REMark

Issue 28 • May 1982

Official magazine for users of Heath/Zenith computer equipment.

on the cover . . . .

Anita Joyelle Swayne
(future software genius)

Tulip Festival 1981
Holland, Michigan

on the stack

>CAT

HUG Goes West 3
Jim Blake

H19 Control Subroutines for the H11 4
Edward Judge

Bullet Proof Software 5
Jim Blake

Oh Those Keys ... They're Special 6
David E. Warnick

A Computer Memory Keyer 8
Dick Jugel

NAVPROGseven Errata 12
Alan Bose

Split Byte Octal/Decimal Addition/Subtraction 13
Robert G. Traub

Buggin' HUG 14

New HUG Software 16

HUG Product List 26

National Conference Filling Up 28
Jim Blake

Heath Related Products 31

Local HUG News 32

Linking Sound-89 to MBASIC 33
Gordon Leon Duke, PHD

PATCH Mysteries Revealed 35
Patrick Swayne

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

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

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

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

Send payment to:

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

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

REMark

HUG GOES WEST

It was my pleasure again to be in attendance at the West Coast Computer Faire (WCCF) held from the 19th to the 21st of March at the Civic Auditorium in San Francisco. I was attending the show with Pat Swayne (the HUG software Wizard) and Jim Blake whom most of you already know just rejoined HUG. There were an incredible 40,000 individuals attending the WCCF over the three day period. Obviously, the action was toe-to-toe with many exciting exhibits for the computer enthusiast to examine.

One of the great pleasures I had was seeing the increased support for the Heath/Zenith computers all over the Faire. Across from the Heathkit display area, I was able to locate some of our good friends at the BUSS booth including Magnolia, DG Electronics, The Software Toolworks, Ultimeth, as well as the crew of BUSS/Sextant. Speaking of Sextant, Charlie Floto and his crew have done a marvelous job of constructing the premiere issue of Sextant. My first look at Sextant, a magazine that supports Heath/Zenith computers, came when I was walking around the Faire near the front entrance. I will warn you now. The front entrance of the WCCF is not the place to begin reading such an excellent publication! Of the many displays one could view at the WCCF, you could literally spend several hours just inside the entrance looking over the many goodies that are offered by the companies found in the BUSS booth. In fact, that is exactly what did happen as I viewed this Heath/Zenith support area from across the isle at the Heath Users' Group display. Down the isle a bit, I located the Commsoft display and CDR was there too!

During our brief stay in San Francisco, it was my pleasure to meet and chat with fellow HUGGIES from all over the United States. I was even able to meet some of our Europeon friends who had traveled to the West Coast just to visit the WCCF.

Saturday presented a little surprise for people attending the Heathkit display as a new product was announced from Zenith known as the ZT-1. The ZT-1 is a new lowcost communications terminal with built in modem and auto-dial features that allow the user to program the numbers to be dialed. These numbers are then placed in non-volatile memory and can be accessed through a menu which can be called during operation. The ZT-1 is also known as the "Pearl" and will sell for about $600.00. The price includes the keyboard/modem and power supply as well as the ZVM-121 Video Monitor....a truly complete communications terminal. One notable feature that could be added is a RS-232 output. As furnished, the ZT-1 now has a TTL output and a direct-connect to the phone.

Sunday morning, from 9:00 AM to noon, was the scheduled HUG meeting where we were able to meet with the membership directly to exchange lies and find out about those exciting new Heath/Zenith projects from the guest speaker, Mr. Barry Watzman. Barry, of course, is the man responsible for new computer products and operating system software for Heath and Zenith. Preceding Barry, HUG had a little slide show that represented a plant tour of the 12 acre facility located in St. Joseph, Michigan. Jim Blake, the current Software Coordinator for the Heath Users' Group and the man with the golden voice, took us on the humorous and entertaining tour with much input from the audience. Even though I had seen the slides, it was great to see the response from our users as Jim tripped and fumbled through the show.

When Barry was introduced, our users were ready! Barry answered question after question where he could and performed an excellent job of "hedging" about the new and mysterious "Z" Machine we have all been hearing rumors about. He indicated that he would provide us with as much detail as he was able (which wasn't much). Anyway, we did catch some key information from Barry such as his response to a question regarding compatibility with older software. Barry mentioned (and explained.), that for the most part, software that we are currently using could be made to operate in the mystery machine. He qualified this statement heavily by indicating some changes to the existing software may be required to exactly duplicate our current program operations, (we've all been through that before....Right!?).

Also, Barry described the new H8 Controller Board that allows the H8, WITH Z80 Board installed, to operate with the H/Z-37 Tandon drives and the Z-67 10 meg hard-disk. This product was displayed (and in operation) in the Heathkit booth, so, we now know that it

Vectored to 28

H19 Control Subroutines for the H11

Edward Judge
30 Autumn Drive RFD
Northampton, MA 01060

This article is basically a program that is a modification of a DECUS program to control the VT-100 CRT Terminal. The program is not fine tuned, but when I fine tuned the VT-100 version, I only gained a couple of hundred byte reduction. It works very well and has proven to be extremely useful in creating professional looking programs. The routines are carefully named to be useful mnemonics that readily relate to their function. They can be called from either FORTRAN or MACRO programs, and make screen formatting a breeze. I did not implement several of the editing functions as I did not need them, but they could be added very easily by following the obvious format.

The Source Macro listing follows, and compiles into a file 3 blocks long. It is fairly well commented, so following it should be easy.

This program was adapted from DECUS 11-424 by David Tindell.

```
     .TITLE H19
     .MCALL    .PRINT
     .ENABL    LC
;-----------------------------------------------------------------
     .GLOBL    GOTOXY
     .GLOBL    ANSI,RESTOR
     .GLOBL    CLRSCR,CLRBOS,CLREOS
     .GLOBL    CLRLIN,CLRBOL,CLREOL
     .GLOBL    KBDSBL,KBENBL
     .GLOBL    CURUP,CURDWN,CURFWD,CURBAK
     .GLOBL    CURRPT,CURSAV,CURRTN
     .GLOBL    BLKCUR,LINCUR
     .GLOBL    CURON,CUROFF
     .GLOBL    L25ON,L25OFF
     .GLOBL    CLKON,CLKOFF
;=================================================================
;-------        ENABLE ANSI MODE
ANSI:     .PRINT    #MSG2
     RTS  PC
;-----------------------------------------------------------------
;-------        CURSOR POSITION REPORT
CURRPT:   .PRINT    #MSG3A
     RTS  PC
;-------        PRESENT CURSOR POSITION - SAVE AND RETURN
CURSAV:   .PRINT    #MSG3B
     RTS  PC
CURRTN:   .PRINT    #MSG3C
     RTS  PC
;-----          RESTORE CRT TO INITIAL POWER-UP SETTINGS
RESTOR:   .PRINT    #MSG3D
     RTS  PC
;-----------------------------------------------------------------
;-----          KEYBOARD DISABLE AND ENABLE
KBDSBL:   .PRINT    #MSG4
     RTS  PC
KBENBL:   .PRINT    #MSG4A
     RTS  PC
;-----------------------------------------------------------------
;-----          CLEAR SCREEN - ALL
CLRSCR:   MOV  #MSG4B-1,R2
     MOVB #'2,3(R2)
     JMP  SCREEN
;-----          CLEAR SCREEN - FROM BEGINNING OF SCREEN TO CURSOR
CLRBOS:   MOV  #MSG4B-1,R2
     MOVB #'1,3(R2)
     JMP  SCREEN                                    Vectored to 29
```

Bullet Proof Software

Since the 'Jingles-in-your-Jeans' program began, several good programs have been submitted, but ALL were written by programmers and MOST were written to be used by programmers. Well, this is not to be critical, but not all users know what you have in mind when they use your program. Lemme 'splain. (This is in reference to MBASIC programs.)

Suppose you have written a game or perhaps a small business applications program and you ask the user for input to a question with the expected answer of 'Y' and you get a 'y'? What's your program going to do? Go off in the weeds? Or, suppose you ask the user to type in a number and you get something else. What does a '?Redo from start' error message mean to Sally Dumb-body? How about 'File not found at line 710? The business man or anyone else who doesn't speak computerese can't deal with that and shouldn't have to. And in our case, it is going to make the phone ring and WE can't deal with THAT. So let's see how to fix these three problems.

YES <> yes <> Y <> y AND NO <> no <> N <> n

```
B1000 ' Get user response
1010 PRINT "Do you want to get rich writing software? Y/N <Y>";:Q$=INPUT$(1)
1020 GOSUB 10000:'  Map Lower to Upper
1030 IF Q$<>"Y" OR Q$<> CHR$(13) THEN PRINT "Sorry to hear that!" ELSE 1050
1040 END or whatever
1050 'Write him a check!
```

See, here, the ONLY valid response is a 'Y' or 'y' OR <CR> AND it only accepts one key stroke.

Now, let us suppose you expect numeric input, such as a numbered selection from a Menu?

```
1000 ' Get Numeric input 1 through 9
1010 PRINT "Which Game would you like to play ? <END> ";:Q$=INPUT$(1)
1020 IF Q$=CHR$(13) THEN SYSTEM
1030 IF VAL(Q$)<1 OR VAL(Q$)>9 THEN PRINT"Invalid choice!:GOTO 1010
1040 ON VAL(Q$) GOTO 2000,2010, etc.
*
*
2000 RUN"GAME ONE"
2010 RUN"GANE TWO"
etc.
```

Now, we will check for a valid string input.

```
1000 '    Get user string input.
1010  PRINT "For which month do you wanttheprint-out? ":M$=INPUT$(3):PRINT M$
1020 M1$="JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC"
1030 N=INSTR(M1$,M$):IF N=0 THEN 1060:IF MID$(M1$,N,3)<>M$ THEN 1060
                                    *
                                    *

1060 PRINT "Invalid Month!":GOTO 1000
```

Ok?... What about the 'File Not Found' problem?

MBASIC has a neat error handling facility that takes care of this. Look in Appendix A on page A-4,5 and 6 for all kinds of ways to trap errors. In this instance, if our program can't find a file, we want to give the user some humaniod answer to the problem or if possible, fix it for him, so....

```
1000 ON ERROR GOTO 5000
1010 OPEN "I",1,"FOO.BAR":IF ERR = 53 THEN 5000

5000 PRINT "The file FOO.BAR does not reside on this diskette. Perhaps"
5010 PRINT "you have the wrong one.":GOTO MENU or something.:' etc.
```

Vectored to 39

Oh Those Keys ... They're Special

David E. Warnick
RD#2, Box 248
Spring Grove, PA 17362

One of the many excellent features of the Heath H89 & H19 is the 8 function keys not found on the competitors units. Unfortunately, when I first tried to make use of these keys, I found that few if any of the owners I spoke with could help. So, now that I've got them working, I'll pass on several methods of programming in BH BASIC which will allow these keys to be decoded. Then I'll include a heavily documented program you can run and apply to your needs.

