens same file and returns you to ED. When you see the asterisk prompt, just #a<cr> and your edited file is back in memory. You should make frequent use of this command when you are doing extensive editing. This way IF something untoward happens, you have the partially edited file on disk, rather than just in memory.

The "n" command. Here is another frequently overlooked command. It is somewhat similar to the "f" (find) command. You may think of "n" as meaning "next". Suppose you have a VERY long file, one that is too large to fit in memory with one append. Suppose that you desire to find some "unique word(s)" somewhere towards the middle or end of the file. You invoke ED in the usual way, but you do NOT have to use the "a" (append) command. From the asterisk prompt, type:

nunique words^z01t<cr>

Let's examine that command since there are several different commands that are all lumped together. We typed n (which is the "next" command), then immediately typed our unique word or words, then terminated the "words we are trying to find" with a CTRL-Z (hold down CTRL key and also depress Z key), then we entered 01t which is a zero and the letter 1 (which means "go to the beginning of the line"), and t for type (see below for more on "01t"). Then we hit Return key. ED will automatically append (and also write if necessary) until it finds a "match" for the unique word(s) entered. It stops, and then 01t moves the pointer to the beginning of that line, and ED types out the line for you. Now, you can do whatever editing you want in this area of the file, then simply e<cr>. There are other ways you can use this command, so experiment with it.

The "x" command is a very useful command which allows you to move "blocks" of text around. It is nicely covered in Hogan's Guide, and a good explanation is also found in Appendix A of ED: A Context Editor for the CP/M Disk System (one of the Digital Research manuals).

Last, before leaving this section on the use of ED.COM, I repeat something from the article in REM issue 11 -

If you are ever in doubt as to the location of the "invisible pointer" while you are editing a file with ED, simply type 01t<cr> (zeroLT<cr>) and the pointer will be just BEFORE the first character of the

line just displayed for you by ED.

DDT

DDT is the "debugger" which is primarily used in checking any assembly language programs you may write. I will not go into detail about this use. DDT can also be used to patch a machine language file. I will give below a patch which you SHOULD make to the CP/M file SUBMIT.COM, IF you have CP/M vers 2.202 from Heath. Please note that SUBMIT.COM in version 2.203 from Heath has ALREADY BEEN PATCHED for you and you do NOT have to make the following patch. It is not difficult to make the patch to vers 2.202. The material in square brackets to the right of the actual "patch" are not a part of the "patch" but merely some of my notes to help you understand what is going on. This "patch" fixes SUBMIT.COM so that it will understand and act upon a CTRL-Z. The CTRL-Z would be entered as an up arrow followed by a Z (^Z). You may have use for this if you are using SUBMIT when doing some editing. Do NOT patch your distribution disk. Copy SUBMIT.COM and DDT.COM to another "bootup" disk and carry out the following procedure. This patch was obtained from Digital Research. What you type is shown in bold print. From the A> prompt:

```
DDT SUBMIT.COM<cr>          [you type DDT SUB-
                            MIT.COM and hit the
                            RETURN key]
DDT VERS 2.2                [DDT displays these
NEXT  PC                    three lines and its
0600 0100                   prompt (-).]
-L441<cr>                   [You type L441 and hit
 0441  SUI  61              RETURN.  DDT will dis-
 0443  STA  0E7D            play these 10 lines.
 0446  MOV  C,A             If the number to the
 0447  MVI  A,19            right of SUI is 41,
 0449  CMP  C               then the patch has
 044A  JNC  0456            already been made.  Do
 044D  LXI  B,019D          a CTRL-C to exit.  If
 0450  CALL 02A7            your screen does not
 0453  JMP  045E            match these lines, do
 0456  LDA  0E7D            not continue.]
-S442<cr>
 0442  61 41<cr>
 0443  32 .<cr>             [Type a period, <cr>.]
-G0                         [Letter G and zero.]
```

A>**SAVE 5 SUBMIT.COM**<cr>

Congratulations! You have just patched a machine language file. What went on in the above was that you used DDT's L (LIST) command to type out 11 lines beginning with memory location 441. After checking listing you used DDT's S (SHOW) command to display contents of memory at HEX address 442. It showed a 61. Then you changed it to 41, ended the S command with a period (.) and used the "G0" command to exit DDT. You simply changed one byte in SUBMIT.COM in memory. Next, you used the SAVE command to write the program from memory to disk (the "5" in the SAVE command is a decimal number indicating that there were 5 256-byte "pages" saved. Done!

SUBMIT

Note: When you use the SUBMIT and the XSUB programs, be sure you are logged in on the A: drive, and ensure that SUBMIT.COM, XSUB.COM, (ED.COM if your procedure involves ED editor and any .LIB files), and any .SUB files ALL are present on the disk in drive A:. Any other utility programs such as PIP.COM should also be on this disk if your .SUB file references them!

SUBMIT is CP/M's version of a "batch processor". What this means is that you will be able to put together (using an editor) a list of commands that you might wish to carry out on more than one occasion. For example, suppose that one daily routine at present consists of booting up CP/M, loading MBASIC and running a program called THIS.BAS. After you are finished with this program you desire to copy a file called IMP.DAT from your "boot disk" in drive A: to a backup disk in B:, then you wish to have the DIR of B: displayed. You simply create (using an editor) the list of commands that you would ordinarily carry out. And, the file that you create MUST have an extension called ".SUB". Let us say that you desire to call this ".SUB" file ABCD.SUB. Using ED, you would (from the monitor prompt A>) type ED ABCD.SUB. Then, when ED gave you its asterisk (*) prompt, you would type i<cr> and then type the following commands:

```
MBASIC THIS<cr>
PIP B:=IMP.DAT<cr>
DIR B:
```
Now you type a CTRL-Z, type, e<cr> and ED puts on disk a file called ABCD.SUB.

Here is how you use SUBMIT with this illustration. From the monitor prompt (A>) type:

SUBMIT ABCD<cr>

SUBMIT will execute your first command which loads MBASIC and runs your program called THIS.BAS. If the last logical statement in your program is SYSTEM (or if program ends, and you type SYSTEM<cr> to exit MBASIC), you can just sit back and watch SUBMIT automatically copy your file and then display the DIR of disk in drive B:. Now, this was an extremely simple illustration. SUBMIT is capable of handling many more commands. All you have to do is have them in a file created by an editor, and remember the file name must have the extension ".SUB". Also note that ALL COMMANDS GIVEN IN THE FILE CREATED BY ED WERE IN **UPPER CASE.** This is really NOT necessary as SUBMIT converts all lower case commands to upper case, but serves as a reminder to user of the following peculiarity of SUBMIT. Not only are your com-

mands converted to upper case, but also, if you are doing some editing with ED using SUBMIT, and are using the "F" (find) command, or the "S" (search and replace) command, SUBMIT also translates the "string to be searched for" to upper case, even though you entered it as lower case. Hence, the "search" will fail! You can NOT use some features of ED, with SUBMIT, unless you are aware of this "flaw". REMEMBER, SUBMIT will translate all entries into UPPER CASE with ED. (SUBMIT will copy files using PIP, however, WITHOUT any lower case conversion.) If you wished to have ED edit a file and wish to insert one or more short lines, here is how you do it (I am indebted to Curt Geske of Digital Research for this information). You do NOT use a <cr> after the Insert command (i), and you use a CTRL-L (entered as ^L) INSTEAD OF a <cr> after each short line, and you use a CTRL-Z to stop the insertion. All of this is entered on one line. Suppose I wished to add 3 short lines to a preexisting text file. Here is what I might have in my .SUB file:

```
XSUB
ED THISFILE
#A
2L
ITHIS IS ADDED^LTHIS TOO^LAND THIS^L^Z
E
```

Each CTRL-L (entered as ^L) in the above file, is interpreted by ED as a carriage return/line feed. The CTRL-Z (entered as ^Z) ends the insertion.

If the above was in a file named XYZ.SUB, and there was a preexisting file named THISFILE, and from the monitor prompt we entered

```
SUBMIT XYZ<cr>
```

then SUBMIT would insert 3 lines into THISFILE, just above the third line of the original contents of THISFILE. Try it!

Please note, however, that the insertion (if entered as lower case in the above .SUB file) would still be translated into upper case by SUBMIT. There just is NO way at present of inserting lower case material using ED with SUBMIT! Sorry!

You can also insert material as follows. You can insert a ".LIB" file into a program being edited by SUBMIT using the "R" command, BUT if the .LIB file contained lower case material, SUBMIT would translate it to UPPER CASE. Sorry again, but that is just the way SUBMIT works.

Here are 2 items that can NOT be in the list of commands/programs that you put into a ".SUB" file. You can NOT put in a CTRL C (even though entered as ^C). You can NOT put a <cr> on a line by itself (to use just a <cr> as a "keyboard response" when using XSUB -- see below)! You can, however, put comments in a SUBMIT file. Just introduce the comment line with a semicolon (;) in the first column, as in this example:

```
XSUB
;THIS LINE IS A COMMENT
ED TESTFILE
ITHIS LINE IS INSERTED^Z
E
```

Comments should not be used within ED or other programs, but can be used any time the command is for CP/M itself, because it is the CCP (Console Command Processor) in CP/M that allows this, and it has nothing to do with SUBMIT. You can even put escape sequences to control your terminal (such as ESC-E to clear the screen) in comment lines as long as the argument is upper case.

Note that if you desire to use SUBMIT with any program that ordinarily requires you to do something from the keyboard, and you wished to put the entries into your .SUB file, then the first entry in your ".SUB" file MUST be XSUB. (XSUB is another CP/M program that allows you to put the desired "keyboard responses", in advance, into your .SUB file.) SUBMIT works with DDT, and with your assembler (ASM.COM). It does NOT work with some advanced word processors, and it does NOT allow you to put "keyboard actions" into your .SUB file when you are using MBASIC. (Example: Your MBASIC program stops and asks you for the date. You could NOT put the date in the .SUB file. MBASIC requires your personal attention to such matters.) Be sure to remember to "patch" SUBMIT to allow CTRL Z action (see above under DDT) if you have CP/M vers 2.202 as mentioned above. Also remember that if you are using ED with SUBMIT, it really doesn't help you UNLESS your file to be edited is all UPPER CASE! Incidentally, you enter a CTRL-Z as ^Z (up arrow z) in your .SUB file if you need to enter it for LATER execution.

Finally, SUBMIT has an extremely handy feature. Suppose that you did not know in advance, certain items such as "filename", which drive you would be using, etc. SUBMIT allows you to use variables in your .SUB file - $1, $2, $3, $4, etc., if needed. You put these into your .SUB file. In our example given above, your file ABCD.SUB could look like this:

```
MBASIC THIS
PIP $1=$2
DIR $3
```

With this example you might enter the following command line to start SUBMIT:

```
SUBMIT ABCD B: IMP.DAT A:<cr>
```

Note the spaces separating the 3 items

AFTER the name of the .SUB file. What happens is that SUBMIT plugs in the first item (B:) where $1 was in the ABCD.SUB file, substitutes the second item for $2 and the third item for $3. So, when SUBMIT plays out this scenario, what happens is:

```
MBASIC THIS
PIP B:=IMP.DAT
DIR A:
```

Note also that you could have $1, and/or $2, and/or $3 in more than one of the commands in any given .SUB file.

MBASIC

I will cover MBASIC vers 5.2 in another article. I do wish to mention here (and will repeat this in that article) the only "flaw" in MBASIC vers 5.2 that I know of. When you "SAVE" a program from MBASIC to disk, using CP/M, always use UPPER CASE for the file name. For example -

SAVE "MYPGM<cr> and not SAVE "mypgm<cr>.

What happens is that the program IS saved and the program name IS put into the directory, if the program name was entered in lower case, BUT the program name is also put into the directory IN LOWER CASE. Microsoft apparently forgot to "mask" the lower case to UPPER CASE before sending it to the DIRectory. If your program name was saved in lower case, a DIR<cr> listing would show it as mypgm.BAS.

A listing such as mypgm.BAS can NOT be erased from the directory using ERA mypgm.BAS<cr>. (You CAN delete it from the directory by going into MBASIC and entering: KILL "mypgm.BAS"<cr>.)

Also, you would only be able to load this program into memory by entering - LOAD "mypgm<cr>. The MBASIC command LOAD "MYPGM<cr> would fail.

Of course, you can easily work around this potential problem. All you have to do is to ALWAYS "save" your programs using UPPER CASE!

Also, ALWAYS enter filename in UPPER CASE when you use the LOAD command! If the file exists in the DIRectory in UPPER CASE, and you enter filename in lower case in LOAD command, then you get error (file not found.)

The same problem is present when you "Open" files. ALWAYS use UPPER CASE when entering file names!

