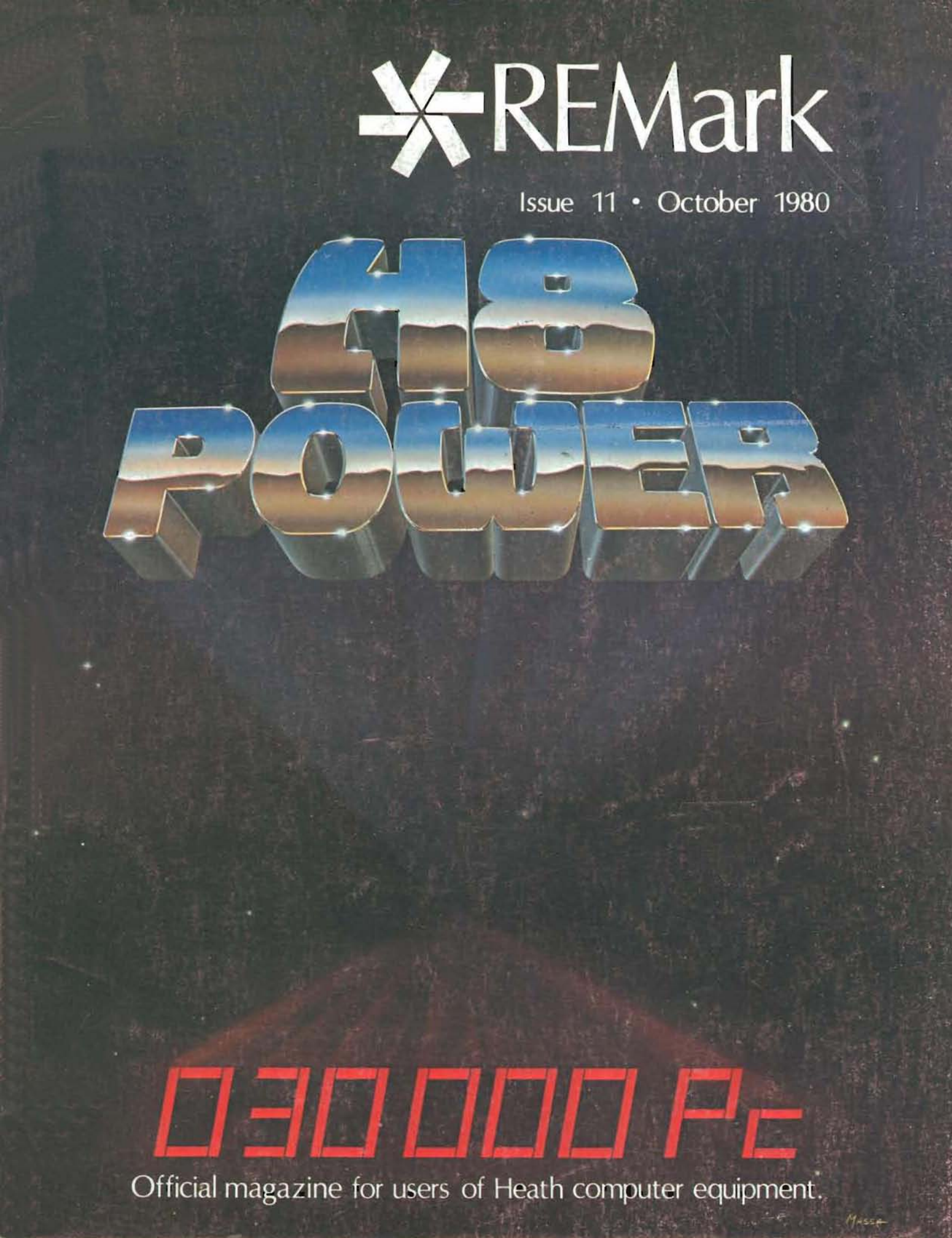
REMark

Issue 11 • October 1980

68 POWER



Official magazine for users of Heath computer equipment.

HOW ABOUT A POSTER FOR YOUR COMPUTER ROOM

on the stack

>CAT

"REMark" is a HUG membership magazine published quarterly. 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 available 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

Although it is a policy to check material placed in REMark for accuracy, HUG offers no warranty, either expressed or implied, and is not responsible for any losses due to the use of any material in this magazine.

Articles submitted by users and published in REMark, which describe hardware modifications, are not supported by Heathkit Electronic Centers or Heath Technical Consultation.

HUG Manager and Editor ..... Jim Blake
Assistant Editor and
Software Developer ..... Patrick Swayne
HUG Secretary ..... Nancy Strunk
Software Developer ..... Gerry Kabelman
Software Developer ..... Jon Falkner

Copyright © 1980, Heath Users' Group

HUG is provided by Heath Company as a service to its members for the purpose of fostering the exchange of ideas to enhance their usage of Heath equipment. As such, little or no evaluation of the programs in the software catalog, REMark or other HUG publications is performed by Heath Company, in general and HUG in particular. The prospective user is hereby put on notice that the programs may contain faults the consequences of which Heath Company in general and HUG in particular cannot be held responsible. The prospective user is, by virtue of obtaining and using these programs, assuming full risk for all consequences.

MicroNET and other SOFTSTUFF

In mid August the nationwide HUG dial up computerized bulletin board got on the air through the facilities of Compu-Serve or better known to most as MicroNET. MicroNET can be accessed from most cities via a local phone call. Initial subsription fee is $9.00 which gets you a users manual, special ID number and password and a list of phone numbers around the country through which MicroNET can be accessed. In addition to the many features available through the NET, HUG members have their own CBBS. In the short time it has been active, there have been almost 1000 sign-ons. Users can leave and retrieve messages to ALL other users or to specific individual. We have seen problems solved, questions answered within minutes. Users also can exchange software, access the HUG software exchange, and very soon, order and down-load some HUG software to your computer right on the spot. Connect time is $5.00 an hour and a major credit card is required for billing purposes.

While the HUG bulletin board is open to all users, a separate, but similar Bulletin Board is maintained on MicroNET by Chuck Sodian exclusively for H-11 users.

Access to all Special Interest Group BB's is accomplished after MicroNET sign-on by typing R SIGS.

As we go to press, the exact procedure for subscribing to MicroNET is not defined, but we expect to be offering it through HUG as a regular product. Call MicroNET at 614-457-8600 for details. Identify yourself as a HUG member.

Equipment needed to take advantage of the system is, at the very minimum some flavor of terminal and modem which would allow you to do everything but save files. To take full advantage, a computer and disk storage is desirable so that you can send and receive files. Some communications software is also needed such as HUG's MCS or a new SOFTSTUFF product called CPS. CPS does full error checking, has auto log-on and employs MicroNET and/or H8/H89 protocol.

The Bulletin Board is open to everyone, but only HUG members are recognized at sign-on and only HUG members will have access some HUG software. Hope to see you on soon.

A new word popped up in our vocabulary a paragraph ago. SOFTSTUFF. Let me tell you about it.

As you know, applications software for the HEATH systems has been provided principally by yourselves, HUG members. And that software is available to HUG members only.

To provide the facility to acquire more software for wider distribution and to reward the author beyond the HUG program, a new 'division' of HEATH was established. SOFTSTUFF. SOFTSTUFF products, may in fact, be software submitted by HUG members. Or software written by HUG members under contract to HEATH or it may be developed from scratch by HEATH software engineers. SOFTSTUFF is called ' affordable software tools for your computer.' They may not have a 100 pages of documentation that you would expect traditionally from HEATH, but sufficient documentation, and each product has been fully checked out and works as advertised. SOFTSTUFF is GOODSTUFF. These products are described page 15. And more neat stuff is coming. It will really RAIN software in the fourth quarter. If you have a piece of software or know of some that you think the rest of the users would be interested in, please contact me at 616-982-3835. It is hoped to offer most products that operate both under HDOS and

(vectored to page 24)

Getting Started with a Text Editor

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

One approach to the understanding and use of a "TEXT EDITOR" is documented using HUG's "ED" for illustrative purposes. The principles discussed can be applied to most microcomputer text editors.

INTRODUCTORY DISCUSSION

A "TEXT EDITOR" is a computer program which assists the user in the preparation of, and revision of, data. The data may be a letter, a manuscript, or any other material the user wishes to write and store for later use. (The original use of a computer "Text Editor" was for the preparation of Assembly language programs for the computer itself.) The data is usually entered into computer memory through a terminal keyboard, and after revision, may be transferred as a "data file" from memory to "permanent" storage (usually a "floppy disk", or cassette tape.) After any additional desired revisions are made, by transferring the data from "permanent storage" back to memory for editing with the Text Editor, the material is again transferred to "permanent" media. This process can be repeated as many times as desired. When all necessary revisions are completed, the SAME DATA may then be manipulated by a "WORD PROCESSOR". A Word Processor is another computer program that provides adjustments of margins, justification (making the right margin even by inserting spaces into the text automatically), centering of headings and other formatting niceties. This is accomplished with minimum effort by the user. ("RUNOFF" - HUG Part No. 885-1025 is such a Word Processor. Note that some relatively expensive "Word Processors" have a built-in text editing function.)

This article, for example, was written and revised using HDOS (Heath Disk Operating System) and HUG's "ED" EDITOR (HUG Part No. 885-1022). This article was revised about 34 times, before being processed by RUNOFF. Of course, many times it is not necessary to use a "Word Processsor" after preparation of text. Most letters do not require any word processing, for example. (But, as originally written, this article averaged about 70 characters per line, and the text you are now reading was the result of the processing by RUNOFF to the formatting standards of REMARK (42 characters wide).

One usually uses a Text Editor in two "stages". First, you use it to INSERT text (data) into computer memory and the data then becomes a disk or cassette file. While using the "Editor" for this purpose, you correct any noted typographical errors or mistakes by "backspacing and typing over". Second, after the text has been inserted into memory, you may REVISE it over and over again as noted above. It is this second use where you utilize most of the commands of the Text Editor that are available to you. It is also this "revision" stage that the newcomer to "Text Editing" finds most troublesome.

It can be very discouraging to the novice when his first attempts at "TEXT EDITING" fail completely, or when certain Editor Commands do not work the way the user thinks they should. I have worked with 7 different Editors and have had my share of frustrations. However, after varying amounts of effort I have been rewarded with success in my use of these Editors. Since the newest Editor I have used is HUG'S "ED", this article deals with this particular Editor, although the same general approach can be used with most Text Editors!

I have found HUG's EDitor to be one of the finest, and certainly the fastest, Editor I have used. I found NO bugs in it!

Any "problems" encountered all disappeared after sufficient experience was accumulated in actual use. HUG's "ED" is a particularly well documented Editor and the documentation covers all the Editor commands. However, in actual use, one uses only a minimal number of certain commands (or combinations of commands) again and again.

You will find that there are numerous single letter commands, and many of these can be used in combination. EACH COMMAND, OR COMBINATION OF COMMANDS, IS TERMINATED AND EXECUTED BY HITTING THE ESC (escape) KEY TWO TIMES.

When one starts out with a new Editor, it is always wise to learn how to use a limited number of commands (or command combinations) and to learn these few commands well.

One thing that may confuse you is that, with HUG's "ED", you execute all the commands with 2 ESCs. When you hit the ESC key, a $ sign is echoed on the terminal screen, so for the rest of this article when you see $, that means that an ESC was entered. All too often I have found myself hitting the RETURN key, expecting that the command would be executed. Not so. It must be 2 ESCs ($$)! This way the return key is saved for use in the text.

GENERAL APPROACH

The VERY FIRST thing to do is to thoroughly review the documentation (instructions) that you received with your Text Editor. In the case of HUG's "ED", the documentation is on the disk that contains the "ED" program. Also on the disk is the complete assembly language source program for "ED" so that advanced programmers may add their own embellishments to the Editor. When I read the documentation, I made short notes on a scratch pad. I specifically looked for, and jotted down, a few VITALLY NECESSARY COMMANDS:

FUNDAMENTAL COMMANDS

Command(s) to "INVOKE" the Editor and name (a new file) or access (an old file).

Command to "READ" a pre-existing old file into memory for revision. Command to "INSERT" text data (write material into computer memory.) Command to "EXIT" the "Insert" mode. Command to "WRITE" data from memory to disk and "EXIT" the Editor.

Here is what I wrote on my scratch pad:

```
>ED SY1:FNAME.EXT SY1:<cr>
#A$$
I--------<cr>
--------<cr>
$$
E$$
```

You invoke ED from the monitor prompt (>) with the command ED, AND you name a pre-existing file (or a new file) immediately after the space after you type ED. You also indicate which disk contains an old file (the first SY1: above, and name the disk which is to receive the revised or newly created file (the space followed by the second SY1: above.) Then you hit RETURN on the keyboard <cr>. (If you omit SY1: the default in either or both instances will be to SY0:.) The program loads, the file is "opened", and ED's prompt (-) (a dash) is displayed.

#A$$ "appends" or reads ALL the data from disk from pre-existing "FNAME.EXT" into computer memory, called "buffer" memory. (You don't need #A$$ if you are starting a new file from scratch.) The 2 dollar signs are automatically echoed by ED when you hit the ESC key 2 times. This is how you execute ALL of ED's commands.

"I" is the command to start INSERTing. EVERYTHING you type after the "I" will be put into computer memory. This will continue until you hit the ESC key 2 times (echoed on the screen as 2 dollar signs.) When you type the 2 ESCs after inserting one or more words, OR one or more lines, this puts you back in ED's command mode. I put several lines of dashes on my scratch pad to indicate several lines, and ended the Insertion with $$ which would terminate the Insertion.

E$$ exits the program ED after automatically writing your text to the diskette, closing the file, and also automatically renaming your original pre-existing file (unchanged) to FNAME.BAK. This is a handy feature.

The above is all there is to "Invoking" the Editor, "Reading" in pre-existing files, "Inserting" text, "Ending" text insertion, and "Exiting" the EDitor.