First, keep your operating manual handy so you can check chapters 6 & 12, they're invaluable. Only those portions of the program associated with the function keys will be explained in this article. However, all operations will be documented with REMark statements.

The inputs we will get are all ESCape sequences as follows:

```
Blue Key    =   ESC P
Red Key     =   ESC Q
White Key   =   ESC R
F1 Key      =   ESC S
F2 Key      =   ESC T
F3 Key      =   ESC U
F4 Key      =   ESC V
F5 Key      =   ESC W
```

We'll assign 8 arbitrary functions. Any function could be used, be it a PRINT statement, GOSUB, mathematical formula, or what have you. For the demonstration program, I've chosen SIN, COS, TAN, SQUARE, SQUARE ROOT, LOG, INTEGER, & STOP.

Now, let's write our program. To input an ESC sequence, we'll need a LINE INPUT. That's simple enough, e.g.:

```
LINE INPUT"What functions shall I perform? ";F$
```

Now to decode our input. Let's verify that it's an ESCape sequence we've got. To do that, we'll use the ASCII value of ESC. Per our operating manual, it's 27, e.g.

```
IF ASC(F$)=27 GOTO nnn
```

To separate the keys, we must look at their differences. This is the letter, and appears on the right of the string. So, we'll use RIGHT$.
We can check individual keys with

```
IF RIGHT$(F$,1)="S" GOTO nnn

IF RIGHT$(F$,1)="W" THEN -------
```

We could also have used

```
IF F$=CHR$(27)+"S" THEN --------

IF F$=CHR$(27)+CHR$(80) GOTO nnn
```

The ASCII values can save us steps here, too. Let's go back to chapter 12 of the manual. The values we find are ASC(P)=80 thru ASC(W)=87. If we subtract 79 from each they equal 1 through 8. Remember ON.....GOTO?

```
ON ASC(RIGHT$(F$,1))-79 GOTO nnn,nnn,nnn,etc.
```

This way each key sends the computer to a specific sub-routine. All that's left is to define each key.

```
REM F1 KEY USED

A=SIN(N/57.296)
```

The division by 57.296 permits inputs in degrees rather than in radians. Finally we'll put it in a demonstration program & run it. We'll use the 25th line to identify the functions we've chosen, lest we forget.

As you set up and run this program, try to remember we've discussed and used:

1) A way to confirm that a function key was used
2) Ways to decode individual keys
3) A shortcut to decode all keys using ON...GOTO

When you apply this info, it's good to remember that there is no best method for all cases. It would be wasteful to decode all keys if only one function is to be programmed. Also, the test for ASCII value 27 lets any key perform the same function. So, think simplest and fewest steps.

Now, here's the complete program.

```
00010 REM **************FUNCTION KEY DEMONSTRATION***************
00020 REM ********************FUNCTION.BAS************************
00030 REM ******************BY DAVID E. WARNICK*******************
00040 REM
00050 REM
00060 REM DEFINE E$ AS ESCAPE
00070 E$=CHR$(27)
00080 REM SAVE CURSOR POSITION
00090 PRINT E$+"j"
00100 REM ENABLE 25TH LINE
00110 PRINT E$+"x1"
00120 REM ENTER REVERSE VIDEO
00130 PRINT E$+"p"
00133 REM MOVE CURSOR TO 25TH LINE
00136 PRINT E$+"Y8 "
00140 REM PRINT 25TH LINE
00150 PRINT "     SIN   COS   TAN   ROOT   SQR             LOG   ";
00160 PRINT "INT    STOP     "
00170 REM EXIT REVERSE VIDEO
00180 PRINT E$+"q"
00190 REM RETURN CURSOR TO POSITION SAVED
00200 PRINT E$+"k"
00210 REM GET FUNCTION
00220 LINE INPUT "WHICH FUNCTION SHALL I PERFORM? ";F$
00230 REM VERIFY THAT INPUT WAS AN ESCAPE SEQUENCE
00240 IF ASC(F$)=27GOTO 280
00250 REM OOPS, IT'S NOT AN ESCAPE
00260 PRINT "YOU DIDN'T TYPE ONE OF MY FUNCTION KEYS.  TRY AGAIN.":GOTO 220
00270 REM CHECK IF STOP FUNCTION WAS SELECTED
00280 IF RIGHT$(F$,1)="R"GOTO 490
00290 REM GET NUMERIC INPUT
00300 INPUT "WHAT NUMBER SHALL I PERFORM THAT FUNCTION ON? ";N
00310 REM SEPARATE THE FUNCTIONS
00320 ON ASC(RIGHT$(F$,1))-79GOTO 440,460,490,340,360,380,400,420
00330 REM F1 KEY
00340 A=SIN(N/57.296):PRINT "THE SIN OF ";N;" IS ";A:GOTO 220
00350 REM F2 KEY
00360 A=COS(N/57.296):PRINT "THE COS OF ";N;" IS ";A:GOTO 220
00370 REM F3 KEY
00380 A=TAN(N/57.296):PRINT "THE TAN OF ";N;" IS ";A:GOTO 220
00390 REM F4 KEY
00400 A=SQR(N):PRINT "THE SQUARE ROOT OF ";N;" IS ";A:GOTO 220
00410 REM F5 KEY
00420 A=N*N:PRINT N;" SQUARED IS ";A:GOTO 220
00430 REM BLUE KEY
00440 A=LOG(N):PRINT "THE LOG OF ";N;" IS ";A:GOTO 220
00450 REM RED KEY
00460 A=INT(N):PRINT "THE INTEGER OF ";N;"IS ";A:GOTO 220
00470 REM COME HERE WHEN STOP IS SELECTED
00480 REM DISABLE THE 25TH LINE
00490 PRINT E$+"y1"
00500 STOP
```

A Computer Memory Keyer

Dick Jugel, K0DG
8014 Taylor Circle
Omaha, NE 68134

After using one of the new memory keyers in the 20 meter CW shack on field day, I was hooked! Never again would I be completely satisfied with a normal keyer or bug, having gotten used to automatic CQ's and contest exchanges. The extra "pause that refreshes" that the keyer provided during memory read-out cycles had the effect of relaxing the constant contest tension, resulting in more pleasurable operation. However, I did not like the $100.00 plus price tag, especially since the almost-new Heath HD-1410 electronic keyer at the home QTH still had a good many dit and dahs left in it.

Then, while running RTTY with my Heath H89 computer one evening, the idea hit me - why not use the computer, interfaced to the keyer, as a memory device? This should allow even more flexibility than any of the currently available stand-alone memory keyers, and would not require a cash outlay for another keyer. The only cost would be for the interface components, since I already had the computer and the electronic keyer. The idea seemed sound enough, so the next weekend was spent building the hardware interface and developing a suitable memory-keying computer program.

The HD-1410 keyer has provision for an external key input, so the computer can be used, in effect, as an external keyer. The program can be easily changed to perform different functions, including automatic message numbering and logging for contest work.

The program discussed in this article is a BASIC memory-keying program, but can easily be modified to perform almost any desired function. It is written in Microsoft BASIC, and should be useable, with some changes, on other types of computers.

The interface from the computer to the keyer is a simple RS-232 to TTL converter, consisting of a 1K resistor, a diode, and an NPN transistor (figure 1). I etched the circuit on a piece of scrap PC board, but perfboard construction is also acceptable. Coaxial cable was used to connect the board between the HD-1410 keyer and an H89 computer serial port. A phono plus is used to connect the interface to the keyer's "EXTERNAL KEY" jack, and a 25 pin RS-232 male plug connects to the computer serial port (I used port 320Q on the H89). The coax shield is connected to the system ground. The interface board can be encased in a small metal box if high levels of RF are present.

In operation, the VOX circuit of a transceiver can be used to provide automatic transmit/receive switching, much like semi-breakin operation. I use the VOX method with a Kenwood TS-520 with good results, as long as the VOX delay is adjusted properly. If the VOX delay is not long enough, the constant dropping in and out of the transmit relay between words or characters can become annoying.

The main program variables can be modified to fit any given hardware configuration or application by simply changing the program source code. I have found this method works well, particularly when there are many more possible uses for the keyer than could be defined and coded into a single large program. For instance, I have modified versions of the program that will support contest operation, keyboard send/receive operation with or without memories, and code practice transmission from pre-stored disk files.

The string variable "C$" at line 60 contains the sending station's call sign. This variable is used in the program to identify the station, without the need to repeatedly key in the station call sign.

The string variables "F1$", "F2$", F3$", F4$", "F5$", "B$", "R$", and "W$" in lines 210 through 280 contain the memory contents for the f1 through f5, blue, red, and white function keys, respectively. These variables may be set to contain any desired string. They can then be "called up" for transmission at any time during program execution by simply hitting the associated function key. Each of these eight memories has a maximum individual storage size of 255 characters, for a combined total of 2,040 characters.

When the program is executed, it will clear the screen, print a heading line on the first line of the CRT, and a legend line on the last (25th) line. The legend line will remain on the screen, but the heading line will eventually scroll upward and off the display. The legend line is a constant reminder of special morse code character assignments, as follows:

CHARACTER	FUNCTION
;	AR
=	SK
:	BK
-	BT
<	KN
>	AS
[	SOS
%BYE	END OF JOB (STOP PROGRAM)

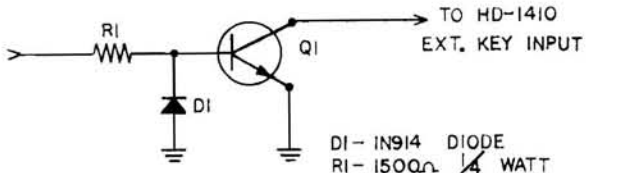


FIGURE 1

When the indicated keyboard characters in the CHARACTER column are entered, the corresponding morse special characters in the FUNCTION column will be generated. These characters may be imbedded anywhere in the text or memory contents. The %BYE command, on the other hand, is used to terminate the execution of the program, and it must start at the beginning of a line and be the only entry on that line.

The program will then ask the operator to enter the keying speed desired, in a range of 1 to 40 words per minute. An invalid entry at this time will cause the system to default to a standard speed of 18 words per minute. However, the speed may be changed at any time during program execution by simply keying in the command S=n, where n is the new speed desired. This command, like the %BYE command, must be the only entry on the line.

The special command C=x, where x is either CQ or another station's call, will cause a standard amateur calling sequence to be performed. For instance, if the string variable "C$" contains "K0DG", the command C=CQ will result in the following string being generated:

CQ CQ CQ DE K0DG K0DG K0DG K

The command C=N0CLW will result in the following string being generated:

N0CLW N0CLW N0CLW DE K0DG K0DG K0DG K

The C= command, like the S=, %BYE, and memory read-out function key commands, must be the only entry on an input line.

The memories are called up and sent by simply hitting their associated function keys as the first entry on a given line. No carriage return is necessary for sending the memories, since they are automatically sent as soon as the function key is pressed. This is the only time that transmission takes place before hitting the carriage return key, since the normal mode allows the entry of an entire line before transmission begins. The automatic transmission of the memories in this manner saves an extra keystroke when using the computer only as a memory storage device and not as a keyboard morse code generator. The operator can send a memory by hitting only one key, the function key, and does not have to also hit the carriage return key.