Now, the good points of MBASIC vers 5.2 are too numerous to mention here. Suffice it to say that "random file" handling is much easier with this version of MBASIC than in HDOS's version of MBASIC. I will cover all this in the next article.

Last, I would like to thank all the folks at HUG, and Barry Watzman, who have "proofed" this article and corrected errors in context. Any remaining errors are entirely mine.

EOF

Local HUG News

Ted Benglen, II announces the formation of a Northern Colorado Heath Users' Group, FT.HUG (Fort HUG). They will meet at least once a month in the Ft. Collins area, and serve users in the Loveland, Greeley, Longmont and Boulder areas. Anyone wishing additional information may contact Ted Benglen, II at 822 E. County Road 30, Ft. Collins, CO 80525. (303) 669-4116

The Naval Postgraduate School Hobby Computer Club would like to invite all Heath users in the Monterey area to attend one of their meetings. They have regular meetings on the first Thursday of each month at 7:00 p.m. in Room 100, Spanagel Hall, Naval Postgraduate School, Monterey CA. There is an informal Heath Users' Group there with members of varied backgrounds. For addition information Heath users may contact: Tom McNair, NPS Hobby Computer Club, Rec. Services, NPS, Monterey, CA 93940.

The PNHUG (Pacific-Northwest HUG) was inadvertently left off the list of local HUG groups in Issue 25 of REMark. They meet the second Tuesday of odd months from 7:00-9:00 p.m. at the Tukwila, WA Heathkit Center and even months they meet the first Monday from 7:00-9:00 p.m. at the Seattle, WA Heathkit Center. Their contact person is Nathan Hall at 10553 41st Pl. NE, Seattle WA 98125 phone: (206) 363-3927. Mailing address for the club is: PNHUG (Pacific-Northwest HUG), c/o Jan Johnson, PO Box 993, Bellevue WA 98009. They also have a 24 hour Bulletin Board at the Tukwila Heathkit Center (206) 246-4468. Presently there are 150 in their group.

Heath Related Products

The nucleus of a clock/calendar PLUS sound board is available for $22 postpaid and can be fully populated for about $30 more. This includes a professionally fabricated printed-circuit board, construction information (including theory and programming data), and listings of programs for reading and writing time/date and using the AY3-8910 sound generator. For an additinal $5 these programs are available on 5" HDOS disk. All parts can be easily ordered by phone from several sources, but most are available locally. The board can be installed on either side of the H89. When installed on the left side a number of signals have to be "stolen" from some right-side board, but hooking it up is straightforward. By the way, I offer a no-quibbling money back guarantee: if buyers give it their best shot and it won't work, I'll buy it back with no questions asked.

Another nifty device I'm offering is an "Auto-booter". This is a small circuit board that installs under the keyboard of a H/Z89 and waits for the power to be turned on. Five or six seconds after power-up, it electronically "pushes" the "B" and the "C/R" keys to begin the boot process. I developed it so that I could turn on the H89 with a BSR X-10 timer, have the machine boot itself and run automatically. The possibilites are even more intriguing since BSR recently introduced a phone-activated controller.

With a clock/calendar board and a suitable prologue, the H89 can check the time and date and know what program to run after waking up. I expect to have it print daily schedules each AM and use its own BSR controller to do other chores.

The one drawback is the small risk of destroying disk files by having a disk in the machine at power-up and down. My own experience has convinced me that this is more myth than reality, but I always have a backup of any disk that I auto-boot with.

The complete Auto-Booter kit sells for $22 postpaid, and can be assembled in an hour.

All of the above can be ordered from:

Ray Albrektson
619 W. Dover
San Bernardino, CA 92407

Send a self addressed stamped envelope for data sheet.

E&H SYSTEMS has developed a product for your H-27 you may be interested in. This product, called "THE 2K KEY", allows the H-11 system to be enhanced by enabling up to 30K words (instead of 28K words) of memory on the LSI-11 bus. This is a gain of 4000 bytes.

This product greatly improves the capabilities of the H-11 to handle larger programs and arrays greatly improving your system usefulness. Here is a brief explanation of the "2K KEY".

THE 2K KEY (new hardware bootstrap for H-27 floppy disk drive)

FEATURES:
Allows enabling of 2K WORDS MORE MEMORY to 30K AUTOMATIC BOOT for most controllers. Takes only 10 MINUTES to install.

SUMMARY DESCRIPTION:
The 'key' to gaining 2000 words (4000 bytes) more memory from a system that had an H-27 disk is to eliminate the existing hardware boot and get another that uses less memory. This hardware allows you to enable 2K more memory (30K words rather than 28K words). This allows you to run programs that wouldn't fit before or use much larger arrays in your programs. You also get a boot program designed to handle RL01'S - RK05'S - DY - and DX devices on the LSI-11 bus. It will automatically search for controllers for the various disks (in a predefined order) and boot the first such device which is found and is operable.

SYSTEM COMPATIBILITY:
Since this bootstrap program is based on the newer DEC standard of having a 256 word hardware bootstrap starting at address 173000, the new bootstrap should be compatible with DEC standards and hardware in every way. It will operate correctly on the LSI-11 Q-BUS, H-11, H-11A, PDP-11/03, PDP-11/23, LSI-11, LSI-11/2, and LSI-11/23 processors.

This product is also software compatible. HT-11, RT-11, and UCSD Pascal operating systems have been tested. Any other operating systems that operate on the Q-BUS should also work.

COST:	ORDER FROM:
$100.00	E&H SYSTEMS
MONEY ORDER - COD	1119 S. Gaylord
$3.00 handling	Denver, CO 80210

WARRANTY:
All parts are warranted for quality and workmanship for a period of 90 days from purchase. If a part should prove to be defective, return within 90 days for a replacement. No other warranty is made or implied.

New HUG Software

885-1053 H11/H19 Support Package $20.00

This H11 disk was announced in Issue 23, December 1981, of REMark. At that time it was announced as the TSTE Modem Package. The disk was released with these additional programs and modification:

EXEC -- This is the latest version of the TSTE modem package which was announced earlier. EXEC is identical to TSTE with the additional feature of containing the CompuServe EXECutive (CSEXEC) protocol. The modem package for the H-11 system requires the Heath Serial I/O Board, H-11-5, and will operate on HT-11 or RT-11 operating systems with a normal telephone modem and console device. EXEC uses a series of control functions to direct the operation. Some options include opening and closing the printer port, transmitting, receiving and closing of files and sending a "BREAK" to the host. EXEC must be reassembled for use with the HT-11 system and the source code is included.

DCOPY -- A disk copy utility. The program will copy between any two of the following drives: DX0, DX1, DX2, and DX3.

BOOTUP -- This version of HT-11 BOOTUP will initialize the system to be cold-started by someone not familiar with the HT-11 system. The routine will prompt for the date if a valid date is not present. It then prompts the user to select either the KMON or BASIC. If BASIC is chosen, BOOTUP causes the BASIC program MENU.BAS to be loaded and run. This gives the HT-11 system a "turn-key" operation.

MENU -- This BASIC menu program uses alphabet characters rather than numbers for program selection. It demonstrates a systematic method for doing ON ... GO TO X,Y,Z... MENU must be modified to run the selections chosen by the user.

SCREEN -- This FORTRAN routine provides a method of initializing ESCAPE codes for screen control on the H-19 terminal. The initialization routine can be limited to only 12 lines of a user program. Filed on a disk, it can be read in with EDIT. Only three TYPE FORMATs are referenced throughout any program for all ESCAPE sequences.

USRLIB -- This library consists of useful subroutines which enhance the capabilites of FORTRAN IV. The routines are written in Assembly and FORTRAN IV. The subroutines will return the date in several formats, convert integer to byte and vice versa, report the cursor position, set, test and clear a bit in a given byte variable, provide a flexible alternative to PAUSE, plus many others.

PAGER -- This FORTRAN program will print any serial ASCII file to a line printer with appropriate pagination based on selected paper length. Any page range may be selected for printing. PAGER is ideal for listing programs and is self prompting.

ESCAPE -- This BASIC routine will extract from an ESCAPE.DAT file the control codes the user wants. It will append line numbers, dimension the string array that is to contain them, insert indices as required and dump the new result into a file. The file may be expanded into a complete program or overlaid into a pre-existing file.

885-1116 HDOS Z80 Debugging Tool $ 20.00

ALDT is a highly sophisticated debugging tool for HDOS 2.0 users. It resides in high memory and appears to the user program as part of HDOS, so that the program can perform all normal HDOS functions, including the loading of device drivers. It gives the programmer full control of all ports, memory locations, and Z80 registers. It includes a built-in Z80 disassembler (Zilog mnemonics). It allows the user to execute his program in different modes, including Trace, and allows execution of loops a specified number of times. Besides the usual debugging functions, ALDT also lets the user check the operating system status, including device drivers, memory usage, and file channels. A help feature lists all available commands while you are running ALDT. It can operate in either the hex or octal/split octal bases.

This program requires HDOS 2.0 and an H8 or H89 computer and at least 48k of RAM (to allow sufficient debugging space). It will run on an 8080 or Z80 processor, and will lock out the Z80 registers if you run it on an 8080 (but it still uses Zilog mnemonics).

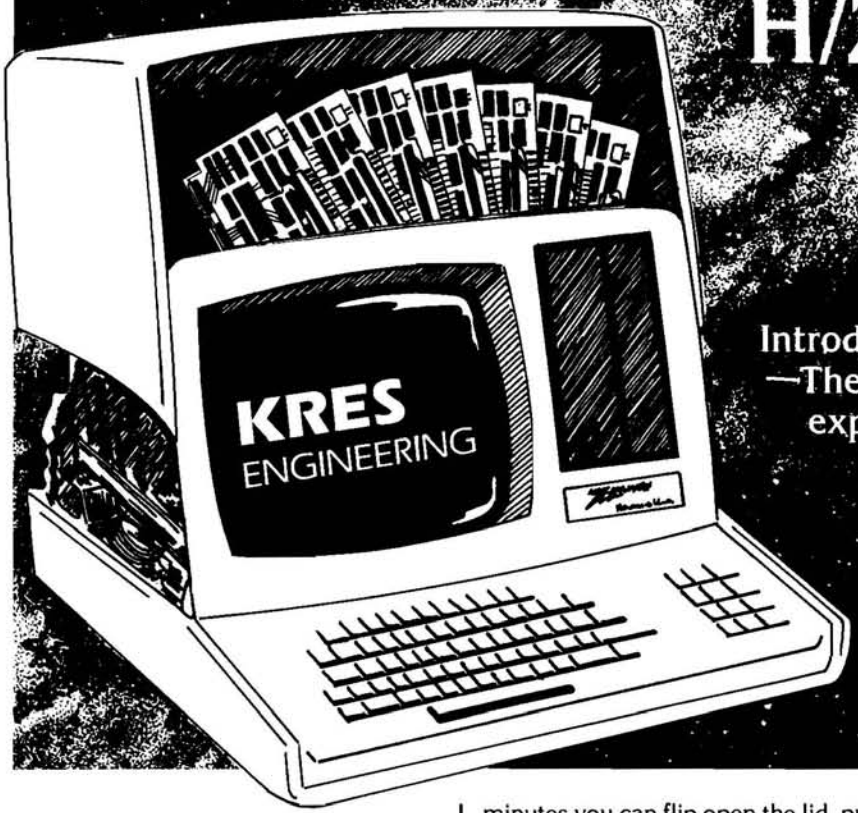
ALDT-RDT Comparison

HUG now offers two debugging programs for HDOS, RDT (885-1092) and ALDT. ALDT is more sophisticated than RDT in the way it positions itself in memory and in its debugging capabilities, and has better breakpoint control. The disassembler prints out SCALLs by name. It has all of the features of RDT except the PRINTER, ASCII, COMPARE, ADD/SUBTRACT, INPUT mode, and MOVE commands. It requires over twice as much memory as RDT, but can be reassembled to leave out some features to use less memory. It is designed for use with HDOS 2.0 only. Documentation is supplied on the disk.

RDT is designed for 8080 debugging, and cannot control the extra Z80 registers or disassemble the extra Z80 instructions. It has all of the features of ALDT except the system status commands, with less

Vectored to 17

KRES Engineering Just Conquered Space in the H/Z-89



Introducing the KRES Expansion™ —The one board that lets you expand the H/Z-89—while keeping it All In One.

MAKING A GOOD COMPUTER BETTER

You bought your H/Z-89 because of its incredible reliability and versatility. You probably already have, or have considered buying, a serial interface, parallel board, floppy controller, hard disk controller, real time clock, sound generator, A/D and D/A conversion card, and possibly a few specialized boards.

The problem is: where do you put them all? The original H/Z-89 only has three right hand slots. Chances are strong, if you're really using your computer, you're already out of slots—or you will be soon.