NEXT, I looked for and noted some important (to me) "POINTER-MOVING" and "TEXT DISPLAY" COMMANDS. My list and notes:

POINTER MOVING-TEXT DISPLAY COMMANDS

B$$ moves pointer to beginning of file. Z$$ moves pointer to end of file just past last character of text.

nT$$ displays 'n' lines. (If 'n' is omitted just displays one line on screen.)

nL or -nL (minus nL) moves pointer forward (or backward) n lines. 'n' is a decimal number. (If n omitted, default is 1 line.)

Funique word(s)$$ finds any unique word(s) in text and moves pointer to that line.

0LT$$ (zeroLT$$) - move pointer to beginning of line (0L) and type line on screen (T). A MOST IMPORTANT COMMAND!

NEXT, I noted the only TEXT REVISION COMMANDS I needed to get started editing:

TEXT REVISION COMMANDS

nK$$ deletes (Kills) n lines, including NL characters. (HDOS automatically puts a "New Line character" in text whenever you hit RETURN key.) If n omitted, default is one line.

Soldword(s)$newword(s)$0LT$$ Substitute new word(s) for old word(s), then move pointer to beginning of line and display the revised line.

```
I-----
-----
```

$$ this is the same Insert mode discussed previously.

LAST, I looked for and noted two "PANIC" COMMANDS:

PANIC COMMANDS

"Control U" deletes an entire command line (before execution with $$) and reprompts. (Note that this command will NOT delete a multiple line insertion.)

Q$$ allows you to "Quit" the edit and RETURN to monitor prompt with everything exactly as it was before you invoked the Editor!

WITH THE ABOVE NOTES FOR REFERENCE I WENT TO THE COMPUTER, SAT DOWN AND BEGAN TO LEARN HOW TO USE ED!

DISCUSSION

I have learned that the ONLY way to really learn how to use an Editor is by actual use, entering a very short listing and manipulating the lines and words. My favorite text for learning purposes is entered and terminated like th :

```
Ithis is the first line
this is the second line
this is the third line
this is the last line
$$
```

The 'I' means that EVERYTHING after this command is treated as text Insertion, UNTIL the insertion is ended by hitting the ESC key 2 times. The 'I' command can also be entered as a lower case 'i' (this is true of all commands - upper or lower case entry is valid for all of them.)

The above is how you insert text into memory. Later, the text can be transferred to disk as a "data file".

THE "POINTER"

The main problem most novices have occurs at this point, AFTER text has been inserted, or AFTER text has been brought in from a previous file for editing. This main problem presents itself to the user who asks himself "Where, in this text, where am I?". The user is usually aware that somewhere in this text that appears on the screen is an illusory and invisible (but nevertheless actual) "POINTER! But, WHERE is this pointer?

THE POSITION OF THE POINTER USUALLY HAS NO RELATIONSHIP TO THE POSITION OF THE CURSOR!! The POINTER POSITION can be determined by having the editor type out (display) a line of text. The pointer position is JUST BEFORE THE FIRST CHARACTER of any single line of text printed out!

HERE IS HOW TO LOCATE THE POINTER. You enter T$$ (T for type, and then 2 ESCs.) Whatever appears on the screen is whatever is present just AFTER the pointer. Suppose NOTHING appears (this is common). Now enter 0LT$$. This combination of commands is one of the most important of any combination of commands. Repeating, it is 'zero'LT$$, and entered as 0LT$$. This puts the pointer at the beginning of the line where the pointer was, and types out the whole line for the user. Now, the pointer is AT and remains at the beginning of this line. You can type T$$ and whatever was just printed out for you will be displayed again.

However, suppose that nothing was typed out following 0LT$$. And, this is a common occurrence. All that this means is that the POINTER is either before any text, after any text, OR is at the beginning of a 'blank' line.

Simply enter BT$$. This puts the pointer at the beginning of the FIRST line of text material and types out that line. If nothing is typed out, simply hit the RETURN key one or more times. This moves the pointer down one line and prints this line, each time the RETURN key is hit! Continue hitting the RETURN key until you are at the beginning of the desired line. NOTE THAT THE POINTER IS ALWAYS JUST TO THE LEFT OF THE FIRST CHARACTER OF THE LINE THAT HAS JUST BEEN DISPLAYED! (Don't forget that you MUST read text with the "#A" command if you are dealing with a pre-existing file, before you can display or edit the text file!)

So, please remember 0LT$$. After you enter most commands, always use 0LT$$. Most of the time this will result in a line of displayed text, AND the pointer is at the beginning of this line.

TEXT REVISION

Suppose you want to enter a line BETWEEN 2 pre-existing lines of text. You position the pointer at the beginning of the line which WILL be located just AFTER your desired insertion. You move the pointer backward through the lines of text with -nLT$$ (minus nL T 2 ESCs) and the n is a numerical value which tells the Editor how many lines back you want to move. Example: -2LT$$ moves the pointer back 2 lines and types out the line for you. The pointer remains at the beginning of the line typed out for you. OR, you can move the pointer forward, line by line, by hitting RETURN key, or you can move it forward n lines (and print out the nth line) with nLT$$. In this latter instance the pointer remains at the beginning of the nth line. So, once you have the line displayed, BEFORE which you wish to enter a new line, simply type Iand insert your new line <cr>$$. In this instance we typed I (for Insert) and our inserted line was "and insert your new line" followed by hitting the RETURN key, and then 2 ESCs were entered to execute the insertion. Now, type 0LT$$ and see

what is printed out. It should be the line AFTER your insertion. So, type -4L8T$$ and this moves the pointer backward 4 lines (the POINTER remains at the beginning of the -4th line) but it and the next 8 lines are printed out for you. Now, you can see your insertion. (You can, of course, insert more than one line simply by continuing to type lines, each line followed by a <cr>, until you terminate the insertion with $$.)

Suppose you wish to add text to the end of your pre-existing text material. Simply go forward with the pointer, moving it with the nLT$$ or Z$$ commands, or hit the RETURN key the desired number of times until you see the LAST line of your pre-existing text displayed. Type 0LT$$ for verification. You should still see the last line of pre-existing text. The pointer is at the BEGINNING of this line!! So, type L$$. This moves the pointer to the beginning of the line AFTER your last line (note that nothing can be displayed at this point so you don't type "T" as part of the command - you can, but nothing can be displayed). Now, type Iand immediately insert your text as usual. Frequently, at this point the first insertion will be a <cr> to leave a blank line between your pre-existing LAST line and the new text, so hit the RETURN key and enter all the desired text until you are finished, hit the RETURN key and type 2 ESCs. Done!

How about correcting mistakes in text? YOU WILL FIND THAT MISTAKES WILL BE CORRECTED QUICKER BY USING ONLY A LIMITED NUMBER OF COMMANDS. YOU DO NOT REALLY EVER HAVE TO USE THE nC OR nD COMMANDS. YOU USUALLY CAN ALTER TEXT FASTER BY USING THE VERY FEW REVISION COMMANDS WE NOTED. Here is how to use these few commands:

Move the pointer to the beginning of the line that needs correcting. Do this by the -nL or nL commands, or the B or Z commands (don't forget that hitting the RETURN key "scrolls" down and displays next line). When you are at the beginning of desired line, look at it and decide whether the correction can easily be done by changing one or more words, OR whether it would be simplest just to do the entire line over. If the latter, simply type K$$ (which deletes the line) and Inow type the desired line<cr>$$. You will find the original line has been deleted and replaced with the desired line. Note that you can do the same thing with K$Idesired line<cr>$$. On the other hand, if correction is only for one or two words, simply use the S command this way (and make sure pointer is at beginning of line first by typing 0LT$$):

Sold word $new word $0LT$$

The S tells Editor we are going to Substitute; enter EXACTLY the old word(s) that need correcting and hit ESC key; enter EXACTLY the new word(s) and hit ESC key; now enter 0LT$$ and corrected line is displayed. From a practical standpoint you just don't ever have to worry about moving the pointer ' rward or backward with the nC or -nC co...ands, nor do you have to nD or -nD to delete characters. It is usually faster just to position the pointer at the beginning of the line in question, and then either delete the whole line using K and Insert whole new line; or, use the S command to substitute one or more words! (Of course, you can use the C and D commands, ' you don't really need to, and most c e time you will find that my suggestic esult in a quicker Edit change!) again, in the S command, that we $$ (zeroLT$$) to move the pointer eginning of the line and type or us. This is very importa se, after the S command, the F s located AFTER the substi1 T moved it to the beginn: ine again. To delete one or more :hout substitution) enter - Swor deleted$$. Here, enter 0LT$$ A ESCs of the S command.

Sometime. ay need to just delete one or more l. in pre-existing text. The command is nK$$. K means "kill". The n is a positive or negative number stating the number of lines backward or forward to be deleted (killed). For example, 2K$$ deletes 2 lines, the line which the pointer is at the beginning of, and the next line.

The other main command you will find extremely useful is the "F" command. This will find any UNIQUE word or words in your text. The F command also leaves the pointer just AFTER the desired unique word or words, so it is always important to use the F command immediately followed by the 0LT command, like this:

BFunique expression$0LT$$

This says "Find the unique expression, then move the pointer to the beginning of that line and print it out". The pointer stays at the beginning of that line, incidentally. The "B" command was used just before the "F" command to ensure that the Editor looked for the unique expression starting at the Beginning of the text.

ONE ADVANCED COMMAND

The documentation explains how you can use a "MACRO" to change a word(s) throughout a long text with just one set of commands. However, the documentation illustration will only change the desired word a maximum of once per line of text. Suppose

you are editing a BASIC program (very long and very complicated) and you have, say, a couple hundred occurrences of a variable called A3. Suppose you desire to change ALL occurrences of A3 to A4! Here is the "Macro" which will do the job:

B5000<5000<SA3$A4$>L>$$

Or if you had a BASIC program that you wanted to change all of the occurances of SY1: to SY0: the command would look like this:

<SSY1:$SY0:$$>$$ Done!

SUMMARY

Whenever you are in the EDitor, you are in the "I" (Insert) mode (which is where you do your entering of data, text, whatever) IF you typed "I" to enter the Insert mode. You terminate the Insertion by hitting the ESC key 2 times (echoed on terminal screen as 2 dollar signs - $$). If you are NOT in the "Insert" mode, then you are in the "Command" mode where you can use all the rest of the "ED" commands for text revision and other manipulations. All these other commands are also terminated and executed by hitting the ESC key two times.

You can successfully and quickly use ED with only minimal commands as long as you know WELL how they are used. "I" (Insert) is always terminated at the end of the insertion (may be many lines, and usually the last inserted 'character' is a <cr>) by 2 ESCs. B$$ will always move the pointer to the very beginning of your text. Z$$ will always move the pointer just past the very LAST character in your text (and remember that the last character in your text is apt to be a NL character cr>). You can also move the pointer backward and forward with -nL or nL, frequently entered as -nLT$$ or nLT$$ (so that you can see the displayed line). Or, you move pointer forward by hitting the RETURN key or by using the FIND command (Funique expression$0LT$$). Use 0LT$$ liberally!! Correct by using the "S" command for small changes, or "K" and "I" commands (delete and insert) to replace the entire line.

Remember that the pointer is always present. You can NOT see it. It may or may not be at the cursor location. But, it is ALWAYS at the beginning of any line which can be displayed with the 0LT$$ combination of commands!!

ENJOY YOUR EDITOR!

A FINAL NOTE

I have an H89 with 48K of memory. If you have a minimum of memory in your system, be sure to check the documentation for a few additional commands such as "W" (Write), and review all of the section on "A" (Append).

EOF

Microsoft BASIC Output Flexibility with Heath-Configured CP/M*

John R. Thomas
9312 Frostburg Way
Gaithersburg, Md. 20760

This article describes a method that can be used with Microsoft Basic to select whether outputs (print statements and program listings) are sent only to a CRT screen or to both a CRT screen and a hard-copy line printer. An often desirable feature in microcomputer operations is the ability to send (copy) output to either a CRT device or to a line-printer device, with device selection under user or program control. Such a selection feature can be employed in program development and debugging, printing selected subsections of programs on demand, or printing a hard copy of program or game instructions only when desired. The selection method outlined here is concerned with Microsoft Basic operating with CP/M as configured for Heath systems.

When operating in the CP/M command system a Control-P typed on the console allows all subsequent console output to be concurrently output on the assigned listing device. Output is sent to both the listing device and the console until the next Control-P is typed. The capability of this important functional feature to switch between output devices as desired is not maintained when operating with Microsoft Basic. One can of course send outputs to either the CRT screen or to the line printer in Microsoft Basic. It is the control over selection that is missing. A program can be easily listed on either the CRT or the listing device by using the Basic "List" or "Llist" statements. However, to output print statements to one or the other, one must use either the "Print" or "Lprint" command. To switch output from the CRT screen to the line printer requires that all print statements in the program be changed to "Lprint" which at the minimum requires the use of an editor if each statement is not to be changed one at a time. Neither of the above two ways of switching output copies output to both the CRT and the line printer. Output is to one or the other.

The capability to switch back and forth as desired between the CRT alone and the CRT plus the line printer can be accomplished in Microsoft Basic operating with a CP/M system by the use of the CP/M IOBYTE function. The IOBYTE function allows for

assignment or reassignment of logical input and output devices to a number of different physical devices. It is through the use of this IOBYTE function by which one had flexibility to switch output devices under program or user control with Microsoft Basic. In the Heath-configured CP/M system from LifeBoat Associates, Version 1.43, the eight-bite IOBYTE function is found at memory location 16899 decimal (4203 hex). The function is initially configured on the CP/M distribution disk such that the system console is assigned to the CRT. Communication with the user, both input and output, is normally through the CRT. Another possible configuration exists in the CP/M User Area under the physical device name of UCI: (user-defined console). When this device is assigned to the console logical device, the user communicates input through the CRT as usual, however, all output that would have gone to the CRT screen now goes to the listing device. (The listing device is configured as the line printer on the distribution disk). The IOBYTE function can be changed in the CP/M command system by the Stat command. The command, STAT CON:=UCl: will change the console output device to the line printer and the command, STAT CON:=CRT: will change it back to the CRT screen. This same type of flexible switching of output devices can be accomplished easily in Microsoft Basic by the use of the "Poke" command to change the IOBYTE value. The initial IOBYTE value as supplied on the distribution disk is set at 129 decimal (81 hex) which assigns the console to the CRT. By executing the command, "Poke 16899, 131", further output will appear at the line printer and by entering the command, "Poke 16899,129", output will again appear on the CRT screen. One can switch back and forth as desired between these two output states in either the direct or the indirect Microsoft command mode.