I would encourage you to experiment with the program and modify it to your own particular needs. The interface board will probably work with other solid state keyers equipped with an external key jack, and the program should not be too difficult to modify for computer systems other than the H89. In addition, I hope this article has given you some ideas for using your own computer in the Ham Shack.

```
10 REM MEMORY KEYER PROGRAM
20 REM BY DICK JUGEL, K0DG, AUGUST 1, 1981
30 WIDTH 255:REM SET CONSOLE WIDTH TO MAXIMUM
40 CLEAR 4500:DEFINT I,T,F,G,H,P,X
50 P=&O320:REM OUTPUT PORT ADDRESS
60 C$="K0DG":REM STATION'S CALL SIGN
70 REM **** SET MEMORY VARIABLES HERE ****
80 REM
90 REM   VARIABLE NAME:    CALLED BY FUNCTION KEY:
100 REM
110 REM       F1$                  F1
120 REM       F2$                  F2
130 REM       F3$                  F3
140 REM       F4$                  F4
150 REM       F5$                  F5
160 REM       B$                   BLUE
170 REM       R$                   RED
180 REM       W$                   WHITE
190 REM
200 REM ********************************
210 F1$="THIS IS MEMORY ONE"
220 F2$="THIS IS MEMORY TWO"
230 F3$="THIS IS MEMORY THREE"
240 F4$="THIS IS MEMORY FOUR"
250 F5$="THIS IS MEMORY FIVE"
260 B$="THIS IS BLUE MEMORY"
270 R$="THIS IS RED MEMORY"
280 W$="THIS IS WHITE MEMORY"
290 REM THE ABOVE MEMORY VARIABLES MAY BE SET TO ANYTHING.
300 REM **** SPEED CONSTANTS FOR BAUD TIMING ROUTINE ****
310 REM **** MAY HAVE TO BE CHANGED TO WORK WITH OTHER THAN H89 COMPUTER ****
320 DATA 1275,585,370,255,190,145,114,90,73,55
330 DATA 45,33,28,26,23,21,19,18,16,15
340 DATA 14,13,12,11,10,9,8,8,7,7
350 DATA 6,6,5,5,5,4,4,4,3,3
360 REM ***** MORSE CHARACTER TABLE *****
370 DATA --..--,-...-,.-.-.-,-..-.,-----,.----,..---,...--,....-,.....
380 DATA -....,--...,---..,----.,-...-.-,.-.-.,-.--.,...-.-,.-...,..--..,.-,-...,-.-.
390 DATA -..,.,..-.,--.,....,..,.---,-.-,.-..,--,-.,---,.--.,--.-,.-.
400 DATA ...,-,..-,...-,.--,-..-,-.--,--..,...---...
410 REM ***** ARRAYS *****
420 DIM C$(47)
430 DIM P7(40)
440 REM ***** FILL ARRAYS *****
450 FOR T=1 TO 40:READ P7(T):NEXT T
460 FOR T=0 TO 47:READ C$(T):NEXT T
470 REM ***** PUT LEGEND ON 25TH LINE - HEATH H19 TERMINAL ONLY *****
480 PRINT CHR$(27);CHR$(106);
490 PRINT CHR$(27);CHR$(120);CHR$(49);CHR$(27);CHR$(89);CHR$(56);CHR$(32)
500 PRINT
510 PRINT "             ;=AR ==SK :=BK -=BT <=KN >=AS [=SOS %BYE=EOJ"
520 PRINT CHR$(27);CHR$(107)
530 REM ***** CLEAR SCREEN AND PRINT HEADING *****
540 PRINT CHR$(27);"E";"M O R S E    M E M O R Y    K E Y E R"
550 PRINT
560 REM ***** GET MORSE SPEED DESIRED *****
570 LINE INPUT "ENTER SPEED IN WORDS-PER-MINUTE (1 TO 40): ";X$
580 GOSUB 1550:REM GO SET MORSE SPEED
590 REM ***** INITIALIZE 8250 A.C.E. *****
600 OUT P+1,&O0:REM CLEAR UART INTERRUPTS
610 OUT P+4,&O40:REM SET UART LOOPBACK MODE
620 OUT P+3,&O200:REM SET ON DATA LATCH ACCESS BIT (DLAB)
630 OUT P+1,&O0:REM 19,200 BAUD MSD
640 OUT P+0,&O6:REM 19,200 BAUD LSD
650 OUT P+3,&O0:REM RESET DATA LATCH ACCESS BIT (DLAB)
660 T=INP(P):REM READ A CHARACTER
670 GOSUB 1350:REM WAIT AT LEAST TWO CHARACTER TIMES
680 T=INP(P):REM READ A SECOND CHARACTER
690 OUT P+4,&O0:REM TAKE UART OUT OF LOOPBACK MODE.
700 REM ***** MAIN KEYBOAD INPUT ROUTINE *****
```

```
710 Z$="":REM NULL INPUT LINE
720 Z8$=INPUT$(1):REM GET AN INPUT BYTE
730 IF Z8$=CHR$(13) GOTO 840:REM EXIT IF CARRIAGE RETURN
740 IF Z8$=CHR$(27) AND LEN(Z$)>0 GOTO 720:REM DUMP IF ESC NOT 1ST CHARACTER
750 IF Z8$=CHR$(27) THEN PRINT "|"; ELSE PRINT Z8$;:REM ECHO CHARACTER
760 IF Z8$<>CHR$(8) THEN 810:REM EXIT IF NOT BACKSPACE
770 IF LEN(Z$) < 1 THEN 720:REM SKIP BACKSPACE IF AT BEGINNING OF LINE
780 Z$=MID$(Z$,1,LEN(Z$)-1):REM DROP LAST CHARACTER IN LINE
790 PRINT " "+CHR$(8);
800 GOTO 720:REM GET NEXT INPUT CHARACTER
810 Z$=Z$+Z8$:REM APPEND INPUT CHARACTER TO INPUT LINE
820 IF LEN(Z$) = 2 AND LEFT$(Z$,1) = CHR$(27) GOTO 840:REM EXIT IF MEMORY KEY
830 GOTO 720:REM GET NEXT INPUT CHARACTER
840 PRINT:REM CARRIAGE RETURN FOR ECHO LINE
850 IF Z$ = "%BYE" THEN 1530:REM %BYE IS END OF JOB!
860 IF Z$="" THEN 710:REM SKIP NULL INPUT LINES
870 IF LEFT$(Z$,1) = CHR$(27) GOTO 930:REM TEST FOR ESC
880 IF LEFT$(Z$,2) <> "S=" GOTO 1070:REM TEST FOR SPEED SET COMMAND
890 X$=MID$(Z$,3,LEN(Z$)-2):REM GET NEW SPEED
900 GOSUB 1550:REM SET NEW SPEED
910 GOTO 710:REM GET NEXT INPUT LINE
920 REM ***** PROCESS MEMORY SEND COMMAND (ONE OF THE FUNCTION KEYS) *****
930 IF LEN(Z$) <> 2 THEN 710:REM ERROR IF LENGTH NOT 2
940 M1$=RIGHT$(Z$,1):REM GET KEY IDENTIFIER
950 IF M1$ < "P" OR M1$ > "W" GOTO 710:REM ERROR IF NOT VALID FUNCTION KEY
960 REM ***** GET SPECIFIED MEMORY CONTENTS *****
970 IF M1$="P" THEN Z$=B$
980 IF M1$="Q" THEN Z$=R$
990 IF M1$="R" THEN Z$=W$
1000 IF M1$="S" THEN Z$=F1$
1010 IF M1$="T" THEN Z$=F2$
1020 IF M1$="U" THEN Z$=F3$
1030 IF M1$="V" THEN Z$=F4$
1040 IF M1$="W" THEN Z$=F5$
1050 GOTO 1120:REM PROCESS MEMORY AS IF IT WERE KEYED INPUT.
1060 REM ***** TEST IF CALL (C=) COMMAND *****
1070 IF LEFT$(Z$,2) <> "C=" GOTO 1130:REM EXIT IF NOT CALL COMMAND
1080 IF LEN(Z$) < 3 THEN 710:REM EXIT IF INVALID
1090 Z$ = RIGHT$(Z$,LEN(Z$)-2):REM GET PARAMETER PORTION OF C= COMMAND
1100 IF RIGHT$(Z$,1) <> " " THEN Z$=Z$+" ":REM INSURE TRAILING BLANK
1110 Z$=Z$+Z$+Z$+"DE "+C$+" "+C$+" "+C$+" K":REM BUILD CALL LINE
1120 PRINT Z$:REM PRINT MEMORY OR CALL LINE
1130 IF RIGHT$(Z$,1) <> " " THEN Z$=Z$+" ":REM INSURE TRAILING BLANK
1140 REM ***** NOW CONVERT THE LINE TO MORSE CODE AND SEND IT. *****
1150 FOR F=1 TO LEN(Z$):REM ITERATE THROUGH EACH CHARACTER IN LINE
1160 G=ASC(MID$(Z$,F,1))-44:REM GET A CHARACTER FROM LINE
1170 IF G<0 OR G>47 THEN 1260:REM DUMP IF INVALID CHARACTER
1180 L$=C$(G):REM GET CORRESPONDING MORSE CHARACTER
1190 IF L$="" THEN 1260:REM DUMP IF NULL CHARACTER
1200 FOR H=1 TO LEN(L$):REM PUT OUT EACH MORSE ELEMENT
1210 IF MID$(L$,H,1)="." THEN GOSUB 1270 ELSE GOSUB 1320:REM DIT OR DAH.
1220 NEXT H:REM ITERATE THROUGH ENTIRE MORSE CHARACTER.
1230 GOSUB 1440:REM WAIT BETWEEN WORDS
1240 NEXT F:REM ITERATE (GET NEXT CHARACTER).
1250 GOTO 710:REM GO GET NEXT INPUT LINE
1260 GOSUB 1420:GOTO 1240
1270 GOSUB 1380:REM TONE ON (MAKING A DIT)
1280 GOSUB 1350:REM WAIT ONE BAUD
1290 GOSUB 1400:REM TONE OFF
1300 GOSUB 1350:REM WAIT ONE BAUD
1310 RETURN
1320 GOSUB 1380:REM TONE ON (MAKING A DAH)
1330 GOSUB 1350:GOSUB 1350:GOSUB 1350:REM WAIT THREE BAUD
1340 GOTO 1290
1350 FOR I = 1 TO X
1360 NEXT I
1370 RETURN
1380 OUT P+3,&O100:REM TURN CW TONE ON Z(REAK BIT SET)
1390 RETURN
1400 OUT P+3,&O0:REM TURN CW TONE OFF (BREAK BIT RESET)
1410 RETURN
```

```
1420 GOSUB 1500:GOSUB 1500:GOSUB 1500:GOSUB 1500
1430 RETURN:REM WORD SPACE
1440 GOSUB 1460:GOSUB 1500:GOSUB 1500:REM LETTER SPACE
1450 RETURN
1460 IF X=X1 THEN RETURN
1470 FOR I=X TO X1
1480 NEXT I
1490 RETURN
1500 FOR I=1 TO X1
1510 NEXT I
1520 RETURN
1530 PRINT CHR$(27);"y1";"END OF JOB. THANK'S FOR DROPPING BYE (HI HI)."
1540 END
1550 X=ABS(INT(VAL(X$)))
1560 IF X<1 THEN X=18
1570 IF X>40 THEN X=18
1580 PRINT "SPEED NOW SET TO";X;"WORDS PER MINUTE."
1590 X=P7(X)
1600 X1=X
1610 IF X>30 THEN X=30
1620 RETURN
```

NAVPROGseven Errata

Alan Bose
2514 Essex Court
St. Joseph, MI 49085

If you chose not to keep the sample data distributed with the NAVPROGseven system, and you delete these files from your disk, you may have some difficulty re-initializing the files to start your own database from scratch. The following changes will fix this and a few other glitches that have been found.

