REMark

Issue 31 • August 1982

THUG

SOFTWARE

Official magazine for users of Heath/Zenith computer equipment.

SMASH THE '89's 64K RAM LIMIT!

128K RAM BOARD — $595

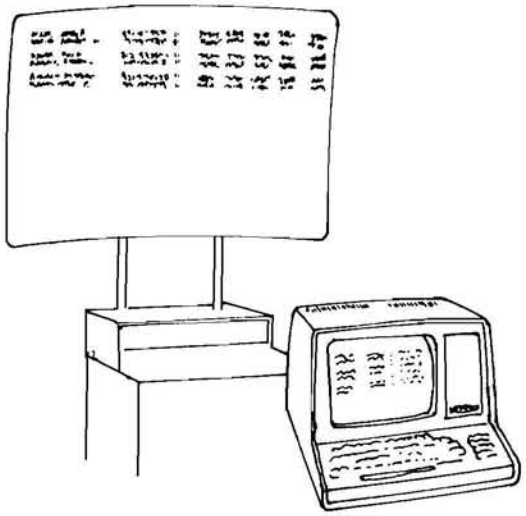
order no. 77318

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

Featuring:

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

Delivery beginning March 15, 1982.



VIDEO OUTPUT — $79

order no. 77319

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

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

Delivery beginning March 15, 1982.

LOWER PRICE! 16K RAM — $125

order no. 77311

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

DOUBLE DENSITY DISK CONTROLLER — $595

order no. 77316

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

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

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

The package includes:

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

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

DOUBLE DENSITY SUBSYSTEMS

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

COMPLETE SYSTEMS

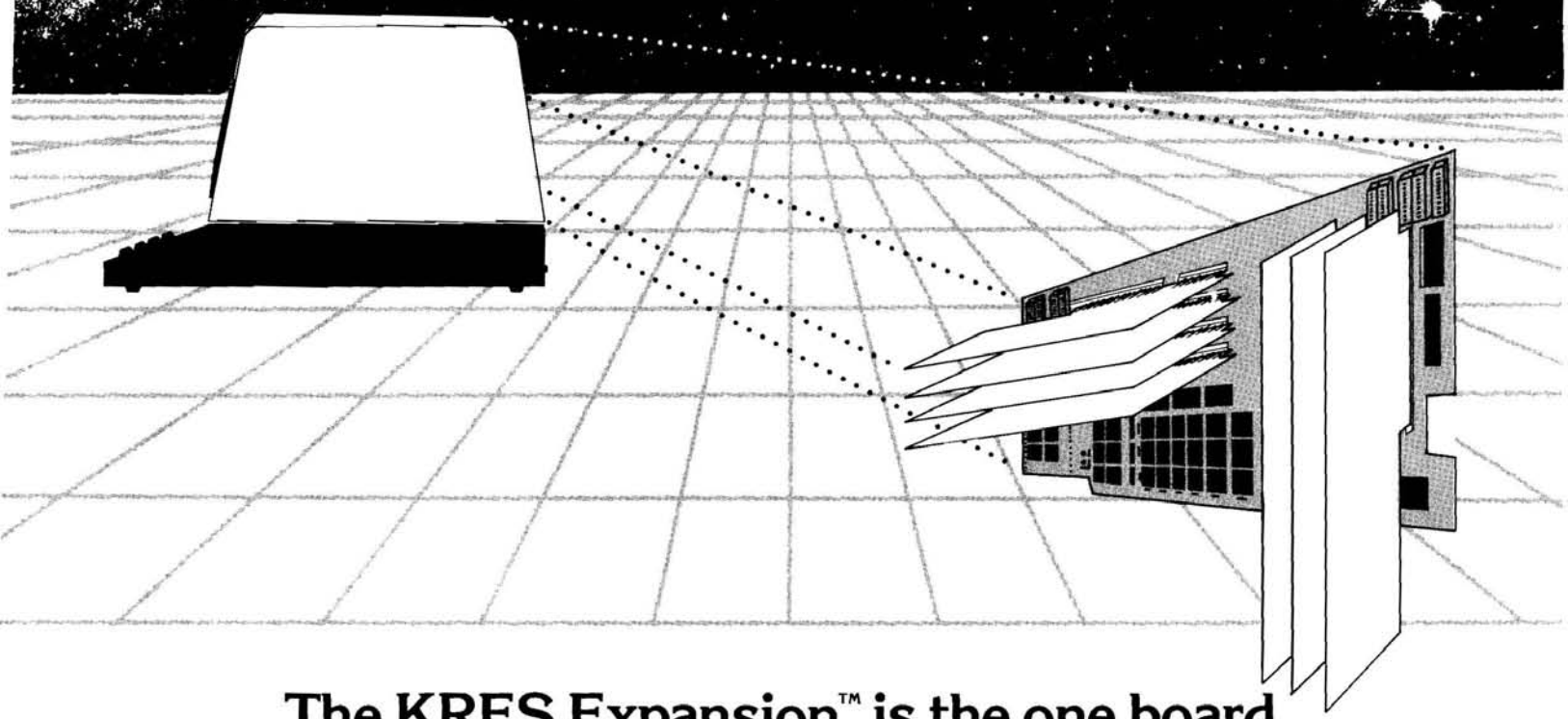
As well as manufacturing enhancements for the '89 [also '88 and '90], we are a Zenith Data Systems OEM, and have all of their hardware and software products available as well. We can provide a completely integrated system, combining the best Zenith products with our own to provide the exact system capabilities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

ORDERING INFORMATION

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

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

KRES Engineering Conquers Space in the H/Z-89



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

You've probably already heard about the KRES Expansion™ that **accommodates up to 7 full-size** (standard Heath) **boards**— more than doubling your present capability. Here are some of the unique features:

EASY INSTALLATION

- Plugs directly into the left and right hand slots, parallel to the CPU card
- No soldering; no trace cutting
- Installs in minutes

SWITCH SELECTED PORT DECODING

- The 4 I/O port select lines in each of the 4 horizontal expansion slots may be individually selected to any 8-port boundary
- Reassigns all 16 I/O lines
- Easily reachable switches are located across top of board

SHADOW OPERATION™

- This KRES exclusive allows you to leave two identically ported boards (that otherwise might conflict) in the system and operate them one at a time.

ENHANCED BUS STRUCTURE

- All 7 slots will accept full-sized right hand cards
- Each horizontal expansion slot will also accept any left hand card
- All 4 horizontal expansion slots give specially designed boards access to a dual bus structure combining all signals on the left and right hand buses
- The data bus on all 7 slots is fully buffered

BOARD XCHG™

- Board exchange feature allows top expansion slot to be electrically interchanged with one of the right hand slots
- Solves bulky cabling problems

AND THERE'S MORE

- Sockets for 16K of RAM are built in, saving you about $100
- The KRES Expansion™ board and its 4 added slots operate from either the existing power supply (with Heath/Zenith 5 Amp regulator) or a secondary supply
- 4 layer printed circuit board for electrically quiet operation, even at higher speeds

MORE TO COME

- KRES will be offering a full line of new boards utilizing both the standard and dual bus structure

GET MORE INFORMATION TODAY

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

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

SEND ME MORE DETAILS!

NAME _____

ADDRESS _____

CITY _____ STATE ____

ZIP _____ PHONE (____) _____

H/Z-89 is a registered trademark of the Heath Co.

REMark Issue 31 • August 1982

on the stack

>CAT

ON THE COVER: This month's cover is an illustration by Ray Massa of the Keyboard Studio Inc., Birmingham, MI, which will also be the cover for our new software catalog to be released this fall.

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

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

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

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

Send payment to: Heath Users' Group
Hilltop Road, St.
Joseph, MI 49085

HUG Manager ..... Bob Ellerton
Software Coordinator and Developer ..... Jim Blake
Software Engineer ..... Pat Swayne
HUG Bulletin Board and Software Developer .. Terry Jensen
HUG Secretary ..... Margaret Bacon

REMark Editor ..... Walt Gillespie
Assistant Editor ..... Nancy Strunk

Introducing - - Walt Gillespie

***As you can see,* REMark** *has changed face considerably. These changes have been brought about by a gentleman by the name of Walt Gillespie. Walt is the newest addition to the HUG staff and has been assigned the task of improving both the quality and quantity of material appearing in REMark as our new Editor. Walt comes to us with a wide background in printing and graphics. Previously, Walt owned and operated Minit-Man Printing located in Otsego, Michigan. He is the man responsible for developing the H-19 Video Layout Sheets along with a bunch of other goodies for the Heath/Zenith Computer Product Line. Walt has outlined a very aggressive direction for REMark that I am sure you will enjoy and encourage. I hope all of you will join with me in welcoming* **Walt Gillespie** *as the new Editor of REMark, avid HUGGIE and staunch supporter of Heath/Zenith Computer Products.*

Bob Ellerton, Manager
Heath Users' Group

TIME MARCHES ON!!!

The inevitable always seems to come to pass. That is, using the old saying, "Nothing remains the same but change."

I know many of you have been with Heath computers from the start, my first was an H-8 plus H9, all 5000 solder connections of it. I can still vividly remember 'cursing the darkness' trying to bring up a program using the tape system. Little help was available from the tech's at Heath as they too were still learning. There was no national users group to cry to for help either. Well, as I said, changes were inevitable and as the growth of Heath and HUG show it's been for the good.

With the first National HUG Conference this year, (it should be a reality by the time you read this), many new ideas will be forth coming for changes.

Change also has come to REMark. Over the past few years REMark has gone from a sporadic publication to a regular monthly magazine. Also the size has increased, plus the addition of limited advertising. Now new changes, as you can see we have gone to typesetting our copy. One reason for typesetting is an increase of approximately 25% in available copy area over the old typewriter method. This additional area will be used to include more articles plus such things as photos and expand the overall graphic design of the magazine.

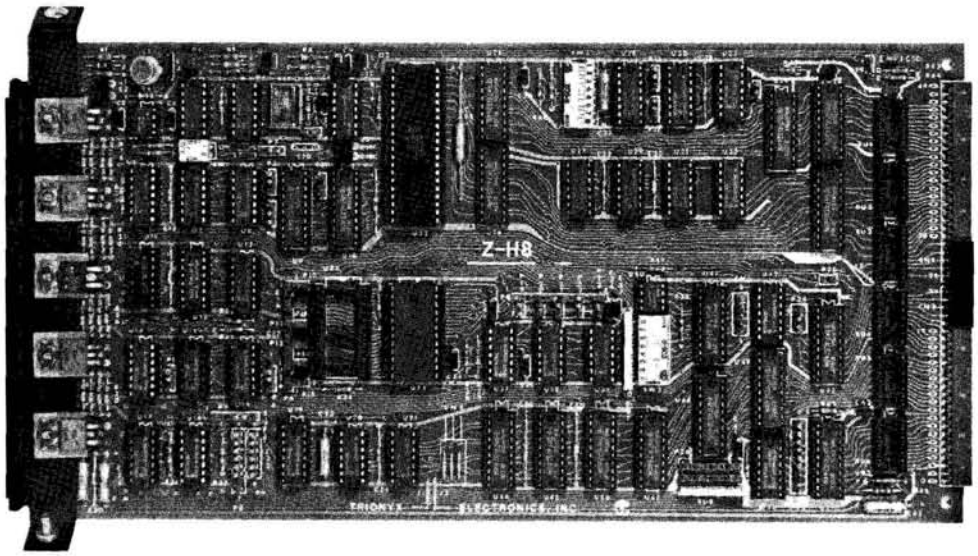
Changes in the future at HUG will come, as in the past, from you the membership. It is your input, questions, comments and gripes that make this happen. Some changes we like, others we don't, but it's only when you give your opinion that your voice is heard and changes made.

The computer industry as a whole is going through great change, with new hardware, operating systems and software changing rapidly. It is with this fact in mind that you should consider your contribution to your organization, HUG. There are many people around who have gained the knowledge of how to best use their computers. Unfortunately there are many who have not yet gained this insight and need help. A number of individuals have contributed much in the way of valuable materials over the past years, that is why HUG has grown. It is with these thoughts in mind that I call on you for your ideas, programs, articles, your input. I hope over the following months to be able to address here in REMark those areas the we need input for and to be more specific as to what need. As always your general input and comments are most welcome.

Walt Gillespie, REMark Editor

Z80 CPU BOARD for the H8*

* H8 is a Registered Trademark of the Heath Company



Model Z-H8

$ 300.00 Assembled $ 250.00 Kit

Programmable Clock Rate — 2 MHz or 4 MHz Operation

Clock Rate under Software Control — May Be Changed at Any Time

Selection of Wait States (None, 1/2, 1 or 2) at 4 MHz

Buss Termination Network • 51 Integrated Circuits • Transistor Power Supply

Includes Heath Extended Configuration (ORG ZERO) Circuits

Exclusive Z80 Front Panel Monitor ROM — Based Upon Heath XCON-8 ROM

Exclusive Software Speed Utilities (For Both HDOS* and CP/M**) on Diskette

---

Fully Compatable With All Heath H8* Hardware and Software

* H8 and HDOS are Registered Trademarks of the Heath Company.
** CP/M is a Registered Trademark of Digital Research.

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

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

TRIONYX ELECTRONICS, INC.

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

Buggin HUG

Terry (or Bob),

The program that enables users to alter flags regardless of any 'L' status which appeared in REMark issue 28 is indeed useful. For the benefit of those who would now proceed to remove all flags from their system files, perhaps a warning is in order. If a file has the 'C' flag set, then you should LEAVE IT ALONE! Removing the 'C' flag from certain system files will render a disk un-bootable.

DALE LAMM 70555,302 (Micro-net)

Dear Bob:

As a novice in computing I have not felt able to contribute anything to REMark, but I have enjoyed the magazine and particularly the articles applicable to CP/M:MBASIC systems, since that is what I have. I have completed both Heath courses in BASIC, but neither of these is particularly strong in the area of sequential and random access files in my opinion. Or to put it another way, I had difficulty in using these kinds of files in writing programs.

A friend recommended a book of the subject which I have found to be very good and I would like to pass the word along to others. It is LeRoy Finkel and Jerald R. Brown, *Data File Programming in BASIC— A Self-Teaching Guide,* 1981, John Wiley and Sons, Inc., 342 pp., paperback, about $12.

It is written particularly for Microsoft BASIC with comments on using TRS-80 BASIC and Northstar BASIC. It begins with a quick review of BASIC statements and programs and then moves into sequential and random access files with lots of exercises to enable you to be sure you understand the ideas. There are a few errors, probably typos, but it was quite evident what was intended. Heavy emphasis is placed on writing self-checking programs so that the program doesn't bomb if a mistake is made entering data. The only significant short-coming I found is that they do not cover placing multiple datasets in one random access record. They say this is explained in most basic reference materials. (Heath's Programming in Microsoft BASIC does a good job on this.)

I hope this book will be as helpful to others as it has been to me.

Sincerly,
Bob Anderson
2608 Winter Street
St. Albans, WV 25177

Dear HUG:

The 5.25 inch hard sector device driver from HUG, SY.DVD, makes HDOS an even more pleasant system to use, by simplifying and speeding up the boot process and eliminating the relentless banging from disk head loads and unloads. For anyone still using the Heath supplied SY.DVD in HDOS 2.0, I heartily suggest that you order HUG's SY.DVD (HUG part no. 885-1095) today. When I first put up SY.DVD, however, I discovered that it did not live up to one of the claims made for it. Rather than the 35% speed improvement in large program loads, I experienced a 35% increase in the time to assemble a program (see BUSS #39 for the results of a timing test I ran.)

Fortunately, in the March issue of HUG (REMark issue 27) there was an article by Pat Swayne, "Improvements to the HUG SY: Device Driver" which used separate timers for head select and deselect. Since my hobby time is limited and since assembling a program as large as SY.DVD on a one disk system is not a trivial matter, I did not get around to installing the fixes until this week.

First of all I would like to report two typographical errors in the listing published in REMark. The Symbol "MBOOT" should be "MFBOOT" and the symbol "FMOFDLY" should be "MFOFDLY". Once these mistakes were corrected the new SY.DVD assembled without errors. Incidently, with no HDOS except SYSCMD and no PIP there were 18 sectors to spare on my single density, single sided, hard sectored disk. Someday soon I must get some larger disks!

Given my experience with the previous version, the first thing I did after I verified that the modified SY.DVD worked was to run a timing test on it. The test I used this time was to time the assembly of SY.DVD itself. I assembled both DKH17 and DKH17I. By typing ahead, I was able to avoid variations due to my typing speed. Hitting an extra return after the last ASM command causes the computer to BEEP when it is done, which keeps me from falling asleep at the stop watch. In any case, the results of my new timings are:

HUG's SY.DVD with SET SY0: SELECT 500@ L 11 minutes 38 seconds.
Heath's SY.DVD@ L 9 minutes 36 seconds.
HUG's SY.DVD with improvements SET SY0: MOTOR 1600@ L 9 minutes 03 seconds.

One extra note: All three tests were run on the same disk which had been initialized with HUG's INITAUTO rather than HEATH's INIT17. INITAUTO is responsible for some of the speed improvements in HUG's SY.DVD and I assume that Heath's SY.DVD also benefits from the change in track formatting.

So, I am pleased to report that HUG's SY.DVD is now an unqualified im-

provement over the Heath driver that it replaces. Thank you HUG and thank Dean Gibson of UltiMeth Corporation for a fine product. I hope that any future release of HDOS from Heath/Zenith will incorporate these improvements.

Dale Wilson
231 Couch Ave.
St. Louis, MO 63122

Dear REMark Editor:

Although I see no listing for 'Letters to the Editor' I suppose there is some mechanism in REMark for the purpose.

Like others who have the somewhat neglected (by Heath) H11 I have upgraded the system to an 11/23 with hard disc and use RT11V4 operating system. This all works very well for our medical billing system running the program in BASIC 11. I wrote for our Radiology practice in the absence of being able to find any available software suitable for our particular practice situation.

Converting programs written for HT11 BASIC for the RT11V4/BASIC 11 system is no problem but I also have the HT11 FORTRAN system which I would like to be able to run on RT11V4. When this is attempted I get a bad load address error message. Possibly others have a similar problem and, like me, are not knowledgeable enough to figure out how the FORTRAN might be modified to run on RT11V4. I think a patch to FORTRAN, so it could be used for RT11V4, would be of interest (if available). Also I think instructions for modifying the H11A backplane for use with 22 bit addressing so that use might be made of some of the new modules and software being put out by DEC would be of interest. I have heard that such a mod is technically relatively simple but have not seen one published and do not have the technical expertise to figure it out myself. Obviously, such a mod should not interfere with use in the original configuration.

Sincerely,
James Monnahan, M.D.
Box 531, Provo, Utah 84603

Vectored to page 30 ☞

QUESTIONS & ANSWERS

(EDITOR'S NOTE: Some of the following Questions & Answers were contributed by Zenith Data Systems Software Consultation to help get this column started. If you need answers to specific questions on software or hardware problems please drop us a note, Heath User's Group, Hilltop Road, St. Joseph, MI 49085. Please keep your questions brief and to the point. We will do our best to answer you here in this column in future issues.)