With the above manipulation of the IOBYTE function, output is sent to either the CRT screen or to the line printer. When output is to the line printer, however, it does not appear simultaneously on the CRT screen. This may present some difficulties as it may be hard to read the current output line on the line printer without advancing the paper. A small change in the line-printer output section in the CPM User Area will allow concurrent output on both the line printer and the CRT screen when the UCl: is invoked. The change involves the addition of a conditional jump from the line-printer routine to the CRT-output routine if the Basic "Poke" statement or the Stat command sets the CON:=UCl:. The modified line-printer output routine for the CP/M User Area follows at the right.

```
                        ;LP OUTPUT ROUTINE
LPTOUT  IN    H84LPT+6  ;8250 MODEM STATUS
        ANI   10H       ;MASK CTS BIT
        JNZ   LPTOUT    ;IS H14 READY
        IN    H84LPT+5  ;IS 8250 READY
        ANI   20H       ;HOLD REG EMPTY BIT
        JZ    LPTOUT    ;WAIT IF NOT READY
        MOV   A,C       ;BOTH ARE READY
        OUT   H84LPT    ;PRINT CHAR ON LP
        LDA   IOBYT     ;GET I/O ASSIGN BYTE
        CPI   83H       ;SEE IF CON:=UCl:
        JZ    CRTOUT    ;IF SO PRINT ON CRT
        RET
```

*CP/M is a registered trademark of Digital Research, Inc.

EOF

Local HUG News

NOTICE TO ALL LOCAL HUG GROUPS:

HUG is in the process of forming a master list of all local HUG groups. Presently we have a list of over thirty groups, however the information on most of these groups is very old and needs to be updated. We need your help in preparing this master list. The information needed is as follows:

Club Name:
Geographical Area:
Club Officers:
Club Mailing Address:
Meeting Location:
Meeting Time:
Approximate Membership:
Phone Number:

An officer of each local group should get together the needed information and mail it to HUG CLUBS, Hilltop Road, St. Joseph, MI 49085.

PUERTO RICO

Anyone interested in forming a local group in Puerto Rico should contact Norberto Collado Rivera, Box 765, Rosario, PR 00746. Norberto is also interested in sharing Spanish programs with any senores or senoras.

CLEVELAND, OHIO

CHUG meets every other Thursday at 7 PM. Contact the Cleveland Heathkit Electronics Center for details at 216-292-7553.

Denver, Colorado

Alferd K. Carr at 621 Cherry Street, Denver, CO 80220 would like to get a local H8 or H89 NOVICE users club started. His phone number is 303-321-6289.

Keep It Simple, Stupid (KISS)

You there, in the back of the class, waving your hand. Did you say that you wanted to know more about the PUT, GET, FLOAD and FDUMP commands? Okay, we'll show you a neat way to imbed them right into the program and let the computer do everything but title the data files and place the cassettes in your cassette recorder (we WILL show you how to include a sort routine and convert this to disk BASIC eventually----honest).

By the way, if you are intending to upgrade to HDOS, don't include the following modifications in your mailing list program. Disk BASIC does not have the same input buffer addresses as Cassette BASIC, so running this program could cause some disastrous crashes.

First, we'll cover some of the basics (no pun intended).

When you're running a cassette-based routine, such as Extended Benton Harbor BASIC, any command (while in the command mode) that you give it is processed by a program called the Console Driver. Every time you strike a key, that character is stored in a memory buffer location called "$INBUF". This is a 31 byte-long storage area where 2nd through 31st locations hold the characters that you type while the first location keeps track of how many characters were entered.

For example, when in the command mode, you type:

```
PUT "THIS IS A DATA FILE"
```

The above six words (string) will be placed in $INBUF where it will be fetched by BASIC. BASIC will then perform the 'PUT' command by dumping the contents of all its variables onto cassette tape.

$INBUF isn't used when processing a program statement when a BASIC program is running. The program statements are located elsewhere in memory.

That is partly why the program statement,

```
10 PUT "THIS IS A DATA FILE"
```

will generate an error message when it is encountered.

If you could somehow slide that 'PUT' command into $INBUF while BASIC is running a program, you can make it work without causing a crash. There is a way you can do it. BASIC has a program-mode statement called POKE that allows you to modify a memory location while the program is running. Its form is:

```
POKE (address),(data)
```

Where the address is the desired location in $INBUF (one location per character, remember) and, in this case, the data is the numerical equivalent of the character we want placed there.

But wait, you can't type "$INBUF" for the address; the POKE command requires a decimal number here. How do you find the right address? Not only that, this address may be different in different versions of Extended Benton Harbor BASIC.

There's an easy way to find this address. Open your Cassette System Software Reference Manual and turn to the "Console Driver Documentation" in the introduction (Chapter 0). Locate the assembly language listing (that's the one that you have to turn the book sideways for) and turn to the last page in this listing. Here you'll find a cross-reference (XREF) that shows, among other things, the locations of the Console Driver subroutines.

Look down the left column until you find $INBUF. Immediately to the right of it will be the address. If you have BASIC version 10.06.00, this will be address 040130 in split-octal. However, the POKE command requires that this address be in decimal.

A quick way to convert this can be found in Appendix D of your Software Reference Manual (Chapter zero). You simply look up the first three numbers of the above address (040) and find the decimal equivalent of '036'. Multiplying this by 256 gives 8192. Now look up the decimal equivalent of octal 130. This turns out to be '088'. Adding 8192 to 88 gives 8280 which is the decimal location of $INBUF.

Okay, you got the address, now how do you convert the characters on the keyboard into numbers that the computer can understand? This too, is easily solved. The characters are represented inside the computer by a type of code known as ASCII (for American Standard Code for Information Interchange). The decimal number representing the ASCII characters can be found in Appendix 'C' of chapter zero of your Software Reference Manual.

So, to use the 'PUT "THIS IS A DATA FILE" ' function, you can write your program like this:

```
5 Z9=8280:REM Z9 = the address for ease of changing with updated BASICs
10 REM The PUT command
20 POKE Z9+1,80 :REM the letter 'P'
30 POKE Z9+2,85 :REM the letter 'U'
35 REM: "POKE (address),84" ('T') isn't needed--Command Completion, remember?
40 POKE Z9+3,84 :REM the letter 'T' (for THIS IS A DATA FILE)
50 POKE Z9+4,72 :REM the letter 'H'
60 POKE Z9+5,73 :REM the letter 'I'
```

And so on until the entire 'THIS IS A DATA FILE' string was POKEd into the buffer. The last POKE instruction you'd give, would be 'POKE Z9,21' which goes into the first address of $INBUF and indicates how many characters were entered.

Even using multiple statements per line, this can eat up a lot of memory. Let's use some FOR-NEXT loops and some fancy string manipulation to cut down on space and make this routine more flexible (so you can use it for other commands). AT LAST! We're getting away from the theory and going to a practical application!

We'll start out by deleting lines ten through 60 above and replacing this area with the commands that we would want to implement in our program.

```
5 Z9=8280:REM $INBUF address (#10.06.00)
10 Z8$="UN"+CHR$(13)+"GE":REM ***GET*** command
15 Z7$="UN"+CHR$(13)+"LOA":REM ***LOAD*** command
20 Z6$=CHR$(13)+"YCON"+CHR$(13):REM ***GET or LOAD*** confirmation
25 Z5$="PU":REM ***PUT***
30 Z4$="DU":REM ***DUMP***
35 Z3$=CHR$(13)+"CON"+CHR$(13):REM ***PUT or DUMP*** continue
```

Line 10 equates the string variable 'Z8$' to the 'GET' command. When in BASICs command mode, you would UNlock the interpreter to allow loading the new variables, hit the RETURN key (represeted by CHR$(13)) and then 'GE' indicates that you want to GET a file. Another portion of the program will supply the file name. Line 15 does the same thing for the LOAD command (Z7$).

Line 20 must be used in conjunction with the GET or LOAD command. It provides the carriage return after the file name that you'd normally supply manually by hitting RETURN. The computer would then ask you if you're sure; you'd respond by tapping the "Y" key. In this line, the first letter of "YCON" will do that for you. "CON" will tell the program to continue after the file is loaded.

Lines 25 and 30 do the same thing for the PUT and DUMP commands, while line 35 will initiate the command by sending a RETURN and the CONTINUE command.

Before getting to the nitty-gritty of POKING these commands into $INBUF, we'll show you how to supply the file name to be used with the above strings. This is done starting at line 5000 below.

```
5000 REM This routine will *GET* a file from tape
5010 LINE INPUT "What is the file name? ";Z1$
5020 Z9$=Z8$+"Z1$"+Z6$:REM *Z8$* & *Z6$* indicates *GET*
5025 REM Z9$ is the same as 'GET "(your string)" (return) Y CONTINUE'
5030 REM The quotes around Z1$ prevents BASIC from generating an error message.
```

The above lines will ask you for a file name (Z1$) and then combine it with Z8$ and Z6$ to form the complete GET operation command. However, it won't do anything yet. To make it do something, we've got to POKE this information into $INBUF and then STOP the program. Stopping the program allows BASIC to look at $INBUF to see if any commands are there. So we'll add the following lines:

```
5040 GOSUB 65000: REM We'll put the POKE subroutine here out of the way.
5050 STOP :REM Allow access to $INBUF
5060 ( return to the main loop )
```

The actual POKING subroutine will start at line 65000. When the computer returns from this routine, it will encounter the STOP command at line 5050 and will stop. The computer will look at $INBUF and execute the GET function. Once the data is loaded, the program will automatically CONTINUE starting at line 5060, which can be a command to return to the main program loop.

Here is how the program would look if you wanted to implement the PUT command:

```
5100 LINE INPUT "What is the file name? ";Z1$
5110 Z9$=Z5$+"Z1$"+Z3$:REM *Z5$* & *Z3$* indicates *PUT*
5120 GOSUB 65000
5130 STOP
5140 ( return to the main program loop )
```

Can you see the major difference between this and the GET routine?

Now we'll show you how one subroutine starting at line 65000 can handle both of these commands.

```
65000 REM Tape manipulation routine
65010 FOR Z0=Z9+1 TO Z9+LEN(Z9$):REM poke in the command
65020 POKE Z0,ASC(MID$(Z9$,Z0-Z9,1)):REM one letter at a time
65030 NEXT Z0:REM almost done
65040 POKE Z9,LEN(Z9$):REM include string length
65050 RETURN :REM all done!!
```

Line 65010 is a FOR-NEXT loop that starts one location after the first $INBUF address and will count out the number of successive memory locations that are required for the command. This number is determined by finding the length of the command (Z9$) set up in the line-5000 routines. The variable, Z9 is the address value of $INBUF as set in line 5 previously.

In line 65020, the data indicated as ASC(MID$(Z9$,Z0-Z9,1)) is POKEd into the current $INBUF address held in variable Z0. A single character from Z9$ is extracted with the MID$(Z9$,Z0-Z9,1) statement. The current address that the FOR-NEXT loop is pointing at (Z0) is subtracted from $INBUF's starting address (Z9) to tell the MID$ command how many characters from the left of the string variable, Z9$, it should look to find the desired letter. The 'ASC(' statement converts this letter into a number acceptable to the POKE command.

Line 65030 increments the Z0 pointer to the next address location and loops if all the specified locations haven't been looked at. This looping continues until all the characters in Z9$ have been POKED in successive memory locations.

Line 65040 will then find the length of Z9$ and store it in the first location of $INBUF before returning to the calling program.