AIRINPUT.BAS -- Add the following lines:

```
115 IF MD=-1 THEN MD=0
3875 IF ERL=3420 AND ERR=9 THEN RESUME NEXT
```

-- the following line should read:

```
3870 IF ERR=53 AND ERL=3900 THEN KY=1:RESUME3990
```

OLDROUTE.BAS -- Add the following line:

```
625 ERASE LI$,R1$,R1,R2$,R2
```

NAVPROG7.BAS -- the following lines should read:

```
1400 NEXT J:NA=CA
1500 IF F3>3 AND F3<50 THEN 1540
2170 IF CA<=Y THEN 2220
```

AUTONAV.BAS -- the following line should read:

```
980 IF CY=1 OR DH<=90 THEN PR(J)=3:GOTO1090
```

AIRALPHA.BAS -- in line 810 the word simultaneously is misspelled.

If you chose to keep the sample data, 115.1 is the correct frequency for the ELX vortac.

EOF

Split Byte Octal/Decimal Addition/Subtraction

by Robert G. Traub
9731-154 Street
Edmonton, Alberta
CANADA T5P 2G4

Anyone who has worked with the split-byte octal notation knows just how difficult a problem addition and subtraction between two 16 bit split-byte octal numbers can be. This is complicated even further if the addition or subtraction is required between a split-byte octal number and a decimal number. At the present time, there are calculators available that will allow one to work with the hexadecimal code, as well as the FULL octal code, but as far as is known, none are available that offer split-byte octal notation. Presented here is a small program that will allow for the addition and subtraction (it does both automatically) of either two split-byte octal numbers or between a split-byte octal number and a decimal number.

A split-byte octal number is achieved by the splitting of a 16 bit octal number into two 8 bit bytes. The 8 bit byte on the left is referred to as the HIGH order byte and the 8 bit byte on the right is referred to as the LOW order byte. With this notation the 16 bits would consist of two consecutive 8 bit bytes, as such:

```
00 100 000   01 000 000

0  4   0      1  0   0
```

where 040 is the high order byte, and 100 is the low order byte. To separate the two bytes, a period or slash character can be used such as:

```
040.100  or  040/100
```

This split-byte notation is not as common as the full octal notation, and therefore has not been included in calculators. The split-byte octal numbering system is not at all difficult to work with, and the program that follows will make working with it a pleasure.

To operate the program simply type the program name (SUBADD.ABS) followed by a carriage return. When the program is loaded into ram and run, it will first display the sign on message "RGT 2.0 :OCTAL ADD/SUB PROGRAM:. The program will then turn up two blank lines and display the message "TO EXIT HIT "H" OR ANY OTHER KEY TO CONTINUE". At this point the user may hit the "H" key to return to HDOS or hit any other key to continue to the first step in the program. This message will be displayed after each additon and subtraction in order to allow the user to exit the program. If the return key for instance was hit, the program will turn up two blank lines and display the message "ENTER FIRST 16-BIT OCTAL NUMBER:". The program will then turn up a blank line and wait for the octal number to be typed in. To enter an 8-bit octal number, all that needs to be done is to fill in the first three digits with "0" such as:

```
000.256
```

It should be noted here that the program will insert a period (.) after the first three octal numbers have been entered (high byte). This is provided in order to separate the high order byte from the low order byte and avoid confusion. As soon as the first split-byte octal number is entered, the program will turn up a blank line and the message "ENTER "D" FOR DECIMAL OR "Q" FOR OCTAL:" will be displayed. Again a blank line will be turned up and the program will wait for either a "D" or "Q" to be typed. If a decimal number is to be subtracted or added to the first split-byte octal number, then a "D" will be entered. If another split-byte octal number is to be subtracted from the first, then enter a "Q" to the above prompt. Assuming first that a "D" was typed in response to the above prompt, the program will then display the message "ENTER DECIMAL NUMBER:"; a new line will be turned up and the program will wait for the decimal number to be entered. If a "Q" had been entered in response to the above prompt, the message "ENTER SECOND 16-BIT SPLIT-BYTE OCTAL NUMBER:" will be shown. The program will turn up a blank line and wait for the input of the second split-byte octal number.

As soon as the second number has been entered, either decimal or split-byte octal, the program will automatically do both the subtraction and addition and display this on the console with the following format:

```
THE RESULTS OF THE SUBTRACTION ARE:
123.123  21331   01010011 01010011

000.173  00123   00000000 01111011

---------------------------------

122.330  21208   01010010 11011000

THE RESULTS OF THE ADDITION ARE:
123.123  21331   01010011 01010011
```

Vectored to 32

BUGGIN' HUG

Dear HUG,

RE: 885-1107 Amateur Radio Logbook and TMS

I find the TMS a very friendly system, and one which I think that (with some supervision) my wife can use in her project, a detailed study of about 140 members of our congregation who are over 65 years of age. I did run into some problems which the documentation did not make clear and I would suggest that these matters be clarified in future releases. I list them below for your consideration:

1) The TMS files are quite large. If it is to fit on a single sided single density disk, all of the non locked system files must be deleted. Then it completely fills the diskette. A good procedure for this is suggested in the Microsoft Fortran documentation.
2) The README.DOC material tells one to copy the 18 .BAS files. TMS has only 16 BASIC files, plus one .ABS file and one .CTL file.
3) The line driver for the line printer must be named AT.DVD and not LP.DVD. If one normally calls his/her line printer LP: then the DVD must be renamed to AT:
4) On page 28 of the TMS documentation, the text refers to a command "TMSRADD". There is no such command in either the menu or the directory. The reference should be corrected to read "RPTADD".
5) The text does not emphasize that each file should be on a separate diskette. While it does suggest that the file TMS.HDR be write protected, it does not warn of the horrors that can occur if you try to add a second file to a disk. I tried it to find out what would happen and found the following items:
    a) The TMS.HDR file is rewritten to include the field data for the new file.
    b) A dummy file or EOF characters is established under the name of newfile.OLD.
    c) The oldfile.OLD remains in memory. However, if you try to start over with NEWFILE a new dummy file will be established with EOF characters destroying the old data.
    d) The old data can be recovered before NEWFILEing but takes a bit of disk swapping.
    e) THE WRITE PROTECT SHOULD BE MADE MANDATORY.
6) The text refers to a TMSR000 on page 36. No such file exists on the distribution disk that I received. However, an examination of TMSR002 shows that it contains all of the subroutines listed on pages 37 and 38 except 11300 and 11400. The one, of course, would contain only a PRINT P$, the other a CHR$(27)+"E".
7) Pages 36 to 40 do not indicate that a "homemade" report generator must be filed on SY1:. Even if there was space for it on SY0: the linking routine in TMSRPTS (stmt 1250) calls the file from SY1:. This requirement should be emphasized.
8) While page 37 lists a few restrictions required in coding a "homemade" report generator, it gives no clues on what are suitable variable names for the homemade program. Since in BASIC all variables are "global" (the curse of what are called subroutines in BASIC), it would be possible to clobber a control variable or other piece of data by using a previously used variable name whose value will be subsequently called for. A cursive scanning seemed to indicate that Mr. Towner had not used any variable names beginning with the

character "W", so I used them in the sample report program enclosed. I have not crashed the program so far when I have used this report generator.

9) In reference to comment 8, it is a good thing to remember that:
   a) N8 contains equal to the number of fields in each record
   b) N$(x) contains the name of field (x)
   c) F0 contains the number of the sorting field
   d) F$ contains the file name
   e) "ZZZZ" is the end of file character

10) You cannot enter a value that begins with "DONE" into any field.

```
00001 REM TMSR003--- TMS COUNTER SUBPROGRAM
00100 PRINT "TMS COUNTING PROGRAM"
00110 PRINT "ENTER FIELD AND KEY WORD AS REQUESTED. THE PROGRAM WILL "
00120 PRINT "LIST THE REQUESTED FIELD OF THE ENTRIES WHERE MATCHES OCCUR "
00122 PRINT "AND THE TOTAL NUMBER OF MATCHES"
00124 PRINT "TO COUNT FOR ONE FIELD/KEY ENTER THE SAME VALUES FOR BOTH "
00126 PRINT "REQUESTS "
00130 W=0                                :REM INITIALIZE SUM
00140 INPUT "FIRST FIELD ",W1
00145 IF W1>N8 THEN PRINT "FIELD NUMBER TOO LARGE" : GOTO 140
00150 INPUT "FIRST KEY WORD ";W3$
00160 INPUT "SECOND FIELD ";W2
00165 IF W2>N8 THEN PRINT "FIELD NUMBER TOO LARGE" : GOTO 160
00170 INPUT "SECOND KEY WORD ";W4$
00180 INPUT "WHAT FIELD DO YOU WISH PRINTED IF TRUE ";W5
00185 IF W5>N8 THEN PRINT "FIELD NUMBER TOO LARGE ":GOTO 180
00210 OPEN "SY1:"+F$+"."+F1$ FOR READ AS FILE #1
00220 OPEN "AT:"FOR WRITE AS FILE #2
00230 PRINT #2,"TMS SUMMING PROGRAM RESULTS"
00240 PRINT #2,"  "
00250 PRINT #2,"FOR ";N$(W1);" EQUALS ";W3$
00260 PRINT #2,"AND ";N$(W2);" EQUALS ";W4$
00270 PRINT #2,"  "
00300 FOR X9=1 TO 9999
00310 GOSUB 12100
00320 IF LEFT$(B$(1),4)="ZZZZ" THEN GOTO 1000
00330 IF (B$(W1)<>W3$) OR (B$(W2)<>W4$)THEN GOTO 360
00340 W=W+1
00350 PRINT #2,B$(W5)
00360 NEXT X9
01000 PRINT #2, "  "
01010 PRINT #2,"NUMBER OF MATCHES EQUALS ";W
01020 CLOSE #2
01030 CLOSE #1
01040 CHAIN "TMSRPTS",10000
12100 REM - - - READ IN A DATA BASE RECORD
12105 IF F7=1 THEN RETURN
12110 FOR X3=1 TO N8
12115 B$(X3)=""
12118 IF F7=1 THEN RETURN
12120 LINE INPUT #1,;B$(X3)
12125 IF LEFT$(B$(X3),4)="ZZZZ" THEN F7=1
12127 IF LEFT$(B$(X3),1)="\" THEN B$(X3)=""
12130 NEXT X3
12140 RETURN
```

Sincerely,

Kenneth Mortimer
352 Green Acres Dr
Valparaiso, IN 46383

Vectored to 27

SPECIAL NOTICE

✱✱✱

The HUG CP/M BIOS for 2.2.03, announced in Issue 26 of REMark and carrying the HUG part number 885-1216, will not be made available through the Heath Users' Group. Those of you who have placed orders for this product will be refunded. We find ourselves in a position that would make it impossible to supply the type of user support of this product that we have tried to accomplish and maintain for other HUG products over the past year. We apologize for the inconvenience that our early announcement of this package has caused. If you would still like to investigate this product, you may contact Livingston Logic Labs directly.