Q: Can I run any CP/M program on my Heath/Zenith computer? If not, how do I know what I can run?

A: CP/M is currently the most popular microcomputer operating system used. It functions in virtually the same manner on most microcomputers. But this doesn't always guarantee compatibility from one computer to another. The only media that is universally recorded in the same format is the single-sided, single-density 8" floppy disk. All other media (especially 5.25" disks) are recorded in the unique technique preferred by the computer's maufacturer. This means that software available in a non-Heath/Zenith format will probably not run on the Heath/Zenith computer. Typically, software available on 8" single-sided, single-density disks will work on Heath/Zenith computers; but to be safe, check with the manufacturer before you buy the software. Software developed by other manufacturers which uses the special function keys of a non-Heath/Zenith video terminal may also fail to function properly because they may not be sending the same codes to the software.

Q: How can I view HEX instead of the SPLIT OCTAL on my computer screen?

A: HEX is available with the new Monitor ROM available from Heath Co.

Q: When I try to assemble the demo program in the HDOS assembly language section I get errors. What am I doing wrong?

A: There is a statement containing XTEX HDOS at the beginning of the program. This tells the assembler to search for the file HDOS.ACM. This file doesn't exist and causes a "U" error to occur at locations in the program. The solution is to create the HDOS.ACM file, using PIP as follows:

```
PIP SY0:HDOS.ACM=SY1:ASCII.ACM,SY1:
                                HOSDEF.ACM,SY1:HOSEQU.ACM
```

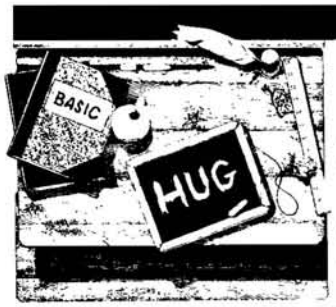
Drive SY1: should contain the Software Tools disk and drive SY0: your working system disk.

Q: I have an H/Z89 with an internal drive and newly installed Z-89-37 controler card with two drives. When I bring up HDOS and try to INITialize a new disk, the system hangs. What is wrong?

A: HDOS comes with a "DK:" device driver for the H/Z-47 installed. When it tries to reset all drives before doing the INITialize, seeing the 37 controller instead of a 47 controller causes the software to "hang". What is needed is the new device update, HOS-5-UP, which comes with only one device driver for disks installed (SY:) and has included device drivers for the new equipment, (H37's, MX-80 and H-25). Also included is modified support utilities for this new equipment such as INIT.ABS.

Q: How can I transfer files between a hard-sectored disk drive and a soft-sectored disk drive?

Vectored to page 30 ☞



A TUTORIAL ON THE USE OF THE ESCAPE CODES WHEN USING THE BASIC LANGUAGE

Kenneth Mortimer PE
352 Green Acres Drive
Valparaiso, IN 46383

EDITORS NOTE: This article was written using Benton Harbor Basic. To use this material with Microsoft Basic some changes must be made. In place of the PAUSE statement use Z$=INPUT$(1), although this is not an exact replacement for the PAUSE statement it will work with the majority of the examples used here.

Many of you have been puzzled when certain print statements are used in basic programs to move the cursor, to enter or leave the reverse video mode, to enter or to leave the graphic mode or to use the twenty-fifth line on the screen of one of the Heath/Zenith terminals. Most of these print statements seem to include the BASIC Expression CHR$(27) or CHR#(27) which may have previously been used in defining a string variable, probably named E$. These programs have been using the terminal escape codes. The codes are described beginning on page 12-10 of my Heath Operating/Service Manual, yours might be different. The ESCAPE key is much like the CONTROL key with which you are familiar. It generates a non printing character which when combined with a printing character gives the computer a command. To see how it works Boot up your machine, enter some material onto the screen and then simultaneously press the ESC key and the "E" key (be sure it is a capital "E"). The screen will clear and the cursor will return to the "home" or upper left hand corner. Any of the escape codes can be entered in this manner. However, what if you want to emphasize a warning note on the screen by printing it in reverse video? In this case you would want to have your BASIC program generate the "enter reverse video" and "exit reverse video" escape sequences.

First,for those of you not familiar with the CHR$ function, this function generates a single character string that corresponds to the ASCII equivalent of the argument. If we look at a table of ASCII codes (Appendix B in the BHBASIC Section of your HDOS manual) you will find that the decimal equivalent of ESC is 27. Therefore the BASIC statement E$=CHR$(27) will generate the escape character and store it as string variable E$.

In order to make this lesson more meaningful for you I will divide it into smaller units and ask you to work along with me at your terminal. Since we will eventually build a table of escape codes I will ask you to use the same line numbers that I use reserving lines 100 to 500 for escape sequences and lines 1000 and above for the body of the program.

(UNIT 1) CLEARING THE SCREEN

1) Enter the following BASIC statements:

```
  100 E$=CHR$(27)     :REM ESCAPE KEY CODE
  150 E1$=E$+"E"      :REM ERASE SCREEN
10000 END
```

2) SAVE this program segment as "ESCAPE" since we will build on it and use it again.
3) Now add the following lines:

```
1000 PRINT E1$
1010 PAUSE
```

4) RUN this program segment. You will note that the screen will clear and the cursor will return to the home position. Hit the return key and you will receive your END message.

(UNIT 2) THE LINE FEED PROBLEM

1) Read the material on the PRINT command in your BASIC manual. You will find that at the end of each PRINT statement a line feed/carriage return code is generated. If you don't want a line feed/carriage return code generated you should end the list with either a comma or a semicolon. When we are using the escape sequences to move the cursor (see units 3 and 4) the line feed at the end of our PRINT statement would move the cursor from the position where our program had placed it.
Let's play with this for a few minutes.
2) Enter the following statements: (Recall ESCAPE if you deleted it from memory.)

```
1000 PRINT E1$
1010 PRINT 1
1020 PRINT 2
1030 PRINT 3
1040 PAUSE
1050 PRINT 4
1060 PRINT 5
```

3) RUN this program. It will pause after printing "3" with the cursor in the left hand column of the line below the "3". The PRINT statement generated the line feed. Hit RETURN and the computer will print out "4" and "5" with a blank line between the "3" and the "4". When you hit the RETURN key you generated another line feed.
4) Let's see if we can correct this. If we place a semicolon after our "3" in line 1030 no line feed code is generated. A comma would cause a skip to the next print zone but not generate a line feed. Rewrite line 1030 as follows

```
1030 PRINT 3;
```

5) RUN this program. It will pause after printing "3" with the cursor on the same line as the "3" and just to the right of it. Now hit RETURN and the "4" will be printed just under the "3". The RETURN generated the line feed. 6) How can I cause the machine to PAUSE so that I can observe an out-

put and then continue on without hitting the RETURN key? If you will look at your BASIC manual you will find that an integral constant is placed after the PAUSE command and the computer will pause for two times this constant in milliseconds and then resume the program. Therefore if I want to have the display hold for two seconds and then proceed I would rewrite line 1040 as follows:

```
1040 PAUSE 1000
```

(ED. For MBASIC use G=1000:I="":FOR H=1 TO G:PRINT I;:NEXT H, increase the value of G for more time delay. If the ';' is left out after the PRINT H statement a line feed will be issued)

7) RUN this program. It will pause for two seconds after it has printed "3". The cursor will still be on the same line as the "3" and when the program resumes by itself the "4" will be printed on the same line as the three. You generated no line feed code.

8) How can you get the computer to print the numbers "1", "2" and "3" as before, PAUSE two seconds and then print "4" and "5" without skipping the line?

I'll let you figure out that one.

(UNIT 3) CURSOR MOVEMENT

1) The cursor may be moved in the four screen directions by the escape codes ESC A, ESC B, ESC C and ESC D. These codes move the cursor one space up, down, right and left respectively. Now add the following BASIC statements to your program.

```
110 A$=E$+"A"     :REM MOVE CURSOR UP
120 B$=E$+"B"     :REM MOVE CURSOR DOWN
130 C$=E$+"C"     :REM MOVE CURSOR RIGHT
140 D$=E$+"D"     :REM MOVE CURSOR LEFT
```

Now SAVE all the statements numbered below 1000.

2) In order to have room to move the cursor around a point as a demonstration we must move the cursor to a point near the center of the screen and place a character there. This time you will have to do it the hard way but in the next unit you will learn a more straight forward way to place the cursor at any point on the screen. Enter the following BASIC statements to move the cursor to the center of the screen

```
1000 PRINT E1$
1010 FOR I=1 TO 11
1020 PRINT
1030 NEXT I       :REM MOVE CURSOR DOWN 12 LINES
1040 PRINT , , ,"X";     :REM PRINT×AT THE CENTER OF THE SCREEN
1050 PAUSE 1000
```

3) Now make the cursor move first up one line, then to the left one column, then down one line and finally one column to the right by adding the following statements.

```
1060 PRINT A$;     :REM MOVE CURSOR UP
1070 PAUSE 1000
1080 PRINT D$;     :REM MOVE CURSOR LEFT
1090 PAUSE 1000
1100 PRINT B$;     :REM MOVE CURSOR DOWN
1110 PAUSE 1000
1120 PRINT C$;     :REM MOVE CURSOR RIGHT
1130 PAUSE 1000
10000 END
```

4) RUN this program and watch the cursor move in a square pattern. The "X" was placed on the screen as a point of reference. Now you might want to have the cursor dance about the screen in some pattern that you like.

(UNIT 4) DIRECT CURSOR ADDRESSING

1) Unit 3 showed a relatively crude way of moving the cursor to a specific point on the screen but this would have erased most of the material passed on its way there. One can move the cursor directly to a point by using the ESC Y code followed by two characters that define the position. The reason that these characters must be represented by a decimal number thirty one greater than their column or line position is explained in your operators manual. Since the variables defining the new cursor position must be included in the command I decided that this operation could be best performed by using the user defined function. Add to your program the following statement

```
260 DEF FN Y$(X,Y)=E$+"Y"+CHR$(31+Y)
    +CHR$(31+X)
```

In this function "X" is the column number counted from the left and "Y" is the line number counted from the top. If you

To see how it works

Boot up your Machine,....

are going to plot equations where you want the line number counted from the bottom you can easily change the statement.

2) Delete lines 1010,1020,1030 and 1040 from the program for unit 3.

Replace them with the following statements:

```
1010 X=40
1020 Y=12
1030 PRINT FN Y$(X,Y);"X";
```

3) Now RUN the program and see what happens. You may now want to change the values of X and Y and start the pattern at a different point. Remember that this program is not "bullet-proofed" so keep X<81 and Y<25.

(UNIT 5) SAVING AND RECALLING THE CURSOR POSITION

1) It is often convenient to be able to remember the position of the cursor, go to another portion of the screen to do some work and then return to the saved cursor position. In order to accomplish this let us add three more escape codes to our library.

```
180 H$=E$+"H"      :REM CURSOR HOME
190 J$=E$+"j"      :REM SAVE CURSOR
    POSITION
210 K1$=E$+"k"     :REM RETURN TO SAVED
    CURSOR POSITION
```

Note that the characters "j" and "k" are lower case letters.

ESC H is the code that moves the cursor to its home position or the upper left hand corner of the screen.

2) Delete lines 1030 through 1130 from the program of unit 4.

Replace them with the following statements

```
1030 PRINT FN Y$(X,Y);"X";J$;     :REM MOVE
     CURSOR AND STORE POSITION
1040 PAUSE 1000
1050 PRINT H$      :REM MOVE CURSOR TO
     HOME
```

Now run the program

and see what happens

```
1060 PAUSE 1000
1070 PRINT K1$;    :REM MOVE CURSOR BACK
     TO SAVED POSITION
1080 PAUSE 1000
```

3) RUN the program. The cursor will be moved from its position after the "X" to the upper left hand corner of the screen. Its previous position will have been stored so after the pause it will move back to its previous position.

(UNIT 6) CURSOR BLANKING

1) It is often desirable to turn the cursor off for a period and then to turn it back on. To do this add the two following statements to your library.

```
250 X5$=E$+"x5"    :REM TURN CURSOR OFF
280 Y5$=E$+"y5"    :REM TURN CURSOR ON
```

Again note that the characters "x" and "y" are lower case letters.

2) Delete lines 1000 through 1080 from the previous program.

Add the following statements

```
1000 PRINT X5$     :REM TURN CURSOR OFF
1010 PAUSE
1020 PRINT Y5$     :REM TURN CURSOR BACK
     ON
1030 PAUSE
```

3) RUN the program. The cursor will disappear. Hit the RETURN key and the cursor will reappear. Hit the RETURN key again to end the program.

(UNIT 7) THE REVERSE VIDEO MODE

1) There are many times that it is desirable to emphasize some material on the screen by printing it in dark letters on a light background. This is called the "reverse video mode". To enter and leave this mode you will add two more escape codes to your library

```
230 P$=E$+"p"      :REM ENTER REVERSE
    VIDEO MODE
240 Q$+E$+"q"      :REM EXIT REVERSE VIDEO
    MODE
```

Again note the "p" and "q" are lower case letters

2) Delete lines 1000 through 1030 from the previous program.

Add the following statements

```
1000 PRINT P$      :REM ENTER REVERSE
     VIDEO MODE
1010 PAUSE
1020 PRINT Q$      :REM EXIT REVERSE VIDEO
     MODE
```

3) RUN the program. When the computer comes to the PAUSE type in a few characters. They will appear in reverse video. Hit the RETURN KEY and an error message will appear in the normal video mode. The reason for the error message is that the system does not know what to do with the code entered during the PAUSE.

(UNIT 8) THE GRAPHIC MODE

1) I will not attempt to justify the use of the Graphic Mode. I'm sure that if you look at the variety of graphic characters presented in your operating manual your imagination will conjure up a few uses, particularly if you are interested in developing some games or in CAI (Computer Aided Instruction). To enter and leave this mode you will have to again add two more escape code sequences to your library. They are

```
160 F$=E$+"F"      :REM ENTER GRAPHIC
    MODE
170 G$=E$+"G"      :REM EXIT GRAPHIC MODE
```

2) Delete lines 1000 through 1020 from the previous pro-

gram and add the following statements.

```
1000 PRINT F$      :REM ENTER GRAPHIC MODE
1010 PAUSE
1020 PRINT G$      :REM EXIT GRAPHIC MODE
```

3) RUN the program. When the computer comes to the PAUSE type in a string of lower case letters. Compare the output with that shown in the operating manual. Hit the RETURN key and again the error message will appear. What would have happened if you had omitted line 1020? How would you correct this without returning to a cold start?

(UNIT 9) THE TWENTY-FIFTH LINE

1) The HEATH advertising copy talks about a 25th line at the bottom of the screen and many of you have seen it used in canned programs that you have purchased or in some of the demonstration programs. How do we open up that line for use? Remember it cannot be used in the same manner as the other twenty four lines. It is most often used to relay information to the operator, to indentify the special function keys or perform some similar task. To use the twentyfifth line one must perform the following steps

a) enable the 25th line

b) move the cursor to the 25th line by direct cursor addressing

c) enter the desired information into the 25th line

d) move the cursor from the 25th line

2) To use the 25th line you must add the following statements to your library

```
240 X1$=E$+"x1"      :REM ENABLE 25TH LINE
270 Y1$=E$+"y1"      :REM DISABLE 25TH LINE
```

3) Delete lines 1000 through 1020 from the previous program. Add the following statements

```
1000 PRINT J$      :REM SAVE CURSOR POSITION
1010 PRINT X1$     :REM ENABLE 25TH LINE
1020 X=1
1030 Y=25
1040 PRINT FN Y$(X,Y)     :REM MOVE CURSOR TO THE 25TH LINE
1050 PRINT "THIS IS THE TWENTY FIFTH LINE"
1060 PRINT K1$     :REM MOVE CURSOR TO SAVED POSITION
```

4) RUN the program. Notice the comment that you have placed on the 25th line.

5) SAVE the program since you will now try to see what will cause the material on the 25th line to be erased. If you do erase the 25th line with any of the next five steps just recall and rerun the program.

6) Depress the OFF LINE key. Simutaneously depress the ESC and E key. Does the erase sequence erase the 25th line? Return to on line mode.

7) Sign off and Reboot. Does this sequence erase the 25th line?

8) Depress the right hand SHIFT key and the RESET key at the same time.

9) BOOT to perform a "cold start". Does this sequence erase the 25th line?

10) Using the command mode in BASIC execute the following command:

```
PRINT Y1$
```

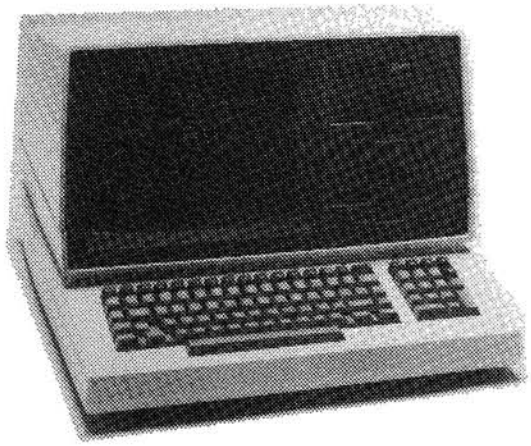
Now you should have a better understanding of the ESCAPE sequences. You might try and change from the underscore cursor to a block cursor, set up a sequence equivalent to the erase key or enter the shifted keypad mode. It's up to your own imagination and ingenuity. One warning should be made, the ESCAPE sequences used in this tutorial are the Heath Escape sequences, if your terminal is set for the ANSI ESCAPE sequences then all of the codes will be different and you will have to look them up in the reference manual and make the necessary changes.

Now you should have a better understanding of the escape sequences

The material in the tutorial is also adaptable to PASCAL, but of course, the CHR function will have to be used instead of CHR$ and the program statements will all be in PASCAL form. Since most versions of FORTRAN compilers written for microcomputers have limited string handling capabilities (FORTRAN wasn't developed with those type of operations in mind) it is beyond the scope of this paper to adapt these codes to FORTRAN. ✱

ZENITH'S FORMULA FOR THE FUTURE

The Products...the People...the Potential...at Zenith Data Systems they're all exceptional. We're a young, dynamic company, developing the next generation of microcomputer and terminal systems.



The Products:

We are developing personal information/ state-of-the-art terminals and desktop computers using the latest in 16-bit technology. Our products feature high resolution graphic displays, networking and advanced ergonomic design.

The People:

Our rapid increase in sales has created an immediate demand for the following professionals:

COMPUTER/TERMINAL DESIGN ENGINEERS

Several positions available in small business computer systems, terminal systems and Continuing Engineering. Individuals are responsible for product design and development. BSEE, MSEE or equivalent degree plus minimum 3 years microcomputer, terminal or data communication systems design experience using 8/16-bit technology required.