And finally, here's how this function can be added to the mailing list program:

```
5 Z9=8280:REM $INBUF address (#10.06.00)
10 Z8$="UN"+CHR$(13)+"GE":REM ***GET*** command
15 Z7$="UN"+CHR$(13)+"LOA":REM ***LOAD*** command
20 Z6$=CHR$(13)+"YCON"+CHR$(13):REM ***GET or LOAD*** confirmation
25 Z5$="PU":REM ***PUT***
30 Z4$="DU":REM ***DUMP***
35 Z3$=CHR$(13)+"CON"+CHR$(13):REM ***PUT or DUMP*** continue
100 IF N>0 THEN 200:REM Why do this?
110 DIM A$(100,6),T$(6)
120 FOR I=1 TO 6:READ T$(I):NEXT I
130 DATA "Name","Address","City  State, Zip","Phone","Bus. Phone","Notes"
```

```
200 REM clear the screen & print the MENU
202 FOR I=0 TO 24:PRINT :NEXT I:REM clear screen
204 PRINT TAB(25);"M E N U"
206 PRINT TAB(25);"-------"
208 PRINT :PRINT
210 PRINT TAB(20);"1)  List all names"
212 PRINT TAB(20);"2)  Add names to the list"
214 PRINT TAB(20);"3)  Edit a record in the list"
216 PRINT TAB(20);"4)  Get a Data File from cassette"
218 PRINT TAB(20);"5)  Put a Data File onto cassette"
220 PRINT TAB(20);"6)  Stop"
222 PRINT :PRINT
224 LINE INPUT "Enter the number for the desired function: ";A$
226 A=VAL(A$):REM help cut down on errors
230 ON A GOTO 1000,2000,1200,5000,5050,3000
295 REM If he doesn't know what to do, give him another chance.
300 PRINT :PRINT :PRINT "I don't understand that (hit RETURN)..."
310 PAUSE :GOTO 200
1000 REM He wants to list all the names; set up a counter
1020 FOR I=1 TO N:REM N should equal how many names we have. Right?
1030 FOR J=1 TO 6:REM we need to use the headings again
1040 PRINT T$(J);":";TAB(14);A$(I,J):REM Line it up in neat columns.
1050 NEXT J:PRINT :NEXT I:REM *print* allows readability
1055 PRINT "Tap the RETURN key when ready...";:PAUSE :REM give'em time
1060 GOTO 200:REM all done!
1200 REM **** EDIT LIST ****
1230 FOR I=1 TO N
1240 FOR J=1 TO 6
1250 PRINT T$(J);":";TAB(14);A$(I,J)
1260 NEXT J
1270 LINE INPUT "Do you want to EDIT this record? <NO> ";Z$
1280 IF LEFT$(Z$,1)="Y"OR LEFT$(Z$,1)="y" THEN 1300:REM allow lower case
1290 NEXT I:GOTO 200
1300 FOR J=1TO 6
1310 PRINT T$(J);":";TAB(14);A$(I,J)
1320 LINE INPUT "New Data: ";A$(I,J)
1330 NEXT J
1340 NEXT I
1400 I=I-1:IF I=N THEN 200:REM prevent crash after edit last record
1405 NEXT I
1410 GOTO 200
2000 N=N+1:PRINT :REM Set up a counter called 'N' and PRINT A BLANK LINE
2010 PRINT T$(1);:LINE INPUT ":  ";A$(N,1):REM Print NAME on screen
2030 IF LEN(A$(N,1))=0 THEN N=N-1:GOTO 200:REM Check to see if user is done.
2040 FOR I=2TO 6:REM Now print address, city state etc.
2060 PRINT T$(I);:LINE INPUT ": ";A$(N,I):REM and file away in memory.
2070 NEXT I:REM keep going until we get to "Notes"
2080 GOTO 2000:REM Bump the counter and get next name.
3000 REM Quit
3010 PRINT "If you change your mind, type CONTINUE..."
3020 STOP
3030 GOTO 5:REM prevents destruction of data in memory
5000 REM *GET* the data file from cassette
5002 PRINT CHR$(7);"This will destroy any current data files in memory!"
5004 LINE INPUT "Do you wish to continue? ";A$
5006 IF A$<>"YES"THEN 200:REM error trap
5007 PRINT "Place the Data File cassette into your player, press the 'PLAY'"
5008 PRINT "button and hit RETURN when ready.":PAUSE
5009 PRINT
5010 LINE INPUT "What is the Data File name? ('RETURN' if not known)";Z1$
5020 Z9$=Z8$+"Z1$"+Z6$:REM *Z8$* & *Z6$* indicates *GET*
5030 GOSUB 65000:REM get the file
5035 STOP :REM necessary during *get* operations
5040 GOTO 200:REM got the file
5050 REM *PUT* the data file onto cassette
5060 PRINT "Place a blank cassette into your recorder, press the 'record'"
5070 PRINT "button and hit RETURN when ready.":PAUSE
5080 PRINT
5090 LINE INPUT "Please enter the Data File name: ";Z1$
5100 IF Z1$="" THEN PRINT CHR$(7);:GOTO 5090:REM should have a name
5110 Z9$=Z5$+"Z1$"+Z3$:REM *Z5$* & *Z3$* indicates *PUT*
```

```
5120 GOSUB 65000:REM put the file
5125 STOP :REM necessary during *put* operations
5130 GOTO 200:REM got the file
65000 REM Tape manipulation routine
65010 FOR Z0=Z9+1 TO Z9+LEN(Z9$):REM poke in the command
65020 POKE Z0,ASC(MID$(Z9$,Z0-Z9,1)):REM one letter at a time
65030 NEXT Z0:REM almost done
65040 POKE Z9,LEN(Z9$):REM include string length
65050 RETURN :REM all done!!
```

We have been doing some editing to this program. The most visible change is to the main program loop starting at line 200. After using a FOR-NEXT loop to clear the screen, we print a MENU to more clearly display the available options. This also makes it easier to add new functions to this program at a later date. After selecting the number for the function that you want, the 'ON GOTO' statement will transfer control to the program line indicated by that number. If it isn't a valid MENU number, the computer will again print the MENU.

The GET and PUT routines starting at line 5000 have been expanded to include prompts for handling the cassettes and some error-checking functions to make sure that you don't accidently destroy any data.

If you've already typed in the mailing list program from the previous issues of REMARK, in addition to adding the new lines above, you should check lines 300, 310, 1050, 1055, 1280, 1400, 1405, 3010, 3020 and 3030 against the above. These have been edited to correct a couple of minor bugs and clean up the display a bit. What each change does is documented with REMs.

Now, how about that sort routine and conversion to disk BASIC?

EOF

Changes to Doc Campbell's Article

(REMark Issue #10)

Note the following changes in Doc Campbell's article on Random files in REMARK Issue #10, May 1980.

Page 4 - Right column, 3rd paragraph.
```
Change: LASTNAME\FIRST\STREET\CITYSTATEZIP
To:     LASTNAME\FIRST\MR\STREET\CITYSTATEZIP
```

Page 4 - Right column, last line.
```
Change: R$="LAST\FIRST\MR\STREET\CITY\STATEZIP"
To:     R$="LASTNAME\FIRST\MR\STREET\CITYSTATEZIP"
```

Page 5 - Right Column, 11th line.
```
Change: "PRINT   2," in lines 120 through
To:     "PRINT  #2," in lines 120 through
```

Page 6 - Right column, 3rd line (Program 3 listing).
Add after line 20 -
```
22 CLEAR 1000
```

Page 9 - Program 7, line 180.
Note that the end of the this line contains "." in the INSTR statement. In some issues of REM, this looks like " ". There should be a period between the quotes.

Page 14- Program 15 at the bottom of page.
Change the numerical value 70 to 90 in lines 100, 140, 145, 190, and 280.

Miscellaneous -
Throughout the text two unfortunate errors occurred. A left bracket was printed wherever a less than symbol (<) was present. A long vertical line was printed wherever the greater than symbol (>) was present in the orignal text. This did not happen in any of the programs, only the text material.

Thanks DOC

New HUG Software

HUG P/N 885-1083 DISK XVI MISCELLANEOUS
H8/H17//H89/ASM/ABS/32K $20.00

NOSPACE will delete the need to type spaces every time you boot up a disk. Requires H89 or H8 using a H8-4 I-O card.

WCT. Wordcount program counts the words in a RUNOFF text file. Only counts words that are to be printed.

TTREAD are for those without printers, it will remove linefeeds and formfeeds from an ASCII file, so it can be read on the console terminal without the jumping that comes from paging.

MOUNTALL will mount all disk drives in your system that have disks in them and the doors are closed. Also will ignore the wrong type of media. For a two or three (five inch) drive system only.

RECOVER will recover a file from a disk a sector at a time and place it on another disk, even if the disk is not mountable. The user will still have to know which sectors to recover in order to save a program from a bad disk.

HUG P/N 885-1206 CP/M GAMES DISK
H8/H17/H19//H89/CPM/BASIC-80/48K $21.00

This disk contains the CP/M Microsoft(tm) version of the games found on HUG disk XII MBASIC games disk (885-1068). Refer to disk XII for information on these programs. Not all programs are included on the CP/M version due to limitations imposed by the CP/M operating system. The following programs are included in this release:

SLOTS.ASC	OTHELLO.ASC
WUMPUS.ASC	DORNBACK.ASC
HANOI.ASC	QUBIC.ASC
NIM.ASC	TICTAC.ASC
HANGMAN.ASC	BATTLE.ASC

All programs make use of the graphics of the H19 terminal (or H89). Most but not all will run in less than 48k of memory.

HDOS 1.6 DEVICE DRIVERS

The Device Driver Disk HUG P/N 885-1019 has resently been updated for version 1.6 of HDOS. A copy of the new version may be ordered using the same HUG part number 885-1019, as all current stock at HUG has been updated to version 1.6. The selling price remains $10.00.

SOFTSTUFF™
Affordable Software Tools

SOFTSTUFF,
established by Heath Company, offers you a selection of software tools at affordable prices. All SOFTSTUFF programs have been checked and confirmed on the hardware indicated. Documentation, though not as extensive as standard Heath Company documentation, has been completely reviewed and judged acceptable. All SOFTSTUFF products come on a 5¼-inch diskette, unless otherwise stated. Specify HDOS or CP/M when ordering. For value and performance...SOFTSTUFF is good stuff.

General Ledger II:
Includes powerful programs for entry, maintenance, reporting and analysis of accounting data. Features include: **Custom Chart of Accounts** for determining account names and numbers. Any numbering system may be used, with or without decimal notation. **Comprehensive Printouts** upon request.96-column. **Double Entry/Automatic Entry Checking** automatically checks equality of debits and credits with each entry. **Simple Data Entry,** with one quick keystroke. **Account Verification** helps guard against mistakes by preventing entries to non-existent accounts and rejecting account numbers already in use. **New Account Facility** lets you open new accounts any time during data entry with no disruption to the transaction being entered. **Balance Reporting** lets you call the balance of any account to the terminal during data entry. All balances are instantaneously updated with entry of new transactions. **Audit Trail** for a source number and free-form description. One of the easiest-to-use, most flexible systems you'll find anywhere. Sample printouts and program listings included. Requires Microsoft BASIC. **HDOS #SF-9004: $124.95. CP/M #SF-9104: $124.95 (8" disk). Manual only, #595-2530: $15.00 (refunded when complete package is purchased).**

Full Screen Editor:
Uses H89 or H19 screen. Cursor motion keys position the cursor so changes can be typed anywhere on the screen. Function keys perform character and line insert and delete, string search, move and copy single and multiple lines, and scrolling of text in the window. For H89 and H8 + H19. **HDOS #SF-9000: $49.95. CP/M #SF-9100: $49.95.**

Text Formatter:
Performs fill and justification (straight right margins) of text previously prepared by your editor. Page numbering, headers and footers, indents, hanging indents, centering and underlining. *INCLUSION* feature allows automatic insertion of up to 26 user defined strings and merging of documents. HOOS/H19/H89. **HDOS #SF-9001: $54.95. CP/M #SF-9101: $54.95.**

Microsoft Macro 80:
8080/Z80 MACRO Assembler. Intel and Zilog Mnemonics supported. Relocatable linkable output. Includes LINK 80 and Cross Reference List utilities. HOOS common deck MACRO included. For H8 and H89. **HDOS #SF-8002: $69.95.**

CPS:
Permits file transfer between the H89 and H8/H19/H17 and Information Services (MicroNET). Features include user defined keys for auto-login, mail check, etc. Full error checking and elapsed time clock on screen. Very easy to use on time sharing systems. **HDOS #SF-9003: $39.95. CP/M #SF-9103: $39.95.**

SORT:
An extremely fast assembly language routine that sorts records up to 255 characters in length with user defined sort fields. Could be called by MBASIC or stand-alone. Source code provided. **HDOS #SF-8004: $29.95.**

Small Business Inventory
For complete inventory analysis. Up to 12-character part numbers (alpha-numeric), 18-character descriptions of parts, 12 items of information on each part include reorder level, usage history by month and year-to-date, much more. Complete printouts. Requires Microsoft BASIC and H19 terminal. **HDOS #SF-9005: $69.95.**

BDS C Compiler
Supports most features of language, including Structures, Arrays, Pointers, recursive function evaluation, overlays. Includes linking loader, library manager, and library containing general purpose, file I/O, and floating point functions. Lacks initializers, statics, floats and longs. Includes "The C PROGRAMMING LANGUAGE" by Kernighan and Ritchie. **CP/M #SF-8106: $119.95.**

CBASIC
Disk extended BASIC– Non-interactive BASIC with pseudo-code compiler and run-time interpreter Supports full file control, chaining, integer and extended precision variables, etc. **CP/M #SF-8107: $139.95.**

Fun for hams...RTTY Communications Processor
Split screen lets you copy incoming while checking and editing outgoing messages. On-screen graphics presents complete system status: time, CW identification, etc. ASCII or Baudot operation. Disk-based autostart. **HDOS #SF-9006: $100.**

To order:
1. Send check or money order to Heath Company, Benton Harbor, MI 49022. Michigan residents add 4% sales tax. Write model numbers clearly.
2. Call toll-free **800-253-0570** and use **VISA** or **Master Card.** In Michigan, Alaska, & Hawaii, call (616) 982-3411.
3. Visit your **Heathkit Electronic Center*** where SOFTSTUFF is on display. See your telephone white pages for the location nearest you. *units of Veritechnology Electronics Corporation.

HEATH ZENITH

CP/M® is a registered trademark of Digital Research Corp.
SOFTSTUFF™ is a trademark of Heath Company.
CBASIC™ is a trademark of Compiler Systems, Inc.