Livingston Logic Labs
P.O. Box 5334
Pasadena, CA 91107

New HUG Software

885-1117 H11/H19 PIRATES ADVENTURE $20.00

Introduction: "PIRATES ADVENTURE" (Copyright Scott Adams, 1978) is the first of Scott Adams' well known series of mission-oriented adventure games. This version was converted from the article and listing which appeared in the December 1980 **BYTE** magazine, pp. 192-212.

This program was submitted to the Heath Users' Group by James C. Wilcox with written permission from Scott Adams.

Requirements: This version of PIRATES ADVENTURE is written for the H-11 computer under **HT-11 BASIC.** The converted program makes use of some **H19** ESCape sequences, which may easily be modified or deleted for use with other terminals.

The program has only been tested on H11's with 32KW of memory. The program is much faster under the HT-11 operating system than with its 8 bit counterpart.

Program Content: The object of PIRATES ADVENTURE is to search for treasures on Treasure Island, with the added "mission" of figuring out how to get there. The way is not easy and the obstacles are many.

The program accepts two word commands, except in special cases. The program compares the entered command with its data table and gives an appropriate response.

The following are examples of legal command structures:

GO NORTH
TAKE BOTTLE
READ MAP

Comments: PIRATES ADVENTURE gives the adventurer a unique experience and a nice change from the popular ADVENTURE game, which so many have played.

885-8001 HDOS Screen Editor (SE) $25.00

Introduction: **SE** is a "screen oriented" text editor for HDOS users. It is modeled after the UCSD Pascal editor.

Requirements: This program is written in **Z80** code and will therefore, execute only on H89's or H8 Z80 CPU's. It requires HDOS, with a minimum of 32K of memory and one drive.

Author: Victor A. Abell

Program Content: The program status and buffer size are displayed on the 25th line of the H89 or H19 terminal. The program uses a series of control characters (e.g. the "CTRL C") to manipulate SE. The following is a list of some of the features or commands of SE:

Append (continue text insertion)
BACK SPACE
Control Characters in Input
Cursor Movement
Edit Direction
Environment (display text buffer)
Find (locate specific text)
Insert (text)
Leading Spaces (modify same)
Margin (set same)
Page (display previous or next page)
Replace (text)
RETURN (end a text line)
Spaces and Tabs
Write (text to disk)

Comments: SE uses control characters to direct the cursor and commands through the text. It does not use the "arrow keys" of the keypad or the special function keys of the H19 terminal.

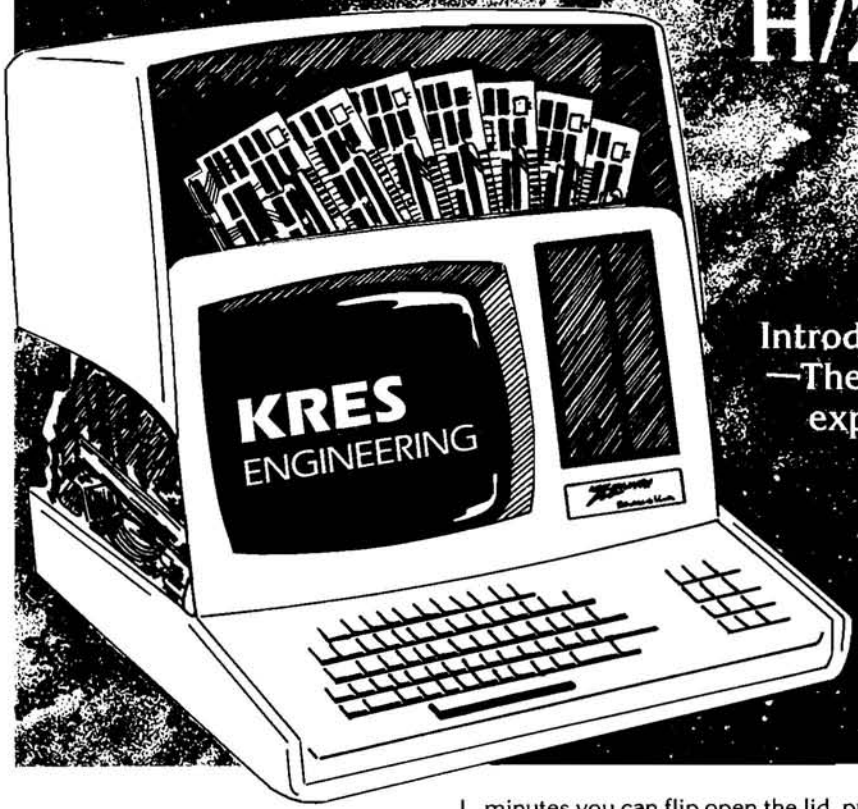
The SE documentation consists of a 30 page manual of step by step operations. SE is recommended to anyone familiar with the UCSD Pascal Editor.

885-8003 BHBASIC to MBASIC (BHTOMB) $25.00

Introduction: BHTOMB is an MBASIC program which will convert any Disk Benton Harbor BASIC (BHBASIC) program to Microsoft BASIC (MBASIC).

Vectored to 22

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



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

MAKING A GOOD COMPUTER BETTER

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

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

INTRODUCING THE KRES Expansion™

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

EXPANDED PORT DECODING

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

QUIET, FAST OPERATION

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

ROOM FOR 16K MEMORY

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

MORE TO COME

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

GET MORE INFORMATION TODAY

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

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

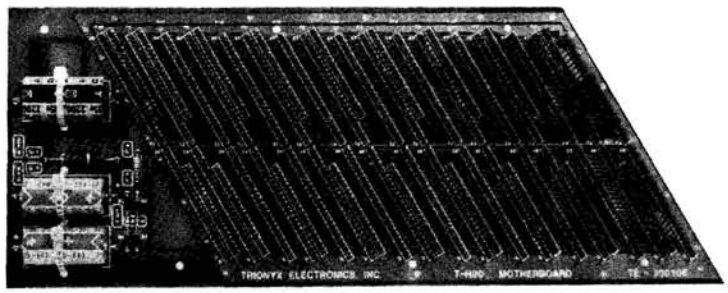
Introducing –

THE NEW H8* COMPUTER

* H8 is a Registered Trademark of the Heath Company

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

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



Fully Assembled Motherboard - $250.00

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

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

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

PC Board $ 75.00

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

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

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

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

Bus Termination Card - Complete Kit $29.50

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

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

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

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

TRIONYX ELECTRONICS, INC.

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

DG IS Heath/Zenith

...with the Fu

H8 PRODUCTS

The Most Extensive Line of Hardware Support for The H8®

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

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

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

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

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

64K STATIC RAM–PROM BOARD

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

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

H8/H89 Software

Software Support for HDOS 2.0

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

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

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

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

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

Computer Support
iture Built-In

NEW SUPER 89

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

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

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

Compatible with all currently available Heath/Zenith hardware devices.

D·G ELECTRONIC DEVELOPMENTS CO.

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

Vectored from 16

Requirements: BHTOMB requires HDOS (1.6 or later) on an H8/H19 or H89 with 48K memory and one disk drive. CP/M will not be applicable with this version. The H19 terminal (H89) is required due to the various ESCape sequences. MBASIC versions 4.7 or 4.82 are compatible.

Author: Robert A. Gregory

Program Content: When BHTOMB executes it displays each line on the terminal as it is converted. All converted lines are saved to a selected file or device.

Any statement not compatible with MBASIC, e.g. LNO, CHAIN, FREEZE, etc., will be listed for further correction by the user, while statements, such as OPEN (which requires an extension), will prompt for any additional information on the 25th line.

This program has converted a 649 line BH BASIC program, with only 16 lines to be converted manually, in about 14 minutes.

Comments: It is impossible to convert all statements of BH BASIC to usable MBASIC code due to the many differences between the two interpreters. It is an excellent package for doing many of the tedious conversion steps that normally would be done by hand.

885-8004 Universal DUMP (UDUMP) $35.00

Introduction: The Universal Disk Utility Manipulation Program (UDUMP) will access and edit any file or sector on a disk.

UDUMP overcomes hardware constraints of 5 1/4" hard sectored mini-floppy disks by using the HDOS device drivers for actual disk access. This permits its use with any directory device supported by a conventional HDOS device driver, regardless of the hardware configuration.

Vectored to 25

SMASH THE '89's 64K RAM LIMIT!

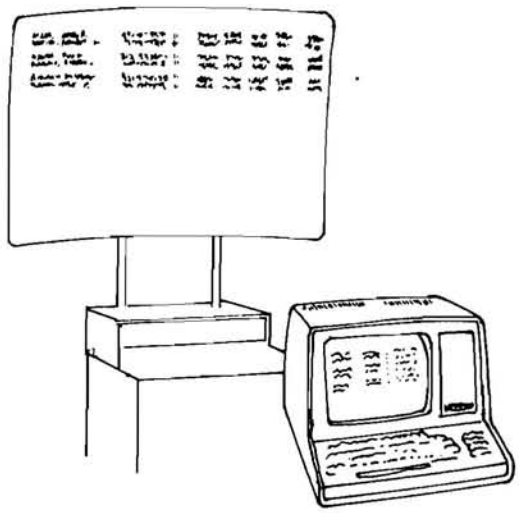
128K RAM BOARD — $595

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.

CP/M and MP/M II are registered trademarks of Digital Research, Pacific Grove, CA.

MAGNOLIA MICROSYSTEMS

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

Sunflower Software, Inc.

Dept. R2
13915 MIDLAND DRIVE
SHAWNEE, KANSAS 66216

RICK KERBEL, *Pres.* **(913) 631-1333**

WORD PROCESSING

WORDPRO

A text editor/formatter heavy on features. When used with "DIABLO", expect proportional word spacing, subscripts and exponents. etc. Takes the work out of word processing. **HDOS $59.95**

SPELGUD/SUPERFOG

Flags misspelled words on your documents or correspondence. / Calculates suggested grade level of comprehension.