SOFTWARE DESIGN ENGINEERS

Engineers needed for enhancement of operating systems for microcomputer products. Responsibilities include utility design, adaptation of existing products to microcomputer systems, and reviewing/ specifying hardware interfaces. BSCS or equivalent degree, demonstrated knowledge of assembly level language (8080/Z80, 68000/Z8000/8086) and experience with operating system development (preferably CP/M, HDOS and UNIX) required. Familiarity with microcomputer hardware desirable.

The Potential:

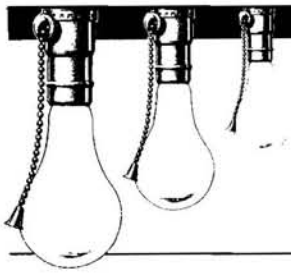
At Zenith Data Systems our engineering teams are small and organized around product lines. That means "hands-on" project control from start to finish...high visibility...and exceptional opportunities for growth! To investigate our competitive salaries and outstanding benefits, including relocation assistance to our beautiful resort-like community of St. Joseph, Michigan—just 90 miles from Chicago—send resume in complete confidence or call collect:

ZENITH | data systems

Kathy Kniola / (616) 982-3844
Hilltop Road
St. Joseph, Michigan 49085

A Wholly Owned Subsidiary of Zenith Radio Corporation

An Equal Opportunity Employer m/f



"Turn on light number one...."

Jim Blake
HUG Software Developer

My wife, Joann, is used to hearing me talk to the computer, and usually my dialog consists of words that should not appear on these pages. The other night, however, she overheard me say..."Turn on light number one." Over and over and over... and light number one did not come on! You see, I was experimenting with the **Housemaster home control system** from Atra. It is a single card that occupies one of the right hand expansion slots in the '89 and performs a variety of functions.

The basic Housemaster unit includes:

1 - A real-time 24 hour clock/calendar.
2 - Voice recognition.
3 - Two AY-3-8910 Sound synthesizers.
4 - BSR X-10 home control interface.
5 - A very nice voice/time control program.
6 - Over 15 additional programs. Some with source.

And a 90 page user manual.

Optionally available is:

* Battery backup for the clock. Why have to set it every time you power up?
* Dual RS-232 communications ports. Another must if you have 5 and 8 inch drives.
* Linear-predictive-coding based voice synthesis module.
* Phoneme-based voice synthesis module.

About the hardware.

The Housemaster unit is packaged on a printed circuit card which occupies one of the two I/O slots in the upper right rear of the '89. Three plugs provide detachable connections from the board to a supplied microphone, an external audio amplifier, and a BSR X-10 transducer. On board is the clock/calendar chip, the two sound generator IC's, a voice recognition circuit, and a small audio amplifier suitable for driving small speakers. The board can optionally be populated with the battery backup, the IC's from your serial card, and a choice of two voice synthesis modules.

The clock/calendar provides seconds, minutes, hour,(12 or 24 hour format) weekday, day-of-month, year and timing pulses for applications such as music synthesis.

The two AY-3-8910's offer a total of six separately programmable sound channels which are divided (three each) into two audio signals for stereo amplification. There are fourteen registers which control; the production of noise or tone, the frequency of the noise or tone, the volume and the control of various amplitude envelopes.

The voice recognition circuit is a dual band pass, zero crossing trigger.

The BSR X-10 home control interface operates through software control of an I/O bit which is interfaced to the ultrasonic tranducer or optionally through an optically-isolated direct connect to the house wiring.

And the Software.

An impressive amount of support and demonstrative software is provided with the system.

The main control program combines the voice recognition, date/time and home interface functions. From a rich menu of options, the user can control household functions from the keyboard, by voice or from a menu of preselected times; even link to user supplied sub-routines. The user can 'teach' it to recognize up to 24 different voice commands.

If the system is in the 'listen' mode (as opposed to the 'time' mode) you could walk in the room, even when the radio is playing, and say - "Wake up computer" and your '89 could ring a bell, play a tune and switch to the 'Voice Control On' mode.

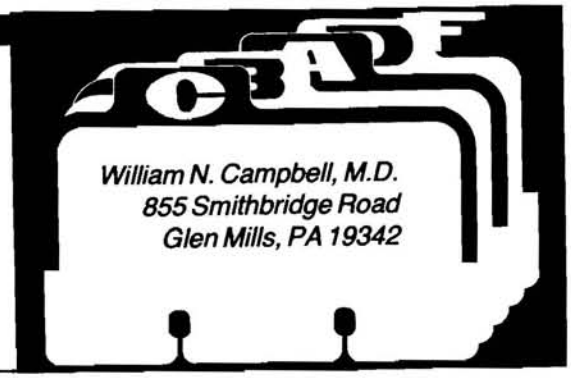
Say - "Turn on lamp one." And on comes the lamp.
Say - "Dim lamp one 50%." And the lamp dims.
Say - "What time is it?" And the speakers chime the hours and minutes.
"Go to sleep computer" And the unit returns to the 'Listen mode'. With the voice synthesis option, he could speak the time.

All this magic is accomplished by speaking the command phrase 4 times, so the program can get an average of what your command 'sounds like'. You can either accept the result, (which is stored) or throw it away and try it again. That's when Joann heard me yelling downstairs. I said "turn on the light" and he goes, "unrecognizable signature." So I try again, and each time, I raise my impatient voice. He, of course, doesn't recognize anything I say. Joann comes down stairs and in a very sexy voice, says "turn on the light." CLICK... the damn thing turns on!

Moving right along. There's a basic test program that tests the sound chips and shows you how to program them. TIME.BAS which retrieves the time and date. CONTROL.ASM for controlling the BSR. VOXCAL.BAS for calibrating the hardware and shows how to use voice recognition and sound detection. (I obviously didn't use this one much!)

Vectored to page 37 ☞

GETTING STARTED WITH RANDOM FILES



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

USING MBASIC (BASIC-80 REV. 5.21) AND CP/M

Abstract -

Some MBASIC programs running under CP/M and dealing with random files are presented and discussed. This article assumes familiarity with my previous article which dealt in considerable detail with sequential file handling using some "mailing list" programs. These random programs, with some modifications, can be altered to handle other data bases. Since random file techniques are significantly different with 5.21 MBASIC, these programs will not run under HDOS MBASIC version 4.82 (see article in REMark issue 10 for random file handling under HDOS).

PRELIMINARY CONSIDERATIONS -

Random files are much more "powerful" than sequential files since you can access any record in a large data base in seconds, for review or updating. One can also quickly append additional records onto the end of an existing random file. I will present some programs to create a random file, change it to a sequential file (and vice versa), sort it, merge it with another file, print out its contents, and rapidly find any record using a "key" (since we are using a mailing list the key will be the last name), using a binary search technique.

Sorting (putting in order, alphabetizing) and merging (combining two already sorted files into one sorted file), makes it necessary to take into account whether the data is in upper or lower case. You MUST have all data in the same "case format". For example, you might elect to have ALL data in upper case if you have a matrix printer. Or, you might choose to have upper AND lower case if you have a "letter quality" printer. For example, my last name could be either 'CAMPBELL' *OR* 'Campbell' if it occurred in your file. Whichever format you decide on is all right, but you MUST be consistent with your data files. One OR the other, but NOT some with one format and some with the other! Otherwise sorting and merging will not work. Also, if you are using a binary search for "fast find" of a last name, the data entry of the last name MUST be in the same "case format" as the last name as it preexists in the disk file. Otherwise, the search will fail! For the sake of simplicity, I will present these programs assuming ALL CAPS. I suggest that you use very small test files and also use ALL CAPS for your data while you are entering these programs and testing them. After you understand the logic in the programs, then you may decide just which "case format" you will use if you use the programs for your own "mailing list data base". The important thing here is that you MUST be consistent.

Remember that a data FILE consists of one or more records. A RECORD consists of one or more data items which are logically grouped together. The individual data items are called FIELDS. In the case of sequential data files, the fields can be separated from one another by a DELIMITER. We used the reverse slash (\) as a delimiter in the mailing list programs presented previously. Those were sequential files. Also, remember that those records were separated from one another by a <cr>. If you think about those records, you will realize that the lengths of the sequential records were VARIABLE. One big difference in random files is that the record lengths are FIXED. You select the record length in advance. Whatever the selected record length is, ALL records must be the same length. If you examine your MBASIC manual, you will see that the programs presented, and the illustrations used also require that each field be of a fixed length. This tends to make the record lengths quite long (as compared to sequential records) since you must allow for the MAXIMUM possible length for EACH field. However, I will present these random file programs using the reverse slash (\) as a delimiter for the fields. This has the advantage of allowing for an AVERAGE shorter record length although we MUST have a fixed record length. This will become apparent to you as we proceed.

Listed below are SUMMARIES of procedures for random file manipulation. (DO NOT MEMORIZE THEM, THEY ARE THERE FOR REFERENCE. YOU WILL RAPIDLY BECOME FAMILIAR WITH THEM!)

Note that ALL data records MUST be of same length. Note also that all programs which access any one data file MUST use the same record length. In all the examples that we will give, our record length will be 85 characters, including a "tag" that is used to mark the end of each record (for our own use).

To write data to random disk file:

1. OPEN disk file for random access (this opens a preexisting or new random file AND creates a "buffer memory area" of the same size as the record length). Example - OPEN "R",1,"RDATA",85 (open random file RDATA as random file number 1, with record length of 85).

2. FIELD the buffer (manual shows how to use multiple fields. I will field the entire "buffer" as one field).

Example - FIELD 1, 85 as R$

3. Enter one data unit (one record). (We will put all data into variable X$.)
4. LSET the data (MBASIC "pads" data, if necessary, to desired length with blanks (spaces) and then puts it into buffer).

Example - LSET R$=X$+CHR$(0) (note that the CHR$(0) is our "tag" which marks end of record for our later use).

5. PUT the data (writes the record from buffer to disk file).

Example - PUT 1,n (n is optional and represents a record number).

6. Repeat steps 3 to 5 as desired.
7. CLOSE the disk file.

Example - CLOSE #1

To access data from a random disk file:

1. OPEN disk file for random access (Ex. OPEN "R",1,"RDATA",85).
2. FIELD the buffer (Ex. FIELD 1, 85 as R$).
3. GET the desired record and put in buffer (Ex. GET 1,n) (n is opt rec #).
4. Retrieve one record from buffer (Ex. X$=R$), optionally remove OUR marker and any trailing blanks (spaces).
5. Do "whatever" (assume you just want to look at record - Ex. PRINT X$).
6. Repeat steps 3 to 5 as desired.
7. CLOSE the disk file. (Ex. CLOSE #1).

Some important notes:

Note that when we opened the random file we also defined the record length. The ',85' above indicates that the records have a length of 85 characters. If you do not put ',n' (n = record length) at the end of the OPEN statement, then MBASIC version 5.21 assumes a default record length of 128. If record length is to be greater than 128, then you MUST so indicate in the OPEN statement, AND you MUST ALSO declare record length using "space/S:n" at the time you load MBASIC from the CP/M monitor prompt, where n = record length. (For example, assume a record length of 200. When you load MBASIC you do so from the A>

```
10 REM  STOR.BAS  convert seq to random file, latter with record length of 85
20 REM
30 ON ERROR GOTO 140
40 INPUT "SEQ INPUT FILE NAME... ";S$:INPUT "RAN OUTPUT FILE NAME... ";T$
50 KILL T$
60 OPEN "I",1,S$
70 OPEN "R",2,T$,85
80 FIELD 2,85 AS R$
90 IF EOF(1) THEN 130
100 LINE INPUT #1,X$:X$=X$+CHR$(0):LSET R$=X$:PRINT X$
110 PUT 2
120 GOTO 90
130 CLOSE:END
140 IF ERR=53 AND ERL=50 THEN RESUME NEXT
150 ON ERROR GOTO 0
```

```
10 REM   TEST.BAS     TEST PGM TO CHECK INPUT METHOD FOR RANDOM FILES
20 REM   This pgm for CP/M and MBASIC vers. 5.2
30 REM
40 DEFINT A-Z
50 INPUT "RANDOM ACCESS FILE NAME (EX. RDATA)... ";T$:PRINT
60 OPEN "R",1,T$,85:'  open random file as file #1 with record length of 85
70 FIELD #1,85 AS R$:' "FIELD" the record as 85 (same as record length)
80 ' next line checks to see if there IS a previous file - see text
90 K=LOF(1):IF K=0 THEN PRINT:PRINT "NO SUCH FILE!!":PRINT:CLOSE:KILL T$:GOTO 50
100 INPUT "TYPE REC # WANTED (IF DONE, TYPE 0)... ";K:IF K=0 THEN 240
110 GET 1,K:' get the desired record and put it in buffer memory - see text
120 X$=R$:' put the contents of buffer memory into string variable X$ - see text
130 IF ASC(LEFT$(X$,1))=0 THEN PRINT:PRINT "NUMBER TOO HIGH!":PRINT:GOTO 100
140 PRINT "Here is contents of record # ";K
150 PRINT X$
160 PRINT "The length of this record is ";LEN(X$)
170 V=INSTR(1,X$,CHR$(0)):' test for position of null and print it
180 PRINT "If a null (CHR$(0)) is present its position is at ";V
190 IF V<>0 THEN X$=LEFT$(X$,V-1)
200 PRINT "Here is record ";K;" with its trailing blanks stripped off"
210 PRINT X$
220 PRINT "The length of this record is now ";LEN(X$)
230 X$="":PRINT:GOTO 100
240 CLOSE:END
```

```
10 REM  PRINT.BAS   Print out on line printer contents of random file and
20 '                preface each record with its absolute file position
30 '                 A CP/M MBASIC program     0 = zero;  O = "OH"
40 '
50 DEFINT A-Z:N=1
60 INPUT "RANDOM FILE NAME... ";P$
70 OPEN "R",1,P$,85
80 FIELD 1, 85 AS R$
90 GET #1
100 IF ASC(R$) = 0 THEN 130
110 Z=INSTR(1,R$,CHR$(0)):X$=LEFT$(R$,Z-1)
120 LPRINT N;X$:N=N+1:GOTO 90
130 CLOSE:END
```

prompt like this - MBASIC /S:200< cr>. Then, any OPEN statement in your program MUST ALSO declare record length with ,200 at the end of the open statement.

The maximum length of any individual FIELD is 255, although record lengths (with several individual fields) can be greater than 255.

The LOF(n)(length of file) statement is only valid with files of 128 records or less. For practical purposes I use it only to determine if a file exists on disk.

Although the manual indicates that the EOF(n)(end of file) statement can be used to determine the total number of records in a file, I have not found it reliable. However, MBASIC returns a 0 (zero) if queried about a nonexistent record using ASC(R$). This seems to be true for ALL records above the last legal record in any given random file. Hence, this can be used in a binary search routine to find the last legal valid record in any given file. This is illustrated in some of the programs which will be given. I am indebted to "WIZ-10" of Compuserve for the fundamental binary search algorithm upon which my MBASIC binary search for EOF is based. The 0 (zero) which MBASIC returns for nonexistent records is a "null" (binary zero) (CHR$(0)) (ASCII 0).

MBASIC pads each record (if necessary) when using LSET and RSET commands with spaces ("blanks") (ASCII 32), to ensure that all records have the desired length. With LSET, the blanks are trailing and with RSET any necessary blanks are put before the data in any given record. I always use LSET.

The "PUT" and "GET" statements may each have an optional record number specified. The number may be actual, or within a numeric variable. If no record number is specified, then either statement acts upon the NEXT record. Thus, PUT #1,14 puts the data in the fielded buffer onto disk as record # 14. PUT #1 simply puts the data in the fielded buffer onto disk as the next record AFTER the last record that was PUT. This is automatic. The programs will illustrate these points.

Assuming that you have a sequential data file of the correct format (created by MAKESEQ.BAS in my last article, or an editor, and shown in FIGURE A of that article) I now suggest you enter and RUN the program STOR.BAS. Be sure to convert any lower case records to upper case first, as I assume upper case entries for all programs. After you become familiar with random files you may use lower case as indicated previously. Whichever you decide to use you MUST be consistent! STOR.BAS simply converts a sequential data file into a new file which is random. (The original sequential file remains unchanged on the disk.) The record length is 85, and the entire record is fielded as 85. Therefore NO record in your sequential file should be longer than 84 characters, including the delimiters associated with each record. (The 85th character is the "null" marker that we will append.) You may have to abbreviate your records for our programs as written. (After you are familiar with random files you would simply increase the record length, but that would complicate matters at this point.) If you have a sequential file called DATA you would enter DATA when STOR.BAS asks for input sequential file name. When the program asks for random file name I called it RDATA just to make it different. Whatever you wish to call it is, of course, acceptable and your choice, (as long as the name is unique).

EXPLANATION OF STOR.BAS -

Line 40 accepts your input for the preexisting disk sequential file name, and name of the random file to be created, into S$ and T$ respectively.

Line 50 erases any preexisting file if it is the same name as the contents of string variable T$. Be careful here. Note that IF the file is NOT found in line 50, then our error routine (lines 30, 140 and 150) "traps" the "file not found" error and resumes program execution on line 60. Line 60 opens the sequential file for input, and line 70 opens the random file (for input OR output) ("R" does this) as file number 2, and with a record length of 85. Note the syntax of these OPEN statements. You will use these statements frequently. Note that when you open a random file, the OPEN statement also creates a memory buffer for you of the same length as the record length given in the open statement. (If you did not include the record length it would default to 128).) Next, line 80 FIELDs the buffer of the random file (#2 in this instance) and creates only ONE field, of length 85. You may use "FIELD 2,85" or "FIELD #2,85". Either way is correct syntax. Line 100 inputs the first record from sequential file named in S$, and puts the contents of this record into string variable X$. Using concatenation (combining 2 or more string variables using '+') we tack on our marker. The marker (tag) is a null, and that is what the CHR$(0) represents. If our record length including delimiters was 74 in the sequential file, the length is now 75 after we add on the CHR$(0). Next (line 100), we use the LSET statement to put the record into the buffer, and this statement also pads the contents of X$ with trailing blanks so that the record becomes precisely 85 'characters' in length. This is all transparent to the user. Last, and still on line 100, we print the contents of X$ on the screen for the user's edification. Line 110 PUTs the contents of the memory buffer (the contents of R$) onto disk as record #1. Note that we did not have to specify the record number as we desired the first record to BE record 1. Each time 'PUT 2' is accessed, it automatically bumps the record number up by a value of 1. This is transparent to the user. Line 120 cycles back to line 90 which tests for EOF in our sequential file, and if NOT EOF then continues with line 100 which repeats all the above steps, except we are now dealing with record #2. This continues until EOF of the sequential file is reached, and then line 90 branches to line 130 which closes both files and ends program. Now, note that we used all the items listed in the summary above dealing with 'writing to a random file'.