INTRODUCING THE KRES Expansion™

Soon you'll be able to plug in the 7-slot KRES Expansion™ board. It's simple: no soldering; no trace cutting. Within minutes you can flip open the lid, pull out your existing boards, pop in the KRES Expansion™ reinsert your boards—including the ones you've had to keep on the sidelines—and be back computing within minutes.

EXPANDED PORT DECODING

To allow the utmost versatility, the KRES Expansion™ offers expanded port decoding on all four added slots. Another benefit is the exclusive KRES board enable for Shadow Operation™

QUIET, FAST OPERATION

The 7 card KRES Expansion™ uses the latest 4-layer technology, allowing electrically quiet operation, even at higher speeds.

ROOM FOR 16K MEMORY

There are sockets for 16K of memory built right into the KRES Expansion™ So you won't need to buy a separate board to upgrade your H/Z-89 to 64K. This alone could save you about $100.

MORE TO COME

All KRES Expansion™ slots can accept any board designed to operate in one of the H/Z-89 original "spare" right hand slots. And if a board you want is not already available, let us know what you want. Because in the months to come, KRES will be announcing an entire series of boards custom designed to fit your H/Z-89.

GET MORE INFORMATION TODAY

The KRES Expansion™ will be available soon. Write for details today or call (714) 559-1047 or (213) 957-6322. Be among the first to conquer space in your H/Z-89. It was never easier!

KRES
ENGINEERING
P.O. Box 17328, Irvine, CA 92713
(714) 559-1047 or (213) 957-6322

Introducing –

THE NEW H8* COMPUTER

* H8 is a Registered Trademark of the Heath Company

PC Board Mounts Additional Pair of Connectors to Plug Into 90 Pin Bus

Optional 40 Pin Connector Assembly Converts H8 Boards to 90 Pin Bus

Fully Assembled Motherboard – $250.00

Featuring the TRIONYX T-H90 MOTHERBOARD
A Professional-Quality Bus for the H8 Computer

- 3-Layer Board Has Center Ground Plane
- Designed For 4 MHZ Bus Operation
- 7 Auxillary Positions For Port Addressable Bus Interface Cards
- Completely Compatable With Original H8 Computer

- 90 Pin Bus Has:
  - Additional Ground Connections For Reliable Operation
  - 8 Additional Data Bits For 16 Bit CPU Board
  - Additional Address Lines For Expanded Memory Capacity
  - Bank Select Lines For Memory Management
  - Additional Lines For Special Control Signals and Future Defined Functions

PC Board $ 75.00

25 Pin Gold Connectors - Set of 20 $45.00
- Build Motherboard to Original H8 Standard

20 Pin Gold Connectors - Set of 18 $34.00
- Add Additional 40 Pins for 90 Pin Bus

25 Pin Gold Connectors - Set of 14 $35.00
- Add 7 Auxillary Card Positions

Power Supply Parts for Motherboard $16.00
- Includes Extra +18 Volt Filter Capacitor

Bus Termination Card - Complete Kit $29.50

PC Board Connector Expansion Kit $15.00
- Adapts Original H8 PC Boards to 90 Pin Bus: Contains PC Board, Two 20 Pin Connectors, Mounting Blocks and Hardware - One Kit Required Per PC Board

The new T-H90 Motherboard has been designed to provide completely reliable operation of the H8 computer through the use of gold-plated connectors and a well grounded bus. The bus has been expanded from 50 to 90 lines and 7 additional card slots have been added. Full implementation of the Motherboard functions will transform the H8 into a commercial grade computer.

Check • Money Order • VISA • MASTERCARD • C.O.D.

Phone Orders Welcome (714) 830-2092 – Send For Free Brochure

TRIONYX ELECTRONICS, INC.

P.O. BOX 5131, SANTA ANA, CA 92704

DG IS Heath/Zenith

...with the F

H8 PRODUCTS

The Most Extensive Line of Hardware Support for The H8®

***DG-80/FP8**
Z80® based CPU including the powerful FP8 monitor—Both only $249.00. The acclaimed FP8 monitor package is now included with the DG-80 CPU!

***DG-64D/64K RAM Board**
Reliable, Low Power, High Capacity Bank-selectable RAM
Priced from $333.00 (0K) to $399.00 (64K)

***DG-64D5 64K/5 volt only RAM Board**
SUPER low power, Reliable, High Capacity, Bank Selectable RAM from $333.00 (0K) to $499.00 (64K)

***DG-32D/32K RAM Board**
Low cost, Dependable RAM for the H8 32K Version Only $179.00.

***DG-ADP4**
H17-4 MHz disk adaptor—$19.95

64K STATIC RAM–PROM BOARD

IF YOU STILL BELIEVE YOU WOULD RATHER USE STATIC MEMORY...OR IF YOU NEED A PROM/EPROM BOARD...
OR IF YOU NEED MORE ROOM ON YOUR H8 MOTHER BOARD...
OR IF YOU HAVE BEEN WAITING FOR STATIC MEMORY FOR THE H8 TO BECOME REASONABLY PRICED.
WE HAVE THE ANSWER—THE STATIC 64 RAM BOARD

FULLY STATIC.
SUPER LOW POWER—Power dissipation typically less than 4 watts (less than any memory board available for the Heath H8). Uses Single Supply 5-Volt memory devices.
CAPACITY—Up to 64K RAM OR Up to 64K EPROM (type 2716/2516) OR any combination of RAM and EPROM up to 64K.
SPEED—4MHz with NO wait states.
Compatible with the OLIVER ADVANCED ENGINEERING PROM Programmer Model #PP-2716.
Bank-select fully compatible with the DG-64D—Hardware selectable in 8K increments/software selectable in 16K increments.
Bank-select port addressable to any of the available 256 I/O ports using solderless jumpers.
On-board ROM-disable port for use with the DG-FP8 Monitor Package
Fully assembled and tested. Priced from $299.00 (0K) to $599.00 (64K). 16K Chip Sets: $125.00.

H8/H89 Software

Software Support for HDOS 2.0

*** Disk Management Utility Package**
Includes "Universal Dump", Intelligent Disk Dump Utility; "Universal Dup", Disk Copy Utility; and "BAD", Bad Disk Recovery Utility. $39.95 (Source: add $40.00)

*** Archive**
Space saving diskette back-up utility. For use with 5¼" or 8" disk systems. $39.95 (Source: add $40.00)

*** SYSCMD/plus**
A must for the serious HDOS user. Enhanced HDOS 2.0 system command processor provides extended commands and capabilities. $39.95 (Source: add $30.00)

*** Preload**
Support utility for systems using multiple, bank selectable memory boards. $29.95 (Source: add $20.00)

*** Advanced H17/H77 Driver**
Software driver for operation of double-sided and/or double-track double density drives with the H-17 and H-77 5¼" disk systems. Low cost alternative to high priced double density disk controllers. $19.95 (Source included).

Computer Support
uture Built-In

NEW SUPER 89

TURN YOUR H/Z-89 INTO A PROFESSIONAL QUALITY COMPUTER
HIGHER RELIABILITY NOW — MORE EXPANDABILITY FOR THE FUTURE

FEATURES	DG-SUPER 89	H/Z 89
USER MEMORY The DG Super 89 comes standard with 64K of user RAM "on-board". This configuration can be increased up to 256K of bank selectable/write protectable RAM.	64K-256K	16K-64K
Ø ORIGIN MODIFICATION No further modification, such as required on the H/Z-89, is necessary to operate in either a standard Heath/Zenith CP/M or HDOS 2.0 Operating System environment.	NOT NECESSARY	ADD-ON
CP/M-HDOS COMPATIBLE	YES	REQUIRES MODIFICATION
PERIPHERAL EXPANSION SLOTS DG Electronics has made provision in the design of the unit not only for compatibility with the standard factory expansion slots, but also for future expansion by doubling the number of available expansion slots on the unit to 6 instead of the standard 3.	6	3
ON-BOARD AMD9511 For those users who perform large amounts of arithmetic computations the DG Super 89 has provision on-board for use of the AMD 9511 arithmetic processor.	YES (PURCHASE SEPARATELY)	NO
CPU CLOCK FREQ. The CPU in the DG Super 89 operates at twice the speed of the standard H/Z-89.	4MHz+	2.048MHz
MULTI-USER CAPABILITY With up to 256K of bank selectable RAM on board the DG Super 89 offers the option of MULTI-USER CONFIGURATIONS of up to 4 users.	YES	NO
ENHANCED MONITOR DG Electronics has developed its own firmware monitor to allow the user greater flexibility and easier access to the advanced capabilities of the Z80 CPU.	YES	NO
REAL TIME CLOCK The DG Super 89 comes standard with an on-board real time clock.	YES	NO
PARITY CHECK ON RAM For those who are sticklers for accuracy, the DG Super 89 has parity check to make the user aware of errors occuring in the RAM during use.	YES	NO
SERIAL PORTS ON BOARD The DG Super 89 offers an additional serial I/O port for greater convenience and flexibility.	2	1

Now you can have all of the features in your H/Z-89 that you have always wanted. High speed and greater expandability are only the beginning of what our NEW DG SUPER 89 has to offer. DG Electronic Developments Co. has given the "89" capabilities of fast number crunching and data verification through parity. We have incorporated into the DG SUPER 89 such necessary items as 64K of user RAM, a powerful Keyboard monitor, and CP/M compatibility, items others require you to "add-on". Add to these features an extra serial port, a realtime clock, three more peripheral expansion slots, and multi-user capability and you have the computer that you really wanted to begin with; for a lot less than you would think.

Compatible with all currently available Heath/Zenith hardware devices.

D·G ELECTRONIC DEVELOPMENTS CO.

Ordering Information: Products listed available from DG Electronic Developments Co., 700 South Armstrong, Denison, Tx. 75020. Check, Money Order, VISA or MasterCard accepted. Phone orders (charge only) call (214) 465-7805. Freight prepaid. Allow 3 weeks for personal checks to clear. Texas residents add 5%. Foreign orders add 30%. Prices subject to change without notice.

Z99-G

(Shown with the optional QT cover and single sheet feeder.)

The Z99-G dot matrix printer features serial and parallel interface, 9-wire print-head for true descenders, dot-addressable graphics for plotting or creating your own character or graphic sets. 80, 96 and 132 column printing with serif style fonts and expanded print modes. High quality correspondence printouts using an 11 x 9 matrix is software selectable. Bi-directional and 100 cps printing with tractor feed and friction feed are standard! $685.

Software

Our software packages can be found at many of the Heathkit stores in North America. Programs vary from educational to personal finance to pure entertainment. Keep track of your personal or business expenses with **MONEY$WORTH** or explore one of the **CAVERNS OF THE DOOMED** (can be used with a Votrax Type 'N Talk). **PROBE** into your own personal database with our ever popular HDOS or CP/M based program. You can even run a remote terminal or operate a bulletin board system with the help of **THE PARALLELER.** And, for Microsoft basic enthusiasts, try one of our three varieties of cross-reference programs to make de-bugging and alterations of programs much easier. Keeping track of periodicals and magazine articles with **KEY-IT!** is a snap. Conserve disk storage space with **COMPACTA.** And we carry several excellent word processing systems that are custom tailored for Heath/Zenith computers.

HOUSEMASTER

This multi-function programmable peripheral has voice recognition, dual sound synthesizers, real-time clock calendar, BSR home controller interface, and optional dual RS-232 serial ports, and ROM or phoneme based voice synthesis for verbal responses! Designed for the H89, this board can control your home's electrical appliances automatically, by voice commands or manually. The sound sysnthesizers provide an unlimited number of sound effects such as lasers, chimes, gun shots, explosions and even the sound of a computer 'thinking'. The voice recognition circuitry will allow you to control functions with verbal commands! Complete software for either CP/M or HDOS is provided with a 90 page manual. Available assembled and tested or in kit form. Call or write for complete details.

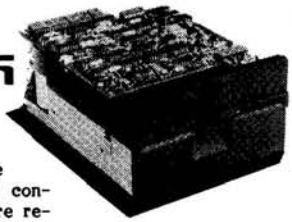
Keyboard Studio inc.

125 Aspen, Birmingham, MI 48009

Call or write for current prices or our free catalog with full details on all products.

(313)-645-5365

Tandon



Tandon TM-100-4 double-sided 96 TPI 5¼" drives can be used with your standard hard-sector or double-density controller. TM-100-1 drives are single-sided 48 TPI, and are replacements for the H77/H17. THINLINE 8" double-sided drives, single or dual-drive enclosures with power supplies also available.

LLL SERIES

Low cost single-density, 8 inch double-sided controllers for the H89 or Z80 based H8 computers. Run industry standard IBM 3740 format. Assembled and tested with necessary CP/M software. HDOS driver is also available.