HDOS or CP/M $45.95

★ ★ ★ ★ ★ ★ ★ ★

EDUCATIONAL

STATES

Teaches state recognition from graphic maps. capitols, nicknames, flowers, birds, trees and largest cities. An excellent geography learning aid.

DUALFORMAT $24.95

MATH

Aids the younger student in all four arithmetic functions. Will solve his school-work or generate arithmetic or word problems for him. Uses large graphic numerals.

DUALFORMAT $24.95

★ ★ ★ ★ ★ ★ ★ ★

GAMES

FIREMAN

FIREMAN is a video action game which requires a good eye and quick reflexes while still being simple enough for the youngsters to enjoy. **DUALFORMAT $19.95**

DATA MANAGEMENT

MAILMATE

Prints Labels, Letters or Lists. Merge Label data into your letters. A host of user definable options. Map driven output lets you determine output formats.

HDOS or CP/M $39.95

SYØ : NAMELIST

Single drive Name/Address/Phone manager. Operates in turn-key fashion. No experience necessary.

HDOS/MBASIC $19.95

UTILITIES

DUALPORT

Modifies HDOS to permit a "2nd" device to duplicate the I/O of your console terminal. Eleven functions give added control and security. **HDOS $39.95**

MAGIC MENU

Converts any system to turn-key operation. Menu calls BASIC/ABS/ or any other program from command mode and all programs automatically return to menu. Even executes multiple command strings. Menu headings are user defined.

HDOS $29.95

S.A.P.

5 utilities (CP/M, only the first 2) Search And Print, MSMAP, PHLAGS, DIRSORT, HMOD2.

HDOS or CP/M $19.95

DISKSORT

Sorts any disk file (external) regardless of size. Select FIXED, VARIABLE, OR MULTIPLE RECORD records. **HDOS $29.95**

USER

is a modification to HDOS that will support Dynamic Disk Partitioning. This function is similar in nature to the USER function supported in CP/M. With 8 User Areas you can define a file to be resident in any/some/all of the Areas. **HDOS $29.95**

Call Us!

"We're User Friendly"

COMPILERS/ TRANSLATORS

S-BASIC

Translates a "structured" Basic Program into B.H. or Microsoft. An excellent introduction to other Structured Language.

HDOS $49.95

CONTROL

A Data Entry/Base Language compiler. Supports data input editing and Random data retreival and update. Write business applications in a fraction of the time.

HDOS $99.95

APPLICATIONS

APPLIED STATISTICS

Complete statistics package for the engineer or student. All statistical functions including histogram.

DUALFORMAT/MBASIC $59.95

Executive Appointment SecretarY

Keeps track of appointments, meetings and schedules. Full screen display for each day. Ideal for Doctors, Lawyers, or anyone with a busy schedule.

HDOS or CP/M $39.95

SYØ:BILLING

Complete balance forward billing system designed for one drive. Accomodates 40 Accounts. Transactions are entered quickly and accurately. **HDOS/MBASIC $39.95**

Ordering Instructions: Sunflower Software can be seen and/or purchased at your nearest Heathkit Electronics Center. When ordering direct, include $2.00 for postage and handling. Kansas residents include sales tax. Specify operating system when applicable.

Attention Programmers: Sunflower Software is interested in high quality software for the H8/H89. Let us evaluate your original programs for our internationally marketed line.

Send For Free Catalog

Vectored from 22

Requirements: UDUMP requires HDOS, version 2.0, on an H8/H19 or H89 with 48K of memory and one of the following drives: H17, H37, H47, H77 or any number of non-Heath disk configurations. The documentation contains a table of the hardware configurations tested with UDUMP.

NOTE: UDUMP will operate on HDOS versions 1.5 and 1.6 for H-17-1, 5 1/4" hard sectored minifloppies. The H19 (H89) is required due to the use of the ESCape codes and function keys.

Author: Michael J. Cogswell

Program Content: UDUMP uses the function keys in conjunction with a menu, to guide the user in making selections. When data is displayed on the screen, the function keys are defined on the 25th line.

The following is a brief explanation of the function keys from the main "menu":

f1 - FILE EDIT -- This option prompts for the filename to edit. It will edit the file sector by sector. The function keys (options) of the File Edit mode are displayed on the 25th line.

See **"Editing with UDUMP"** below.

f2 - Track/Sector Edit -- This option permits the editing of disk sectors on an absolute track and sector basis without regard to normal HDOS file structures. With this option the disk need not be mounted or even mountable. This is the only option which will edit a corrupted disk.

The mode options are displayed on the 25th line.

f3 - Mount/Dismount/Cat -- This key will display a sub-menu of disk utilities. By selecting one of the function keys (in this mode) the user can mount, dismount, reset and catalog disks without exiting UDUMP.

f4 - Remove Flags -- This option gives the user a sub-menu to remove any desired flag from every file on the disk. UDUMP will remove various combinations of the Lock, Write protect and System flags. It also will remove **ALL** flags from the selected disk.

f5 - Configure UDUMP -- This option sets the desired defaults for UDUMP. The function keys once again are used to enter the desired changes.

Editing with UDUMP -- As explained above the File Edit and Track/Sector ("f1" and "f2", respectively) are the two modes for editing a disk sector. Editing may be done in OCTAL, HEX or ASCII formats.

The cursor will be placed under the first byte of data in the sector and the program will be ready to accept data. The cursor may be moved from byte to byte by using the cursor control keys of the keypad. After the desired changes have been made, the edit may be saved to the disk or exited without making changes.

Searching with UDUMP -- While in an Edit mode, a search may be made on a string of characters. The string may be entered in OCTAL, HEX or ASCII formats.

UDUMP will begin searching from the current cursor position within the sector and search forward to the end of the sector. If the string is not found, the next sector will be read and the search will continue until either the string is located or UDUMP reaches the end of the file or disk.

If the string is located, the cursor will be placed under the first byte of the string. UDUMP will be in Edit mode, replacing the new data directly over the old. After making the changes, the search may be repeated.

Assembling with UDUMP -- One of the most unique features of UDUMP is the ability to assemble any valid 8080 mnemonic directly into a sector of the disk. The 25th line will display the **"Assemble Mode:"** prompt. The restrictions of this mode are detailed in the documentation of UDUMP.

Comments: This is one of the finest utilities that HUG has ever released. In essence, UDUMP is a disk "screen editor".

The documentation contains the complete explanation of UDUMP, and should be used as the source for details of the program.

The following program, also written by Mr. Cogswell, is included on the disk with UDUMP:

FAKEMNT -- This program is a small utility program which will mount a disk that has a corrupt structure. The program will recover **most** of the files on the disk.

When attempting to recover all files on a disk, **at least one file will be truncated.**

It is possible, if the disk structure is badly damaged, that no files can be completely recovered. Please note, these programs work with disks that are logically damaged, not physically damaged.

Comments: The details of FAKEMNT are contained in the documentation that comes with the UDUMP disk package.

EOF

HUG Product List

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

Part Number	Description		Selling Price	REM #
CASSETTE SOFTWARE (H8 and H88)				
885-1008	Volume I Documentation and Program Listings (some for H11)		$ 9.00	
885-1009	Tape I Cassette		$ 7.00	
885-1013	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 SUBMIT, CLIST, FDUMP, ABSDUMP, etc.		$18.00	
885-1061	TMI Cassette to Disk H8 only		$18.00	
885-1062	Disk VIII H8/H89 (2 disks) MEMTEST, DUP, DUMP, DSM		$25.00	
885-1063	Floating Point Disk H8/H89		$18.00	
885-1065	Fixed Point Package H8/H89		$18.00	10
885-1075	HDOS Support Package H8/H89		$60.00	
885-1077	TXTCON/BASCON H8/H89		$18.00	
885-1079	HDOS Page Editor		$25.00	15
885-1080	EDITX H8/H19/H89		$20.00	
885-1082	Programs for Printers H8/H89		$20.00	
885-1083	Disk XVI RECOVER, etc.		$20.00	11
885-1089	MACRO, CTOH, and misc Utilities		$20.00	20
885-1090	Misc. HDOS Utilities CCAT, HPLINK, AH, MBSORT, etc.		$20.00	22
885-1092	RDT Debugging Tool H8/H89		$30.00	14
885-1095	HUG SY: Device Driver HDOS 2.0		$30.00	18
885-1098	H8/HA-8-3 Color .ABS/.ASM		$20.00	19
885-1099	H8/HA-8-3 Color in Tiny Pascal		$20.00	19
885-1105	HDOS 2.0 Device Drivers MX-80, Paper Tiger, Clock, etc.		$20.00	24
885-1116	HDOS Z80 Debugging Tool		$20.00	27
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
PROGRAMMING LANGUAGES				
885-1038	WISE on Disk H8/H89		$18.00	
885-1042	PILOT H8/H89		$19.00	
885-1059	FOCAL-8 H8/H89		$25.00	13
885-1078	HDOS Z80 Assembler		$25.00	21
885-1085	PILOT Documentation		$ 9.00	
885-1086	Tiny Pascal H8/H89		$20.00	13
885-1094	HUG Fig-Forth H8/H89 2 Disks		$40.00	18
BUSINESS, FINANCE AND EDUCATION				
885-1047	Stocks H8/H89		$18.00	
885-1048	Personal Account H8/H89		$18.00	
885-1049	Income Tax Records H8/H89		$18.00	
885-1051	Payroll H8/H89		$50.00	
885-1055	Inventory H8/H89	*	$30.00	
885-1056	Mail List H8/H89	*	$30.00	
885-1070	Disk XIV Home Finance H8/H89		$18.00	
885-1071	SmBusPkg III 3 Disks H8/H19 or H89	*	$75.00	17
885-1091	Grade and Score Keeping	*	$30.00	14
885-1097	Educational Quiz Disk H89 or H8/H19	*	$20.00	18
DATA BASE MANAGEMENT SYSTEMS (DBMS)				
885-1107	Amateur Radio Logbook and TMS		$30.00	23
885-1108	Telephone/Mail Info. System	*	$30.00	23
885-1109	Retriever (2 disks)		$40.00	23
885-1110	Autofile		$30.00	23
885-1115	Aircraft Navigation DBMS H8/H89		$20.00	25
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	

Vectored to 28

Vectored from 15

Dear HUG,

There is a small bug in the EDIT program of HDOS. If you are editing a large file, you will get a double echo of a control-C at the end of an insert command which puts something into the file near the beginning. This occurs because the processing of a deferred control-C returns to the wrong point in the interrupt service routine. The processing must be deferred until the moving around of the entire text buffer following the insertion is completed. Thus, if a long file is being edited it is likely that you will be able to hit control-C before the move is complete, and the processing is deferred.

To fix this bug, you need only patch two bytes of the EDIT.ABS file to return to the processor after the code which echos the control-C. For Version 2.0 of HDOS, the appropriate patch is as follows:

SECTOR	BYTE	OLD	NEW
8	B8	FC	01
8	B9	22	23

The sector number and byte addresses are based on using the HUG Disk VIII patch program, where the first sector of the file is 0, second file sector is 1, etc. The contents are given in hexadecimal. In octal, they would be OLD 374 042, NEW 001 043.