TEST.BAS EXPLANATION -

Please enter and run program TEST.BAS now. I believe the program itself contains enough comments to

explain it. However, I do suggest that you study this program and run it until you well understand ALL the ramifications as it will surely help you understand random files. Therefore I will spend some time explaining certain things and emphasizing others now. This is probably the most important program in this article. Line 40 defines all numeric variables as integers, and you will see this frequently in my programs. It makes programs run faster. In certain of the programs it is a necessity. Line 90 needs some elucidation. The LOF(n) statement's use is quite limited in MBASIC. It returns the last record number of a random file as long as there are no more than 128 records in the file. But, if there are NO records in a file, then for practical purposes the file does NOT exist. In this instance, LOF(n) returns a 0 (zero). We utilize this in line 90 by testing for the presence or absence of 0. If 0, then there is no such file (except we just created it when we opened it, so we CLOSE it and then KILL it and cycle back to line 60 so that the user can enter a valid random file name). Any value other than 0 means that there is such a random file and it does contain some records, although we may not know, and actually do not care how many records are present in this program.

You are able to rapidly access ANY record in a random file as long as you know the record number (record number = the absolute POSITION of the record in any given random file). Presumably you ran the program PRDATA.BAS in my last article and already have hard copy printout of each record along with its absolute position in the data file. The positions in the random file are identical. Line 100 asks the user to supply the record number wanted. If you don't have hard copy printout just enter 1<cr>. K (numerical variable) accepts your input, and then line 110 GETs that record and puts it into the random file buffer created by line 70. This buffer was fielded as 85 characters in length and is accessed through string variable R$. So, line 120 puts the contents of R$ (the fielded buffer) into string variable called X$. Note that at this point the record length MUST be (and IS) 85, and the contents of X$ are our record + our marker (null) + 0 or more spaces. Line 130 tests X$ to see if its FIRST character is a null. (If it IS, then the record number entered into K in line 110 was higher than the last valid record in the file - MBASIC transparently returns a 0 (zero) when we test for it in this fashion. Note that we could also have tested for it by using ASC(R$) instead of the LEFT$ statement here. Same results, and same implications. Assuming that the record # entered was valid and not past the end of file, however, the rest of the program is dedicated to showing you the contents of the record, its length, the presence of the null that WE appended when we created this random file, and how we use the marker 'null' to strip away it AND any following trailing blanks (spaces) that may have been padded with the LSET statement when the file was created. Now, you should study this program until you feel VERY comfortable with all of its lines. Line 230 'clears' the string variable X$, and cycles back to 100 so you can enter another record number.

HERE is one of the IMPORTANT POINTs illustrated in this program - AFTER line 190, the string variable X$ contains your record EXACTLY as it was when you were using X$ in the sequential file programs previously presented. And, if you desired, you could take it apart to access each individual field just as you did in the sequential program LABEL.BAS! You should now see what we are driving at. RANDOM FILE HANDLING, as I am presenting it, REQUIRES certain steps which were listed in the summaries given previously. There are not many of these steps. The END RESULT, however is that you can manipulate the record string just as you did in the SEQUENTIAL programs (AFTER you strip away the marker (CHR$(0)) and any trailing blanks)! We tested for the marker in line 170, put its position in variable V and then line 190 strips away the marker and any trailing blanks. Pay good attention to this as you will find frequent use of these techniques.

Also, if you check TEST.BAS against the second summary given earlier, you can see that we utilized all the steps given, finally closing the file in line 240, after 0 (zero) was entered by the user as input in line 100.

You should have noted in this program that we did NOT manipulate R$. In all the programs presented in this article I have used "R$" ONLY to define the contents of the buffer created by the open statement. You should NEVER try to use the string variable associated with the random memory buffer (R$ in these programs) for ANY other purpose since its "pointers" reference the buffer, and NOT ordinary string space in memory. (You CAN test it, with IF as we have done.) Instead, put the contents of R$ into another string variable (X$ for example) and then you can manipulate the latter. If you DID use R$ as a regular string variable, then it would no longer reference the random memory buffer space, and your programs may CRASH. Therefore, do not manipulate R$! Let it only define the buffer in the FIELD statement, as we have done in these programs. (The same thing applies if you use multiple string variables to define multiple fields in the FIELD statement. Do not use these string variables for other purposes in your programs.)

If you examine TEST.BAS you will see that it contains all the needed routines to access any random file whose length is 85, and which is also fielded as 85. You simply delete lines 140-160, 180, and 200. The string X$ in line 210 contains only the actual record accessed by its absolute position in the file, without the marker and without any trailing blanks.

Last, you may have figured out that MBASIC finds your desired record number in the random file by counting. MBASIC knows how long each record is since WE supplied that information when we created, and/or accessed any given random file in the OPEN statement (or, it defaulted to 128 if we did not supply the record length). Regardless of the number of records in any given file, you will find that MBASIC's "counting" is extremely fast, and only takes a second or two. And, it is also for this reason that any given set of programs that work with one par-

ticular data base and record length should ALWAYS use the same record length for EACH of its programs. So, in all of our programs given in this article we will use 85 as the record length.

PRINT.BAS EXPLANATION -

This short program simply OPENs a random file of your choice (record length of 85), FIELDs it with R$ and also with field length of 85, GETs the first record of the file (file #1), locates our marker strip (null) and strips it and the trailing blanks away, then prints the record number (1, for the first record) and also prints the contents of the record number, then cycles back to line 90 to repeat the process for the next higher record number. You will recall that if the GET statement does not specify a record (it does not in line 90) then MBASIC automatically increments the record number by 1 for each "GET". Line 100 checks each record for the presence of a binary null in the FIRST position of the record. If none is present, it is a valid record. If it IS present, then MBASIC is informing us that the record is one more than the last valid record, and program branches to 130, CLOSEs the file and ENDs. If you do not have a printer, simply change the LPRINT statement in 120 to PRINT, and all will be scrolled out on your terminal screen. You can stop the scrolling at any point with a CTRL S, and start it again with another CTRL S. If you examine your printout you probably note that the file is NOT in alphabetical order by last name or any other part of the record that you wish to use as a 'key' for sorting. Which brings us to -

EXPLANATION OF SMSORT.BAS -

"Sorting" means to put in order. Numbers can be put in increasing OR decreasing order as far as their values are concerned. Usually, with strings, such as last names in a mailing list, one alphabetizes the names in increasing alphabetical order. There are many different sort routines. One usually taught is the "bubble" sort. This is a very inefficient sort for large numbers of records. The Shell-Metzner (two folks who were the originators of this sort) sort is fast and a good general purpose sort. Frankly, I have never taken the time to try to fathom the logic of this algorithm (reason why the "bubble sort" is so commonly used for teaching purposes is that it CAN be understood by most mortals). However, it is very reliable and quite fast. The sort presented here in MBASIC is quite a bit faster than an assembly language sort using the bubble sort (with 250 - 500 or more records)! The actual sort routine is from lines 180 through 230. As noted, I leave it to you to figure out how it works. But I will explain the rest of the program. Line 40 defines all numerical variables beginning with all letters of the alphabet as integers. We have previously discussed this. Line 50 shows how to dimension a string array. Here we have allowed for a maximum of 250 strings to be sorted. The actual maximum value depends on the length of the strings being sorted as well as the amount of memory available in your computer. (We will discuss other ways of sorting to get around the memory problem later in this article.) Line 50 contains part of our error routine, along with lines 290 and 300. We discussed this in the sequential file handling article. Line 70 erases any preexisting file with same name as the one you entered in line 60 for the output random file name, so be careful here! If file did not exist our error routine traps the error and allows the program to continue on line 80. We will use the variable K (in the GET statement in line 120) and, since we desire to begin with the first record in the file we set K to 1 in line 80. Lines 90 and 100 then OPEN the input and output random files, both with a record length of 85; then line 110 FIELDs the input file buffer as 85 (we fielded the output file buffer with value of 85 in line 250, but we could have fielded it at this point in the program). Line 120 GETs record K from random file opened as #1 (the input random file). Line 130 checks each record for EOF. Whenever a null (binary zero) is

Vectored to page 22 ☞

```
10 REM  SMSORT.BAS RANDOM FILE TO RANDOM FILE SHELL-METZNER SORT
20 ' A CP/M MBASIC program
30 '
40 DEFINT A-Z
50 DIM A$(250):L =1:ON ERROR GOTO 290
60 INPUT "RAN FILE NAME TO BE SORTED.. ";P$:INPUT "RAN OUTPUT FILE NAME... ";T$
70 KILL T$
80 K =1
90 OPEN "R",1,P$,85
100 OPEN "R",2,T$,85
110 FIELD #1, 85 AS R$
120 GET #1,K
130 IF ASC(R$)=0 THEN 180
140 Z=INSTR(1,R$,CHR$(0)):X$=LEFT$(R$,Z-1)
150 A$(L)=X$:L =L+1
160 K=K+1:GOTO 110
170 ' beginning of sort. When C=0 sort is finished. Print C to show progress.
180 L=L-1:C=L:B=L
190 C=INT(C/2):PRINT C:IF C=0 THEN 240 ELSE D=1:E=B-C
200 F=D
210 G=F+C:IF A$(F)<=A$(G) THEN 230
220 SWAP A$(F),A$(G):F=F-C:IF F<1 THEN 230 ELSE 210
230 D=D+1:IF D>E  THEN 190 ELSE 200
240 X=1
250 FIELD #2, 85 AS R$
260 IF X =L+1 THEN 280 ELSE LSET R$=A$(X)+CHR$(0):X =X+1
270 PUT 2:GOTO 260
280 CLOSE:END
290 IF ERR=53 AND ERL=70 THEN RESUME NEXT
300 ON ERROR GOTO 0
```

DG IS Heath/Zenith

...with the F

CONGRATULATIONS!

DG congratulates HUG on the first NATIONAL HUG CONFERENCE. In celebration DG is offering special price discounts on all H8 hardware products during the month of August. The sale ends August 31, 1982. Call for details or see us in Chicago.

H8 PRODUCTS

The Most Extensive Line of Hardware Support for The H8®

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

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

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

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

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

THE DG STATIC 64

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

SUPER LOW POWER—Power dissipation typically less than 4 watts (less than any memory board available for the Heath H8). Uses Single Supply 5-Volt memory devices.

CAPACITY—Up to 64K RAM OR Up to 64K EPROM (type 2716/2516) OR any combination of RAM and EPROM up to 64K.

SPEED—4MHz with NO wait states.

Bank-select fully compatible with the DG-64D.

Fully assembled and tested. Priced from $299.00 (0K) to $599.00 (64K). 16K Chip Sets: $125.00.

H8/H89 Software

Software Support for HDOS 2.0

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

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

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

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

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

n Computer Support uture Built-In

SUPER 89

TURN YOUR H/Z 88/89 INTO A PROFESSIONAL QUALITY COMPUTER HIGHER RELIABILITY NOW — MORE EXPANDABILITY FOR THE FUTURE

FEATURES	DG-SUPER 89	H/Z 88/89
USER MEMORY The DG Super 89 comes standard with 64K of user RAM "on-board". This configuration can be increased up to 256K of bank selectable/write protectable RAM.	64K-256K	16K-64K
Ø ORIGIN MODIFICATION No further modification, such as required on the H/Z-89, is necessary to operate in either a standard Heath/Zenith CP/M or HDOS 2.0 Operating System environment.	NOT NECESSARY	ADD-ON
CP/M-HDOS COMPATIBLE	YES	REQUIRES MODIFICATION
PERIPHERAL EXPANSION SLOTS DG Electronics has made provision in the design of the unit not only for compatibility with the standard factory expansion slots, but also for future expansion by doubling the number of available expansion slots on the unit to 6 instead of the standard 3.	6	3
ON-BOARD AMD9511 For those users who perform large amounts of arithmetic computations the DG Super 89 has provision on-board for use of the AMD 9511 arithmetic processor.	YES (PURCHASE SEPARATELY)	NO
CPU CLOCK FREQ. The CPU in the DG Super 89 operates at twice the speed of the standard H/Z-89.	4MHz+	2.048MHz
MULTI-USER CAPABILITY With up to 256K of bank selectable RAM on board the DG Super 89 offers the option of MULTI-USER CONFIGURATIONS of up to 4 users.	YES	NO
ENHANCED MONITOR Enhanced monitor supports the advanced features of the Super 89.	YES	NO
REAL TIME CLOCK The DG Super 89 comes standard with an on-board real time clock.	YES	NO
PARITY CHECK ON RAM For those who are sticklers for accuracy, the DG Super 89 has parity check to make the user aware of errors occuring in the RAM during use.	YES	NO
SERIAL PORTS ON BOARD The DG Super 89 offers an additional serial I/O port for greater convenience and flexibility.	2	1

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

Compatible with all currently available Heath/Zenith hardware devices.

D·G ELECTRONIC DEVELOPMENTS CO.

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

Vectored from page 19

returned, we are at "end of file" and proceed to line 180 which is the beginning of the sort routine. Assuming a valid record, line 140 finds OUR binary null marker (the one we put on as a tag at the end of our record when we created the random file), and strips it and any trailing blanks away, leaving only our record with the delimiters in X$. Lines 150 and 160 do the following. The string variable X$ containing our record is placed into A$(L) which is in the array. L and K are each incremented by 1, and we loop back to get the next record. This looping, and the routine, continues until we reach the EOF in line 130 and then branch to line 180. The array is ready to sort.

The only thing I have added to this sort routine (which is a standard Shell-Metzner sort) is the "PRINT C" in line 190. This simply displays the value of C as the sort loops through line 190 and gives you something to watch as the sort proceeds. You will see that C keeps dropping in value until 0 is reached, at which time the sort is over and line 190 branches to line 240. At this point the array which contains our records is now sorted. Next, we simply put the contents of the now sorted array into the new random file (the output file) with lines 260 and 270. We retrieve the array using numeric variable X which is incremented by 1 at the end of line 260, and place each sorted record in array into its proper location in the output file after tagging it with our marker and LSETing it. When the end of the records in the variable is reached we branch to 280 and CLOSE.

This would be a good place to emphasize something. The marker we are using is a binary null. This has NOTHING to do with the binary null that MBASIC returns at EOF. This is sheer coincidence. I was using CHR$(0) for a marker before I found out MBASIC returned this value at end of file as we previously have discussed.

In order to sort larger files than we can using MBASIC, I subdivide larger files (if necessary) and create one or more sequential files using a program such as RTOS.BAS. If I have a 1000 record file with average record length of 80 to 85, I OPEN a second sequential output file as #3, naming the sequential files simply A and B. Then I change line 120 to read as follows:

IF LEFT$(X$,1) < "M" THEN PRINT #2,X$:GOTO 80 ELSE PRINT #3,X$:GOTO 80

This creates 2 files on disk, A containing all LAST names beginning with A (through L), and B containing all LAST names beginning with M (through Z).

I then exit MBASIC and use a machine language sort. HUG sells an excellent CP/M machine language sort authored by Bill Moss and Pat Swayne. (Ed: Doc refers here to HUG part number 885-1212 - cost $20.00). I then sort file A, then sort file B to output files C and D respectively. Then, I either concatenate the 2 sorted files C and D, or merge them. If the former, I then put them back into a new random file using STOR.BAS (to which I have added the pertinent lines from PRINT.BAS so that I can get hard copy printout of the new file with each record preceded by its record number its absolute position in the file) at the same time as the new random file is being created! If merging 2 sorted files, I use MERGE.BAS.

The above procedure works very well. Suppose, however, that you desired to sort your file by zip code, rather than by last name. This is NOT difficult and here is how to do it. Refer back to the last article on sequential file handling and review how to manipulate strings using the INSTR function to isolate the various reverse slash (\) delimiters. After you get the desired key (zip code) into a string variable name of your choice, simply put it at the beginning of X$. Suppose the variable name of your choice was ZIP$. You then say X$=ZIP$+X$. This routine would go after line 110, just before line 120 in program RTOS.BAS. Then, after sorting, when you use STOR.BAS to put the sorted file(s) back into a random file, you grab off the zip code from the beginning of the record, discard any delimiter if you used one (you don't need one to separate the zip from last name since the zip is ALWAYS 5 characters long), and put the zip code back in its correct location in the record. This is a good way to learn string handling techniques, incidentally!

You will be very pleasantly surprised at how fast HUG's CP/M sort program is. Just seconds for a 500 record file! It is also a Shell-Metzner sort.

RTOS.BAS -

You should have no trouble understanding this program as it simply converts a random file on disk to a sequential file, leaving the random file as it was. However, line 110 "packs" the sequential file as it is being created. I will explain this in the "FINAL REMARKS" at the end of this article.

```
10 REM  RTOS.BAS   convert random (record length 85) to sequential file
20 '  A CP/M MBASIC program      0 = zero;  O = "OH"
30 ' also "packs" file (removes any "deleted" records)
40 '
50 INPUT "RAN INPUT FILE NAME... ";P$:INPUT "SEQ OUTPUT FILE NAME... ";T$
60 OPEN "R",1,P$,85:OPEN "O",2,T$
70 FIELD 1, 85 AS R$
80 GET #1
90 IF ASC(R$) = 0 THEN 130:'  looking for EOF
100 Z=INSTR(1,R$,CHR$(0)):X$=LEFT$(R$,Z-1)
110 IF LEFT$(X$,1)="*" THEN 80:' do NOT put any record with 1st char * to disk
120 PRINT #2,X$:GOTO 80
130 CLOSE:END
```

EXPLANATION OF MERGE.BAS -

It is not difficult to understand the logic of this program. First, one must have 2 sorted sequential files on disk. The object is to combine them into one sorted random disk file. After INPUTting the names of the 3 files in lines 50 through 70, and placing the names into string variables A$, B$, and C$ respectively, lines 80 through 100 OPEN the 3 disk data files. The random file is given a record length of 85 in line 100 as part of the OPEN statement, and the as-

```
10 REM  MERGE.BAS  merges 2 SORTED seq files into 1 SORTED random file
20 ' A CP/M MBASIC program.  Also provides hardcopy output with absolute rec #
30 '              (0=zero    O="OH")
40 '
50 INPUT "FIRST SEQ SORTED FILE NAME (EX. FILE1)..... ";A$
60 INPUT "SECOND SEQ SORTED FILE NAME (EX. FILE2)..... ";B$
70 INPUT "OUTPUT MERGED RAN FILE NAME (EX. RANCOMB)..... ";C$
80 OPEN "I",1,A$
90 OPEN "I",2,B$
100 OPEN "R",3,C$,85
110 N=1
120 FIELD 3, 85 AS R$
130 IF EOF(1) THEN 190
140 LINE INPUT#1,X$:X$=X$+CHR$(0)
150 IF Z=-1 THEN Z=0:GOTO 180
160 IF EOF(2) THEN 240
170 LINE INPUT#2,Y$:Y$=Y$+CHR$(0)
180 IF X$<Y$ THEN WRITE #3,X$:PUT #3,N:LPRINT N;" "+X$:N=N+1:Z=-1:
          GOTO 130 ELSE  WRITE #3,Y$:PUT #3,N:LPRINT N;" "+Y$:N=N+1:GOTO 160
190 WRITE #3,Y$:PUT #3,N:LPRINT N;" "+Y$:N=N+1
200 IF EOF(2) THEN 290
210 LINE INPUT#2,Y$:Y$=Y$+CHR$(0)
220 WRITE #3,Y$:PUT #3,N:LPRINT N;" "+Y$:N=N+1
230 GOTO 200
240 WRITE #3,X$:PUT #3,N:LPRINT N;" "+X$:N=N+1
250 IF EOF(1) THEN 290
260 LINE INPUT#1,X$:X$=X$+CHR$(0)
270 WRITE #3,X$:PUT #3,N:LPRINT N;" "+X$:N=N+1
280 GOTO 250
290 PRINT
300 PRINT "MERGED DATA FILES ARE NOW IN FILE CALLED ";C$;"."
310 PRINT "HAVE A GOOD DAY!"
320 CLOSE #1: CLOSE #2: CLOSE #3
330 END
```

sociated memory buffer of this random file is FIELDed as one field with a length of 85 characters, using R$ in line 120.