Do Your Own Thing

If you are an assembly language programmer who writes programs that run under HDOS, there may come a time when you want to do console I/O that is completely independent of HDOS. For example, you may wish HDOS to ignore all control characters, the terminal width, etc. One way to accomplish this is to write your own console I/O routines. Since the HDOS routines are interrupt driven, you can effectively "turn them off" by disabling the interrupts generated by the hardware. The following example shows how to do this, and includes single character input and output routines.

```
        ORG     (begin) BEGINNING OF PGM
.EXIT   EQU     0       EXIT TO HDOS
IOFF    EQU     0       INTERRUPT OFF BYTE
ION     EQU     1       INTERRUPT ON BYTE
IPORT   EQU     351Q    INTERRUPT CONT PORT
SPORT   EQU     355Q    CONSOLE STATUS PORT
DPORT   EQU     350Q    CONSOLE DATA PORT
INBIT   EQU     1       INPUT STATUS BIT
OUTBIT  EQU     40Q     OUTPUT STATUS BIT

* TURN OFF CONSOLE INTERRUPTS
        MVI     A,IOFF  DISABLE CONSOLE
        OUT     IPORT   INTERRUPTS
        JMP     MNPGM   GO TO MAIN PGM

* SINGLE CHARACTER INPUT ROUTINE
* RETURNS CHARACTER IN A
SCIN    IN      SPORT   CHECK CONSOLE STATUS
        ANI     INBIT   READY TO RECIEVE?
        JZ      SCIN    LOOP UNTIL READY
        IN      DPORT   GET DATA
        ANI     177Q    STRIP PARITY BIT
        RET             RETURN WITH CHAR.

* SINGLE CHARACTER OUTPUT ROUTINE
* EXPECTS CHARACTER IN A
SCOUT   PUSH    PSW     SAVE CHARACTER
SCOUT0  IN      SPORT   CHECK CONSOLE STATUS
        ANI     OUTBIT  BUFFER EMPTY?
        JZ      SCOUT0  LOOP UNTIL IT IS
        POP     PSW     RESTORE CHARACTER
        OUT     DPORT   TRANSMIT CHARACTER
        RET             RETURN TO CALLER

* START MAIN PROGRAM HERE

MNPGM   EQU     *

* AND WHEN YOU RETURN TO HDOS, USE:
EXIT    MVI     A,ION   RESTORE CONSOLE
        OUT     IPORT   INTERRUPTS
        XRA     A
        SCALL   .EXIT
```

The above will work on an H89 or an H8 using the H8-4 interface. If you use the H8-5, change the EQU's as follows:

```
IOFF    EQU     25Q     INTERRUPT OFF BYTE
ION     EQU     27Q     INTERRUPT ON BYTE
IPORT   EQU     373Q    INTERRUPT CONT PORT
SPORT   EQU     373Q    CONSOLE STATUS PORT
DPORT   EQU     372Q    CONSOLE DATA PORT
INBIT   EQU     2       INPUT STATUS BIT
OUTBIT  EQU     1       OUTPUT STATUS BIT
```

-- Patrick Swayne, hereafter known as PS:

HUG Parts List 01-Oct-80

P/N	Description	Price
885-0017	H8 Poster	$ 2.95
885-0018	H89 Poster	$ 2.95
885-0019	Color Graphics Poster	$ 2.95
885-1008	Volume I Documentation	$ 9.00
885-1009	Tape I Cassette	$ 7.00
885-1010	Adventure Disk H8/H89	$ 10.00
885-1012	Tape II BASIC Cassette	$ 9.00
885-1013	Volume II Documentation	$ 12.00
885-1014	Tape II ASM Cassette H8 Only	$ 9.00
885-1015	Volume III Documentation	$ 12.00
885-1019	Device Driver Disk H8/H89	$ 10.00
885-1022	HUG Editor (ED) Disk H8/H89	$ 15.00
885-1023	RTTY Disk H8 Only	$ 22.00
885-1024	Disk I H8/H89	$ 18.00
885-1025	Runoff Disk H8/H89	$ 35.00
885-1026	Tape III Cassette	$ 9.00
885-1027	Morse8 Cassette H8 Only	$ 14.00
885-1028	RTTY Cassette H8 Only	$ 11.00
885-1029	Disk II Games 1 H8/H89	$ 18.00
885-1030	Disk III Games 2 H8/H89	$ 18.00
885-1031	Disk IV MUSIC H8 Only	$ 23.00
885-1032	Disk V H8/H89	$ 18.00
885-1033	HT-11 Disk I	$ 19.00
885-1034	Character Ed Cassette H8 Only	$ 11.00
885-1035	ED/ASM/DEBUG Cassette H8 Only	$ 11.00
885-1036	Tape IV Cassette	$ 9.00
885-1037	Volume IV Documentation	$ 12.00
885-1038	WISE on Disk H8/H89	$ 18.00
885-1039	WISE on Cassette H8 Only	$ 9.00
885-1040	PILOT On Cassette H8 Only	$ 11.00
885-1042	PILOT on Disk H8/H89	$ 19.00
885-1043	MODEM Heath to Heath H8/H89	$ 21.00
885-1044	Disk VI H8/H89	$ 18.00
885-1045	FOCAL Cassette H8 Only	$ 11.00
885-1047	Stocks H8/H89 Disk	$ 18.00
885-1048	Personal Account H8/H89 Disk	$ 18.00
885-1049	Income Tax Records H8/H89 Disk	$ 18.00
885-1050	Modem Comm. System for H8/H89	$ 18.00
885-1051	Payroll H8/H89 Disk	$ 50.00
885-1052	Morse8 Disk H8 Only	$ 18.00
885-1054	SmBusPkg II 3 Disk H8/H19/H89	$ 60.00
885-1055	MBASIC Inventory Disk H8/H89	$ 30.00
885-1056	MBASIC Mail List H8/H89 Disk	$ 30.00
885-1057	Tape V Cassette	$ 9.00
885-1058	Volume V Documentation	$ 12.00
885-1060	Disk VII H8/H89	$ 18.00
885-1061	TMI Load H8 ONLY Disk	$ 18.00
885-1062	Disk VIII H8/H89 (2 Disks)	$ 25.00
885-1063	Floating Point Disk H8/H89	$ 18.00
885-1064	Disk IX H8/H89 Disk	$ 18.00
885-1065	Fix Point Package H8/H89 Disk	$ 18.00
885-1066	Disk X H8/H89	$ 18.00
885-1068	Disk XII MBASIC Graphic Games	$ 18.00
885-1069	Disk XIII Misc H8/H89	$ 18.00
885-1075	HDOS Support Package H8/H89	$ 60.00
885-1077	TXTCON/BASCON H8/H89 Disk	$ 18.00
885-1078	HDOS Z80 Assembler	$ 25.00
885-1079	HDOS Page Editor	$ 25.00
885-1080	EDITX H8/H19/H89 Disk	$ 20.00
885-1082	Programs for Printers H8/H89	$ 20.00
885-1083	Disk XVI Misc H8/H89	$ 20.00
885-1201	CP/M Volumes H1 & H2	$ 21.00
885-1202	CP/M Volumes 4 & 21 C	$ 21.00
885-1203	CP/M Volumes 21 A & B	$ 21.00
885-1204	CP/M Volumes 26/27 A & B	$ 21.00
885-1205	CP/M Volumes 26/27 C & D	$ 21.00
885-1206	CP/M Games Disk	$ 21.00

Using Printers with the H 11A

H11A ---- H14/H24/H34/H44 COMPATABLE ?

In order to interface the H14 to the H11A as a line printer, a serial interface is required. Either the H11-5, WH11-5, or the WHA11-5 will work. No modification is necessary to the H14. The listed interface cards will also require no modification. All that is necessary is make sure that the card is addressed and vectored properly. The card address should be 177510. The card vector should be 200. The rest of the card should be set up as per the manual for RS-232 communication and the baud rate should of course be the same as that set up on the printer. The WH11-51 adapter cable is required for proper electrical connection between the interface and printer.

In order to interface the DEC LA-34 printer to the H11A as a line printer, the same requirements necessary for the H14 apply. Since this printer runs at baud rates up to and including 300, no handshake signal is necessary.

The TI-810 "OMNI" printer is a very high speed printer and requires a handshake signal to the computer to tell "him" when the printer cannot accept any characters. If the printer was purchased from Heath Company, no modifications are necessary to the printer. The busy signal is of the proper polarity. For the least amount of "grief", the WHA11-5 interface card is required. Simply connect the two units together by way of the 134-1093 cable (supplied with the printer) and the WH11-51 adapter cable. Also make sure the baud rates are the same.

If you are interfacing the printer to the H11-5 or WH11-5 card, modifications are necessary. A description of the modifications that need to be performed, can be found in issue 8 of 'REMARK' magazine. This article was written by Randy Borchardt and will essentially turn your H11-5 or WH11-5 interface into a WHA11-5 as received from the factory. The aritcle describes how to perform this 'MOD', and still remain 'H14' compatible. This simply means you can plug either your TI-810 or H14 into the same interface without making any modifications and still have either work properly.

If you do not wish to remain 'H14' compatable, the second from the last paragraph describes how to modify the WHA11-5 board for this purpose. In addition to this jumper wire change (Pin 5 to Pin 4 on IC-21), an additional foil must be cut and jumper wire installed. This modification is as following and will be referred to as 'MOD44'.

MOD44

Objective: Isolate Pin 2 of IC-8 then ground it.

Reason: Make the Software think the printer is busy when the Software interrogates the printer status bit.

Procedure: While looking at the board with the component side facing you, and the 'PULL' handle at the top, locate the foil that runs vertically between Pins 30 and 31 of IC-20 (THE UART). Cut this foil at any convenient location. Doing this will isolate Pin 2 of IC-8.

Solder a jumper wire between Pins 2 and 8 of IC-8. Doing this will ground Pin 2 of IC-8.

NOTE: MOD44 can be performed on any WHA11-5 interface whether or not you decide to remain 'H14' compatible. This mod will not hinder the operation of the card in any way. It will simply make it more versatile.

H44 DIABLO PRINTER

The first thing that must be done to the diablo printer when it is received, is to make sure the busy signal is on the proper pin on the J2 connector on the HPRO4 processor board in the printer. The printer usually comes from the factory with the yellow wire installed into Pin 4. This must be moved to Pin 2. Once this is done, the busy signal will appear on Pin 4 of the RS-232 'D' style connector. The 'BUSY' condition of this signal will be active low (-12V). This of course is the opposite of that of the H14 printer. In order to interface this printer to the WHA11-5 card, two things must be done:

1. Make sure the card you have is a WHA11-5. If it is not, you must perform Randy Borchardts' modification.

2. The jumper wire on the back of the board going to Pin 5 on IC-21 must be moved to Pin 4 on the same IC.

3. 'MOD44' must be performed.

Interconnect the two units by way of the WH11-51 adapter cable and the cable that comes with the diablo printer. No other cables or adapters are necessary. Unfortunately there is no way to remain H14 compatible when using the diablo printer unless you wish to modify the H14. All that is necessary in this case is to interchange the two wires going to Pins 7 and 4 on S102 in the H14 printer. The H14 'BUSY' modification will also have to be installed.

Jim Buszkiewicz

H 8-2 Interface

Sometimes information is passed around via circular ways...and this may be one of those times. Here's information on the H8-2 handshake requirement that I received from Jim Buszkiewicz that I have not seen elsewhere...in the Heath manual nor in previous REMark issues. And if it is something that had been previously published, then you can forget this letter completely.

I had a problem getting my H8-2 and the ESCON Microcomputer/Selectric Interface to talk to each other. I asked Heath what could be wrong. According to Jim, the DATA TAKEN pulse (RDAV) must be AT LEAST 10 uSECONDS wide. This piece of information is not readily apparent in the Heath manual. You can figure it out if you step through the circuitry, look up the handshake 74120 bit synchronizer specifications and capabilities, etc. Anyway, the ESCON parallel interface DATA TAKEN=DATA ACCEPTED=RDAV pulse is 7 us...they tell you that in their book. I assembled a 74121 "pulse stretcher" and put it in the DATA ACCEPTED (ESCON'S name for the Heath-named DATA TAKEN acknowledgement) line (pin 12 of the ESCON EIA input connector). For anyone interested, the values for R and C I used were 2200 ohms and 0.01 UF (disc ceramic). According to the chart in the Texas Instruments TTL Data Handbook, 2nd edition, Fig. 6, pg.6-64, this gives a pulse about 15-20 us wide. The 74121 pin 1, the not-Q output, is used for RDAV. A schematic diagram of this standard circuit is attached. I used a disc ceramic because it was handy. The circuit is a standard one-shot, with the negative-going 7 us strobe used to trigger the one-shot. Pins 4 and 5 are pulled high by Vcc via the 1K resistor. The RDAV pulse must be negative going. The handshake requirements for DAV and RDAV, except for the minimum pulse width requirement for RDAV, are explained in the manual, although not very thoroughly.

I made a small 1x2 printed circuit board and used double-adhesive foam tape to mount the board just under the EIA connector. In summary, the RDAV pulse for the H8-2 must be negative-going, at least 10 us wide. In addition, Jim advised me to jumper H1-H2 for the particular channel and connect that channel to INT3.

I do not have jumpers A1-A2 (which defeats the handshake and places the H8-2 in continuous mode) and do not have B1-B2 jumpered. With the pulse stretcher in the RDAV line, the ESCON interface works well with the H8-2.

Aloha,

Wallace K. Izuo
960 Ala Lehua Street
Honolulu, Hawaii 96818

Consultant's CORNER

In this column of REMark, the Heath Technical Consultants provide answers to the most commonly asked questions on Heath computer products, both hardware and software.

Q: My H-14 works fine, but when using the 96 character per line size, the print becomes very light. Is there something I can do to create darker print in this mode?

A: Yes, there is. Install a 10K ohm resistor (Heath p/n 6-103-12) in parallel with the existing 1K at R171, and replace R192 with a 30.1 ohm resistor (6-3019-12). You will then have to re-calibrate the temperature sensing adjustment.

Q: I am using Extended Benton Harbor BASIC versions 10.02 or higher and I need to be able to load the programs that were written under the 10.01 version. How do I do this?

A: A very simple solution is to use the OLDLOAD command instead of the LOAD command. The OLDLOAD command only needs to be used when loading the program in from an old tape. Once the program has been loaded into memory the program can be saved and it will now be in the new format used by the higher versions of Extended Benton Harbor BASIC.

Q: I need more information about how HDOS accesses the disk drives. Where can I get this information?

A: The HDOS Programmers Guide (597-1973) and the Addendum (597-2194) are available from Heath Company's PARTS DEPARTMENT. These two books provide the needed system information for the assembly language programmer to allow access to the disk files and other HDOS functions.

EOF

File Handling in BASIC

Dear Gerry,

Here are two subroutines and two short programs for the next issue of REMark. The programs are specially designed for those who prefer to handle files in BASIC, using a minimum of control disk space for maximum versatility. The ideas are not new, but the programs are better than any I have been able to find.