ZENITH

We carry most of the Zenith line of hardware and software at reduced prices. Products such as Z90 computers, 16K expansion boards, Z25 printers, double-density controllers and Z19 terminals. Our Heath/Zenith software includes the CP/M operating system, Mbasic, Fortran, Pascal, Magic Wand, Supersort, DataStar, SuperCalc, Cbasic, Wordstar, Cobol, Basic compilers, Peachtree accounting packages and much more!

Heathkit/Zenith Educational Systems

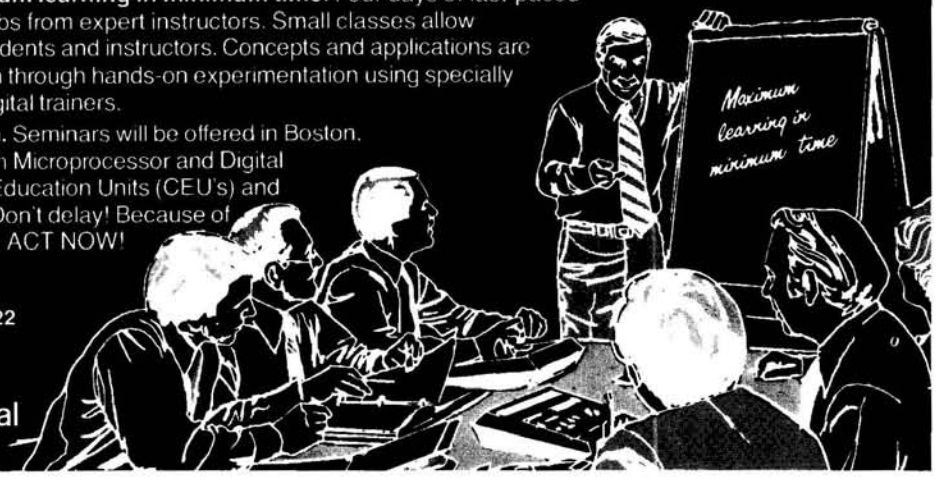
Seminars in state-of-the-art electronics

Heathkit/Zenith introduces challenging seminars in digital technology and microprocessors. Fast-track seminars for professionals who need updating to advance or meet new responsibilities. In today's fast moving world, the half-life of electronics knowledge is only 5 years. If the rapid advances in technology are leaving you behind, catch up quickly and completely in one of these four-day, interactive, workshop seminars.

Intensive seminars for maximum learning in minimum time. Four days of fast-paced lectures, demonstrations, and labs from expert instructors. Small classes allow personal interaction between students and instructors. Concepts and applications are reinforced for maximum retention through hands-on experimentation using specially designed microprocessor and digital trainers.

Send for complete information. Seminars will be offered in Boston, Chicago and San Francisco. Both Microprocessor and Digital seminars qualify for Continuing Education Units (CEU's) and offer potential for college credit. Don't delay! Because of small size, classes will fill quickly. ACT NOW!

Write: HEATH CO.
SEMINARS
Benton Harbor, Mich. 49022
ATTN: Bruce Capes



Heathkit
ZENITH Educational Systems

SMASH THE '89's 64K RAM LIMIT!

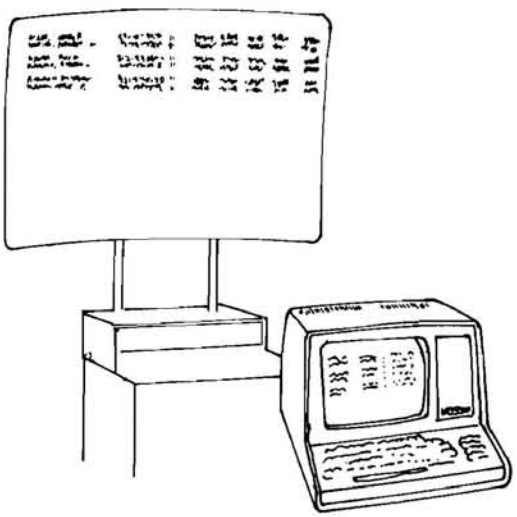
128K RAM BOARD — $595

order no. 77318

Now have 176K of fast dynamic RAM at your disposal in your Heath/Zenith '88, '89, or '90.
Featuring:

- 128K of 200nSec Dynamic RAM, added to the 48K on your CPU board, for a total of 176K.
- Full compatability is retained for MMS and Zenith CP/M® [64K], as well as HDOS [56K].
- Versatile bank switching technique supports three full MP/M II® compatible banks.
- 112K "Electronic Disk" BIOS module for MMS CP/M included, complete with source code.
- Ultimeth Corp. supplies HDOS support.

Delivery beginning March 15, 1982.



VIDEO OUTPUT — $79

order no. 77319

Add an industry standard video output to allow reproduction of the CRT image on another monitor or projection TV system to enhance the usefulness of your terminal in classroom and other educational applications. Allows simultaneous viewing of the display in group situations.

The auxillary display unit should be capable of high resolution for satisfactory performance.

Delivery beginning March 15, 1982.

LOWER PRICE! 16K RAM — $125

order no. 77311

Magnolia's 16K Add-On RAM board has been so popular we lowered the price.

DOUBLE DENSITY DISK CONTROLLER — $595

order no. 77316

Complete hardware and software support for FOUR 8" single or double sided drives and FOUR 5" Single or Double sided, 48tpi [40 track] or 96tpi [80 track] drives, in addition to the three 5" drives supported by your Heath/Zenith controller.

Plus, the obvious advantage of being able to use Single Density 8" media for program and data interchange.

Full compatability is retained with MMS support of the '89s built-in 5" floppy, as well as several Winchester hard disk subsystems.

The package includes:

- Double Density Controller Card.
- Cables for both 5" and 8" disk drives
- CP/M 2.2 on both 5" and 8" media
- New I/O Decoder and Monitor PROMs
- Ultimeth Corp. supplies HDOS support.

If your '89 isn't ORG-0 CP/M compatible yet, our modification is available for $50 additional.

DOUBLE DENSITY SUBSYSTEMS

Dual 8"	DS	48tpi [2.4M]	order no. D8DS	$2695
	SS	48tpi [1.2M]	order no. D8SS	$1995
Single 5"	SS	48tpi [162K]	order no. S540SS	$ 945
	DS	48tpi [343K]	order no. S540DS	$1095
	DS	96tpi [700K]	order no. S580DS	$1295
Dual 5"	DS	96tpi [1.4M]	order no. D580DS	$1995

COMPLETE SYSTEMS

As well as manufacturing enhancements for the '89 [also '88 and '90], we are a Zenith Data Systems OEM, and have all of their hardware and software products available as well. We can provide a completely integrated system, combining the best Zenith products with our own to provide the exact system capabilities to best satisfy your requirements.

ORDERING INFORMATION

Our products are available from many Heathkit Electronic Centers and independent computer stores throughout the United States. If your local dealer doesn't stock our products, you may order direct or request further information by calling our Sales Department on our toll-free number, [800] 426-2841.

MAGNOLIA MICROSYSTEMS
2264-15th Avenue W. • Seattle, WA 98119
(206) 285-7266 (800) 426-2841

PIP Patch

If you use both HDOS and CP/M, then you are aware that when you copy several files at a time with PIP in CP/M, the file names are listed on your CRT screen as the files are copied, but in HDOS the file names are not listed. HUG member Dr. William Moss sent us a patch for the HDOS PIP program that causes it to print file names when you are copying or deleting files. The patch was adapted from an idea by John Stetson, and I have modified it slightly. To install the patch, you need an unmodified copy of PATCH.ABS (as it comes from your HDOS 2.0 distribution disk), and PIP.ABS should also be unmodified. The patch session is shown below, with your entries shown in bold print. PIP.ABS and PATCH.ABS are both on SY0:.

```
>PATCH
PATCH Issue #50.06.00.

File Name? PIP.ABS
Patch ID? IFOJIC
Prerequesite Code? IFBEIADPGEFFCF
Address? 56275
056275 = 052/315
056276 = 030/162
056277 = 064/063
056300 = 021/^D      (You type CTRL-D)
Address? 63162
063162 = 040/052
063163 = 112/315
063164 = 107/056
063165 = 114/043
063166 = 051/043
063167 = 051/043
063170 = 012/043
063171 = 012/176
063172 = 000/377
063173 = 012/002
063174 = 012/043
063175 = 103/247
063176 = 157/302
063177 = 160/171
063200 = 171/063
063201 = 162/076
063202 = 151/012
063203 = 147/377
063204 = 150/002
063205 = 164/052
063206 = 040/030
063207 = 050/064
063210 = 103/311
063211 = 051/^D
Address? ^D
Patch Check Code? PDJELHDH

PATCH Issue #50.06.00.

File Name? ^D

Enter the patch carefully. If you make a
mistake, PATCH will not accept the Check
```

Vectored from 16

sophisticated breakpoint control. Device drivers required by a program must be loaded before running RDT. It requires less memory than ALDT, and can be assembled to use even less by leaving out the disassembler. It will run under HDOS 1.5, 1.6, or 2.0. The documentation is supplied in printed form.

HUG Product List

NOTE: The number in the REM # column refers to the issue of REMark containing a description of the software. Usually, it refers to the "New HUG Sofware" column, but it may refer to an article.

Part Number	Description		Selling Price	REM #
CASSETTE SOFTWARE (H8 and H88)				
885-1008	Volume I Documentation and Program Listings (some for H11)		$ 9.00	
885-1009	Tape I Cassette		$ 7.00	
885-1012	Tape II BASIC Cassette		$ 9.00	
885-1013	Volume II Documentation and Program Listings		$12.00	
885-1014	Tape II ASM Cassette H8 Only		$ 9.00	
885-1015	Volume III Documentation and Program Listings		$12.00	
885-1026	Tape III Cassette		$ 9.00	
885-1036	Tape IV Cassette		$ 9.00	8
885-1037	Volume IV Documentation and Program Listings		$12.00	8
885-1039	WISE on Cassette H8 Only		$ 9.00	
885-1057	Tape V Cassette		$ 9.00	
885-1058	Volume V Documentation and Program Listings		$12.00	
HDOS SOFTWARE (H8/H17 or H89 -- 5-inch only)				
MISCELLANEOUS COLLECTIONS				
885-1024	Disk I H8/H89		$18.00	6
885-1032	Disk V H8/H89		$18.00	8
885-1044	Disk VI H8/H89		$18.00	
885-1064	Disk IX H8/H89		$18.00	
885-1066	Disk X H8/H89		$18.00	10
885-1069	Disk XIII Misc H8/H89		$18.00	
GAMES				
885-1010	Adventure Disk H8/H89		$10.00	4
885-1029	Disk II Games 1 H8/H89		$18.00	8
885-1030	Disk III Games 2 H8/H89		$18.00	8
885-1031	Music 8 & 89 H8/H19 and H89		$20.00	25
885-1067	Disk XI Graphic Games .ABS and B H BASIC (H19/H89)		$18.00	12
885-1068	Graphic Games (H19/H89)	*	$18.00	10
885-1088	Graphic Games (H19/H89)	*	$20.00	14
885-1093	Dungeons and Dragons Game Requires H89 or H8/H19	*	$20.00	16
885-1096	Action Games (H19/H89)	*	$20.00	18
885-1103	Sea Battle Game (H19/H89)		$20.00	20
885-1111	HDOS MBASIC Graphic Games	*	$20.00	23
885-1112	HDOS Graphic Games		$20.00	23
885-1113	HDOS Fast Action Games		$20.00	23
885-1114	Color Raiders and Goop (HA-8-3)		$20.00	23
UTILITIES				
885-1019	Device Drivers (HDOS 1.6)		$10.00	6
885-1022	HUG Editor (ED) Disk H8/H89		$15.00	20
885-1025	Runoff Disk H8/H89		$35.00	
885-1043	MODEM Heath to Heath H8/H89		$21.00	
885-1050	M.C.S. Modem for H8/H89		$18.00	
885-1060	Disk VII H8/H89 SUBMIT, CLIST, FDUMP, ABSDUMP, etc.		$18.00	
885-1061	TMI Cassette to Disk H8 only		$18.00	
885-1062	Disk VIII H8/H89 (2 disks) MEMTEST, DUP, DUMP, DSM		$25.00	
885-1063	Floating Point Disk H8/H89		$18.00	
885-1065	Fixed Point Package H8/H89		$18.00	10
885-1075	HDOS Support Package H8/H89		$60.00	
885-1077	TXTCON/BASCON H8/H89		$18.00	
885-1079	HDOS Page Editor		$25.00	15
885-1080	EDITX H8/H19/H89		$20.00	
885-1082	Programs for Printers H8/H89		$20.00	
885-1083	Disk XVI RECOVER, etc.		$20.00	11
885-1089	MACRO, CTOH, and misc Utilities		$20.00	20
885-1090	Misc. HDOS Utilities CCAT, HPLINK, AH, MBSORT, etc.		$20.00	22
885-1092	RDT Debugging Tool H8/H89		$30.00	14
885-1095	HUG SY: Device Driver HDOS 2.0		$30.00	18
885-1098	H8/HA-8-3 Color .ABS/.ASM		$20.00	19
885-1099	H8/HA-8-3 Color in Tiny Pascal		$20.00	19
885-1105	HDOS 2.0 Device Drivers MX-80, Paper Tiger, Clock, etc.		$20.00	24
885-1116	HDOS Z80 Debugging Tool		$20.00	27
PROGRAMMING LANGUAGES				
885-1038	WISE on Disk H8/H89		$18.00	
885-1042	PILOT H8/H89		$19.00	
885-1059	FOCAL-8 H8/H89		$25.00	13
885-1078	HDOS Z80 Assembler		$25.00	21
885-1085	PILOT Documentation		$ 9.00	
885-1086	Tiny Pascal H8/H89		$20.00	13
885-1094	HUG Fig-Forth H8/H89 2 Disks		$40.00	18
BUSINESS, FINANCE AND EDUCATION				
885-1047	Stocks H8/H89		$18.00	
885-1048	Personal Account H8/H89		$18.00	
885-1049	Income Tax Records H8/H89		$18.00	
885-1051	Payroll H8/H89		$50.00	
885-1055	Inventory H8/H89	*	$30.00	
885-1056	Mail List H8/H89	*	$30.00	
885-1070	Disk XIV Home Finance H8/H89		$18.00	
885-1071	SmBusPkg III 3 Disks H8/H19 or H89	*	$75.00	17
885-1091	Grade and Score Keeping	*	$30.00	14
885-1097	Educational Quiz Disk H89 or H8/H19	*	$20.00	18
DATA BASE MANAGEMENT SYSTEMS (DBMS)				
885-1107	Amateur Radio Logbook and TMS		$30.00	23
885-1108	Telephone/Mail Info. System	*	$30.00	23
885-1109	Retriever (2 disks)		$40.00	23
885-1110	Autofile		$30.00	23
885-1115	Aircraft Navigation DBMS H8/H89		$20.00	25