The fundamental logic involved here is simply to compare the first record of each of the 2 sequential files, and to put the lesser of the alphabetized records to disk as the first record of the new random file. Then, bring in another record from this "lesser" file and compare it to the other record already brought in. The lesser of these 2 records is then written to disk as the second record of the random file. Etc. Etc. Line 140 pulls in the first record from file #1 (A$) and line 170 brings in the first record of the file #2 (B$), both using LINE INPUT statements. Both records then have our "null" marker appended to them concatenating both X$ and Y$ with CHR$(0), the latter being the "null" marker. Line 180 then does the comparison.

Line 180 may need some elucidation. This line is simply an IF...THEN...ELSE statement. Some may not realize that you can have several statements between the 'THEN' and the 'ELSE'. It is perfectly legal in MBASIC as this line demonstrates. Translated, it simply means that if the record in string variable X$ is less than the string variable Y$, THEN put X$ onto disk in the random file and print its absolute position (N), a space, and the record's contents on the printer, increment the value of N by 1, set "Z" to -1 and goto line 130 ELSE we do exactly the same thing with Y$ and goto line 160 (without the flag being set to -1).

Try it, it works nicely. We intentionally used "WRITE #3,X$" (or Y$) here to demonstrate another statement that can be used with random files. The WRITE command here is the same thing as saying LSET R$=X$ (or Y$). It does exactly the same thing. (If you don't have a printer, just change the LPRINTs in this program to PRINTs.)

The only tricky part about a merge program is that you have to allow for several possible eventualities, which depend on which file's "EOF" is reached first. Hence, all the seeming duplication of lines in the program. They simply take care of what must be done whenever EOF occurs in one or the other of the 2 input sequential files. A little study will show you exactly what happens. So, this program as written will successfully merge any 2 already ordered sequential files REGARDLESS of how many records are in either of the files, into one sorted random disk file.

Line 320 could also have been written as "320 CLOSE", as that would have closed all 3 files also.

EXPLANATION OF MAKERAN.BAS

This program is quite similar to MAKESEQ.BAS which was presented and discussed in my last article, except that it has been modified to create a random file similar to the other random files we have been discussing. Many of the lines should be quite familiar to you by now as we have discussed them before in previous portions of this article. Therefore I will only point out the new things that are part of this program. If you scan the various lines you will see the already discussed features of random file manipulation.

Line 160 gosubs to a binary search routine which finds the last legal record of a desired random file. The binary search routine RETURNs from line 650 with the numerical value of the last legal record of the random file in variable K.

A "binary search" is familiar to all of us. In essence, it is what we all do when we look up a word in the dictionary (at least what we did the first time we used a dictionary). We went to the middle of the book, then (depending on what we found there) we went either halfway towards the front of the book OR halfway towards the back of the book. We

continued this process until we found the desired word. This is sometimes called "divide by two, then compare". A binary search with a computer requires that we know the top record number and the bottom record number to begin with. We know that the first record is record number 1, so we put 1 in variable I. We do NOT know the top number, so we pick an arbitrarily large number (I chose 25000) and put its value into N. Then we simply "divide and conquer" in line 600 and get THAT record number. Line 610 looks for the presence or absence of the binary zero (null) that MBASIC returns to the ASC(R$) IF statement when any record is at EOF or above. Line 610 then readjusts the "top" record and the "bottom" record appropriately. Line 620 inquires if I=J. Whenever I DOES equal J, then we are finished, and the value of K at this time is either the absolute position of the last legal record in the random file, OR the position of the first record PAST the last legal number. Line 620 then gosubs to 640 where we test for a binary null in the record in R$ at whatever position K represents. IF a binary null IS present, then we are one past the actual absolute last record and 1 is subtracted from K. Then we RETURN to line 620 which then RETURNs to line 160 where (in this program) K is incremented by 1 which is the first record number for any records we desire to append to this random file (assuming there are records in the file).

Lines 190 through 340 are similar to the ones we used in MAKESEQ.BAS. X$ contains the record which is to be placed in the random file at record number K position. We do that with the usual random statements after appending OUR "null" character to the end of X$. And, line 390 increments K by 1; line 400 branches back to 190 for any more records we desired to append. I believe you will see how the rest of the lines work without further explanation.

Note that I left out the "edit" routine that we used in MAKESEQ.BAS. You SHOULD put the "edit" routine in this program. It should start just past line 330, before line 340. This would be a good place for you to learn how to use

```
10 REM MAKERAN.BAS  Creates random file from scratch or appends to existing one
20 '                This program for CP/M and MBASIC vers 5.2
30 '                Note the LACK of dividing up ONE 255 byte record into
40 '                sub-records as was done with 4.82 MBASIC under HDOS
50 '                (0=zero     O="OH")
60 DEFINT A-Z:' make all numeric variables integers - pgms run faster
70 INPUT "RANDOM  FILE NAME (EX. RDATA)... ";T$
80 ' note record length of 85 defined with ',85' at end of open statement
90 ' in line 130. You can do this with record lengths up to 128 characters'
100 ' However, for OVER 128 bytes, you MUST ALSO declare record length at'
110 ' the time you load MBASIC. This is easily done using a submit file to'
120 ' load MBASIC and do the necessary declaration! See text.
130 OPEN "R",1,T$,85
140 FIELD 1,85 AS R$
150 K=LOF(1):IF K=0 THEN 420:' if LOF(1)=0 there is NO file by that name!
160 GOSUB 590:K=K+1
170 '
180 '
190 PRINT CHR$(27)+"E"
200 PRINT:PRINT "If done, type DONE & hit RETURN key":PRINT
210 LINE INPUT "'Salutation (Mr Mrs Ms)'.... ";A$
220 IF A$="DONE" OR A$="done" OR A$="Done" THEN CLOSE:END
230 LINE INPUT "First Name & Middle Initial.... ";B$
240 LINE INPUT "Last Name.... ";C$
250 PRINT:PRINT "You may type 1 or 2 lines of address, PLUS City, State, Zip"
260 PRINT:LINE INPUT "1st line of address... ";D$
270 LINE INPUT "Optional 2nd line of address.... ";E$
280 LINE INPUT "Town or City.... ";F$
290 LINE INPUT "State (2 letter abbreviation)....) ";G$
300 IF LEN(G$)<>2 THEN PRINT:PRINT "INVALID ENTRY!":PRINT:GOTO 290
310 LINE INPUT "Zip Code (5 digits)..... ";H$
320 IF LEN(H$)<>5 THEN PRINT:PRINT "INVALID ENTRY!":PRINT:GOTO 310
330 LINE INPUT "Opt phone number, OR just RETURN.. ";P$
340 X$=C$+"\"+B$+"\"+A$+"\"+D$+"\"+E$+"\"+F$+G$+H$
350 ' next line tacks on a "binary zero" (null) to mark end of record - see text
360 IF LEN(P$)=0 THEN X$=X$+CHR$(0) ELSE X$=X$+"\"+P$+CHR$(0)
370 IF LEN(X$)>85 THEN GOSUB 510:GOTO 190
380 LSET R$=X$
390 PUT #1,K:K=K+1
400 GOTO 190
410 ' Routine if NO file found with entered name
420 PRINT:PRINT TAB(10)"THERE IS NO FILE BY THAT NAME!"
430 PRINT TAB(10)"IF YOU DESIRE TO START A NEW FILE,"
440 PRINT TAB(10)"TYPE   YES   AND <cr>. TO ENTER A"
450 PRINT TAB(10)"DIFFERENT FILE NAME, JUST <cr>... ";
460 INPUT S$
470 IF LEN(S$)=0 THEN CLOSE:KILL T$:PRINT:GOTO 70
480 IF LEFT$(S$,1)="Y" OR LEFT$(S$,1)="y" THEN K=1:GOTO 190
490 GOTO 420
500 ' Routine to check length of whole record including "tacked on null"
510 PRINT:PRINT TAB(10)"ENTRY TOO LONG BY ";LEN(X$)-85;" CHARACTERS!!"
520 PRINT CHR$(7):PRINT TAB(10)"PLEASE ABBREVIATE AND ENTER AGAIN...":PRINT
530 LINE INPUT "         HIT RETURN TO CONTINUE.... ";Z9$
540 Z9$="":PRINT
550 RETURN
560 ' Following routine is binary search for EOF, looking for first "null"
570 '  in the FIRST position of a record - this is one MORE than last actual
580 '  preexisting record. See text.
```

```
590 N=25000:I=1:J=N:PRINT:PRINT "Checking file, please wait a second...":PRINT
600 K = (I + J)/2:GET #1,K
610 IF ASC(R$)<>0 THEN I = K ELSE J = K - 1
620 IF I = J THEN GOSUB 640:RETURN
630 GOTO 600
640 IF ASC(R$)=0 THEN K = K - 1
650 RETURN
```

MBASIC's RENUM statement-command, if you do not already know how to use it. It is adequately explained in the manual. The point here is that by using RENUM, you can renumber only lines 340 through 650, changing 340 to 1000 for example, and incrementing lines from 350 through 650 by 10 ABOVE 1000. And, all the GOTO's etc are correctly changed for you. Then you would have plenty of available space for the line numbers between 330 and 1000 that could contain the edit routine. Note that when you type in the edit routine, all line numbers associated with any GOTO's or GOSUB's should be changed to the applicable line numbers. When done simply type RENUM and you now have a nice neat program that is numbered in increments of 10!

BIN85.BAS -

This is the last program in this series. The purpose of this program is to "fast find" any record in your mailing list random file when you know the last name, but do NOT know the record number. This program also presents all names having the SAME spelling as your desired last name to you for your inspection. For example, if there are 5 folks in the file named JONES, then all 5 will be displayed, along with their absolute record numbers. Note that the file MUST be correctly ordered (alphabetized or sorted) before you run this program AND the file must be "packed" (there can be NO blank records and there can be NO "deleted" records - see below). This program actually does 2 binary searches. The first one is used to find the highest numbered legal record in the file and this was discussed above under MAKERAN.BAS. Then the other binary search is done to find the "match" of the desired last name with any given record. Again, it is fundamentally a divide by 2 operation, resetting the top and/or the bottom record after each trial "match" is carried out. Line 240 does the "matching", and when there IS a match the program branches to line 300. Lines 300 to 330 go back record by record to find any similar matches. When one is NOT found, then we go to line 340 which increments the record number in N by 1

```
10 REM    BIN85.BAS       binary search routine - finds record using last name
20 '     A CP/M MBASIC program.              (0=zero    O='OH')
30 '   note that file must be ordered and packed; assumed ALL CAPS
40 '   fielded as 85, record length of 85
50 '   Will find ALL records with same last name. Note name MUST be all CAPS
60 '
70 DEFINT A-Z
80 PRINT "Complete name of random, ordered file to be searched (ALL CAPS) ? ";
90 LINE INPUT P$
100 IF LEN(P$)=0 THEN 80
110 PRINT:PRINT "Type desired last name in the SORTED random file. USE ALL CAPS"
120 PRINT "If done type 0 (zero) and return"
130 LINE INPUT "Enter now..... ";X$
140 IF X$="0" THEN 410
150 IF LEN(X$)=0 THEN 110
160 X$=X$+"\"
170 OPEN "R",1,P$,85
180 FIELD 1,85 AS R$
190 GOSUB 430:H=K:' gosub to get highest legal record & put its position in H
200 L=1
210 N=INT((L+H)/2)
220 IF N=0 THEN 290
230 GOSUB 390
240 IF INSTR(1,LEFT$(Y$,LEN(X$)),X$)<>0 THEN 300:' IF <> 0 then we have a match!
250 IF L>=H THEN 290
260 IF Y$>X$ THEN 280
270 L=N+1:GOTO 210
280 H=N-1:GOTO 210
290 PRINT:PRINT MID$(X$,1,LEN(X$)-1);" is not in file":CLOSE:GOTO 110
300 N=N-1:' now go back, record by record looking for more possible matches
310 IF N=0 THEN 340
320 GOSUB 390
330 IF INSTR(1,LEFT$(Y$,LEN(X$)),X$)<> 0 THEN 300
340 N=N+1
350 GOSUB 390
360 ' next line prints first match, loops to 340, goes forward in file for more?
370 IF INSTR(1,LEFT$(Y$,LEN(X$)),X$)<>0 THEN 380 ELSE CLOSE:GOTO 110
380 PRINT:PRINT "Record #";N;"is ";Y$:GOTO 340
390 'following line puts contents of record N in fielded buffer, then into Y$
400 GET 1,N:Y$=R$:RETURN
410 PRINT:PRINT "Bye-bye!":END
420 ' following is binary search routine to find EOF
430 N=25000:I=1:J=N:PRINT:PRINT "Checking file, please wait a second...":PRINT
440 K = (I + J)/2:GET #1,K
450 IF ASC(R$)<>0 THEN I = K ELSE J = K - 1
460 IF I = J THEN GOSUB 480:RETURN
470 GOTO 440
480 IF ASC(R$)=0 THEN K=K-1
490 RETURN
```

and prints it out on the screen along with its contents for our inspection, goes forward by 1 in file for any more "positive" matches, else closes the file and branches to line 110 to see if we desire to look for any other last names in the file. Note that if you desire hard copy printout, just change PRINT's in line 380 to LPRINT's.

FINAL REMARKS -

There are a couple of programs that you SHOULD now write to complete this series of random programs, if you desire to make them into a workable mailing list set of programs. One is an EDIT program, and another is a DELETE program. Hopefully, you will be able to do so without much difficulty. The EDIT program is constructed along the same lines as many of the above programs. You OPEN, FIELD, GET the desired record and its contents into R$, transfer to X$, remove the null marker and any trailing blanks. Then, using the same procedures as in the LABEL.BAS program you dissect X$ by finding the position of the delimiters with the INSTR function, and then place the contents of the various fields into string variables of your choice. Then display the various fields on the screen as we did in MAKESEQ.BAS, allowing for user input to get any field and alter it, and then redisplay all until the record is correctly edited. Then reassemble the various individual fields and put them back into X$ along with the delimiters, and our null marker; LSET R$=X$ and the record is replaced back into the random file.

The delete program is most easily carried out by making it a part of the EDIT program which we have just discussed. The method I use is to simply access the record I want to delete as discussed above, and then when it is "in X$", I simply use the LEFT$ and MID$ statements to replace the first character of the record in X$ with an asterisk (*). What this does of course is to replace the first character of the last name with an asterisk. Then, if you desired, you could add one line to any program where you didn't want to see this "functionally" deleted record. That would be an IF statement to the effect that IF the first character of any record was an asterisk, then to branch to "wherever" so that it would not be used by the program in question. The program called RTOS.BAS which creates a sequential disk file from a random disk file has this line, and it prevents any record whose first character is an asterisk from being written to disk. It is line 110 in that program. This results in the new file written to disk having none of the "deleted" records. They are now "physically" deleted. This process is called "packing the file".

You will also want to convert the LABEL.BAS program, which was discussed in the last article dealing with sequential files, to a random file. Again, this should not prove too difficult for you at this point.

The last program I suggest you write to complete the "package" is MENU.BAS. This program would simply display the various options available and ask user to select any desired program option by unique number or unique letter. Use Q$=INPUT$(1) or a LINE INPUT "prompt and statement" to capture the unique character or number which user enters and put it in Q$. Then have a series of IF statements to run any selected program. IF Q$="2" THEN RUN "BIN85 would run the "last name BIN85" fast search program, assuming user typed 2. In ALL of the other programs change END to RUN "MENU. The last option displayed in the menu should return user to the CP/M monitor prompt if user types its unique number or character. Assuming that your menu displayed "8 - whatever; 9 - Return to CP/M" then you would need an IF statement in your program like this -> IF Q$="9" THEN SYSTEM. Then, if you used a line input statement -> LINE INPUT "Your choice is number..... ";Q$, and user typed 9<cr>, user would be returned to CP/M system. (All of these statements, of course, must have line numbers before them.)

I wish you good luck and enjoyment with CP/M, MBASIC and random files.