Subroutine to convert name or address line A$ to lower case:

```
10 B=0:T$="":FOR A=1 TO LEN(A$):N=ASC(MID$(A$,A,1):IF N<65 OR N>90 THEN B=0
20 T$=T$+CHR$(N+B):B=32:IF N<>32 GOTO 40
30 IF MID$(A$,A,5)=" AND " OR MID$(A$,A,4)=" OR " THEN B=0
40 NEXT A:A$=T$:RETURN
```

Subroutine to recognize single key answer A$ without carriage return:

```
10 V0=PEEK(8242)+256*PEEK(8243):V0=V0+2029:REM - Buffer pointer
20 U0=PEEK(V0):U1=PEEK(V0+1):V=U0+256*U1:REM - Buffer address
30 POKE V,0:PRINT "? ";:REM - Clear first byte and prompt (Go here if invalid)
40 U=PEEK(V):IF U=0 GOTO 40:REM - Loop awaiting keystroke
50 PRINT:POKE V0,U0:POKE V0+1:A$=CHR$(U):RETURN:REM - Restore pointer
```

Fraternally, Jim Tennant
P.O. Box 7176
Ketchikan, Alaska 99901

```
00010 REM       'REM.BAS', BY JIM TENNANT; H8/(H9 or H19)/H17
00020 REM    Separate or Rejoin Remark Statements in BASIC Files
00030 REM                HDOS 1.6; BASIC #110.05.00
00040 REM
00050 CLEAR :V0$="Lines":V1$="Bytes":V2$="(Saved)":V3$="(Deleted)"
00060 F0$="REM"+" ":F2$=".BAS":F3$=".REM":LINE INPUT "File Name: ";F$
00070 F4$=".PGM":IF LEN(F$)>4 THEN IF MID$(F$,LEN(F$)-3,1)="." GOTO 90
00080 F$=F$+F2$:REM - Default to SY0: and .BAS
00090 IF MID$(F$,4,1)<>":" THEN F$="SY0:"+F$
00100 F1$=LEFT$(F$,LEN(F$)-4):OPEN F$ FOR READ AS FILE #1
00110 F2$=F1$+F2$:F3$=F1$+F3$:F4$=F1$+F4$:REM - Filename Family
00120 IF RIGHT$(F$,3)="PGM" GOTO 300:REM - Merge if .PGM file
00130 REM
00140 REM    Sort: Replace SY#:XXX.BAS with SY#:XXX.PGM & SY#:XXX.REM
00150 REM
00160 OPEN F4$ FOR WRITE AS FILE #2:OPEN F3$ FOR WRITE AS FILE #3
00170 E=CIN(1):IF E<=0 THEN CLOSE #1,#2,#3:GOTO 230
00180 B=B+1:LINE INPUT #1,;T$:T$=CHR$(E)+T$:N=MATCH(T$,F0$,7):G1=G1+1+LEN(T$)
00190 IF N=0 THEN P=P+1:PRINT #2,;T$:G2=G2+1+LEN(T$):GOTO 170
00200 R=R+1:IF N<8 THEN PRINT #3,;T$:G3=G3+1+LEN(T$):GOTO 170
00210 PRINT #3,;LEFT$(T$,6);RIGHT$(T$,LEN(T$)-N+2):G3=G3+LEN(T$)+9-N
00220 P=P+1:PRINT #2,;LEFT$(T$,N-2):G2=G2+N-1:GOTO 170
00230 IF R=0 THEN PRINT "No remarks in '";F$;"'.":UNSAVE F4$:UNSAVE F3$:END
00240 UNSAVE F$:PRINT TAB(5)F$;TAB(30)F4$;TAB(55)F3$
00250 PRINT TAB(5)B;V0$;TAB(30)P;V0$;TAB(55)R;V0$:PRINT TAB(5)G1;V1$;TAB(30);
00260 PRINT G2;V1$;TAB(55)G3;V1$:PRINT TAB(6)V3$;TAB(31)V2$;TAB(56)V2$:END
00270 REM
00280 REM    Merge: Replace SY#:XXX.PGM & SY#:XXX.REM with SY#:XXX.BAS
00290 REM
00300 OPEN F3$ FOR READ AS FILE #2:E1=CIN(1):E2=CIN(2)
00310 OPEN F2$ FOR WRITE AS FILE #3:GOSUB 320:GOTO 350
00320 IF E1<=0 THEN V1=65535:RETURN
00330 P=P+1:LINE INPUT #1,;T1$:T1$=CHR$(E1)+T1$:V1=VAL(T1$):E1=CIN(1)
00340 G1=G1+1+LEN(T1$):RETURN
00350 IF E2<=0 THEN V2=65535:GOTO 380
00360 R=R+1:LINE INPUT #2,;T2$:T2$=CHR$(E2)+T2$:V2=VAL(T2$):E2=CIN(2)
00370 G2=G2+1+LEN(T2$)
00380 IF V1=V2 GOTO 410
```

```
00390 B=B+1:IF V1<V2 THEN PRINT #3,;T1$:G3=G3+1+LEN(T1$):GOSUB 320:GOTO 380
00400 PRINT #3,;T2$:G3=G3+1+LEN(T2$):GOTO 350
00410 IF V1=65535 GOTO 440:REM - 65535 must not be a line number
00420 B=B+1:PRINT #3,;T1$;RIGHT$(T2$,LEN(T2$)-6)
00430 G3=G3+LEN(T1$)+LEN(T2$)-5:GOSUB 320:GOTO 350
00440 CLOSE #1,#2,#3:UNSAVE F$:UNSAVE F3$:PRINT TAB(5)F2$;TAB(30)F$;TAB(55);F3$
00450 PRINT TAB(5)B;V0$;TAB(30)P;V0$;TAB(55)R;V0$:PRINT TAB(5)G3;V1$;TAB(30);
00460 PRINT G1;V1$;TAB(55)G2;V1$:PRINT TAB(6)V2$;TAB(31)V3$;TAB(56)V3$:END
```

```
00010 REM     'PRINT.BAS', BY JIM TENNANT (H8/H19/H17)
00015 REM      (H9 Users, delete odd-numbered lines)
00020 REM           HDOS 1.6; BASIC #110.05.00
00030 REM
00040 CLEAR :PRINT :LINE INPUT "File Name: ";F$:IF F$="" THEN END
00050 IF LEN(F$)>4 THEN IF MID$(F$,LEN(F$)-3,1)="." GOTO 70
00060 F$=F$+".DAT":REM - Defaults
00070 IF MID$(F$,4,1)<>":" THEN F$="SY0:"+F$
00080 LINE INPUT "Output Device: ";Q$:IF RIGHT$(Q$,1)<>":" THEN Q$=Q$+":"
00090 INPUT "Lines per page: ";Q:IF Q$="TT:" GOTO 300
00100 REM
00110 REM     PRINT ANY LENGTH PAPER WITH DATE & PAGE NUMBERS
00120 REM
00130 S1$="Set paper to first print line ":S2$="and RETURN."
00140 FOR A=8383 TO 8391:T$=T$+CHR$(PEEK(A)):NEXT A:D$=MID$(T$,4,3)
00150 IF D$="May" GOTO 190
00160 IF D$="Jun" THEN D$=D$+"e":GOTO 190
00170 IF D$="Jul" THEN D$=D$+"y":GOTO 190
00180 D$=D$+"."
00190 D$=D$+LEFT$(STR$(VAL(T$)),LEN(STR$(VAL(T$)))-1)+", 19"+RIGHT$(T$,2)
00200 P=1:OPEN F$ FOR READ AS FILE #1:GOSUB 220:PRINT S1$;S2$;
00210 PAUSE :PRINT #2,;F$;TAB(60);D$:PRINT #2,:GOTO 240
00220 OPEN Q$ FOR WRITE AS FILE #2:RETURN
00230 PRINT S1$;"of Page";P;S2$;:PAUSE :PRINT #2,;F$;TAB(65)"Page";P:PRINT #2,
00240 FOR A=1 TO Q:E=CIN(1):IF E<=0 GOTO 280
00250 IF E=10 THEN A=A+1:REM - skip leading zero next
00260 LINE INPUT #1,;T$:IF RIGHT$(F$,3)<>"BAS" OR E<>48 THEN T$=CHR$(E)+T$
00270 PRINT #2,;T$:NEXT A:CLOSE #2:P=P+1:GOSUB 220:GOTO 230
00280 CLOSE #1:PRINT #2,:PRINT #2,TAB(31)"** eof **":CLOSE #2:GOTO 40
00290 REM
00300 REM     NO-SCROLL DISPLAY WITH LINE & BYTE COUNTS
00310 REM
00313 DIM E$(4):FOR A=1 TO 4:E$(A)=CHR$(27):READ C
00317 FOR B=1 TO C:READ E:E$(A)=E$(A)+CHR$(E):NEXT B:NEXT A
00320 OPEN F$ FOR READ AS FILE #1:I=CIN(1)
00330 A=0
00335 PRINT E$(3):REM - blank cursor & erase
00340 LINE INPUT #1,;I$:I$=CHR$(I)+I$:L=L+1:A=A-1:IF LEN(I$)>=79 THEN A=A-1
00350 G=G+1+LEN(I$):PRINT I$;:I=CIN(1):IF I=10 THEN A=A-1:L=L+1
00360 IF I>0 GOTO 410
00370 IF A>=1 THEN FOR B=1 TO A:PRINT :NEXT B
00380 T$=" End '"+F$+"' at Line"+LEFT$(STR$(L),LEN(STR$(L))-1)+".  RETURN"
00390 GOSUB 440
00395 PRINT E$(3);E$(4):REM - erase & reset escapes
00400 CLOSE #1:GOTO 40
00410 IF A>=1 THEN PRINT :IF A>1 GOTO 340
00420 T$=" RETURN for Line"+STR$(L+1)+"of '"+F$+"'":GOSUB 440
00425 PRINT E$(2):REM - exit reverse video & restore cursor
00430 GOTO 330
00440 T$=T$+".  ("+RIGHT$(STR$(G),LEN(STR$(G))-1)+"Bytes) "
00445 PRINT E$(1);:REM - save cursor, enable line 25, set cursor, reverse video
00450 PRINT TAB(10)T$;:PAUSE :RETURN
00455 DATA 10,106,27,120,49,27,89,56,42,27,112,3,113,27,107,4,120,53,27,69,1,122
```

EOF

BUGGIN' HUG

Dear Jim,

The ".LOADD" scall, as illustrated in example C.2 of the addendum to the HDOS system programmer's guide (597-2194), is insufficient. Just invoking the scall will load a device driver, but not lock it in memory. (If you excute the prologue, and then RESET SY0:, the driver(s) will not be saved.

To lock in the driver(s), add the following called subroutine following the main program logic:

```
SLOAD   LHLD    S.SYSM
        SHLD    040356A
        LHLD    041053A
        LXI     D,2
        DAD     D
        MOV     A,M
        ORI     2
        MOV     M,A
        RET
```

Then re-assemble the prologue so that this subroutine is called after any successful .LOADD, I.E.-

```
LOAD1   LXI     H-PROAA
        SCALL   .LOADD
        JC      LOAD2
        CALL    SLOAD   Lock the DVD
        CALL    $TYPTX
        DB      'LP: LOADED',ENL

LOAD2   LXI     H,PROAB
        SCALL   .LOADD
        JC      LOAD3
        CALL    SLOAD   Lock the DVD
        CALL    $TYPTX

       . . .Etc.
```

The "SLOAD" subroutine is taken directly from the current SYSCMD.SYS. It is tested and recommended for use (in a prologue or program) ONLY with HDOS version 1.6.

Keep up the good work! Al Heigl

Dear Gerry,

I was disappointed to hear that you have so little software available from the users of the ET-3400 and ETA-3400. I am enclosing a small program that does checkbook validation. It might be of interest to some ETA-3400 users.

George Brown
2428 Eck Drive
Raleigh, NC 27604

```
1 PR"CHECK VALIDATION"
2 PR"PROGRAM BY GEORGE BROWN 7/80"
3 PR"INPUT DOLLARS,CENTS."
4 PR"USE MINUS FOR WITHDRAWALS."
5 PR"START WITH INITIAL BALANCE"
10 LET X=0
20 LET Y=0
25 PR"TRANSACTION"
30 INPUT D,C
40 IF D<0 GOTO 170
50 X=X+D
60 Y=Y+C
70 IF C=0 GOTO 190
80 IF Y=0 GOTO 190
85 IF Y<0 GOTO 140
90 Z=Y-100
100 IF Z<0 GOTO 190
110 X=X+1
120 Y=Y-100
130 GOTO 190
140 Y=Y+100
150 X=X-1
160 GOTO 190
170 C=-1*C
180 GOTO 50
190 PR"          BALANCE $";X;".";Y
200 GOTO 25
```

Dear HUG,

If there are any other HUG members that are interested in getting two H8's running together and talking to each other, I would be happy to exchange ideas. The set up I have is geared towards a RTTY "Mailbox" that I run on a local two meter repeater. It is set up around a specific set of programs but it is easliy adaptable to run almost any machine language program and allows direct access to the disk drives of the main system from the second computer. If there are any other members working in this area I would appreciate hearing from them.

Glenn R. Beard
236 East 6th Street
Red Hill, PA 18076

Dear HUG,

My computer system includes a H-14 line printer. As you know, one of the criticisms of the H-14 is that it does not produce a very dark printout, hence the printout does not reproduce very well.

I took this problem to a local ribbon manufacturer. They made up a special ribbon for me to try, and it produces an appreciably darker printout than I have been getting with ordinary typewriter ribbons.

These ribbons may be ordered from:

Frankel Manufacturing Company
285 Rio Grande Boulevard
Denver, CO 80223
att: Mr. John Murray

Specifications:

dual 4 spools
18 yards Super Kemlon ribbon
black Matrix ink

The ribbons are $23.00 per box of one dozen. Minimum order is one box. I am sure other H-14 owners will be interested to know this.