Vectored to 25

Chain LISP

by Stanley Schwartz, M.D.
60 Edgehill Road
Providence, RI 02906

With the publication of LISP/80 by the Software Toolworks, HDOS users have access at a very low cost to a microcomputer version of what may be the most powerful general language available on mainframes - LISP. LISP/80 will not turn your H89 into M.I.T., but even within the limitations of this implementation you will find LISP infinitely fascinating as well as superior to BASIC for many tasks, particularly those involving the manipulation of data bases. The purpose of this article is to present several LISP functions that will greatly aid you in the composition of large programs in LISP/80. With these functions you will be able to run LISP programs limited in size only by the available disk space and write well-structured, easily debugged code. This article is not an introduction to LISP and explains basic LISP concepts only as much as is necessary to understand the use of the CHAIN-like functions below.

Every programmer, on any size machine, sooner or later runs out of memory. In BASIC we have a halting way around this limitation with the CHAIN command, which erases the current program in memory, loads in another from disk, and begins execution at a specified line. Keeping track of variables is difficult with this method. Frequently used subroutines must be duplicated in all segments of the program since the entire contents of program memory are lost.

The structure of a LISP program is fundamentally different from a BASIC program and allows a way around these problems. The LISP system consists of a number of built-in functions such as PLUS or LENGTH and a special function, DEFINE, which lets us add our own functions to the system. These functions are similar to BASIC functions - each returns a value. Once DEFINEd, our user functions are treated just like the built-in ones. A LISP program consists of many small functions which call themselves and one another. Programs in LISP/80 are read in from disk files by the LOAD function precisely as if they had been typed at the terminal. Thus the disk files must contain DEFINE functions to add new functions to the system.

This makes programs highly modular by nature. The functions are similar to tiny subroutines. Functions that use one another or form a logical group (such as SCREEN-CONTROL, SORTING, etc.) can be kept in different disk files and LOADed when needed. The main program can consist of little more than a series of LOAD functions. There is one problem with this picture - the LISP/80 documentation offers no obvious way to get rid of those functions that are no longer needed and implies that it is not possible! Without such a method the maximum size of a program is limited.

```
PUTPROP(SMASH PROGRAMS (SMASH Smashl Smash VIEW))

DEFINE ((

(SMASH (NLAMBDA LIST (Smashl LIST)))

(Smashl (LAMBDA (LIST)
      (COND ((NULL LIST) NIL)
            ((GETPROP (CAR LIST) 'PROGRAMS)
                (MAPCAR (GETPROP (CAR LIST) 'PROGRAMS) 'Smash)
                (Smash (CAR LIST)) (Smashl (CDR LIST)))
            (T (Smash (CAR LIST)) (Smashl (CDR LIST))))))

(Smash (LAMBDA (ATOM)
      (MAPCAR (GETPROPLIST ATOM) '(LAMBDA (A) (REMPROP ATOM A)))))

(VIEW (NLAMBDA (NAME)
        (SELECTQ NAME
           ((ATOMS A) (MAPATOMS '(LAMBDA (A)
                (COND ((GETPROP A 'VALUE% CELL) (PRINl A) (TAB 20)
```

```
                    (PRINT (GETPROPLIST A)))))))
          ((FUNCTIONS F) (MAPATOMS '(LAMBDA (A) (COND
               ((OR (GETPROP A 'EXPR) (GETPROP A 'FEXPR)) (PRINT A))))))
          ((PROGRAMS P) (MAPATOMS '(LAMBDA (A) (COND
               ((GETPROP A 'PROGRAMS) (PRINl A) (TAB 10)
                    (PRINT (GETPROP A 'PROGRAMS)))))))
        '(ERROR! WRONG ARGUMENT (A F P)))))
))
```

This is a copy of the file SMASH.LSP from my system. It consists of two statements - a PUTPROP and a DEFINE. The purpose of the DEFINE is to add the four functions SMASH, Smashl, Smash and VIEW to the system. The PUTPROP statement is a programming aid, but understanding it is the key to understanding the action of the SMASH functions.

All information stored in the computer in LISP consists of two basic types. The first, called an 'ATOM' is simply a variable length string. The second, a 'LIST', consists of a list of atoms or other lists. (Atoms may be literal or numeric, but we need not be concerned with that here.) An atom is similar to a variable in BASIC and may have a value assigned to it by a 'SET' function similar to the BASIC 'LET'. This value may be an atom or list. For example, the value of the atom FRUIT might be the list ((APPLE (IDA_RED GOLDEN_DELICIOUS)) ORANGE (GRAPE (WINE TABLE))).
You have probably noticed that the basic form of the list FRUIT is very similar to the SMASH function definitions. LISP does not make any distinction between programs and data. Whether a list is interpreted as data, as a function call or function definition depends only on the context. Now we are getting close to the meat of the matter! Just where and how does LISP connect the atom FRUIT with its value and the definition of SMASH with the name (atom) "SMASH"? If we type (GETPROPLIST 'SMASH) to the LISP interpreter, it displays on the screen the following list:

(FEXPR (NLAMBDA LIST (Smashl LIST)) PROGRAMS (SMASH Smashl Smash VIEW)) This is the 'Property list' of the atom "SMASH". All data in LISP/80 is stored as part of the property list of an atom. Function definitions are stored under the EXPR and FEXPR properties. The PUTPROP expression above created a 'PROGRAMS' property for the atom SMASH where we could keep a list of the SMASH functions. SMASH is also the name of one of those functions, with the function definition stored under the FEXPR property of the atom SMASH. The 'value' of an atom is kept in the VALUE CELL property of the atom.

The function Smash gets the property list of the atom and removes each property. An atom with no properties (as opposed to one with the properties set to NIL or 0) ceases to exist and the space is recovered. The functions Smashl and SMASH process input to Smash. For example, if you type (SMASH HORSE DOG CAT) to the intepreter, where HORSE is an atom, DOG is a function and CAT is a PROGRAM of several functions, they all will be deleted.

The last function, VIEW, is a utility to allow you to inspect the active atoms, functions and programs.

One note of caution: the built in LISP functions can also be SMASHED by this method. If you accidentally SMASH one of them your program may crash. To recover, just reload LISP from the disk.

This article does not cover the most interesting aspects of LISP, which have made it the language of choice for complex Artificial Intelligence programs, or even those aspects which increase programmer productivity far beyond the bounds of BASIC. It illustrates an important point for microcomputer users: because of the great inherent flexibility of LISP, even a sparce implementation can be configured by the user to duplicate almost any feature available in any LISP or indeed in any programming language.

EOF

BUGGIN' HUG

Dear HUG,

This letter is directed to the H-11 crowd out there, somewhere. I want to inform you about something that my own small circle of H-11 consider a great step.

Fairbrother Associates is offering a controller board for the Heath H-11 (DEC LSI-11) computer system. The controller will convert the standard non-DMA single-density floppy drive system with 256,000 bytes/floppy of storage to a full DMA double-density floppy drive system with 512,000 bytes/floppy of storage. A 50-conductor cable with standard Shugart connectors is necessary to hook up the controller to the floppy disks.

We would like to know if anyone would be interested in such a product if offered for sale.

The controller is a top quality dual-height board with integral bootstrap that may be disabled. It is completely RX01/RX02 compatible, and comes completely burned in and tested, and is backed by a one year warranty. The board will run DEC supplied diagnostics unchanged.

At the present, RT-11 Version V-3B or V-4 is necessary to run the system as the Heath supplied DX: handler does not support DMA transfers.

At a later date, we will have an enhancement to the V3B/V4 supplied DY: handler that will allow the controller board, which is an 18 bit board, to access the memory above 64K with only the 11-03 CPU board, and not the 11-23 CPU. With this enhancement, the CONTROLLER will use all memory above 64K (to the current limit of 256 KB) as an electronic disk called DY7:.

Because of the method in which DEC encodes the double-density information on the disk, regular single-density disks can be formatted with a system utility called FORMAT and used at double-density. This is what DEC uses, and it can save a lot of money, as you can use your old disks.

To quickly explain 'DMA' for those of you who are not familiar with the term, it is an abbreviation for Dirct Memory Access. It is a much faster way of reading and writing from the floppy disk to and from memory. This allows the controller to place the information DIRECTLY into memory, at up to 20 times the speed of the Heath controller. BASIC, any other programs, and any other disk accesses, are brought into memory MUCH faster.

The controller is tentatively priced at $995.00. We will make a cable available at near cost, around $35.00. A 256 KB dual wide memory board is available at $1395.00.

We can also supply 11-23 CPU's, and disc and tape controllers and drives. Timesharing operating systems, database, editing, word processing, sorting, and other utility programs are available, all runnable on the H-11.

If you are interested in the controller, or have ideas that could help the success of this product, or other products that you feel would be of interest to the H-11 user, write to us with your thoughts on the matter.

Your commitment, comments, and/or ideas will be greatly appreciated.

Edward Judge
Fairbrother Associates
PO Box 685
Northampton, MA 01060

Dear HUG,

While going through my back issues of REMark magazine, I found Ray Massa's article on converting decimal numbers to binary (Issue 15-page 27). After running his second program, I found it took four pages on the line printer to print. I worked on it a while and came up with a MBASIC program that would print the binary equivalent of decimal 0-255 on one page. Hope your readers can make use of it. Keep those REMarks coming, I enjoy them!

```
10            -BINARY.BAS-
20  PRINT CHR$(27)+"E";CHR$(12);CHR$(7)
30  PRINT"SET LP: TO 132 COLUMNS --HIT";@
    LINE INPUT"RETURN WHEN READY-- ";A5$
40  OPEN"O",1,"LP:"
50  PRINT#1,TAB(48);"---DECIMAL TO";@
    PRINT#1,"BINARY CONVERSION---":@
    PRINT#1,:PRINT#1,
60  FOR M1=0TO42
70  FOR M=M1TO255 STEP 43
80  Z=M
```

```
 90  IF Z=0 THEN BI$="00000000"
100  T=Z/2
110  IF T<1 THEN 150
120  IF T=INT(T) THEN BINARY$="0"+BI$@
     ELSE BI$="1"=BI$
130  Z=INT(T)
140  GOTO 100
150  BI$="00000001"+BI$
160  BI$=RIGHT$(BI$,8)
170  PRINT#1,"     ";USING"###";M;
180  PRINT#1," *** ";USING"\      \";@
     BI$;:I=I+1
190  IF M>213 THEN I=6
200  IF I=6 THEN I=0:PRINT#1
210  BI$=""
220  NEXT M
230  NEXT M1
240  CLOSE
250  END
```

Lynn Wakefield
108 Greenwood
Helena, MT 59601

Dear HUG,

For those who have and enjoy playing the game of "OTHELLO" by Richard Musgrave distributed on HUG p/n 885-1068, Disk XII, MBASIC GAMES, but who like me have a tough time telling my pieces from the computers, here is an easy fix. Just list and then retype lines 70 and 770 as follows:

```
70 Z=RND(-PEEK(8219)*PEEK(8219))@
   :H1$="wwww":H2$="iiii"

770 PRINT Y$" aOTHELLO"Y$CHR$(34)@
    "\ME = ii   YOU = ww
```

I've underlined the part of the lines which are actually changed. I think you'll find these graphics a lot easier on the eyes.