EOF

Current Local HUG Clubs

(NOTE: This listing is of July 1, 1982. If your club is not listed or you are forming a new club and you would like to have it included in our list please send the proper information to: Heath Users Group, attn: Nancy Strunk, Hilltop road, St. Joseph, MI 49085)

AK, Eagle River
Alaska HUG
P.O. Box 951
Eagle River, AK 99577
907-694-9908 Group Size 20
Contact Person: Ben Sevier

AZ, Phoenix

2727 W Indian School Rd
Phoenix, AZ 85017
602 991-2515 Group Size
Contact Person: John A Grosberg, President
2nd Tues at 7:00 p.m. at HEC

AZ, Tucson
Tucson HUG
7109 E Broadway
Tucson, AZ 85710
602 885-6773 Group Size 25
Contact Person: Chuck Lucking
Meet first Thursday each month 7:00 p.m.
Meetings held at Heathkit Tucson

CA, Anaheim
ANAHUG (Anaheim HUG)
330 E. Ball Road
Anaheim, CA 92805
714-776-9420 Group Size 120
Contact Person: John Belsher, President
3rd Thursday 7:00 PM at HEC

CA, El Cerrito
ECHUG (El Cerrito HUG)
6000 Potrero Avenue
El Cerrito, CA 94530
415-236-8870
Contact Person: Alan Biocca
4th Wednesday at HEC

CA, El Monte
ETUG (ET/ETA 3400 Users' Group)
11231 Oak Street
El Monte, CA 91731
Group Size 100
Contact Person: Charles Van Dyke
Newsletter 4 times year

CA, Fresno
FresHUG (Fresno HUG)
4833 East Santa Ana
Fresno, CA 93726
209-291-6258 Group Size 4
Contact Person: Harlen Collins

CA, Glendora
Southern CA H11 Users Group
430 W. Highland Avenue
Redlands, CA 92373
714-886-4766 Group Size 40
Contact Person: Dr. M.J. Di Girolamo
Meets at 625 E. Palm, Glendora, CA

CA, Los Angeles
Los Angeles HUG
24025 Fernlake Drive
Harbor City, CA 90710
213-539-4276 Group Size 20
Contact Person: Dean Gibson c/o Ultimeth Co.
1st Thursday 7:00 PM at HEC

CA, Pomona
Pomona HUG
1555 N Orange Grove Ave
Pomona, CA 91767
714 985-5303 Group Size
Contact Person: Herb Friedman
Meet 4th Thursday each month
at 7:30p.m. at HEC

CA, Redwood City
BAHUG Bay Area HUG
2001 Middlefield Road
Redwood City, CA 94063
415-365-4915 Group Size 219
Contact Person: Bob Bance, Sec.
2nd Tuesday 7:00 PM at HEC

CA, Riverside
Tri-HUG
5705 Via Sotelo
Riverside, CA 92506
714-683-2929 Group Size 20
Contact Person: Kenny Adcock

CA, Sacramento
SHUG (Sacramento HUG)
1860 Fulton Avenue
Sacramento, CA 95825
91 662-7220 Group Size 35
Contact Person: Gloria Stewart, Sec.
Meet 2nd Wed 7:30pm at Sacramento HEC

CA, San Diego
San Diego HUG
12202 Kingford Court
El Cajon, CA 92021
714-561-2540 Group Size 170
Contact Person: Richard Cobb
1st Wednesday 7:00 PM
at Parkway Jr HS La Mesa

CA, Woodland Hills
LUVAHUG
22504 Ventura Blvd.
Woodland Hills, CA 91364
213-883-0531 Group Size 40
Contact Person: Paul S. Townsend
2nd Thursday 7:00 PM at HEC

CO, Colorado Springs
CSHUG (Colorado Springs HUG)
Colorado Springs, CO 80906
303 632-3019 Group Size 25
Contact Person: Richard Evers
Meet last Thurs each month 7:00 pm
Have 24hr BB (303) 634-1158

CO, Denver
DENHUG (Denver HUG)
P.O. Box 20422
Denver, CO 80220
303-394-2082 Group Size 120
Contact Person: Alfred K. Carr, Sec./Tres.
BB 303-423-3224 (24 hrs)
2nd Monday 7:00 PM at HEC

CO, Ft. Collins
FT.HUG (Fort HUG)
822 E. County Road 30
Ft. Collins, CO 80525
303 669-4116 Group Size
Contact Person: Ted Benglen, II
Meet once a month at present

CT, Avon
CONHUG (Connecticut HUG)
8 Huckleberry Lane
W. Simsbury, CT 06092
203-658-2944 Group Size 35
Contact Person: Tom Carbone
1st Wednesday at HEC
H11 Special Interest Group

FL, Fort Myers

P.O. Box 05-37
Tice, FL 33905
Contact Person: Robert Sloat
Just getting started

FL, Miami
Miami Amateur Computer Club
4705 W. 16th Avenue
Hialeah, FL 33012
305-823-2280 Group Size 35
Contact Person: Ralph Boyd
At HEC

FL, Orlando
HUG of Central FL Computer Sc.
135 Stonyridge Drive
Longwood, FL 32750
305-339-8853 Group Size 34
Contact Person: Jim Donlon, President
4th Wednesday at various locations

GA, Atlanta
ATHUG (Atlanta HUG)
5285 Roswell Road
Atlanta, GA 30342
404 436-3677 Group Size 50
Contact Person: Leon Trulove
Meet first & third Thurs 7:00-9:00 p.m.
BB 404 252-4342 6:00pm to 8:00am

Ga, Augusta
CSRA Computer Club
PO Box 284
Augusta, GA 30903
404 860-2934 Group Size
Contact Person: Paul Pennington
Meet 4th Monday at 7:30pm
at Campus Com puter Sys
3830 Washington Rd Martinez,GA 30907

HI, Honolulu
HUGH (HUG Hawaii)
1255 Nuuanu Avenue # 1405
Hononlulu, HI 96817
808-531-8843 Group Size 45
Contact Person: Jim Branchaud, President
3rd Saturday at Mililani,
1st Wednesday at Kalihi

IL, Champaign
CCCC (Champaign Cty Comp Club)
412 Dorchester
Mahomet, IL 61835
312-586-5100 Group Size 12
Contact Person: Roger Fraumann

IL, Downers Grove
I-HUG (Illinois HUG)
6116 Lane
Downers Grove, IL 60516
312-971-1660 Group Size 25 Contact Person:
Len Bateman
3rd Wednesday at various locations

IL, Downers Grove
HUG Metro (Local Chicago)
15 W. 780 Fillmore
Elmhurst, IL 60126
312 985-2381 Group Size 30
Contact Person: Larry Shipinski, President
Meet 2nd Monday of each Month
7:30pm at HEC

IL, Peoria
CIHUG (Central Illinois HUG)
408 Bess Street
Washington, IL 61571
309-745-8313 Group Size 17
Contact Person: Ronald Morgan, President
3rd Sunday at 3 PM (Jan, Mar, May, Jul, etc.)

IL, Rockford
Blackhawk Bit Burners
325 Beacon Drive
Belvidere, IL 61008
815-544-5206 Group Size 35
Contact Person: Frank D. Dougherty

IN, Indianapolis
Indianapolis HUG (IHUG)
3390 Peppermill Drive #2C
West Lafayette, IN 47906
317-257-4321 Group Size 60
Contact Person: Robert Wild, President
2nd Wednesday 7:15 PM at HEC

KS, Wichita
Wichita HUG
1909 Siefkin
Wichita, KS 67208
316-681-3456 Group Size 18
Contact Person: David Horwitz
2nd Sunday of ODD months
2:00 PM at E. Pike Bldg.
Corner of Webb and Kellog in Wichita

LA, New Orleans
NOHUG
1900 Veterans Blvd.
Kenner, LA 70062
504-467-6321 Group Size 60
Contact Person: Nathan Gifford
1st Wednesday at 7:30 PM at HEC

MA, Northampton
Hampshire Computer Club
Box 685
Northampton, MA 01061
617-584-7159 Group Size 100
Contact Person: Ed Judge, Secretary
2nd Tuesday 7 PM
at McConnel Hall Smith College
Beginners Group 1st Tuesday

MA, Peabody
HUG North Shore
6 Susan Drive
Saugus, MA 01906
617-233-2941 Group Size 60
Contact Person: Hal Messinger, President
BB 617-531-9332 24 hours
2nd Wednesday Hilltech Bldg Danvers

MA, Pittsfield
BERCHUG (Berkshire County HUG)
73 Waverly Street
Pittsfield, MA 01201
Contact Person: Paul E. Ouellette, President

MA, Wellesley
HUG'EM
165 Worcester Ave
Wellesley, MA 02181
617 237-1510 Group Size 100
Contact Person: Malcolm Partridge, Director
3rd Wed 7:00 p.m. at HEC

MD, Baltimore
Baltimore HUG
6106 Marlora Road
Baltimore, MD 21239
301-323-6093 Group Size 70
Contact Person: William Frey
3rd Thursday 7:00 PM at HEC

MI, Detroit
Metro Detroit Area HUG
7716 Winona
Allen Park, MI 48101
313-928-7423 Group Size 50
Contact Person: Chuck Dattolo

MI, Kalamazoo
SMHUG (Southwest Michigan HUG)
623 Wildwood Place
Kalamazoo, MI 49008
616-349-3535 Group Size 50
Contact Person: Al Jacobs,
Secretary/Treasurer
4th Saturday 1 PM at Western Michigan
University
Moore Hall, Rm 1034, News Letter

MI, Saint Joseph
BLHUG (Blossomland HUG)
P.O. Box 414
Saint Joseph, MI 49085
Group Size 33
Contact Person: Vance Fisher, Chair Person
1st Tuesday 7:00 PM at various locations
Check HEC for place of meeting.

MN, St. Paul-Minneapolis
SMUGH
8895 72nd Street
Cottage Grove, MN 55016
612-459-4382 Group Size 100+
Contact Person: Steve Howard, President
Last Monday at 7:00 PM (Alt. St Paul & Mpls)

MO, St. Louis
SLHUG (St. Louis HUG)
3794 McKelvey Road
Bridgeton, MO 63044
314-291-1850 Group Size 120
Contact Person: Mike Davis, President
BB 314-291-1854 after hours ONLY
2nd Wednesday 7:30 PM at HEC

NC, Charlotte
HUG Charlotte
2721 Picardy Place
Charlotte, NC 28209
704-374-6997
Contact Person: Jim Simpson

NC, Fayetteville
Cape Fear Computer & HUG
2454 Vandemere Avenue
Fayetteville, NC 28304
919-485-4586 Group Size 11
Contact Person: Jerry Mills, President
Bi-Weekly 2:00 PM on Sundays at homes.

NC, Hillsborough
HUG-RTP
Rt 3, Box 39A
Hillsborough, NC 27278
919 73-6678 Group Size
Contact Person: Joe Williams
Meeting place and time unknown

NE, Omaha
OMAHUG (Omaha HUG)
9207 Maple Street
Omaha, NE 68134
402-391-2071 Group Size 200
Contact Person: Chuck Juvenal, Chairman
3rd Sunday 6:30 PM at HEC

NJ, Fairlawn
HUGNJ (HUG of New Jersey)
3507 Broadway
Fairlawn, NJ 07410
201-791-6938 Group Size 85
Contact Person: Mel Beiman
BB 201-791-3015 24 hours
3rd Monday 8 PM at HEC

NJ, Ocean
SHUG (South Jersery HUG)
1013 State Hwy 35
Ocean, NJ 07712
201 775-1231 Group Size 71
Contact Person: James J Jones Jr. (Sec)
Meet 1st Wed 7:30 p.m. at Ocean HEC
BB 201 775-8705 24hrs.

NY, Buffalo
BUG (Buffalo Users Group)
19 Mayfield Lane
Tonawanda, NY 14150
716-662-7122 Group Size 70+
Contact Person: Dick Rose, Newsletter Editor
3rd Sunday 1 PM at HEC

NY, Long Island
Jeri-HUG (Jericho HUG)
15 Jericho Turnpike
Long Island, NY 11753
513-334-8181 Group Size 80
Contact Person: Bob Lippman
2nd Thursday 7:30 PM at HEC

NY, Rochester
RHUG (Rochester HUG)
937 Jefferson Road
Rochester, NY 14623
716-773-0193
Contact Person: Joanne Lang, Chairperson
Last Tuesday at 7:00 PM at HEC

OH, Cincinnati
Cincinnati HUG
10133 Springfield Pike
Woodlawn, OH 45215
513-771-8850 Group Size 50
Contact Person: Roger Svoboda
2nd Tuesday 6:30 PM at HEC,
$10.00 Dues/year
Newsletter I/O Port

OH, Cleveland
NOHUG (Northeastern Ohio HUG)
4705 Tanglewood Place
Lorain, OH 44053
Group Size 40
Contact Person: Art Petkosek
2nd & 4th Thursday
7:00 PM at Maple Hts. Library

OH, Cleveland
Cleveland HUG
28100 Chagrin Blvd
Cleveland, OH 44122
216 291-1612 Group Size 10
Contact Person: Gerry Ciganko
First Thurs 7:00 p.m. at HEC
BB 216 292-7553 24 hours

OH, Columbus
Columbus HUG
2500 Morse Road
Columbus, OH 43229
614-475-7200 Group Size 50
At HEC

OH, Dayton
Wright-Patterson HUG
4110 Spruce Pine Court
Dayton, OH 45424
513-236-4915 Group Size 36
Contact Person: Jim Moore, President
1st Thursday 4 PM at Wright-Patterson AFB

OH, Toledo
THUG (Toledo HUG)
4804 Mt. Airy Road
Sylvania, OH 43560
419-882-3626 Group Size 30
Contact Person: John F. Priebe, President
Last Sunday 8 PM

OK, Oklahoma City
OKC TUGS
2727 NW Expressway
Oklahoma City, OK 73112
405-848-7593 Group Size 40
Contact Person: Bob Perry
2nd Sunday at 1:00 PM at HEC
BBS 405-848-9329 24 hours

OKINAWA
OKIHUG Okinawa Heath Users'Gp
Box 376, USAFSO
APO San Francisco, CA 96331
Group Size 13
Contact Person: Carl H. Eaton
Meet on Fridays monthly at 7:30 pm
Meeting place varies

PA, Frazer
FUG (Frazer Users Group)
1641 Princess Anne Drive
Lancaster, PA 17601
717-397-3146 Group Size 60
Contact Person: Dave Hendrie, President
1st Sunday 4 PM at HEC

PA, Philadelphia
Philadelphia Heath Users'Group
6318 Roosevelt Blvd
Philadelphia, PA 19149
215 288-0180 Group Size 135
Contact Person: Henry F. Beechhold, President
Meet 2nd Wed. each month
7:00 p.m. at HEC 8

PA, Pittsburgh
PittsburgHUG
3482 William Penn Highway
Pittsburgh, PA 15235
412-824-3564 Group Size 35
Contact Person: John C. Schultz, President
3rd Thursday 7:00 PM at HEC

RI, Warwick
HUG-RI' (HUG of Rhode Island)
558 Greenwich Avenue
Warwick, RI 02886
401-738-5152 Group Size 150
Contact Person: Walt Phaneaf
2nd Wednesday 8 PM at HEC

TN, Memphis
Memphis HUG
6874 Kirby Brooks Drive
Memphis, TN 38115
901-362-8860 Group Size 4
Contact Person: Morris Proctor
Meets at National Cotton Council

TX, Dallas
DFW HUG (Dallas-Fort Worth)
2715 Ross Avenue
Dallas, TX 75201
214-826-4053 Group Size 70
Contact Person: Henry Gardiner, President
1st Thursday and 15 days later (Wed.) at 7:30 PM
At HEC BB 214-742-1380

TX, Ft. Worth
FWHUG
6825A Green Oakes Road
Ft. Worth, TX 76116
817 737-8822 Group Size 26
Contact Person:
Meets 2nd and 4th Tues

TX, Houston
HUG-H
7798 Braniff
Houston, TX 77061
713-644-5689 Group Size 75
Contact Person: Tom McCormick, President

TX, San Antonio
San Antonio (SAHUG)
7111 Blanco Road
San Antonio, TX 78216
512-341-8876 Group Size 65
Contact Person: Tom Schneider
First Tuesday at HEC, 7:30 PM

TX, Wichita Falls
Nortex HUG (North Texas HUG)
4510 Allendale Road
Wichita Falls, TX 76310
817-692-1241
Contact Person: Alan D. Martin

UT, Midvale
UHUG (Utah HUG)
58 E. 7200 South
Midvale, UT 84047
801-566-4628 Group Size 75
Contact Person: Don Greene, President
2nd Wednesday 7:00 PM at HEC

VA, Fairfax
CHUG (Capital HUG)
P.O. Box 2653
Fairfax, VA 22031
703-591-7861 Group Size 500 +
Contact Person: Mike Frieders, President
3rd Monday 7:30 PM at Fairfax High School
Large software library (135 + disks)

VA, Richmond
Richmond HUG
1724 Blakemore Road
Richmond, VA 23225
804-232-2925 Group Size 8
Contact Person: Jim Scott
2nd Monday at various locations

VA, Virginia Beach
THUG (Tidewater HUG)
1055 Independence Blvd.
Virginia Beach, VA 23455
804-460-0997 Group Size 90
Contact Person: John E. Smith, President
1st & 3rd Tuesday at 7:00 PM at HEC

WA, Bellevue
Pacific Northwest HUG
C/o Jan Johnson PO Box 993
Bellevue, WA 98009
206 363-3927 Group Size 150
Contact Person: Nathan Hall
Meet 2nd Thurs odd months 6:00 Tukwila HEC
Meet Ist Mon even months Seattle HEC

WA, Spokane
SPOHUG (Spokane HUG)
RFD 1 Box 676
Spokane, WA 99204
509-448-9727 Group Size 18
Contact Person: Charles Ballinger
Newsletter

WA, Vancouver
Portland-Vancouver HUG
516 SE Chkalov Drive
Vancouver, WA 98663
206-254-44
Group Size 70
Contact Person: Evert Jan Stokking

PANAMA CANAL
Canal HUG
P.O. Box 1112
APO Miami, FL 34001
84-4094 Group Size 6
Contact Person: Michael Gulick, President
1st Tuesday 7:30 PM at Howard Air Force Base

PUERTO RICO, Rosario
PRHUG (Puerto Rico HUG)
P.O. Box 765
Rosario, PR 00746
809-892-4677 Group Size 5
Contact Person: Norberto Collado Rivera

W. GERMANY, Frankfurt
Frankfurt HUG
American Consulate General FRDCO
APO NY, NY 09757
566187 Group Size 3
Contact Person: Carl Lovett

W. GERMANY, Sprendlingen
HUG-Deutschland
Robert-Bosch-Strasse 32-38
D-6072 Dreieich W. GERMANY
06103/3808 Group Size 200
Contact Person: Egon Becker/Lydia Luguet

Vectored from page 8

A: Both hard-sectored and soft-sectored disk controller cards must be installed in your computer. With appropriate disk drives and the proper BIOS for this combination, transferring files between the two is as easy as the transfer of files between two hard-sectored disk drives. If your hard-sectored disk controller is supporting two drives and you BOOTed from the hard-sectored drive, the hard-sectored drives are called A:, B:, and C:. The soft-sectored drives will then be D:, E:, and F:. An example of a file transfer from A: to E: would be:

A>PIP E:=A:FILENAME.EXT<CR>

This example assumes that the PIP file is on the disk in drive A. For additional information on drive name determination, refer to Appendix F in the Heath/Zenith CP/M manual.

Q: How many Software Consultants does it take to change a light bulb?

A: None! It's obviously a hardware problem.....

Vectored from page 8

Dear HUG:

Here's a note for H88 operators who wish to use the MX-80 printer. One method of interfacing the two is shown with enough explanation to allow other alternatives to be chosen.

The equipment here is the MX-80 printer with the 8141 RS232C Serial Interface card, the H88 with H88-3 Serial Interface and cassette operating system release number XX.06.00, and an RS232C cable modified by Heath to connect pin 20 of the cable connector at the printer end to pin 4 at the computer-end connector. For 9600 baud operation the switches are set as follows. Before inserting the 8141 Serial Interface card, a control circuit board with two switches is exposed. Set SW1 at 1 on, 2 on, 3 on, 4 off, 5 on, 6 on, 7 off, 8 on, and SW2 at 1 on, 2 on, 3 off, 4 off. On the 8141 board set the switch at 1 on, 2 on 3 off, 4 off, 5 on, 6 off, 7 off, 8 n/a.

The main problem is now to insure that the printer is ready to receive a character before the H88 outputs one to the printer. A NOT-RDY signal from the printer enters the printer's serial interface board. With a jumper at JNOR (factory installed) this signal is inverted once as it is changed from TLL to RS232C voltage levels and is routed to pin 20 on the DB25 connector. It is now the DTR signal. In the Heath-modified cable the signal is routed to pin 4 of the connector at the computer. It is now the CTS (clear to send) signal and is routed to pin 7 of P603 on the H88-3 serial interface card. On the card the signal is inverted and presented to the 8250 IC as a NOT-CTS. The 8250 expects an inverted signal here to provide a positive CTS signal at bit 4 of the MODEM Status Register (port 346).

However, the console driver of the cassette operating system expects an

inverted CTS at this register. Therefore the logic of the search loop looking for an inverted or low CTS signal must be changed. The change is:

LOCATION	OLD	NEW
042000	302(JNZ)	312(JZ)

With this change all the system tapes will transmit controls or characters with ASCII values of 0 to 127. Instructions are in the cassette system manual. One point, to LIST in BASIC, use the following commands:

*PORT OUT 224,9600:LIST

This is necessary as on any return to the command mode, the video terminal is set as the output terminal.