If you are using a SUPERZAP or another program which allows patching by program address, the appropriate addresses are 053060A and 053061A.

In terms of the HDOS Source Listings HOS-1-SL, 595-2466-01, Vol. 4, what has been done is change the JNZ INTRPT on line 1982 to a jump to the instruction LDA CCFLG in line 478, following the echo code in lines 476 and 477. This bypasses the second echo when the deferred control-C is actually honored.

This bug drove me just about nuts thinking that my terminal had key bounce causing a second control-C to be entered. Finally I followed it through in the source listings and found out what happended.

Very truly yours,

Robert H. French
1919 Rochelle Ave. Apt 1831
Districts Heights, MD 20747

Dear HUG,

I would like to add a word to Roy Reichert's article on "The BASIC Memory Grabber", REMark #25, February '82. In his article, Mr. Reichert explored the consequences of branching out of FOR/NEXT loops. A simple way of forcing the FOR/NEXT loop to go the full range is to set the loop index equal to the range. Witness:

```
1000 FOR T=1 TO 10
     ....
1500    IF (exp) THEN T=10: GOTO 2000
     ....
2000 NEXT T
```

This technique is logically equivalent to branching out of the loop and also ensures that the FOR/NEXT stack is restored!

Bob Hall
2603 Wayne Street
Bellevue, NE 68005

Dear HUG,

After reading a previous letter to HUG concerning the use of a POKE statement in MBASIC to allow output to screen or printer using only a PRINT statement (while under CP/M), I decided to see if other values could be used for other devices. Here are some:

```
X=169  for LPT: at PORT 340Q
X= 41  for TTY: at PORT 320Q
X=105  for CRT: at PORT 350Q
```

A simple MBASIC program can then be written to change the I/Obyte:

```
 10 REM I/OBYTE POKE SUBROUTINE
 20 P=PEEK(3)
 30 LINE INPUT "Output to Screen (S),@
    Printer (P), or TTY (T) <S>: ";X$
 40 IF X$="P"THEN X=169:GOTO 70
 50 IF X$="T"THEN X=4  :GOTO 70
 60 X=P
 70 ON ERROR GOTO 100
 80 POKE 3,X
 90 PRINT"TEST!"
100 POKE 3,P:REM RETURN CONTROL TO CRT:
110 END
```

Since this program changes the current CRT: device, other values may be substituted to change the current LST: device in order that an LPRINT command may be issued for I/O.

NOTE: This program does not alter any subsequent STAT DEV: commands as the default device(s) will still be reported. Care must be taken to return control of the CON: to the CRT:, hence the ON ERROR GOTO statement.

Vectored to 31

Vectored from 3

really does exist. We can expect to see this product sometime this year. He suggested that priorities were required to ensure the successful introduction of the new computer and that some projects were slipping since the manpower was unavailable to move every project out the door. He did, however, indicate that strong support for both the H8 and the H89 would continue (I, for one, was glad to hear that remark!).

Attending the WCCF is a real experience. I have 80 pounds of literature that I haven't even gotten to yet. I'm still a little groggy from "jet-lag", however, this is one event that is worth every second that you are there. I hope to meet even more of you next year!

BE:

National Conference Filling Up

Bill Parrott had a pretty good idea. He mentioned on MicroNet one night that we should have a HUG get-together.

Those of you who have registered will be receiving a 'care package' any day that includes a hotel registration card and your very own guaranteed-to-get-you-in name tag. (You pick up the badge holder at the door.) That's part of what you get for your fifteen bucks and without it you get used Kool-Aid for lunch and dinner...in the parking lot...and are not eligible for the freebie door prizes...As Bob mentioned last month, the registration fee goes up to $20 June 1. The main reason for this, is to encourage early registration so we can get a handle on the probable attendance and to minimize a big hassle for our Nancy and Margaret Friday evening and early Saturday morning. (Seems they wanna fool around.) So, get the cards and letters coming. When we reach 1000 attendees--sorry.

The agenda hasn't changed much from what was published last month, except that some of the 'hopeful' speakers are now firm, and by coincidence, two of them have the same first name, Gordon. As for the tour of Heath, President Bill Johnson will cover much of the site in his slide presentation, but a formal trek to Benton Harbor seems too ambitious. We'll have to be prepared to play that one by ear and see how many want to drive over. (Two hours.) We'll arrange something, if there aren't too many.

Exhibit space is all gone. Some of your friends showing their goodies are: D-G Electronics, Sunflower Software, Commsoft, Software Toolworks, Evryware Software, Kres Engineering, Software WIZARDRY, Keyboard Studio, Generic Software, Logic Labs, Buss/Sextant, Micro-Interface, Ultimeth Inc. and Trionyx. And each of them will contribute something to the door prize pot!

==========

...it's going to be a great weekend!

==========

Answers to questions you've asked.

Yes, if your spouse, youngsters, or pets plan to attend the gathering and attend the dinner, that's another $15 each. No, the room rate does not apply at the downtown Hyatt. No, you don't have to stay at the Hyatt Regency O'Hare. Of course, you have to be present to win a prize!

:JB:

Vectored from 26

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

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

MISCELLANEOUS

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

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

```
Vectored from 4

;-----          CLEAR SCREEN - FROM CURSOR TO END OF SCREEN
CLREOS:   MOV  #MSG4B-1,R2
     MOVB #'0,3(R2)
     JMP  SCREEN
;-----------------------------------------------------------------
;-----          CLEAR LINE - ALL
CLRLIN:   MOV  #MSG4B-1,R2
     MOVB #'2,3(R2)
     JMP  LINE
;-----          CLEAR LINE - FROM BEGINNING OF LINE TO CURSOR
CLRBOL:   MOV  #MSG4B-1,R2
     MOVB #'1,3(R2)
     JMP  LINE
;-----          CLEAR LINE - FROM CURSOR TO END OF LINE
CLREOL:   MOV  #MSG4B-1,R2
     MOVB #'0,3(R2)
     JMP  LINE
SCREEN:   MOVB #'J,4(R2)
     JMP  PRINT4
LINE:     MOVB #'K,4(R2)
     JMP  PRINT4
PRINT4:   .PRINT    #MSG4B
     RTS  PC
;-----------------------------------------------------------------
;-----          PRINT THE FOLLOWING IN NORMAL OR REVERSE VIDEO
NRMVID:   .PRINT    #MSG4C
     RTS  PC
REVVID:   .PRINT    #MSG4D
     RTS PC
;-----------------------------------------------------------------
;-----          25TH LINE ENABLED
L25OFF:   MOV  #MSG5A-1,R2
     MOVB #'1,4(R2)
     JMP  PRNT5A
;-----          KEYCLICK OFF
CLKOFF:   MOV  #MSG5A-1,R2
     MOVB #'2,4(R2)
     JMP  PRNT5A
;-----          BLOCK CURSOR ON
BLKCUR:   MOV  #MSG5A-1,R2
     MOVB #'4,4(R2)
     JMP  PRNT5A
;-----          CURSOR OFF
CUROFF:   MOV  #MSG5A-1,R2
     MOVB #'5,4(R2)
     JMP  PRNT5A
;-----          25TH LINE ENABLED
L25ON:    MOV  #MSG5B-1,R2
     MOVB #'1,4(R2)
     JMP  PRNT5B
;-----          KEYCLICK ON
CLKON:    MOV  #MSG5B-1,R2
     MOVB #'2,4(R2)
     JMP  PRNT5B
;-----          LINE CURSOR ON
LINCUR:   MOV  #MSG5B-1,R2
     MOVB #'4,4(R2)
     JMP  PRNT5B
;-----          CURSOR ON
CURON:    MOV  #MSG5B-1,R2
     MOVB #'5,4(R2)
     JMP  PRNT5B
PRNT5A:   .PRINT    #MSG5A
     RTS  PC
PRNT5B:   .PRINT    #MSG5B
     RTS  PC
;=================================================================
;-----          CURSOR UP-DOWN-FORWARD-BACKWARDS
CURUP:    MOV  #MSG5C-1,R2
```

```
      MOVB #'A,5(R2)
      JMP  TRANS
CURDWN:   MOV  #MSG5C-1,R2
      MOVB #'B,5(R2)
      JMP  TRANS
CURFWD:   MOV  #MSG5C-1,R2
      MOVB #'C,5(R2)
      JMP  TRANS
CURBAK:   MOV  #MSG5C-1,R2
      MOVB #'D,5(R2)
      JMP  TRANS
TRANS:    MOV  (R5)+,R4
      BIC  #177400,R4
      JSR  PC,DECODE
      MOVB RO,3(R2)
      MOVB R1,4(R2)
      .PRINT    #MSG5C
      RTS  PC
;=====================================
;-----     GO TO A GIVEN X-Y COORDINATE
GOTOXY:   MOV  #MSG8-1,R2
      MOV  (R5)+,R4
      BIC  177400,R4
      JSR  PC,DECODE
      MOVB RO,3(R2)
      MOVB R1,4(R2)
      JSR  PC,DECODE
      MOVB RO,6(R2)
      MOVB R1,7(R2)
      .PRINT    #MSG8
      RTS  PC
;=====================================
DECODE:   MOV  #'0,R0
      MOV  #'1,R1
      TST  R4
      BGT  1$
      RTS  PC   ;IF NO ARG, HOME
;
1$:   DEC  R4
      MOV  (R5)+,R3
      CMP  R3,#-1
      BNE  2$
      RTS  PC   ;SET DEFAULTS 0,1
;
2$:   MOV  @R3,R1
      CMP  R1,#1
      BGE  3$
      MOV  #'1,R1
      RTS  PC
;
3$:   CMP  R1,#80.
      BLE  4$
      MOV  #'8,R0
      MOV  #'0,R1
      RTS  PC
;
4$:   CLR  R0
      DIV  #12,R0
      ADD  #60,R0
      ADD  #60,R1
      RTS  PC
;=====================================
MSG2:     .ASCII    <33>/</<200>
      .EVEN
MSG3A:    .ASCII    <33>/[n/<200>
      .EVEN
MSG3B:    .ASCII    <33>/[s/<200>
      .EVEN
MSG3C:    .ASCII    <33>/[u/<200>
      .EVEN
MSG3D:    .ASCII    <33>/[z/<200>
```

```
      .EVEN
MSG4:      .ASCII     <33>/[2h/<200>
      .EVEN
MSG4A:     .ASCII     <33>/[2l/<200>
      .EVEN
MSG4B:     .ASCII     <33>/[2J/<200>
      .EVEN
MSG4C:     .ASCII     <33>/[0m/<200>
      .EVEN
MSG4D:     .ASCII     <33>/[7m/<200>
      .EVEN
MSG5A:     .ASCII     <33>/[>1h/<200>
      .EVEN
MSG5B:     .ASCII     <33>/[>1l/<200>
      .EVEN
MSG5C:     .ASCII     <33>/[01A/<200>
      .EVEN
MSG9:      .ASCII     <33>/[00;00H/<200>
      .EVEN
      .END
```

* underlined l=small L EOF

Heath Related Products

I am pleased to announce a new product for the H-8 computer with the color graphics board.