David E Warnick
RD#2, Box 2484
Spring Grove, PA 17362

Note from Ted Stowe of the La Mesa, California Heathkit Center:

Problem/Service Hint: HA-8-6. No operation with systems that had been running prior to the HA-8-6 with DG-64D Dynamic RAM Boards.

Solution/Method: On Heath Z80 Board, remove jumper from E49-E51 - also jumper selection for the DG-64D/DG-6405 should be set as described in the appropriate DG manual.

```
 #: 27919      Sec.0
Sb: Another HDOS Bug!!
Fm: Bill Parrott 70070,462
To: Syso/All (X)
```

Terry: While tracking down a problem for Dale Lamm, I discovered a catastrophic bug in HDOS 2.0 (the bug is also present in 1.6). Please see my message #27918 to Dale for details.

```
Sb: #27913-#help, Mr. Wizard!
Fm: Bill Parrott 70070,462
To: Dale lamm 70555,302 (X)
 #: 27918      Sec. 0
```

Dale: Congratulations!!! You are the proud father of a genuine HDOS

*** BUG ***

The .NAME processor goes like this:

```
NAME    PUSH    H
        PUSH    D      ;save user registers
        CALL    FCC    ;fetch channel info
        JNC     NAME1  ;got it fine
        POP     D
        RET
```

As you can see, HDOS is kind enough to return to the user's (HL) rather than to the user's program. (WHO LEFT OUT THE OTHER POP INSTRUCTION?!?!?!?!) Ah, well... the only way around this absolutely catastrophic error is to do something like this (I realize it's crude, but it should work:

```
        LXI     H,NAMADD
        LXI     D,DEVEXT
        MVI     A,CHAN
        SCALL   .NAME
        JC      wherever

NAMADD  RET
        DS      8
DEVEXT  DS      6
```

Since HDOS returns to our 'name' area, this should work. If the call succeeds, fine. If it fails, return to a return and back to us finally!

Good work, Sherlock...

Vectored from Ad page

mistake, PATCH will not accept the Check Code given, and you will have to type CTRL-C and start over. Test the patch by copying files. One small difference between this patch and CP/M PIP is that file names are listed before the file is copied in CP/M, and afterwards here. Also, this patch will list the file name even if only one file is copied, and regardless of where the file is copied to/from, including TT:. (Watch next month's REMark for an article on how I figured out those PATCH ID codes!)

PS:

Dear HUG,

I have been using the ABSDUMP program, supplied on DISC VII. It has been very useful, but I find that a lot of patches and modifications that are published in REMark, BUSS, etc., are listed by track and sector. This is the format for the DUMP program on DISC VIII. Since ABSDUMP works on a cluster basis, I have to convert these track/sector numbers to clusters.

Following is a short MBASIC program which prints a list of tracks, clusters, and sectors. A segment of the output is also enclosed. Feel free to print it in REMark if you think it's useful.

Joe Whalley
23450 Coyote Springs Dr.
Diamond Bar, CA 91765

```
10 OPEN "O",1,"LP:"
20 PRINT#1,"TRACK","CLUSTER"," SECTOR (LOW,HIGH)","TOTAL SECTOR"
30 PRINT#1,"-------------------------------------------------------------------"
40 FOR CL=0 TO 199
50 TR=INT(CL/5)
60 IF CL=0 THEN GOTO90
70 LS=(CL-(TR*5))*2
80 GOTO 100
90 LS=0
100 HS=LS+1
110 TS=LS+(TR*10)
120 PRINT#1, TR,CL,LS;",";HS,,TS;" ";TS+1
130 IF HS=9 THEN PRINT#1,"-------------------------------------------------------------------"
140 NEXT CL
```

TRACK	CLUSTER	SECTOR (LOW,HIGH)	TOTAL SECTOR
0	0	0 , 1	0 1
0	1	2 , 3	2 3
0	2	4 , 5	4 5
0	3	6 , 7	6 7
0	4	8 , 9	8 9
1	5	0 , 1	10 11
1	6	2 , 3	12 13
1	7	4 , 5	14 15
1	8	6 , 7	16 17
1	9	8 , 9	18 19
2	10	0 , 1	20 21
2	11	2 , 3	22 23
2	12	4 , 5	24 25
2	13	6 , 7	26 27
2	14	8 , 9	28 29
3	15	0 , 1	30 31
3	16	2 , 3	32 33
3	17	4 , 5	34 35
3	18	6 , 7	36 37
3	19	8 , 9	38 39
4	20	0 , 1	40 41
4	21	2 , 3	42 43
4	22	4 , 5	44 45
4	23	6 , 7	46 47
4	24	8 , 9	48 49

Add LPRINT, PEEK, POKE, USR, and INPUT$ to BASIC-E

The more I play with BASIC-E (which HUG now offers as part no. 885-1215), the more I see that it is quite adequate for most of what I do in BASIC (which isn't much). It is fast and saves disk space, since all you need to run a program are the 12k runtime interpreter and your compiled program, which will usually be 2/3 to 1/2 the size of the original source program. My only objection to it is that is has no LPRINT or POKE commands and no PEEK function. However, it does have an OUT command and an INP function, and it seemed to me that it would be very easy to modify them somehow to accomplish what I wanted.

If you have a BASIC that has a POKE command and a PEEK function, then you have all it takes to make your own LPRINT command if your operating system is CP/M. All you have to do is change the value of the IOBYTE (a byte in low memory which matches physical input/output devices to CP/M's logical I/O devices) so that anything you output to the logical console device (CON:) goes to the printer instead of to the CRT terminal. In MBASIC, you would do it this way:

```
10 PRINT "THIS LINE IS PRINTED ON THE CRT"
20 REM  SWITCH CON: OUTPUT TO THE PRINTER
30 POKE 3,PEEK(3) AND 252 OR 2
40 PRINT "THIS LINE GOES TO THE PRINTER"
50 REM  SWITCH CON: OUTPUT BACK TO THE CRT
60 POKE 3,PEEK(3) AND 252 OR 1
70 PRINT "WE'RE BACK ON THE CRT AGAIN"
```

You may prefer to use this method of printing even when you use a BASIC that has LPRINT because it is more versatile. For example, suppose that you wanted to give the user of your program the option to list some data on the CRT or on a printer. Using LPRINT, you would have to write two routines, one using PRINT statements and the other using LPRINT. But if you use the IOBYTE method, you only need one routine using PRINT statements, and you simply set up the IOBYTE before you execute it. Just one word of caution: Be sure that you set CON: output back to the CRT as soon as you finish printing, because until you do, you are locked out from entering anything at your keyboard.

The IOBYTE

Before I explain how to implement the things I have discussed above in BASIC-E, I thought you might like to know a little more about the IOBYTE and how the BASIC routine above works. The IOBYTE is explained on pages 15 and 16 of the CP/M 2.0 Alteration Guide, one of the Digital Research manuals supplied with CP/M. As it states there, the IOBYTE is an optional feature that may not be implemented on everyone's version of CP/M. But it is implemented on most versions these days (including Heath's) because of the increased use of printers, modems, etc. The IOBYTE (an 8-bit byte) is divided into 4 fields of two bits each. The least two significant bits make up the Console field, which is the only one we are concerned with. Normally these bits are set to 01 (one), which means that the Console logical device is assigned to the CRT physical device. If you set them to 10 (two), it sets what is called the Batch mode, in which Console input is set to the Reader device and Console output is set to the List device. The List device is itself a logical device and is controlled by bits 6 and 7 of the IOBYTE. Although I haven't tried it, it should be possible to select one of up to three printers by changing the bits in the List field. This would give you the power to not only turn printing off and on in BASIC, but to select which printer you want to use as well. I may try this some time, and if I do I will write about it in REMark.

Our little MBASIC routine works by first PEEKing the value of the IOBYTE at location 3. Then it ANDs this value with 252 to reset the console field bits, and ORs it with the value we want to set (either 1 for the CRT or 2 for the printer). The result is POKEd back to the IOBYTE. NOTE: This procedure will also work in Heath CP/M if you set the console field bits to 11 (three) which sets it to logical device UC1:. In Heath CP/M, UC1: is defined as CON: for input and LST: for output, so the keyboard would still work while in this mode. However, it may not work in non-Heath CP/M.

Modifying BASIC-E

The part of BASIC-E that I modified is the run time interpreter, RUN.COM. If you put it and DDT.COM on a disk in drive A: and enter the following, you can see the area to modify. In these examples what you type is in bold print.

```
A>DDT RUN.COM
DDT VERS 2.2
NEXT  PC
XXXX 0100
-L1F8
  01F8  LXI  H,01FD
  01FB  MOV  M,C
  01FC  IN   00
  01FE  RET
  01FF  LXI  H,0205
```

```
0202  MOV  M,C
0203  MOV  A,E
0204  OUT  00
0206  RET
0207  JMP  0939
020A  MOV  A,M
```

Addresses 01F8 through 01FE are part of the INP function, and addresses 01FF through 0206 are part of the OUT command. By locating and inspecting the parts of the RUN program that call these routines, I found out that they are entered with the HL register pair containing the address of a binary number which is decoded from the port address given to the command. For example, if you say OUT 100,32, address 01FF is entered with HL pointing to an address containing 100 in 16-bit binary form. Since BASIC-E does not check to see if the arguments to OUT and INP are greater than 255 (the highest port number), it is easy to modify these routines to perform memory reads and writes instead of port reads and writes. That is just what I did at first, and that gave me PEEK, POKE, and LPRINT (using the IOBYTE), but in the process I lost INP, and OUT. So instead of modifying the INP routine to PEEK, I made it into a USR function, which gave me the ability to add INP, OUT, and PEEK in BASIC itself. Here are the actual modifications I made (using DDT):

```
-A1F8
01F8  INX H
01F9  MOV H,M
01FA  MOV L,C
01FB  PCHL
01FC  .
-A1FF
01FF  INX H
0200  MOV H,M
0201  MOV L,C
0202  MOV M,E
0203  RET
0204  .
```

You can make these modifications by typing the parts in bold print, after you have loaded RUN.COM with DDT as shown before. After you make the modifications, exit DDT and save the new RUN.COM as follows:

```
-G0        (The letter G and a zero)
A>SAVE 46 RUN.COM
```

The INP function will now function as a USR function, and the OUT command will function as a POKE command. Note that the syntax is not changed, only the meaning of INP and OUT. Before the modification, if you said

```
OUT 100,32
```

it meant, "Write the value 32 to port address 100." Now it means, "Write the value 32 to memory address 100." The old INP function used to read a value from the specified port address and return the value to BASIC-E in the A register. The new INP function jumps to whatever address is specified, and when the user subroutine RETurns, it returns to BASIC-E. Since BASIC-E is expecting something in the A register, the user subroutine can return any byte value you wish to BASIC-E by placing it in the A register. This makes it possible to write our own port input and output routines to get back what we lost, along with PEEK, INPUT$, INKEY$, and whatever else we might think of. CP/M provides a perfect place to put small subroutines in the default FCB area that starts at 5C hex. This area is not used by RUN.COM while a program is running. Here are some useful subroutines.

```
     REM  USER SUBROUTINES FOR MODIFIED
     REM  BASIC-E.

999990    REM  PEEK FUNCTION
     REM  ENTER WITH "ADDRESS" SET TO \
          PLACE TO PEEK. \
          RETURNS "BYTE" WITH VALUE READ.
     OUT 96,58: OUT 97,ADDRESS
     OUT 98,INT(ADDRESS/256): OUT 99,201
     BYTE=INP(96)
     RETURN

999991    REM  INP FUNCTION
     REM  ENTER WITH "ADDRESS" SET TO \
          PORT TO READ. \
          RETURNS "BYTE" WITH VALUE READ.
     OUT 96,219: OUT 97,ADDRESS
     OUT 98,201
     BYTE=INP(96)
     RETURN

999992    REM  OUT COMMAND
     REM  ENTER WITH "ADDRESS" SET TO \
          PLACE TO WRITE TO, AND "BYTE" \
          SET TO VALUE TO BE WRITTEN.
     OUT 96,62: OUT 97,BYTE: OUT 98,211
     OUT 99,ADDRESS: OUT 100,201
     BYTE=INP(96)
     RETURN

999993    REM  INPUT$(1) SIMULATOR
     REM  RETURNS CHARACTER IN "CHAR$" \
          WHEN A KEY IS STRUCK. \
          CP/M ECHOES THE CHARACTER.
     OUT 96,14: OUT 97,1: OUT 98,195
     OUT 99,5: OUT 100,0
     CHAR$=CHR$(INP(96))
     RETURN

999994    REM  INKEY$ FUNCTION
     REM  RETURNS CHARACTER IN "CHAR$" \
          IF A CHARACTER HAS BEEN \
          STRUCK.  OTHERWISE, RETURNS \
          A NULL.  THE CHARACTER IS NOT \
          ECHOED.
     OUT 96,14: OUT 97,6: OUT 98,30
     OUT 99,255: OUT 100,195
     OUT 101,5: OUT 102,0
     CHAR$=CHR$(INP(96))
     RETURN
```