If the cable is not modified, then the DTR signal arrived at pin 20 of the connector on the H88. It is routed to pin 14 of P603 on the H88-3. It is inverted, presented to the 8250 a NOT-DSR (not data set ready) signal, and produces a DSR signal in bit 5 of the MODEM Status Register. Thus if an unmodified cable is used, the mask used for testing must be changed as well as the logic of the search loop, or:

LOCATION	OLD	NEW
041377	020(4th bit mask)	040(5th bit mask)
042000	302(JNZ)	312(JZ)

Fine. But what about the MX-80 graphics? This requires transmitting to the MX-80 ASCII characters from 128 to 255. There are two masks built into BASIC to prevent this. To remove these for graphics, make the following patches and copy before entering the program with GO:

LOCATION	OLD	NEW
057303	346(ANI)	000(NOP)
057304	177(MSB mask)	000(NOP)
111031	346(ANI)	000(NOP)
111032	177(MSB mask)	000(NOP)

Now the cassette operating system works with no hardware patches. Also assembly language programs are straight forward. Following is a printer initialization routine and a print character routine for the system with the modified cable.

An alternative to patching the program (JNZ to JZ) is the hardware change of installing a jumper in JREV on the

```
**************************************************
*
* COLLECTION OF SUBROUTINES
* TAB,1,8,14,25,53
*
**************************************************


**************************************************
*
*MX-80 LINE PRINTER INITIALIZATION
*
**************************************************
LPINT   XRA     A           SET A=0
        OUT     341Q        CLEAR IER
        MVI     A,020Q      SET LOOPBACK MODE
        OUT     344Q
        MVI     A,200Q      SET DLAB IN LCR
        OUT     343Q
        MVI     A,014Q      LSB OF BAUD RATE DIVISOR
        OUT     340Q
        MVI     A,000Q      MSB OF BAUD RATE DIVISOR
        OUT     341Q
        MVI     A,0030Q     RESET DLAB; SET WL=8;
        OUT     343Q        1SB; NO PARITY
        IN      340Q        CLEAR JUNK
        LXI     H,65000A    GET DELAY VALUE
DELAY2  DCR     L           THIS TAKES ABOUT
        JNZ     DELAY2      100 MS
        DCR     H
        JNZ     DELAY2
        IN      340Q        CLEAR JUNK
        MVI     A,0         CLEAR LOOPBACK
        OUT     344Q
        MVI     A,127       DEL(CLR PRINTER BUFFER)
        CALL    PRCHR       SEND IT
        RET                 RETURN TO MAIN PROGRAM

**************************************************
*
*SUBROUTINE TO PRINT A CHARACTER
*CHARACTER IN A REGISTER
*
**************************************************
PRCHR   PUSH    PSW         SAVE CHARACTER
THRE    IN      345Q        THRE
        ANI     040Q        MASK THRE BIT
        JZ      THRE
CTS     IN      346Q        CLEAR TO SEND
        ANI     020Q        MASK CTS BIT
        JZ      CTS         LOOP TIL READY
        POP     PSW         GET CHARACTER BACK
        OUT     340Q        TO PRINTER BUFFER
        RET                 RETURN TO MAIN PROGRAM
```

8141 Serial Interface board and cutting the jumper at JNOR. This inverts the CTS.DSR signal and eliminates the need for the patch. However, the logic will be inverted for assembly language programs or for any others not taking this peculiarity into account. Also other patches will probably be made (e.g. for graphics of CTS/DSR routing).

I hope the above explanation is also useful to H89 owners.

Frederick M. Galloway
6507 Blue Wing Drive
Alexandria, VA 22307

Dear HUG,

I am a high school computer teacher. We have four Apple II+ computers that were purchased soon after I had my Heath system. I prefer the Heath over the Apple, but the Apple has some nice features that I wished the Heath software was capable of doing - namely, the ability to use the cursor keys to backspace over text and then retype the end of the line using the cursor right key. The above feature really helps out if you make a one letter error at the beginning of the line. One uses the cursor left key to back up to the letter that is incorrect, types the correct letter, and then uses the cursor right key to return to where you left off. An additional feature is the blanking of the rest of the line if you don't continue to the end of what you already typed. Thus, you see exactly what was stored in the memory.

I have come up with a patch to TED-8 #03.06.00 cassette software to do the following:

1. The backspace key now erases the character it backspaces over.
2. The cursor left key will back up the cursor without erasing the characters in the line.
3. The cursor right key will 'retype' the characters in the line up to the last character originally typed.
4. When the cursor is in the middle of a line and RETURN is hit, the rest of the line from the cursor to the end is blanked out.

I should note that with this patch one loses the ability to exit the INSERT with the ESC key. However, CTRL-C will still exit from INSERT.

The distribution copy of TED-8 should be loaded into the computer and the following changes made (summarized from assembled patch comments):

ADDRESS	ENTRY
047136	132
052146	132
054003	132
055133	315 021 067
055137	000
055152	271 066
055322	303 372 066
055325	000 000 000
061240	315 041 067
067131	000

This modifies TED-8 so the following patch residing at 066271 to 067056 will do the four new functions outlined above. Now enter the patch 066271 to 067056 and configure TED-8 as usual.

Richard W. Washburn
Box 3035 USNS
FPO Miami, FL 34051

```
PATCH   ORG     066271A

*** Assembly Constants for TED-8 03.06.00
**
*
ALT2    EQU     055137A
BKSP    EQU     010Q
BELL    EQU     007Q
CCL     EQU     055372A
COLNO   EQU     040226A
ESC     EQU     033Q
LINE    EQU     065075A
LINPTR  EQU     065064A
$CRLF   EQU     056103A
$RCHAR  EQU     040106A
$TYPC.  EQU     060134A
$TYPCH  EQU     060130A
$WCHAR  EQU     040111A

*** This gives TED-8 a retype ability.  The
*   use of the ESC key to exit the INSERT
*   mode is lost.  User must use CTRL-C.
*   The right and left cursor controls
*   are used to move the cursor.  Right
*   retypes and left non-destructively
*   backspaces over the text.

*** The following locations of TED-8 must
*   be changed as follows:
*   ADDRESS   ENTER
*   055133    315 ADDRESS OF PTCH1.2
*   055137    000
*   055152    271 066

*** Because the following patches take up
**  more than the first patch space in
*   TED-8 the start of the text buffer must
*   be moved down and the references to the
*   start changed in TED-8.  The buffer now
*   starts at 067132.
*   ADDRESS   ENTER
*   047136    132
*   052146    132
*   054003    132
*   067131    000

PTCH1.0 CALL $RCHAR
        CPI  'D'
        JE   NDBKSP     NON-DESTRUCTIVE BACKSPACE
        CPI  'C'
        JNE  EXITBAD
RETYP   LHLD LINPTR
```

```
        MVI   A,#LINE+120Q
        CMP   L           SEE IF AT END OF LINE
        JE    EXITBAD     YES RING BELL
        MOV   A,M
        ANA   A           AT END OF ENTERED TEXT
        JZ    EXITBAD     YES RING BELL
        CALL  $TYPC.
        INX   H
        JMP   ALT2
NDBKSP  LHLD  LINPTR
        MVI   A,#LINE
        CMP   L           SEE IF AT FRONT OF LINE
        JE    EXITBAD     YES RING BELL
        DCX   H
        MVI   A,BKSP
        CALL  $WCHAR
        PUSH  H           SAVE LINPTR
        LXI   H,COLNO
        DCR   M           BACK UP COL COUNTER
        POP   H           GET LINPTR
        JMP   ALT2
EXITBAD CALL  $TYPCH
        DB    BELL
        JMP   ALT2

*** The exit of the ATL - ACCEPT TEXT LINE
**  must be modified to allow retyping.
*   Make the following changes:
*   ADDRESS   ENTER
*   055322    303 ADDRESS OF PTCH1.1
*   055325    000 000 000

PTCH1.1 PUSH  H           SAVE LINE
        LHLD  LINPTR
        MVI   M,0         PUT 0 AT END OF LINE
        MVI   A,ESC       BLANK OUT REST OF LINE
        CALL  $WCHAR
        MVI   A,'K'
        CALL  $WCHAR
        POP   H           GET LINE FOR CCL ROUTINE
        CALL  $CRLF
        JMP   CCL

*** This routine zeros the line buffer before
**  the ATL- ACCEPT TEXT LINE routine of TED-8
*   is entered.

PTCH1.2 LXI   H,LINE
        MVI   A,120Q
LOOP    MVI   M,0
        INX   H
        DCR   A
        JNZ   LOOP
        LXI   H,LINE-1
        RET

*** This patch deletes after the backspace key
**  is hit.  The characters in the line buffer
*   are set to null, 0.  The following changes
*   are made to TED-8:
*   ADDRESS   ENTER
*   061240    315 ADDRESS OF PTCH2.0

PTCH2.0 SHLD  LINPTR
        MVI   A,040Q
        CALL  $WCHAR
        MVI   A,BKSP
        CALL  $WCHAR
        RET
        END   PATCH
```

*

Looking for help!

Ben Whitehurst of 4609 Tia Encatada N., Albuguerque, New Mexico 87109 is looking for other H89 users in his area for starting a local users group and also for helping with his questions.

New HUG Products

☞ SPECIAL NOTICE ☜

It has come to our attention that some of our readers have misinterpreted our notice regarding the cancellation of our plans to market the Livingston Logic Labs **BIOS-80** (**REMark** Issue 28, page 16). We would like to make it clear that this product was dropped due to a failure to negotiate a sales contract which was agreeable to both parties involved, and not due to any bug or defect in the **BIOS-80** software. If you are interested in a **BIOS** that is capable of operating the large capacity drives via the hard sectored controller, you may wish to contact Livingston Logic Labs directly.

885-1219

NAVPROGseven CP/M $20.00

NAVPROGseven Aircraft Navigation
& Flight Planning

©Copyright 1982 Alan Bose
President, Taildragger Flyers, Inc.
Ross Field, Benton Harbor, MI

Changes made for CP/M and MBASIC 5.21
©1982 Glen E. Hassebrock, Jr.

"It's here! A CP/M version of Alan Bose's famed NAVPROG7 . . for Heath/Zenith CP/M systems. All the fancy graphics and handy navigational features have been retained; nothing was lost in the translation." Glen E. Hassebrock, Jr.

Requirements: For this version, NAVPROG7 requires CP/M 2.2+, Microsoft BASIC 5.21 on an H8/H19/H17 or H89 with 64K of memory and two disk drives. An 80 character line printer is also required to take full advantage of all the features.

NOTE: This package requires the H19 (H89) terminal graphic features.

Introduction: NAVPROGseven is a database management system designed for pilots flying cross-country. The system is built around a latitude-longitude referenced navigation program designed to prepare a flight log that is ready for use in the cockpit.

The system stores performance data about each aircraft the user flys, navigation data about each checkpoint, airport or navaid the user flys over, and saves this information for easy access on subsequent flights.

Program Content: The details of this product are identical to the HDOS version, refer to the February Issue 25 of REMark. This abstract will contain simply a brief overview.

There are eight programs called by seven user-selectable items on the master menu. The master menu includes the following commands:

Directions & Guidance
Input/Revise Airport & Navaid Data
Input Aircraft Performance Data
Automatic Route Preparation
Air Navigation & Flight Planning
Navigate Pre-planned Route
Sort & List Data on File

NAVPROG7 contains a comprehensive flight log and comes complete with sample aircraft and route data, and the airport/navaid database has over 100 airports, navaids, intersections, and checkpoints already on file.

Comments: The system is menu driven and may be set for "turnkey" operation. The programs are self-prompting with one-key responses and many safety checks that allow the user to go back to the menu and start over.

885-1218

HUG Payroll (CP/M) $60.00

Introduction: The HUG Payroll package can handle 30 employees on a single low density drive while maintaining up to 100 employees on two low density drives such as those used in the H77, Z87 or H-17.

Requirements: The CP/M version of the HUG Payroll Package requires the use of a 64K H/Z89 (or Z90) or a 64K H8 using the H/Z19 terminal. A printer that can be interfaced with your computer is a must. (The H/Z25 is highly recommended since the Payroll Package was developed using this printer.)

This CP/M version was written in Microsoft BASIC version 5.2 or later.

Disk Contents:

PAYNEW	.BAS
PAYCHECK	.BAS
PAYCLOSE	.BAS
PAYMENU	.BAS
PAYDEDUC	.BAS
PAYDELET	.BAS
PAYADD	.BAS
PAYDISPL	.BAS
PAYTABLE	.BAS
PAYDAY	.BAS
PAYCHANG	.BAS
PAYSORT	.BAS
PAYCAL	.BAS
PAYREPOR	.BAS
TABLES	.TAX

Program Content: The CP/M HUG Payroll Package is identical to the operations of the HDOS version. Refer to the July, Issue 30 of REMark for details.

The HUG Payroll contains several powerful features that allow for user friendly operation including a startup routine for entering new data. A Master Menu is used and linked to several sub-menus for ease of operation. Complete instructions are supplied with the package.

Comments: As with any business package, the HUG Payroll will have its limitations. However, the software in this case is very solid. Problem areas, such as changing tax deductions or using the tax table have been simplified using specific programs that can be updated as necessary.

This particular piece of software has been used by several companies for tests to ensure the best results with minimal problems.

885-8006

HDOS SUBMIT $20.00

Introduction: SUBMIT is a utility which allows the creating of "command files". The command file consists of HDOS or other utility commands which are passed to HDOS.

SUBMIT allows the user to create within a file a number of HDOS command lines, which can be executed by simply entering a single SUBMIT command. This may be used in many applications for HDOS users.

Requirements: SUBMIT was written under HDOS version 2.0, with 8080 assembly code and includes some Z80 instructions, and therefore will be limited to Z80 machines; i.e. H/Z89 or an H8 with a Z80 board.

Only one hard sectored H17 or H77 drive is required. The H19 terminal type is needed for the graphic requirements of SUBMIT. No other hardware devices are required, but obviously the instruction commands of SUBMIT will be limited to the devices of the system.

Author: Mr. Brian Downs

Program Content: As mentioned above, a SUBMIT "command file" consists of HDOS or other utility (e.g. MBASIC) commands which are passed to HDOS for execution. These command files are created by an editor such as with the HUG EDitor P/N 885-1022.

This SUBMIT program is different from other SUBMIT files, in that SUBMIT is actually a program which can contain certain logical condition statements. This will be explained later.

The invocation of SUBMIT from HDOS is as follows:

>SUBMIT filename/parm0,parm1,..parm9

The SUBMIT file commands may be complete in themselves. They may contain "parameter substitution holders" which will be replaced by the parameters specified on the SUBMIT invocation (above) before they are executed.

SUBMIT also recognizes special line types which allows the writing of "user friendly" and truly powerful command files. SUBMIT provides the following five special command lines to create a SUBMIT file:

1) Display Message Command
2) Parameter Test Command
3) Transfer Command
4) Label/Comment Line
5) Query Command

The following is a brief look at these command lines:

1) Display Message Command — This command will display a message on the terminal. The message is limited to 74 characters.

The message line begins with an "!". Two additional characters have special meaning when used in the message command; the "<" and ">" will turn the reverse video on and off, respectively.

2) Parameter Test Command — This command will test if a parameter has been specified at all or if it (the parameter) equals a particular value.

3) Transfer Command — The transfer command will alter the sequential execution of the command lines. The following is a transfer command:

↑ \LABEL\

This line will transfer control (GOTO) to the line immediately following the line containing the label "\LABEL\" and then continue executing commands. Used in combination with option 2) above, SUBMIT will create a conditional transfer (or GOTO).

SUBMIT may be terminated by a transfer to an unknown label.

4) Label/Comment Line — This line causes no action and is considered to be either a label or comment. (There is no difference between a label or comment.) A use of the "\LABEL\" was explained in number 3) above.

5) Query and File Existence Test Command — This command (ASK) allows the user to either, one, ask a question that can be answered by a "Y" or a "N", or two, test for the existence of a file.

The following is a simple program that could be created from an editor for SUBMIT:

```
ASK 'Do you want to mount SY1:?'
↑ \MOUNT SY1:\      (yes, mount)
↑ \BYE\             (no, quit)
\MOUNT SY1:\        (label)
MOUNT SY1:          (HDOS command)
ASK SY1:FILE.EXT    (exist?)
↑ \TYPE FILE\       (yes, type)
!FILE.EXT does not exist on SY1:
DISMOUNT SY1:       (HDOS command)
↑ \BYE\             (quit)
\TYPE FILE\         (label)
TYPE SY1:FILE.EXT   (HDOS command)
DISMOUNT SY1:       (HDOS command)
↑ \BYE\             (quit)
```

NOTE: There are other syntax instructions which have not been included in this example. This example program is intended to show the user one simple example of using conditional statements which are legal in this version of SUBMIT.

Command File Debugging: Since the logic contained in a SUBMIT file can become complex, there are two facilities for command file debugging.

The first allows the user to instruct SUBMIT to suspend execution after the expanded commands have been written to the executable (.SUB) file. The user can then inspect the .SUB file in order to see how the parameter substitution was accomplished.

The second debug facility is a tracing capability. This lets the user observe what commands are being executed as the command file is being processed.

SUBMIT has one further convenient capability, that is a "one line submit". This allows the user to enter a single command line that will do any number of HDOS commands. The command line is limited to 99 characters.

Comments: The written documentation contains the detailed information of SUBMIT. Some features of SUBMIT have not been included here.

SUBMIT will not allow interactive sessions within a sequence of a SUBMIT command file. E.G. SUBMIT can run MBASIC, load a file, print the file to the LP:, exit MBASIC to HDOS, and continue, but nothing can be typed from the terminal at any time. That would

mess up the type-ahead buffer.

This is an excellent package that gives an added dimension to executing a SUBMIT file.

HUG PRODUCTS LIST

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

Part numbers shown in **bold print** are available in soft sector 5.25-inch format. Add -37 to the part number to order soft sector. For example, to order 885-1206 in soft sector, use 885-1206-37.