F. B. McLaughlin
8915 Piney Creek Road
Parker, CO 80134

Dear JB:

I would like to announce the Pf Glare Screen; now available for the Heath H89 - H19. These units significantly reduce the discomfort and fatigue associated with using CRT display terminals. Originally produced for our own use, the Glare Screens were so effective it was decided to produce and market them publicly.

The units are constructed of 1/8 inch grey acrylic plastic with neoprene inserts centered on each side. Installation is accomplished by merly pressing the Glare Screen into the face plate aperature. The neprence inserts provide a friction fit against the edge of the face plate.

The price of $13.49 includes postage and handling charges. A club discount of $1.00 per screen is offered on orders of five or more. California residents add sales tax. Additional charges for shipping outside continental United States.

Send check or money order payable to:

Pf Research
866 Hummingbird Drive
San Jose, California 95125

Dear Mr. Blake,

H89 users who have two disk drives in the separate H17 cabinet may be interestd in the easy way that I found to add a third drive to my system. As you may recall, Heath made a special offer of the H17 to H89 owners in late 1979. "SYO:" is removed from the H89 cabinet, placed in the H17 cabinet, and the two drives are then connected to the H89 controller board with a long 34 wire ribbon cable.

A third drive may be added to this configuration by installing it in the space vacated by "SYO:" in the H89 cabinet, and connecting it to the long ribbon cable via a clamp-on "T" connector. A disk drive can be obtained from Heath (part H17-1), from a mail-order supply house (e. g., Advanced Computer Products), or one can get lucky and find an unwanted, barely used one at a local computer store, as I did.

The clamp-on "T" connector is a 3M no. 3463-0001. I determined its proper position on the long ribbon cable by using the original "SYO:" connector as a guide. I even used the black sleeving that originally protected that connector. One must be careful in orienting the "T" on the long ribbon cable, because that cable is inverted in comparison to the original cable. The best way to determine the proper position is to plug in the long ribbon cable and study its orientation with respect to the new drive very carefully. One final note: after the "T" connector has been firmly clamped on the ribbon cable, using a hammer to seat the clamp firmly, the clamp should be locked in place by melting its ends to the body of the connector with a soldering iron.

I hope my experience will be useful to some other HUG member. The whole operation should be well within the ability of anyone who has already assembled the H89 and connected the H17 add-on.

Sincerely,

Victor A. Abell
1715 Summit Drive
West Lafayette, IN 47906

NEW HUG GROUP

A new local HUG group has been formed in Warwick, Rhode Island. The first meeting was scheduled for October 8 with the theme "Become Aquainted with Your Fellow Heath Users". Meatings will be held at the Heathkit Electronics Center, 558 Greenwich Ave., Warwick RI.

New Products from non-Heath Sources

STRETCH-8

The STRETCH-8 expansion for the Heath H8 computer more than doubles the capacity of the H8 computer by proving eight additional slots. With the front panel and CPU boards in place you have 15 slots left for memory, I/O and special purpose cards. A supplementary 7.5 amp, 8 volt power supply and the oak sides panels make the STRETCH-8 only 14 inches longer than the present size of the H8.

Only $200.00 by cashiers check, money order or VISA/Mastercard
(California Residents + 6% Tax)

STRETCH-8
P.O. Box 1120
Burbank, California 91507

NEW from DG Electronics

DG Electronic Development Co. has announced the availabilty of their DG-64D dynamic 64K RAM card. The features included on the DG-64D make it the most flexible memory available for the Heath H8 computer. Low power requirements (less than 8 watts) enable utilization of 64K in memory while allowing a full compliment of other peripherals.

The DG-64D is compatible with a standard Heath configuration or with DG's NEW DG-80, Z80 based CPU. The low cost of the DG-64D RAM is $529; fully assembled and burned-in.

The NEW DG-80 Z80 CPU card for the H8 computer was also resently announced at a selling price of $249.00. The DG-80 allows the H8 user all the power, speed and flexibility offered by the Z80 microprocessor. Operational speeds to 4MHz along with the full compliment of Z80 instructions give the user flexibilty never before seen in the H8.

DG Electronic Development Co. announces the DG-ADP4 plug-in hardware modification to allow operation of the H17 disk system at 4MHz when using the NEW DG-80, Z80 CPU card in the H8 along with the DG-ADP4. The DG-ADP4 also requires the NEW DG-FP8 descripted below. This modification plugs into an IC socket on the H17 controller board and reqiuires no further modification. The selling price for the DC-ADP4 is $19.95.

A NEW monitor for the H8 is also annouced as the DG-FP8 Monitor selling for $69.95. This monitor gives the user the full scope of the present PAM-8 panel monitor, lus added features to make full use of the DG-80 the Z80 CPU card. Some of the features include: Hex or octal display and entry; two keystroke display of memory contents pointed to by any register; automatic start-up of program counter for disk bootup; display and alter contents of primary and alternate resister; allow operation of panel monitor in Z80 alternate interrupt modes and others.

The combination of the DG-80 CPU, the DG-FP8 Monitor, and the DG-ADP4 4MHz Adapter give advanced power, flexibility and reliability to the H8 for personal, business, OEM and engineering applications.

DG Electronic Development Co. has also announced that it will be offering standard CP/M Version 2.2 for the H8 computer. This version of CP/M does require the DG-80 Z80 CPU card and the DG-FP8 Monitor package. The expected price is $130.00.

For more information and complete specification sheets contact:

DC ELECTRONIC DEVELOPMENTS CO.
P.O. 1124
Denison, Texas 75020

VIDEO LAYOUT PADS for H19, H88 or H89 are now available from Walt Gillespie at MINT-MAN Printing, 211 E. Allegan Street, Otsego, Michigan 49078. The VIDEO LAYOUT PAD is 11 by 14.5 inches and has both the column and line numbers in their postion value, decimal value and ASCII value. The layout area is setup 80 across and 25 down simulating the screen of the H19, H88 or H89. The thirty-three different graphic characters are shown on the bottom of each sheet making them very useful for creating graphic displays. Again both the decimal and ASCII values are provided for these characters. The pads are selling for $2.25 for a pad of 50 sheets (including postage and handling). The minimum order is 5 pads for $11.25. Michigan residents add 4% sales tax.

HT-11 ACCOUNTING SYSTEM

An accounting system for HT-11 is available from Radio Sales and Service Company at 2000 12th Avenue, Columbus, Georgia 31901. This system includes both accounts receivable and accounts payable. It also includes payroll, general ledger, letter writing and label printing programs. The package runs on a H11 system with a H9, H27, H36 and 24K of memory but the terminal and printer can be most any type. The program is on six disks selling for $200.00 or just the listings and instructions for $100.00.

(vectored from page 3)

ORG 0 CP/M *. The HUG software program continues as usual.

Speaking of HUG software, as summertime closes in most areas of the country software submissions traditionally increase so let's have another contest. Here's a $500 gift certificate for the best piece of software submitted by November 30, 1980. What makes a 'best piece of software'? Most important is the demand and how much work HUG would have to put into the product before it is released to the rest of the membership. By demand, we mean how many other HUG members would want your program? And, is it easy to use?

Submit your program and documentation on tape or disk. A sample run isn't necessary, nor is a hardcopy listing. Just explain what the program does on the submission form and enclose it along with tape or disk.

Some hints: Any program that uses the features of the H19/H89 or that can be modified by us for that purpose.

Any kind of small business software.

Utilities... Despooler.
Word processing.
Disk recovery routines.

POSTERS

Not long ago, Ray Massa from the'Detroit area sent us a slide of a proposed H 8 poster he had designed. We thought it was pretty neat, so we asked him and his wife Nancy to design one for the H89 and the H 8 color graphics board. The posters are 18x24" and sell for $2.95. The H 8 poster is depicted on the front cover.

MEET PATRICK SWAYNE

We are very fortunate to have Patrick Swayne join the staff as of the 15th of September. Pat not only writes good code, he will assist us in the preparation of REMark, assuring its regular delivery to the membership each month. If you want to contribute to the success of REMark, send your ideas and articles (preferably on tape or disk) to either myself or Pat. Welcome aboard Pat!

JB:

Changes for HDOS BOOT-UP

...some things we learned from the Capital Heath Users' Group

John Stetson has passed around information on a short patch that would by-pass the requirement for a "CR" at the begining of the "BOOT" of a disk. Along with this modification are a few more from the "CHUG" news letter.

Here's what you can do:

1. Eliminate the requirement to " Type spaces to determine the BAUD RATE".

2. Eliminate the requirement for a "CR" to initiate "BOOT".

3. If the date was entered when you mounted the first disk of the day, and power stays on the system, you can eliminate the need for the required "CR" at the "DATE (DD-MM-YY)?" line during mounting of the next disks.

4. Modify the TT: print-out to reflect these previous modifications.

In order to implement these modifications to the operating system you will need either the program "Dump" from HUG disk VIII (885-1062) or one of the similar programs that can access and change a byte stored on a particular sector and track of the disk to be modified. The locations provided below are based upon the HUG program. These modifications will work for both the H8/H17/H19 or the H89.

ELIMINATE SPACES ON BOOT-UP

This modification presumes that you will be operating your system at a baud rate of 9600. Should you have implemented this modification and change the baud rate using the program "BAUD" (from HUG disk 885-1060) you will need to return the console terminal to the 9600 preset rate before you attempt to re-boot.

TRACK 0 SECTOR 4

LOCATION	OLD VALUE	NEW VALUE
54H	3A	21
55H	08	0C (06 for 19200 baud)
56H	20	00
57H	F5	C9

ELIMINATE BOOT "CR"

This will automatically bypass the need for a "CR" after the baud rate has been determined. You will lose the ability to do Checksums (big-deal) or use the Ignore command that normally is available at the BOOT of a disk.

TRACK 0 SECTOR 0

LOCATION	OLD VALUE	NEW VALUE
10H	CD	C3
11H	2D	22
12H	25	23

BYPASS DATE

If a date already exists in memory from a previous entry, this modification will bypass the "CR" that would normally be required.

TRACK 2 SECTOR 0

LOCATION	OLD VALUE	NEW VALUE
4F	20	3A
50	A8	A0
76	C3	C9
8D	29	7F

CHANGE TT: PRINTOUT

These changes to the operating system on the disk are not necessary for operation, but since they now become somewhat meaningless it looks a little neater when they are modified.

TRACK 0 SECTOR 0

AT LOCATION A5,A6,A7 AND A8 REPLACE THE EXISTING VALUES WITH 00.

TRACK 2 SECTOR 9

AT LOCATION 6B TO AND INCLUDING 7E REPLACE THE EXISTING VALUES WITH 00.

TRACK 2 SECTOR 9

LOCATION	OLD VALUE	NEW VALUE
7F	6D	39
80	69	36
81	6E	30
82	65	30

EOF

Revision To HUG's Modem Communication System

```
**        REVISIONS TO;  HUG 885-1050  'MODEM COMMUNICATIONS SYSTEM'
*
*              MCS REVISIONS BY: A. RICHARD TINDER
*              FOR INDIANOLA HIGH SCHOOL, INDIANOLA IA, 50125
*              APRIL 4, 1980
*
**        THESE CHANGES TO 'MCS' ALLOW SENDING FILES TO TIME-SHARE
*         SYSTEMS, AND TO REMOTE TERMINALS NOT USING 'MCS'.
*
*         THE 'SEND' COMMAND WILL ASK IF IT IS TO WAIT FOR A SECOND
*         REFLECTED CHARACTER (PRESUMABLY A LINE FEED) AFTER SENDING
*         A  CR  BEFORE TRANSMITTING THE NEXT LINE.  THIS IS REQUIRED
*         BY MOST TIME-SHARE INSTALLATIONS.
*         IF YOU ANSWER 'NO', 'SEND' THEN ASKS IF IT IS TO TRANSMIT
*         A  LF  AFTER EACH  CR  .  THIS IS NECESSARY IF YOU ARE SEND-
*         ING THE FILE TO A REMOTE TERMINAL, RATHER THAN A COMPUTER.
*
*         ANSWERING 'NO' TO BOTH QUESTIONS LEAVES 'MCS' IN ITS ORIGINAL
*         CONFIGURATION, EXCEPT THAT IT CHECKS FOR REFLECTION OF EACH
*         TRANSMITTED CHARACTER.  SET YOUR MODEM FOR HALF-DUPLEX UNLESS
*         YOU ARE COMMUNICATING WITH A FULL-DUPLEX REMOTE COMPUTER.
*         REMOTE TERMINALS SHOULD BE SET FOR HALF-DUPLEX, ALSO.
*
**        A DELAY HAS BEEN INSERTED AT THE BEGINNING OF 'CONVERSATION'
*         TO ALLOW THE PROMPT '.OK' TO PRINT COMPLETELY BEFORE THE 'DI'
*         INSTRUCTION IS EXECUTED.
*
****      WARNING!       THESE CHANGES TO 'MCS' HAVE BEEN TESTED ON
*                        THE 8251 USART (H8-5, H8-2) VERSION ONLY!!
*
**   IN 'STORAGE' ADD THESE LINES:
*
F.ECWT    DB        1          FLAG: 1= WAIT FOR CHAR. AFTER  CR
F.XMLF    DB        0          FLAG: 1= SEND  LF  AFTER  CR

**   IN 'M E S S A G E S' ADD THESE LINES:
*
M.WLF     DB        A.NL,'WAIT FOR ECHO  LF  AFTER  CR  ?',' '+200Q
M.XMLF    DB        A.NL,'SEND  LF  AFTER  CR  ?',' '+200Q

**   RIGHT AFTER 'C.CONV  EQU      *', INSERT:
*
*         WAIT FOR PROMPT PRINTING
*
          MVI       A,60
          CALL      DLY

**   RIGHT AFTER 'C.SEND  EQU      *' HEADER, INSERT:
*
*         SET/RESET  LF  FLAGS
*
C.SENDO   EQU       *
          MVI       A,TRUE
          STA       F.ECWT         SET WAIT AFTER  CR  FLAG
          MVI       A,FALSE
          STA       F.XMLF         CLEAR SEND  LF  FLAG
          LXI       H,M.WLF
          SCALL     .PRINT         ASK: WAIT FOR  LF  ?
          SCALL     .SCIN
          JC        *-2
          CPI       'Y'            SHALL WE LEAVE THE FLAG SET?
          JE        C.SENOX        YES, AND SKIP TO EXIT
          MVI       A,FALSE        NO,
          STA       F.ECWT              CLEAR IT AND ASK NEXT QUES.
          SCALL     .CLRCO
```

```
          LXI       H,M.XMLF
          SCALL     .PRINT          ASK: SEND  LF  ?
          SCALL     .SCIN
          JC        *-2
          CPI       'Y'             SHALL WE SET THE FLAG?
          JNE       C.SENOX         NO, LEAVE IT CLEAR AND SKIP
          MVI       A,TRUE          YES,
          STA       F.XMLF               SET SEND  LF  FLAG
C.SENOX   SCALL     .CLRCO          EMPTY CONSOLE BUFFER
*         OLD CODE RESUMES HERE WITH 'LXI     H,M.EFN', ETC.
**   REPLACE ALL THE CODE FROM 'C.SEND5'+3
*                                    'CALL     PUTM'
*                               TO 'C.SEND6'-1'
*                                    'ENDIF'
*                               ** INCLUSIVE **
*    WITH THESE LINES:
*
LFLOOP1   CALL      PUTM            TRANSMIT CHAR.
LFLOOP2   CALL      GETM            READ MODEM FOR ECHO
          JC        *-3                  AND WAIT FOR IT
          CPI       A.CR            WAS IT A  CR  ?
          JNE       C.SEND6         IF NOT, SKIP ALL THIS STUFF
          LDA       F.ECWT               ELSE, CHECK WAIT FLAG
          ORA       A
          JNZ       LFLOOP2         IF SET, GO BACK AND READ MODEM
          LDA       F.XMLF          NOW CHECK SEND  LF  FLAG
          ORA       A
          JZ        C.SEND6         IF CLEAR, SKIP
          MVI       A,A,LF               ELSE,
          JMP       LFLOOP1              GO SEND  LF
*    OLD CODE RESUMES HERE WITH:   'C.SEND6  EQU       *', ETC.
```