With these subroutines, we can write a BASIC-E version of the MBASIC program

presented at the beginning of this article.

```
PRINT "THIS LINE IS PRINTED ON THE CRT"
REM  SWITCH CON: OUTPUT TO THE PRINTER
     ADDRESS=3:GOSUB 999990
     OUT 3,BYTE AND 252 OR 1
PRINT "THIS LINE GOES TO THE PRINTER"
PRINT
REM  SWITCH CON: OUTPUT BACK TO THE CRT
     OUT 3,BYTE AND 252 OR 2
PRINT "WE'RE BACK ON THE CRT AGAIN"
```

The PEEK subroutine would, of course, have to be included with this little program for it to work. Notice the extra PRINT statement just below "THIS LINE GOES TO THE PRINTER". It is required because of a peculiarity of BASIC-E. A line of text is not actually printed until a carriage return character is encountered, and a carriage return is not printed until another PRINT statement or the end of the program is encountered. So the line is not sent to the printer (or console, for that matter) until the CR at the end is reached, and the CR (which a line printer must see before it will print anything) is not sent until the extra PRINT statement is executed. If there are several lines of print to be sent to a printer, you only need one extra PRINT statement, which should be the last statement before you turn off the printer.

This pecularity also affects the INPUT$ subroutine, as is shown in this example.

```
10    PRINT "ENTER A CHARACTER:"
      GOSUB 999993
      IF CHAR$=CHR$(27) THEN STOP
      PRINT "THE CHARACTER IS ";CHAR$
      GOTO 10
```

When you run this little program, the line

```
ENTER A CHARACTER:_
```

appears on the screen. The underline indicates where the cursor would be. You might think that you would have to put a semicolon at the end of the first PRINT line to prevent the cursor from appearing on the next line. If you did, the program would sit and wait for you to enter a character before the prompt line appeared at all! Another peculiarity is that trailing spaces are not printed until something else is encountered. For example, if your prompt line was

```
10    PRINT "ENTER A CHARACTER:  "
```

you would still get

```
ENTER A CHARACTER:_
```

on the screen. It is easy, however, to "fool" BASIC-E as follows:

```
10   PRINT "ENTER A CHARACTER:  ";CHR$(127)
```

The CHR$(127) is a Delete character, which is not acted upon by most video terminals. This time, the prompt would appear as

```
ENTER A CHARACTER:  _
```

with the cursor spaced over as intended. The third line in the program illustrates something you should consider whenever you write programs: Always provide a way out of an infinite loop. In this program, if you type an escape character, it stops.

The INPUT$ subroutine used in this program calls BDOS function no. 1 in CP/M, which waits until you type a key, echoes it, and returns with its value in the A register. The INKEY$ subroutine also makes a BDOS call, to function no. 6, which does not wait for a key to be struck, but returns with zero in the A register if no key was struck before the call, and returns with the value of the key if one was struck. There are many other things you can accomplish in subroutines by making BDOS calls, giving BASIC-E far more power and flexibility than it had before.

PS:

Protected Input in HDOS MBASIC

by Raymond H. Thompson
12260 Welcome Drive
San Antonio, TX 78233

The following MBASIC routine is designed to provide very highly protected input into a user program. The routine is designed for Microsoft Basic version 4.82 running under HDOS.

Some of the features of this routine are the use of arrow keys to position the cursor and copy material. IC (insert character) and DC (delete character) are also functional. Characters that input by the user are tightly restricted to avoid the problem of entering string data when numerics are required, etc.

The real reason for the routine being developed was to provide protected input when numbers were required. Most of the ready solutions involved inputting a string and doing a VAL$() on the string. This however did not alllow for letters being input as VAL$() would return zero. If some check was done after the INPUT function and letters were found in the input some type of error had to be generated and the user was required to re-input the whole thing. The very clean solution seemed to be to reject each input character as it was typed and this is the approach that I took.

The routine does indeed return a string of the input data in all cases. The string is built up from each individual character. The one big difference is that if only numbers are to be allowed the string will only consist of numbers, and will conform to a rigid format.

Several special keys are recognized by the routine. The left arrow will move the cursor left without disturbing the display and erase one character from the input buffer. The right arrow will move the cursor right one space and copy the character moved over into the input buffer. The IC (insert character) will insert one blank at the current cursor position and shift any information to the right of the cursor right dropping the right-most character if necessary. The DC (delete character) will delete one character at the current cursor and shift everything left duplicating the next to last character at the end of the display. The BACKSPACE will function the same as the left arrow key except it will erase the character it backspaces over. The DELETE key will position the cursor to the start of the input area and clear out the input buffer.

The routine as it is written is probably not what most of you would call very readable. Well I have to agree with you on that point. The routine was written for speed with no concern to readability. In BASIC almost everything extra causes some sort of speed loss. Hence no extra spaces and an almost total lack of comments. (**Editor's Note:** We have re-written the routine with spaces after key words and with the lines truncated (using @) to fit in our magazine column. We recommend that you remove the spaces and type the lines without breaking them up when you implement the routine.)

Before I describe the routine I would like to cover the interface to the user program. The first line contains a CLEAR statment. You can change the value to any appropriate value that you need. The GOSUB in the first line must immediately follow the CLEAR. This will establish variables at the head of BASIC's symbol table allowing for faster lookup. The GOTO should be to the first line in your program.

One very important point must be stressed. Locate the routine as close to the front of your program as is possible. This will minimize line number search times required by BASIC. If you place the routine at the end of your program, you will have a real dog on your hands. The routine uses the following variables. Some variables must not be reused for any purpose, others can be reused but will be destroyed by the routine.

Do not reuse the following variable

VARIABLE	USE
U0%(.)	Real Time input routine

The following variables may be re-used

VARIABLE	USE
ML%	Maximum length of input to allow
MU%	If =1 map lower case to upper case
TP%	Type of input to allow (see text)
WK$	Copy material
I%	General working variable
A$	Current character
L%	Number of characters input so far
LN%	Length of input data
IN$	Input data

Now I will go into more detail of how to interface the routine to your program and what to pass to the routine and what you get back.

The following variables must be set before doing a GOSUB to the start of the routine (which in the listing is line # 1040).

WK$ Set this variable to any copy material that you would like. This would be used to correct a name without the user having to retype the whole thing. Do not print any copy material as the routine will print it. If the copy material is longer than the maximum length it is truncated on the right.

MU% If set to a one (1) will convert all lower case characters to upper case before any processing. If other than one (1), characters are taken as is.

TP% Specifies the type of input to allow. The following values are supported.

0 Means to allow all characters except control characters.

1 Specifies to allow only letters and numbers. Any special characters will be rejected.

2 Says to only allow letters. Space is considered a letter in this case.

3 Will only allow numbers. If a sign is needed it must be first. Only one decimal point will be allowed.

The input restrictions will apply on any copied material input by use of the arrow key.

Upon return from the routine the following variables will be returned.

LN% is the length of what was input. In some cases this will not match the actual length of the returned variable.

IN$ is what was input. It will match exactly what is shown on the screen.

To call the routine merely set up the proper parameters, place the cursor where the input is to begin and do a GOSUB to the first line of the routine. To make life a little easier some useful routines have been provided.

If all you require is normal input (all characters TP%=0) with no copy material then merely set ML% to the size to allow and do a GOSUB to line 1460. A prompt of asterisks will define the input size on the screen. Also when <RETURN> is pressed any information beyond the cursor up to the size of the input will be erased.

If you require straight numeric input with no copy material then merely set ML% to the size of the input and do a GOSUB to line 1540. If <RETURN> is pressed and no data was entered then a zero will be printed. The remainder of the input area is also blanked.

If you would like the rest of the input area blanked then set the variables as for a full call but instead do a GOSUB to line 1500.

The routine works by using a real time input routine as described in previous issues of REMark. Each character that is fetched from the keyboard is examined to determine its function. Since the USR0 function returns a number each character is checked against its ASCII value. The character fetched is not printed until it has been determined that it was indeed a valid character.

The copy material works by extracting the characters copied over from the string variable WK$. Inserts and deletes also function against the string WK$. All characters fetched are placed in WK$. When <RETURN> is pressed then the input data is taken from WK$.

The cursor position is saved by the routine by using one of the functions of H-19/H-89 terminal. If you are saving the cursor this way then use caution when using this routine as your saved position may be destroyed.

On special keys (IC, DC, etc) they return two keystrokes, the first being an ESCape, the second being some letter. If the first character is an ESCape then immediately an attempt is made to get another character. If no character is available or is not a function the the routine recognizes, the routine will go back into its main waiting loop.

Placing this routine in your program will make the input very professional and make it consistent every time. Nothing annoys me more than to have the program say ?RE-ENTER when trying to input something.

KEYIN.ASC

```
1000 CLEAR 1000:GOSUB 1610:GOTO 1890
1010 '*
1020 '*** Keyboard Input Routine ***
1030 '*
1040 TP%=ABS(TP%):ML%=ABS(ML%):IF ML%>80@
     THEN ML%=80
1050 IF LEN(WK$)<>ML% THEN@
     WK$=LEFT$(WK$+SPACE$(ML%),ML%)
1060 GOSUB 1380:PRINT WK$;:GOSUB 1420:L%=0
1070 DEF USR0=VARPTR(U0%(0))
1080 I%=USR0(0):IF I%=0 THEN 1080 ELSE@
     IF I%=27 THEN 1210
1090 IF I%=10 THEN 1340 ELSE IF I%=8@
     THEN 1290 ELSE IF FI%=127 THEN 1260
```

```
1100 IF MU%=1 AND I%>96 AND I%<123 THEN@
     I%=I%-32
1110 A$=CHR$(I%):IF L%>=ML% THEN 1070
1120 ON TP% GOTO 1150,1160,1180
1130 IF I%<32 OR I%>127 THEN 1070
1140 PRINT A$;:L%=L%+1:MID$(WK$,L%,1)=A$:@
     GOTO 1070
1150 IF (I%>=48 AND I%<=57) OR I%=32@
     THEN 1140
1160 IF (I%>=65 AND I%<=90) OR I%=32@
     THEN 1140
1170 IF I%>=97 AND I%<=122 THEN 1140@
     ELSE 1070
1180 IF I%=43 OR I%=45 THEN IF L%<1@
     THEN 1140 ELSE 1070
1190 IF I%=46 THEN IF@
     INSTR(LEFT$(WK$,L%),A$)@
     THEN 1070 ELSE 1140
1200 IF I%>=48 AND I%<=57 THEN 1140@
     ELSE 1070
1210 DEF USR0=VARPTR(U0%(0))
1220 I%=USR0(0):IF I%=0 THEN 1070 ELSE@
     IF I%=68 THEN 1270
1230 IF L%>=ML% THEN 1070
1240 IF I%=67 THEN 1250 ELSE IF I%=64@
     THEN 1310 ELSE IF I%=78 THEN 1330@
     ELSE 1070
1250 A$=MID$(WK$,L%+1,1):I%=ASC(A$):@
     GOTO 1120
1260 IF L%>0 THEN FOR L%=L%TO1STEP-1:@
     PRINT CHR$(8);:NEXT L%:GOTO 1070
1270 IF L%<1 THEN 1070
1280 L%=L%-1:PRINT CHR$(8);:GOTO 1070
1290 IF L%>0 THEN MID$(WK$,L%)=" ":@
     PRINT CHR$(8);" ";:GOTO 1280
1300 GOTO 1070
1310 WK$=LEFT$(LEFT$(WK$,L%)+" "+@
     MID$(WK$,L%+1),ML%)
1320 GOSUB 1380:PRINT MID$(WK$,L%+1);:@
     GOSUB 1420:GOTO 1070
1330 WK$=LEFT$(WK$,L%)+MID$(WK$,L%+2)+@
     RIGHT$(WK$,1):GOTO 1320
1340 LN%=L%:IN$=LEFT$(WK$,L%):WK$="":@
     RETURN
1350 '*
1360 '*** Save Cursor Address ***
1370 '*
1380 PRINT CHR$(27);"j";:RETURN
1390 '*
1400 '*** Restore Cursor Address ***
1410 '*
1420 PRINT CHR$(27);"k";:RETURN
1430 '*
1440 '*** Alpha Input With Prompt ***
1450 '*
1460 TP%=0:WK$=STRING$(ML%,"*")
1470 '*
1480 '*** User Provided Input ***
1490 '*
1500 GOSUB 1040:GOTO 1560
1510 '*
1520 '*** Numeric Input With Prompt ***
1530 '*
1540 TP%=3:WK$=STRING$(ML%,"."):GOSUB 1040
1550 IF LN%<1 THEN IN$="0":PRINT IN$;
1560 IF LEN(IN$)<ML% THEN@
     PRINT SPACE$(ML%-LEN(IN$));
1570 RETURN
1580 '*
1590 '*** Establish Common Routine@
      Constants ***
1600 '*
1610 I%=0:L%=I%:TP%=I%:MU%=I%:A$="":@
     WK$=A$:WIDTH 255:DIM U0%(3)
1620 DEF FNPS$(X%,Y%)=CHR$(27)+"Y"+@
     CHR$(X%+31)+CHR$(Y%+31)
1630 U0%(0)=&H36:U0%(1)=&H1FF:@
     U0%(2)=&H77D8:U0%(3)=&HC9
1640 RETURN
1650 '*--------------
1660 '* REQUIRES:
1670 '* ML% = MAXIMUM LENGHT OF INPUT
1680 '* TP% = TYPE OF INPUT
1690 '*       0 = ALLOW ALL
1700 '*       1 = NO SPECIAL CHARACTERS
1710 '*       2 = LETTERS ONLY
1720 '*       3 = NUMBERS ONLY
1730 '* MU% = IF  SET TO 1 MAP LOWER CASE@
             TO UPPER
1740 '* WK$ = ANY COPY MATERIAL
1750 '* -------------
1760 '* RETURNS:
1770 '* LN% = LENGTH THAT WAS INPUT
1780 '* IN$ = DATA THAT WAS INPUT
1790 '* -------------
1800 '* USES:
1810 '* I%  = GENERAL LOOP COUNTER
1820 '* A$  = CURRENT CHARACTER
1830 '* L%  = LENGTH OF CURRENT INPUT
1840 '* U0% = ARRAY OF MACHINE LANGUAGE@
             ROUTINE
1850 '*--------------
1860 '*
1870 '*** Your program can start here ***
1880 '*
1890 END
```