Part Number	Description	Selling Price	REM #
CASSETTE SOFTWARE (H8 and H88)			
885-1008	Volume I Documentation and Program Listings (some for H11)	$ 9.00	
885-1009	Tape I Cassette	$ 7.00	
885-1013	Volume II Documentation and Program Listings	$12.00	
885-1014	Tape II ASM Cassette H8 Only	$ 9.00	
885-1015	Volume III Documentation and Program Listings	$12.00	
885-1026	Tape III Cassette	$ 9.00	
885-1036	Tape IV Cassette	$ 9.00	8
885-1037	Volume IV Documentation and Program Listings	$12.00	8
885-1039	WISE on Cassette H8 Only	$ 9.00	
885-1057	Tape V Cassette	$ 9.00	
885-1058	Volume V Documentation and Program Listings	$12.00	
HDOS SOFTWARE H8/H17 or H89 — 5-inch only)			
MISCELLANEOUS COLLECTIONS			
885-1024	Disk I H8/H89	$18.00	6
885-1032	Disk V H8/H89	$18.00	8
885-1044	Disk VI H8/H89	$18.00	
885-1064	Disk IX H8/H89	$18.00	
885-1066	Disk X H8/H89	$18.00	10
885-1069	Disk XIII Misc H8/H89	$18.00	
GAMES			
885-1010	Adventure Disk H8/H89	$10.00	4
885-1029	Disk II Games 1 H8/H89	$18.00	8
885-1030	Disk III Games 2 H8/H89	$18.00	8
885-1031	Music 8 & 89 H8/H19 and H89	$20.00	25
885-1067	Disk XI Graphic Games .ABS and B H BASIC (H19/H89)	$18.00	12
885-1068	Graphic Games (H19/H89)	* $18.00	10
885-1088	Graphic Games (H19/H89)	* $20.00	14
885-1093	Dungeons and Dragons Game Requires H89 or H8/H19	* $20.00	16
885-1096	Action Games (H19/H89)	* $20.00	18
885-1103	Sea Battle Game (H19/H89)	$20.00	20
885-1111	HDOS MBASIC Graphic Games	* $20.00	23
885-1112	HDOS Graphic Games	$20.00	23
885-1113	HDOS Fast Action Games	$20.00	23
885-1114	Color Raiders and Goop (HA-8-3)	$20.00	23
UTILITIES			
885-1019	Device Drivers (HDOS 1.6)	$10.00	6
885-1022	HUG Editor (ED) Disk H8/H89	$15.00	20
885-1025	Runoff Disk H8/H89	$35.00	
885-1050	M.C.S. Modem for H8/H89	$18.00	
885-1060	Disk VII H8/H89	$18.00	
	SUBMIT, CLIST, FDUMP, ABSDUMP, etc.		
885-1061	TMI Cassette to Disk H8 only	$18.00	
885-1062	Disk VIII H8/H89 (2 disks) MEMTEST, DUP, DUMP, DSM	$25.00	
885-1063	Floating Point Disk H8/H89	$18.00	
885-1065	Fixed Point Package H8/H89	$18.00	10
885-1075	HDOS Support Package H8/H89	$60.00	
885-1077	TXTCON/BASCON H8/H89	$18.00	
885-1079	HDOS Page Editor	$25.00	15
885-1080	EDITX H8/H19/H89	$20.00	
885-1082	Programs for Printers H8/H89	$20.00	
885-1083	Disk XVI RECOVER, etc.	$20.00	11
885-1089	MACRO, CTOH, and misc Utilities	$20.00	20
885-1090	Misc. HDOS Utilities CCAT, HPLINK, AH, MBSORT, etc.	$20.00	22
885-1092	RDT Debugging Tool H8/H89	$30.00	14
885-1095	HUG SY: Device Driver HDOS 2.0	$30.00	18
885-1098	H8/HA-8-3 Color .ABS/.ASM	$20.00	19
885-1099	H8/HA-8-3 Color in Tiny Pascal	$20.00	19
885-1105	HDOS 2.0 Device Drivers MX-80, Paper Tiger, Clock, etc.	$20.00	24
885-1116	HDOS Z80 Debugging Tool	$20.00	27
885-1119	B H BASIC Support H8/H19 or H89	$20.00	29
885-8001	SE UCSD-Style Screen Editor	$25.00	28
885-8003	B H BASIC to MBASIC Converter	$25.00	28
885-8004	UDUMP and FAKEMNT Disk Manipulation Utilities	$35.00	28
885-8005	MAPLE Modem Program	$35.00	29
885-8006	HDOS SUBMIT	$20.00	31
885-8007	EZI-TRANSfer	$30.00	30
PROGRAMMING LANGUAGES			
885-1038	WISE on Disk H8/H89	$18.00	
885-1042	PILOT H8/H89	$19.00	
885-1059	FOCAL-8 H8/H89	$25.00	13
885-1078	HDOS Z80 Assembler	$25.00	21
885-1085	PILOT Documentation	$ 9.00	
885-1086	Tiny Pascal H8/H89	$20.00	13
885-1094	HUG Fig-Forth H8/H89 2 Disks	$40.00	18
BUSINESS, FINANCE AND EDUCATION			
885-1047	Stocks H8/H89	$18.00	
885-1048	Personal Account H8/H89	$18.00	
885-1049	Income Tax Records H8/H89	$18.00	
885-1055	Inventory H8/H89	* $30.00	
885-1056	Mail List H8/H89	* $30.00	
885-1070	Disk XIV Home Finance H8/H89	$18.00	
885-1071	Small Business Package III 3 Disks H8/H19 or H89	* $75.00	17
885-1091	Grade and Score Keeping	* $30.00	14
885-1097	Educational Quiz Disk H89 or H8/H19	* $20.00	18
885-1118	Payroll	* $60.00	30
DATA BASE MANAGEMENT SYSTEMS (DBMS)			
885-1107	Amateur Radio Logbook and TMS	$30.00	23
885-1108	Telephone/Mail Info. System	* $30.00	23
885-1109	Retriever (2 disks)	$40.00	23
885-1110	Autofile	$30.00	23
885-1115	Aircraft Navigation H8/H89	* $20.00	25
A85-8008	Farm Accounting System	* $45.00	30
AMATEUR RADIO			
885-1023	RTTY Disk H8 Only	$22.00	6
885-1106	Morse-89 H8/H19 or H89	$20.00	22

* Means MBASIC is required

H11 SOFTWARE

885-1008	Volume I Documentation and Program Listings (some for H11)	$ 9.00	
885-1033	HT-11 Disk I	$19.00	
885-1053	H11/H19 Support Package EXEC Modem Software, etc.	$20.00	27
885-1117	Pirate's Adventure for H11/H19	$20.00	28

CP/M SOFTWARE (5-inch only)

885-1201	CP/M (TM) Volumes H1 and H2	% $21.00	
885-1202	CP/M Volumes 4 and 21-C	%% $21.00	
885-1203	CP/M Volumes 21-A and B	%% $21.00	
885-1204	CP/M Volumes 26/27-A and B	%% $21.00	
885-1205	CP/M Volumes 26/27-C and D	%% $21.00	
The above CP/M products are 2 disks each.			
885-1206	CP/M Games Disk	* $20.00	11
885-1207	TERM and H8COPY	$20.00	26
885-1208	HUG Fig-Forth H8/H89 2 Disks	$40.00	18
885-1209	Dungeons and Dragons Game MBASIC and H89 or H8/H19	* $20.00	19
885-1210	HUG Editor	$20.00	20
885-1211	Sea Battle Game for CP/M	$20.00	20
885-1212	CP/M Utilities I	$20.00	21
885-1213	CP/M Disk Utilities	$20.00	22
885-1214	Amateur Radio Logbook	* $30.00	23
885-1215	BASIC-E	$20.00	26
885-1217	HUG Disk Duplication Utilities	$20.00	26
885-1218	CP/M MBASIC Payroll	* $60.00	31
885-1219	CP/M Aircraft Navigation H8/H89	* $20.00	31

% Means CP/M 1.43 only (ORG-4200).
%% Means CP/M 1.43 or 2.2 (Heath).
MBASIC programs on these disks are for version 4.8 or earlier.
Other CP/M disks are for 2.2.
* means MBASIC is required.

MISCELLANEOUS

885-4	HUG Binder	$ 5.75	
885-4001	REMark VOLUME I	$20.00	23
885-4002	REMark VOLUME II	$20.00	

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

Vectored from page 14

And there are 10 different sound effects, which are guaranteed to make you very unpopular if someone is trying to watch TV. The neatest program is one called SOUNDBLD.BAS that lets you mess around with the sound chips and build various effects, then save it for use by other programs. As the brochure says, you feel like an artist with a paint brush. I didn't know I was so talented! All the software is available on HDOS or CP/M and in both BH BASIC and MBASIC.

Summary: This system is well designed and documented, and it's educational as well as useful. We all have different uses for our systems, and an accessory such as this is very desirable. However, I don't want my seven thousand dollar box sitting around waiting for time to turn on the yard light. Therefore, some background software is needed and probably has already been developed by someone.

The HOUSEMASTER is available as a Kit or assembled from ATRA P.O. Box 653 Arlingtom Va 2216 and comes with a 30 day money-back-no-questions-asked warranty as well as the standard 90 day warranty on parts and labor unless you mess it up.

Heath Related Products

Announcing MORTTY

MORTTY an advanced communications program for the H8, H89 or Z89 with HDOS. MORTTY is a general-purpose communications program adaptable to almost any set of conventions in current use. It includes ASCII and Baudot code capabilities at any baud rate of which the equipment is capable. There are 17 parameters for adaptation to particular conventions such as full-screen, split-screen with type-ahead buffer, full or half duplex, and may more. There are 15 disk operations, including disk-file send, receive, direct hex upload and download, automatic message storage with file sequencing, and auto-answer from disk.

Other features include advanced string storage and transfer system with many convenient applications, such as auto-logon, automatic programming of smart modems, printers, and the like, CW ID, and user-definable functions executable by a single key stroke.

Equipment required:
1a. H89 or Z89 with min. 32K of memory plus H88-3 serial interface

or

1b. H8 with min. 32K memory, H19 terminal, H-17 disk system and H8-4 serial interface

and

2. Telephone modem or equivalent modulators, demodulators, transmitters, receivers, etc., for other types of communications.

Optional equipment: A printer, extra memory and extra disk drives all can be used.

Software required: HDOS ver. 2.0. Price: $100.00 ppd., includes 5 1/4" hard-sectored disk with MORTTY.ABS, a hardcopy 30 page user's guide and any updates issued within one year of purchase date.

Send check or money order, specify "MORTTY program", to:

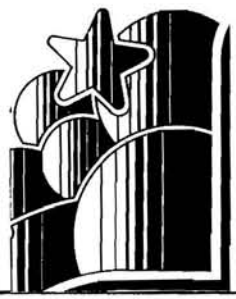
Phillip L. Emerson
3707 Blanche
Cleveland Heights, OH 44118

GRAFIX25: A Developmental Utility

Robert B. Hall
2603 Wayne Street
Bellevue, NE 68005

Have you ever written a BASIC program whose output included a chart? Were you able to remember which letter in the graphics (ESC-F) mode stood for the left corner? Or the right corner? If you are like me and have a poor memory for "trivial" details like these; if you detest referring to the Heathkit Operations Manual everytime you need a new corner; then take heart! The following program may be of some help. Run **GRAFIX25** before you start developing your chart-producing program. GRAFIX25 will print 11 common graphic characters alongside their alphabetic conuterparts on the 25th line. They are safe (relatively) from accidental erasure and will not interfere with the developmental process and they are conveniently in plain sight! DELETE GRAFIX25 and you are ready to continue with your own program.

```
890   '
900   ' GRAFIX25:  A DEVELOPMENTAL UTILITY
910   '
1000  E$=CHR$(27)
1200  DEF FN P$(X,Y)=E$+"Y"+CHR$(31+Y)+CHR$(31+X)
1300  PRINT E$"x1" FN P$(1,25)
1400  DATA "`",96
1500  DATA "a",97
1600  DATA "b",98
1700  DATA "c",99
1800  DATA "d",100
1900  DATA "e",101
2000  DATA "f",102
2100  DATA "s",115
2200  DATA "t",116
2300  DATA "u",117
2400  DATA "v",118
2500  '
2600  PRINT FN P$(1,25) E$ "1"
2700  FOR I=1 TO 11
2800     READ A$,A
2900     PRINT FN P$(I*7-6,25) A$ "= " E$ "F" CHR$(A) E$ "G"
3000  NEXT I
3010  PRINT FN P$(1,1);E$"E"
```



NEXT MONTH
Spot Light on the FIRST HUG NATIONAL CONFERENCE

More on Improvements to B H BASIC

Patrick Swayne
HUG Software Engineer

In my articles on improvements to Benton Harbor Disk BASIC in REMark issues 29 and 30 I forgot the HDOS 1.6 user in presenting the source listings for LBASIC, BEDIT, and ELOADR. The HDOS 1.6 user does not have access to the .ACM files in the XTEXT statements, so the programs will not assemble without errors. To correct this, remove the XTEXT statements from the programs and replace them with definitions as follows.

In the program LBASIC.ASM, add the following definitions (after removing the XTEXT statements).

```
.EXIT   EQU  0          .NAME   EQU  54Q
.SCIN   EQU  1          S.SYSM  EQU  40320A
.CLRCO  EQU  7          S.OMAX  EQU  40324A
.LINK   EQU  40Q        S.DLINK EQU  40346A
```

In BEDIT.ASM, add these definitions.

```
.EXIT   EQU  0          .CLRCO  EQU  7
.SCIN   EQU  1          .OPENR  EQU  42Q
.SCOUT  EQU  2          .CLOSE  EQU  46Q
.PRINT  EQU  3          .NAME   EQU  54Q
.READ   EQU  4          .ERROR  EQU  57Q
.CONSL  EQU  6          USERFWA EQU  42200Q
```

In ELOADR.ASM, add these definitions.

```
.EXIT   EQU  0
.READ   EQU  4
.LOADO  EQU  10Q
.SETTP  EQU  52Q
.ERROR  EQU  57Q
```

Additional changes must be made to BEDIT.ASM to assemble it with the HDOS 1.6 assembler. When it is assembling a CODE PIC program, it does not understand references to the origin pointer ("*-2" in BEDIT.ASM), so they must be replaced with labels. For example, where the program has

```
EDIT    SCALL   .SCIN
        JNC     *-2
```

it must be replaced with the following.

```
EDIT    SCALL   .SCIN
        JNC     EDIT
```

In two places you will have to add a new label. Just put it in the label column of the line before the "*-2", then replace the "*-2" with that label. You can use LABEL1 and LABEL2 for your labels.

If you are thinking of buying HUG disk 885-1119, you should know that LBASIC and EDBASIC (BEDIT plus ELOADR) are supplied in assembled forms for both HDOS 1.6 and 2.0. I used the HDOS 2.0 assembler to assemble the HDOS 1.6 programs, so I missed the definitions and label problems, but the programs on the disk work.

Listing to a Printer

In Benton Harbor Disk BASIC, you can list a program to a printer by typing

```
*REPLACE "LP:"
```

(assuming that your printer device driver is LP.DVD). This causes the entire program currently in memory to be listed. But suppose you want to list only a few lines of the program on a printer. Well, BASIC has an undocumented feature that allows you to do it. The LIST command has the ability to list to a device in a manner similar to the PRINT command. For example, if you want to list lines 10 through 100 on a printer, you could do it as follows.

```
*OPEN "LP:" FOR WRITE AS FILE #1
*LIST #1,10,100
*CLOSE #1
```

Part of the lines will be printed when you enter the second statement, and the rest will be listed when you CLOSE the printer. This is because BASIC only sends data to the printer in 256-byte lumps until the printer is closed.

Since disk files are treated the same way as the printer in B H BASIC, you can also LIST part of a program to a file. If you have a program that has a subroutine starting at line 2000 and ending with line 2120, you can save it as a separate file this way.

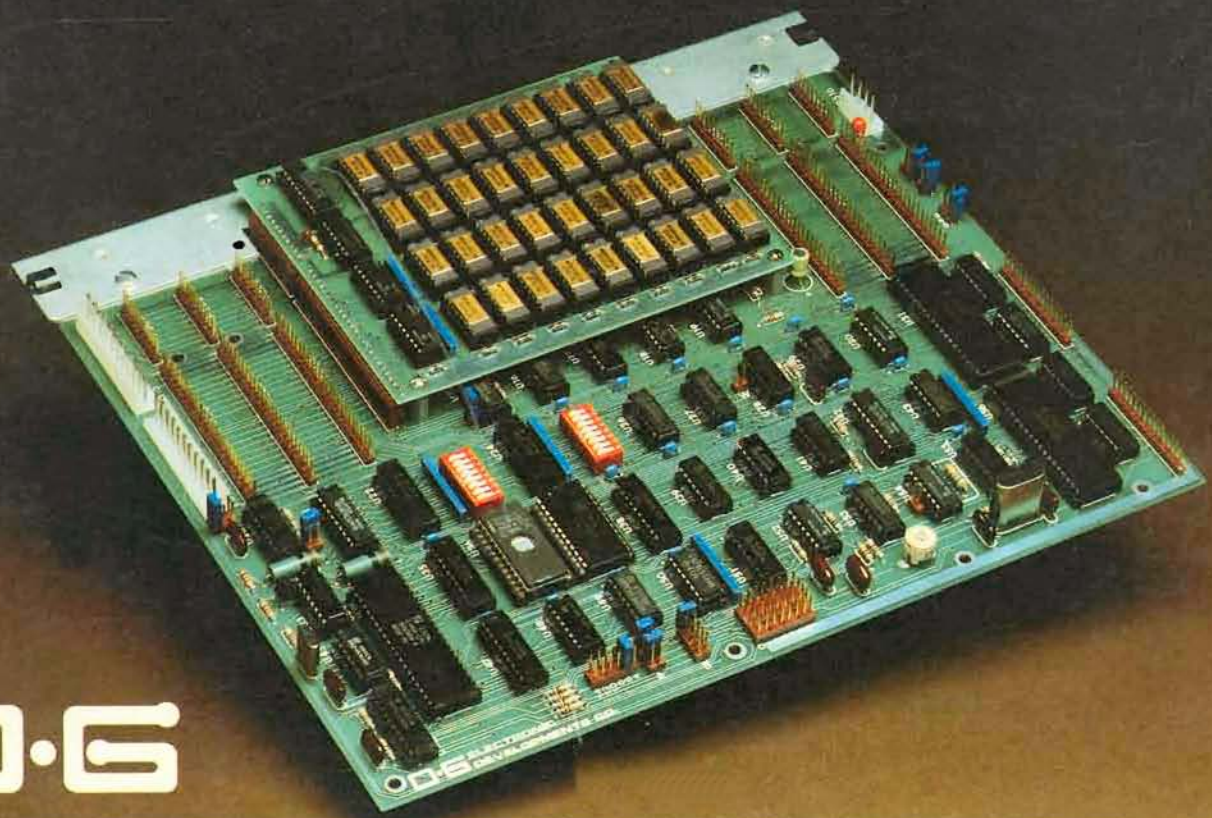
```
*OPEN "SUB.BAS" FOR WRITE AS FILE #1
*LIST #1,2000,2120
*CLOSE #1
```

Response has been good to my improvements to B H BASIC, indicating that there are a lot of you using it out there. If I make more discoveries or write more support programs, they will appear in REMark, and there may be more support disks.

PS:

The SUPER 89

- Multi-User Capability
- Twice the Computing Speed
- Up to 256K of Bank Selectable RAM
- Data Verification with Parity Check
- More Expansion Slots
- Real Time Clock
- Arithmetic Processor Provision
- Fully Heath Compatible



D·G

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

Heath
Users'
Group
Hilltop Road
St. Joseph MI 49085

POSTMASTER: If undeliverable,
please do not return.

885-2031