EOF

Type-A Head Buffer for HDOS 1.6

LOCATION	OCTAL	DECIMAL
HDOS Starting address (low byte)	040 320	8400
HDOS Starting address (high byte)	040 321	8401
Offset to Line Counter Byte	012 147	2663
Offset to Queue Tail Pointer (low byte)	012 153	2667
Offset to Queue Tail Pointer (high byte)	012 154	2668
Offset to Queue Head Pointer (low byte)	012 155	2669
Offset to Queue Head Pointer (high byte)	012 156	2670
Offset to Pointer to Buffer Start (low byte)	012 157	2671
Offset to Pointer to Buffer Start (high byte)	012 160	2672
Offset to Pointer to Buffer End + 1 (low byte)	012 161	2673
Offset to Pointer to Buffer End + 1 (high byte)	012 162	2674
Offset to Buffer Start	012 166	2678
Offset to Buffer End	012 332	2778

THANKS GOES TO: Robert C. Johnson
21st Floor
One Financial Plaza,
Hartford CT. 06103

EOF

Addendum to the Microsoft BASIC Software Reference Manual

APPENDIX E -- ASSEMBLY LANGUAGE SUBROUTINES

The USR command in MICROSOFT BASIC allows access to assembly language programs. For example, the following BASIC program calls the "ALARM" routine that sounds the horn on your computer. This routine is located at 002.136 in split-octal notation. In normal octal, that is &01136. (Octal is used for ease of conversion. Decimal or hex could be used.)

```
10 DEF USR4 = &01136
20 FOR I=1 TO 10
30   X = USR4(X)
40        FOR J=1 TO 200
50        NEXT J
60 NEXT I
```

Line 10 in the program defines where the USR starts. The alarm program is permanently stored in your computer's firmware. Check the listing of the firmware in your manual for more detail.

Lines 20 through 60 form a loop that will sound the horn ten times. The actual call to the USR is on line 30. In this example, no values are passed in the variable "X", but "X" is needed to give a proper function form.

Lines 40 and 50 form a timing loop that gives a delay between the beeps. You can slow or speed the beeping by changing the loop count for J.

In this simple example, the code to sound the horn is already in your computer's memory as firmware. You do not have to enter the assembly language program or reserve space for it. Normally you will have to do both.

The next sample shows the normal procedure where you have to enter the assembly language program yourself. You also have to reserve the necessary memory.

Memory can be reserved when you start MBASIC with the "/M" option. It can also be reserved with the CLEAR command. HDOS is loaded just above the maximum memory limit of MBASIC, so never set the limit higher than the top of MBASIC.

There are two ways to get data between a BASIC program and an assembly language program. The normal way is to pass arguments through the Floating Point Accumulator (FAC). You can also use PEEK and POKE, but this can be very risky. If you are forced to use POKE, be sure to only POKE into the area that you have reserved for your assembly language program.

The FAC occupies eight bytes in memory - enough for a double precision number. The highest byte in memory of the FAC contains the exponent of single or double precision numbers. The next lower byte contains the most significant byte of the fraction, and lower bytes contain less and less significant bytes. More detail on the floating point format is given at the end of this appendix.

When a USR is entered, the H,L pair points to FAC-3, which is the least significant byte of a single precision number. It is also the lower byte of a two byte integer.

Numbers can be passed to USR's in all three forms - integer (%), single precision (!), and double precision (#). The value of the A-register tells what type of argument is in the FAC. The list below tells how each type is sent to a USR routine.

A-REG. TYPE OF VARIABLE

2 INTEGERS (%) are sent in FAC-2 and FAC-3, where FAC-2 has the most significant byte. Note that the number is stored in two's complement form.

4 SINGLE PRECISION (!) values are sent in FAC-0 through FAC-3. FAC-0 contains the exponent, FAC-1 contains the most significant byte of the fraction, and so on.

8 DOUBLE PRECISION (#) values are sent in FAC-Ø through FAC-7. FAC-Ø through FAC-3 are as above, and FAC-4 through FAC-7 are additional bytes of precision for the fraction.

The next sample program is designed to accept a two byte integer in the FAC and return the two bytes interchanged. It makes no check that the argument is an integer, so it will also swap the bytes in FAC-2 and FAC-3 for a floating point value. This operation is not useful, but it is shown.

An assembly language listing of the USR is shown below. It starts at 174.000A=&076000=31744, and can be run on a system with only 32K of RAM.

```
                    ORG  174000A

174.000   106       SWAP MOV  B,M  LOW BYTE TO B-REG
174.001   043            INX  H    ADVANCE TO HI BYTE
174.002   116            MOV  C,M  HI BYTE TO C-REG
174.003   160            MOV  M,B  PUT LOW INTO HI
174.004   053            DEC  H    RETREAT TO LOW BYTE
174.005   161            MOV  M,C  PUT HI INTO LOW
174.006   311            RET      GO BACK TO MBASIC

                         END  SWAP
```

The MBASIC program and a sample output are:

```
10 CLEAR 100,&075777
20 DATA &0106,&0043,&0116,&0160,&0053,&0161,&0311
30 FOR I=0 TO 6
40   READ D
50   POKE &076000+I, D
60 NEXT I
70 DEF USR = &076000
80 PRINT HEX$( USR( &H1234 ) )
90 PRINT USR0( .502 )
100 END
RUN
3412
 .500282
OK
```

Line 10 reserves 100 bytes for strings and sets the top of MBASIC's memory at &075777. In line 20, the instruction bytes for the program are given in octal. The loop on lines 30 through 60 read the program from the data statement and POKE it into memory at the proper place. On line 50, 076000+I is the address where the instruction is to be stored.

Line 70 defines where the USR routine (SWAP) starts in memory. Lines 80 and 90 call the USR and print the result that is returned.

The results of running the program are shown after the listing. In the first call to the USR on line 80, the HEX value &H1234 is sent into the USR. The output of this call is printed in HEX and shows that the two bytes have been swapped. (HEX is used in the example because two HEX digits form one byte. Thus, you can see that the bytes have been swapped.)

Note that USR and USRØ have been used in the program. They are equivalent.

The second call to the USR uses a single precision floating point number. There is no error message because the USR does not test if it has been sent an integer, it simply swaps the two bytes in FAC-2 and FAC-3. It is not clear from the output, but that is what has happened.

An USR that processes strings cannot use the same communication method between the BASIC program and the assembly language program. When the USR's argument is a string, the A register contains the value 3, and the H,L pair still points to FAC-3. However, the D,E pair points to a "string descriptor" that describes the string argument.

This "string descriptor" consists of three bytes. The first byte is the length of the string, and the second two are the lower and upper bytes, respectively, of the starting address of the string.

The result of a string USR is returned to the BASIC program IN THE SAME LOCATIONS THAT THE ARGUMENT WAS IN! Because of this, there are two precautions that you must observe when you use string USR's.

1. You must not increase the length of a string in an USR, although, you may shorten a string. If you do increase the length of a string, part of memory will be used for two purposes, and it will probably crash MBASIC.

2. You must not use a string constant when a string USR is called. If you do, the modification that the USR makes will be to the string constant and your source program will be changed. You MUST force MBASIC to use a temporary string. Concatenating a null string will work. For example:

```
PRINT USR( "INPUT TEXT"+"" )
```

A sample string USR follows. This example also starts at &076000 and can be used with only 28K of memory. This USR example will trim the trailing spaces off the end of a string and return only the leading characters.

```
10 CLEAR 100,&075777
20 DATA &353,&006,&000,&116,&171,&270,&310,&043
30 DATA &136,&043,&126,&353,&011,&053,&176,&376
40 DATA &040,&302,&030,&174,&015,&302,&015,&174
50 DATA &353,&053,&161,&311
60 FOR I=0 TO 28
70         READ D
80         POKE &076000+I,D
90 NEXT I
100 DEF USR = &076000
120 A$ = "TEST    "
130 PRINT "+";A$;"+"
140 PRINT "+";USR(A$);"++++++"
RUN
+TEST    +
+TEST++++++
```

The sample program first pokes in the data. Then it sets the string A$ with trailing blanks. A$ is then printed with the trailing spaces and "+" signs to show the beginning and end of the string. The last print shows the result after the trailing spaces have been trimmed.

The assembly language program listing is given below and a description of it follows.

```
                    ORG  174000A

174.000   353            TRIM XCHG      D,E TO H,L
174.001   006  000            MVI  B,0
174.003   116                 MOV  C,M  COUNT TO B,C
174.004   171                 MOV  A,C
174.005   270                 CMP  B    TEST IF ZERO LENGTH
174.006   310                 RZ        NOTHING TO DO
174.007   043                 INX  H
174.010   136                 MOV  E,M  LO ADDRS TO E-REG
174.011   043                 INX  H
174.012   126                 MOV  D,M  HI ADDRS TO D-REG
174.013   353                 XCHG      ADDRS TO H,L
174.014   011                 DAD  B    ADDRS OF LAST+1
174.015   053            LOOP DCX  H
174.016   176                 MOV  A,M  GRAB CHAR
174.017   376  040            CPI  ' '  TEST IF BLANK
174.021   302  030  174          JNZ  FIN  NO, WE'RE DONE
```

```
174.024   015                   DCR  C
174.025   302  015  174         JNZ  LOOP ALL CHARS DONE

174.030   353              FIN  XCHG
174.031   053                   DCX  H     BACKUP TO
174.032   053                   DCX  H           COUNT ADDRS
174.033   161                   MOV  M,C  RESET COUNT
174.034   311                   RET       GO HOME

                                END  TRIM
```

This program first moves the D,E pair into H,L and then fetches the string's length and address. The length is put into B,C and tested for zero. If it is zero, nothing needs to be done. After the address is loaded into D,E it is swapped into H,L. Then the length is added to give the address of the end of the string plus one.

In the loop, the character on the end of the string is tested to see if it is a blank. If not, the program finishes. If so, the length count is decreased and the program loops back to check the next previous character.

When the program finds the first non-blank from the end of the string (or the end of the string), it stores the reduced count in the string descriptor. Note that it is necessary to "back-up" to the location of the count before the new value is stored. Finally, the program exits to MBASIC.

Additional information on the use of assembly language subroutines will be included in the next release of MBASIC. Development work is presently underway by Heath's software developers. Be patient.

EOF

Spinwriter for the H89

Spinwriter model 5510 may be used with the H89 computer by adding a jumper between pin 19 on the printer connector to pin 4 on the H89 connector. The switches on the Spinwriter should be set with switch numbers 2, 7, and 8 in the UP position and all the others down. This information was provided by Blain Schmidt of Salt Lake City, Utah.

Changing your address? Be sure and let us know since the software catalog and REMark are mailed bulk rate and it is not forwarded or returned.

CUT ALONG THIS LINE

HUG MEMBERSHIP RENEWAL FORM

When was the last time you renewed?

Check your ID card for your expiration date.

IS THE INFORMATION ON THE REVERSE SIDE CORRECT?
IF NOT FILL IN BELOW.

Name _____

Address _____

ity-State _____

Zip _____

REMEBER — ENCLOSE CHECK OR MONEY ORDER

CHECK THE APPROPIATE BOX AND RETURN TO HUG

RENEWAL RATES		NEW MEMBERSHIP FEE IS:
US DOMESTIC	$15 ☐	$18 ☐
CANADA	$17 ☐ US FUNDS	$20 ☐
INTERNAT'L*	$22 ☐ US FUNDS	$28 ☐

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

Heath
Users'
Group
Hilltop Road
St. Joseph MI 49085

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

POSTMASTER: If undeliverable,
plese do not return.

885-2011