EOF

Making Sense of Making Sense

My Article "Making Sense of MAKEBIOS" in REMark #26 has two small errors. In the listing of MAKE.SUB, change $R/W to $$R/W, and $DIR to $$DIR.

PS:

Vectored from 25

885-1217 HUG Disk Duplication Utilities $20.00 26

% Means CP/M 1.43 only (ORG-4200)
%% Means CP/M 1.43 or 2.2 (Heath)
Other CP/M disks are for 2.2

MISCELLANEOUS

885-0017	H8 Poster	$ 2.95
885-0018	H89 Poster	$ 2.95
885-0019	Color Graphics Poster	$ 2.95
885-4	HUG Binder	$ 5.75
885-4001	REMark VOLUME I	$20.00 23
885-4002	REMark VOLUME II	$20.00

CP/M is a registered trademark of Digital Research Corp.

Running DECUS Programs Under HT11

by: Dr. Peter Vijlbrief
Schoonoord 26
2215 ED VOORHOUT
The Netherlands

ADVENTURE:
The DECUS program 11-340 "ADVENTURE" can be ordered, if you are a member of DECUS (and why shouldn't you be?) from them on a floppy (KA) and don't forget to order the Write-Up (AA) too, the latter costs very little and gives you some hints to install the program and to run it. But of course there are no special hints to run the program under the HT-11 software system. Here they are.

You get the game on a diskette in the FORTRAN source language and one module in ASSEMBLER source.

You need the HT-11 Assembler and Fortran to compile the game. My system has 32K words of memory (and the arithmetic chip, but that is not necessary in this case).

Because of the length of some source modules, it is wise to copy them one by one on a scratch diskette (in DK) and after compiling into an object program, copy them on a spare diskette using PIP.

The FORTRAN modules need the /V switch when they are compiled, e.g.:

```
.R FORTRA<cr>
*ADVENT=ADVENT/V<cr>
*
```

After all xxxx.FOR modules are compiled, copy AMAC.MAC on DK, and using EXPAND and ASEMBL, compile to AMAC.OBJ. Then LINK as follows: create a special system-diskette, containing the SYSTEM programs, PIP.SAV, LINK.SAV and FORLIB.OBJ, make it bootable and put it in SY:, the diskette containing the collected xxxx.OBJ modules is in DK:, then:

```
.R LINK<cr>
*SY:ADVENT=DK:ADVENT,ASUB,AIOSUB,ASUSP,AMAC/F/C<cr>
*AINIT/0:1.C<cr>
*AMAIN/0:1<cr>
*
```

After successful linking ADVENT.SAV in this way, link the "starter":

```
.R LINK<cr>
*AINDX.SAV=ABUILD,ASUSP/F
*
```

Then RUN AINDX, this will produce AINDX.DAT. Put AINDX.DAT and ATEXT.TXT on a new diskette in DK:, and the SY:-diskette with ADVENT.SAV in SY:, .R ADVENT<cr> will initialize the program, just as described in the DECUS Write-Up. Besides containing the necessary SYSTEM programs, my SY:-diskette has PIP.SAV and ADVENT.SAV also a copy of ATEXT.TXT and AINDX.DAT. Then it is possible when you get a run-time problem (because of mains-failure, etc.), you can always copy ATEXT.TXT and AIDX.DAT to DK: using PIP and start the game again.

When you want to stop, don't use <ctrl C>, but type QUIT and don't erase DK:, because you need the ATEXT.DAT and AINDX.DAT on DK: to restart the game again. All that is described in the DECUS Write-Up I mentioned before. This is a rather cheap way to get the famous ADVENTURE game on your H-11!

OTHER LANGUAGES:
When you like to experiment with some high-level languages and not spend too many dollars, you can use DECUS ALGOL (11-231A: 3 floppy diskettes [KC] and don't forget to order the Write-Up [AC].

You can copy from the original diskette #3 on an empty floppy:

```
ALGOL.ALG      ALGOL .ERR
ALGOL.SAV      ALGRTS.ERR
SWAP .ALG      DEMO  .SRC
```

Put this diskette in DK: and the normal HT-11 SYSTEM diskette in SY:. If you start running ALGOL.SAV (.RUN ALGOL) all seems okay. After: CODE FILE: _ give only <cr> and the ALGOL-compiler (ALGOL.ALG) comes in action, prompting with #.

You then can enter DEMO,DEMO=DEMO/L, but no luck! The computer indicates a time-out error with: ?M-TRAP TO 4 013356. That means: the PC has arrived at location 13356, but something wrong has happened. By disassembling that section I found that from position 13350 to position 13354 the PDP-11 Console Switch & Display Register (177570) was set to zero. On a LSI-11 the Console Switch & Display Register is not installed. (See Richard H. Eckhouse Jr., L. Robert Morris: Minicomputer Systems, Organization, Programming and Applications [PDP-11], second edition 1979, Prentice-Hall). If the software attempts to deposit data into a non existent memory location, the "time-out" feature will cause an error flag. The data register will then reflect location 4, the trap location, for references to nonexistent locations. To cure this problem I patched three NOP instructions (nonoperating instructions) in the error-causing part of ALGOL.SAV. If you use the unmodified program as is present on diskette #3, then PATCH your copy at addresses 13344, 13346 and 13350 with 240 (=NOP). Start the program again and it will run on your H-11 under HT-11 without further problems.

Of course you can LINK the ALGRT2.OBJ module according to the instructins in the Write-Up:

```
.R LINK<cr>
*ALGOL,ALGOL=ALGRT2/B:400/M<cr>
*STACK ADDRESS= TMPSTK<cr>
*
```

Then you get an ALGOL.SAV which make use of the arithmetic chip (EIS/FIS). The addresses yu have to PATCH in this program are located at: 13214, 13216 and 13220.

Then run ALGOL:

```
.RUN ALGOL<cr>
ALGOL INTERPRETER V06.6.002
CODE FILE: _A<cr>
ALGOL V06.6.019#
#DEMO,DEMO=DEMO/L<cr>

CODE FILE: DEMO<cr>
```

and then you see the DEMO.SRC, compiled to DEMO.ALG running under the ALGOL-interpreter.

In this way you can experiment with a middle-class ALGOL without spending too much money.

EOF

Enhancements for Cassette BASIC

by: Donald F. Mason
559 Brookleigh Road
Victoria, BC
Canada V8Z 3K1

The availability of source listings for issue 10.06.00 of (Cassette) Extended BH BASIC has allowed me to make three changes that I have wanted to do ever since I first used the system. These are relatively trivial but are, I feel, none the less worth sharing with other HUG members.

As supplied by Heath, the BUILD command always requires parameters for initial line number and increment even though the first often, and the second usually, have values of 10. The first change allows the command to be entered in three forms:

```
BUILD iexpl,iexp2   (Standard format)
BUILD iexpl         (Same effect as BUILD
                     iexpl,10)
BUILD               (Same effect as BUILD
                     10,10)
```

In many implementations of BASIC a function is supplied which permits input from

the console terminal without a Carriage Return being given, i.e. the INPUT$(function in Microsoft BASIC. The second enhancement changes the PIN(function in BH BASIC to give this facility. Reference to PIN(0), an illegal form in standard BH BASIC, causes execution to stop until a key is struck, then returns the ASCII Decimal code for that key and resumes execution. To give a trivial example of the use of this important function consider the single character response required to a display such as SURE? (Y/N). This may be coded:

```
2000 PRINT "SURE? (Y/N) ";:A=PIN(0)
2010 PRINT CHR$(A)
```

The program will take off when a key is struck without waiting for a Carriage Return. There are of course many more sophisticated uses for this tool. Note that the character is not echoed back to the console screen and that this must be done by the program if it is to happen.

The final change blanks out characters on the console screen as they are deleted by Backspacing in the same way as the Disk version of BH BASIC.

These changes make use of a patch area, 64 bytes long, left in BASIC at address 112 175. To enter the changes, first load the original unconfigured version from Heath, then make the changes shown below through the front panel of the H8, or with the substitute command on an H88, go through the configuration routine in the normal way and finally make a copy of the configured version.

Change #1; BUILD command defaults

Address	Enter
046 044	315 175 112
112 175	021 012 000 142 153 315 117 075
112 205	247 310 315 211 057 353 315 117
112 215	075 247 310 367 272 315 211 057
112 225	311

Change #2; implement PIN(0)

Address	Enter
060 317	303 226 112
112 226	315 142 074 173 247 302 322 060
112 236	315 313 040 303 303 060

Change #3; delete after Backspace

Address	Enter
111 053	314 244 112 000
112 244	076 040 315 360 040 076 010 315
112 254	360 040 065 311

These enhancements have now been in use for two to three months without any problems and can be considered stable. They have proved very useful to me in programming a variety of applications, including an elementary word processor being used to write this article.

Unfortunately, the changes as shown are only applicable to version 10.06.00 of Extended BH BASIC. Similar changes could, of course, be made to earlier versions if the listings were available.

In conclusion, I would like to draw the attention of tape system users to an excellent article by James F. Teixiera in "Microcomputing", July 1980. This article documents a program which adds a line editor to BH BASIC and occupies the top 350 bytes of memory. If you have not already implemented this on your machine, I recommend that you do so. It saves an awful lot of re-typing and frustration.

EOF

NATIONAL HUG CONFERENCE
Registration Card

Name _____

Address _____

City _____ State ____ Zip _____

|__| Yes! I'll attend the **First National HUG Conference** and here's my 15 bucks. ($20 after June 1, 1982)

|__| Can't make up my mind that far in advance.

|__| No chance! But tell me about it in REMark.

|__| Send me Hotel reservation information.

Fold, staple, mutilate and send to us right away! All payments must be by check or money order. Send to National HUG Conference, Heath Users' Group, Hilltop Road, St. Joseph, MI 49085.

You may copy this page.

BULK RATE
U.S. Postage
PAID
Heath Users' Group

Heath
Users'
Group
Hilltop Road
St. Joseph MI 49085

POSTMASTER: If undeliverable, please do not return.

885-2027