I have a disk with two (2) games for sale along with instructions to hook up a simple inexpensive contact closure-type joystick to the color graphics board to operate the games.

Game #1 is Color Raiders, which is an exciting visual target game which simulates the motion of a jet fighter as it goes through space. It has eight (8) different skill levels.

Game #2 is Star Battle, which is a game that has the player destroy star debris as it floats across the screen. The object is to destroy all the debris before a chunk of it hits you, or the alien saucer comes out to attack your ship. The difficulty of the game increases with the score.

Both games and instructions on the HDOS 2.0 compatible disk sell for twenty-nine ninety five ($29.95).

Thank you,

Daniel Guernsey
219 West River Road North
RD#1, Box 12
Fulton, NY 13069

HOME PHONE: (315) 598-3264
WORK PHONE: (315) 592-5156

New Orleans General Data Services is about to start production of a Color Graphics Board for the H-89/Z-90 computers. The board will display 16 colors with a resolution of 192 by 256 dots using the TMS-9918A Video Display Processor. Besides graphics, the board hosts a variety of other features, including:

-- an AY-3-8910, 3-channel Programmable Sound Generator with two 8-bit parallel I/O ports;

-- an ADC-0808, 8-channel, 8-bit A/D Converter;

-- an 8253, triple 16-bit Counter-Timer; and

-- an 8259 Priority Interrupt Controller.

In addition, the following are available as options:

-- a 9511A Floating-Point Math Chip;

-- two AD7542, 12-bit D/A Converters with a precision reference; and

-- a SC-01A Votrax Speech Synthesizer.

For complete information contact:

New Orleans General Data Services, Inc.
7230 Chadbourne Drive
New Orleans, LA 70126
(504) 241-9495

Vectored from 27

Additionally, I would like to expand to the CP/M MBASIC RESET command.

If it is necessary to reset more than one drive, try logging onto another drive besides drive A:. RESET will read new directories from the Boot drive and the Current drive (also any other active drives):

```
A>B:
B>A:MBASIC
Ok
RESET
```

Regards,

Brian Brown
PO Box 791
Benton Harbor, MI 49022

Local HUG News

Wayne Emanuel, who lives at RR#1 Sulphur Springs, AR 72768, which is in the extreme northwest corner of Arkansas would like contact with other Huggies in his area. Those interested can contact Wayne at (501) 298-3442 or write.

Ruben Rodriguez and some friends who own Heath computers are very interested in organizing a HUG in Puerto Rico. Those interested should contact Ruben at (809) 854-4360 or write PO Box 1135 Manati, PR 00701.

The ATHUG (Atlanta HUG) meets on the first and third Thursday of each month at the Heathkit Center at 5285 Roswell Road Atlanta, GA 30342 from 7:00 pm to 9:00 pm. Contact person is Leon Trulove at 880 Walnut Circle Marietta, GA 30060 - (404) 436-3677. Operation hours on their Bulletin Board are 6:00 pm to 8:00 am daily and all day Sunday. The BB number is (404) 252-4342.

The SHUG (South Jersey HUG) has formed and meets the first Wednesday each month at the Heathkit Center at 1013 State Hwy. 35 Ocean, NJ 07712 at 7:30 pm. Their contact person and acting secretary is James J. Jones Jr. 272 Stocktons Bridge Road Pemberton, NJ 08068. Bulletin Board operates 24 hours at (201) 775-8705. There are approximately 71 members and they have a newsletter on the BB.

Alfred K. Carr, Sec/Tres/Contact Person of the Denver HUG reports the following changes in their group: BB# (303) 423-3224 - 24 hours. Newsletter/library exchange - group size is 120. They have been successful in gaining an exemption from the federal income tax. Other HUGs may contact DENHUG for information how this was done.

Information has come to us regarding a new computer group from Paul Pennington, contact person for the CSRA Computer Club PO Box 284 Augusta, GA 30903. They meet at 7:30 pm the fourth Monday of each month at Campus Computer Systems 3830 Washington Road Martinez, GA 30907. Contact Paul at home (404) 860-2934 or work 863-4819.

The SHUG (Sacramento HUG) has formed and meets the second Wednesday each month at 7:30 pm at the Heathkit Center at 1860 Fulton Avenue Sacramento, CA 95825 - (916) 662-7220. Contact person/Sec is Gloria Stewart. Group size - 35.

If you were not aware, there is a new Heathkit Center located at 8262 Arlington Expressway Jacksonville, FL 32211. Paul V. Sullivan of 1134 Montego Rd West Jacksonville, FL 32216 is interested in starting a HUG in this area. Those interested should either write to Paul or phone him at (904) 724-4343.

Ted G. Benglen, II reports that a very unpleasant situation has cropped up in the form of a fraudulent HUG. It seems a party in Abilene Texas has used HUG to fraud suppliers. The formation of the new Abilene users group was announced in the August 1981 Issue of REMark. A Mr. D.R. Kight of PO Box 1651 Abilene TX 07604 with the nickname "Aug" announced the new HUG. Recently, the phone number has been changed and is unlisted. Ted has been able to confirm that the party still lives in Abilene, but all efforts to contact them have been ignored. He has also been told that the Abilene HUG no longer exists, and really never did. It is unfortunate that this person saw fit to use a HUG as a means of fraud.

```
Vectored from 13
     000.173  00123   00000000 01111011

     ---------------------------------

     123.316  21454    01010011 11001110

     TO EXIT HIT "H" FOR "HDOS" OR ANY

     OTHERKEY TO CONTINUE TO FIRST STEP.
```

The display will show each number entered as well as the results in split-byte octal, as well as in decimal and in binary. At the end of the display, the program will again print the message "TO EXIT HIT "H" FOR "HDOS" OR ANY OTHER KEY TO CONTINUE." The program will now wait for a response to this prompt. To exit the program, simply hit the letter "H", and to do another addition, or subtraction, you just hit any other key on the keyboard (space bar, return etc.). There is nothing at all preventing the changing of any or all of these messages, so please feel free to do so if desired.

This program has proven to be a great help over the months, and its continued use is expected for many years. This program will save hours of work when trying to determine the results between two split-byte octal numbers or between a split-byte octal and a decimal number. There are many uses for such a program, and its presence on disk will greatly enhance working in split-byte octal.

EOF

Linking Sound-89 to MBASIC

Gordon Leon Duke, PHD
3444 46th Avenue South
Minneapolis, MN 55406

I have just installed Bob Ellerton's noisemaker on my H89 (Re: REMark 23 page 18). This is an excellent modification, and it was cheap and easy. Bob's directions are very clear and I have been thrilled with the quality and variety of sounds that it is possible to produce.

The only thing missing was the ability to run the noisemaker under MBASIC. Terry Jensen rescues us with his article "VARPTR in MBASIC" (Re: REMark 21 page 23). Using Terry's directions, I was able to convert the machine code required to run the noisemaker into a MBASIC array. The following program will execute the warble subroutine (beginning line 60 of Bob's assembler program).

```
100 DEFINT I-Z
110 DIM U0(18)
120 FOR I=0 TO 18: READ U0(I): NEXT I
130 DATA &H280E, &H8006, &H043E, &HF2D3, &H3D78
140 DATA &HC200, &H0000, &H003E, &HF2D3, &H3D78
150 DATA &HC200, &H0000, &HC205, &H0000, &HC20D
160 DATA &H0000, &H023E, &HF2D3, &H00C9
170 Z=VARPTR(U0(0))
180 U0(6)=Z+9: U0(11)=Z+19: U0(13)=Z+4: U0(15)=Z+2
190 DEF USR0=VARPTR(U0(0))
200 PRINT USR(0)
210 STOP
```

The first step in creating the MBASIC program was to extract the relevant subroutines from the assembler code given in Bob's article. I chose the warble subroutine and rewrote the code as given below.

```
016 050          WARBLE    MVI  C,050Q
006 200          WO2       MVI  B,200Q
076 004          WO1       MVI  A,004Q
323 362                    OUT  362Q
170                        MOV  A,B
075              ONLOOP    DCR  A
302 ??? ???                JNZ  ONLOOP
076 000                    MVI  A,0
323 362                    OUT  362Q
170                        MOV  A,B
075              OFFLOOP   DCR  A
302 ??? ???                JNZ  OFFLOOP
005                        DCR  B
302 ??? ???                JNZ  WO1
015                        DCR  C
302 ??? ???                JNZ  WO2
076 002                    MVI  A,02
323 362                    OUT  362Q
311                        RET
```

To convert the machine code, I used the format of the table given in Terry's article.

Numbered Address	Stored In Memory	Bytes Reversed	Hex Values	MBASIC Variable
RELATIVE	016	050 016	280E	U0(0)
1	050			
2	006	200 006	8006	U0(1)
3	200			
4	076	004 076	043E	U0(2)
5	004			
6	323	362 323	F2D3	U0(3)
7	362			
8	170	075 170	3D78	U0(4)
9	075			
+ 10	000	302 000	C200	U0(5)
11	302			
12	???	??? ???	????	U0(6)
13	???			
14	076	000 076	003E	U0(7)
15	000			
16	323	362 323	F2D3	U0(8)
17	362			
18	170	075 170	3D78	U0(9)
19	075			
+ 20	000	302 000	C200	U0(10)
21	302			
22	???	??? ???	????	U0(11)
23	???			
24	005	302 005	C205	U0(12)
25	302			
26	???	??? ???	????	U0(13)
27	???			
28	015	302 015	C20D	U0(14)
29	302			
30	???	??? ???	????	U0(15)
31	???			
32	076	002 076	023E	U0(16)
33	002			
34	323	362 323	F2D3	U0(17)
35	362			
36	311	000 311	00C9	U0(18)
37	000			

(NOTE: "+" INDICATES A REQUIRED NOP.)

Now all that remained was to compute the relative jumps. There are four labelled addresses used in the assembler code. (The first label, "warble", is not used in this version.) The first byte of the instruction found at these four addresses is underlined in the above table. Column one of that table gives the values to be added to VARPTR(U0(0)) in line 190 of the MBASIC program. The following table summarizes this process.

LABEL	FIRST BYTE	RELATIVE ADDRESS	MBASIC VARIABLE REFERENCING THE ADDRESS
WO2	006	Z+2	U0(15)
WO1	076	Z+4	U0(13)
ONLOOP	075	Z+9	U0(6)
OFFLOOP	075	Z+19	U0(11)

Other noisemaker routines can be converted in a similar manner. However, the conversion of assembler code to a MBASIC routine is tedious. Is anyone aware of a MBASIC program that could help? Ideally such a program would read 8080 mnemonics and produce the MBASIC code.

I hope this helps other HUG members to enjoy a little sound from their H89's.

EOF

PATCH Mysteries Revealed

In the last issue of REMark, I presented a patch to the PIP program in HDOS that causes it to list file names copied or deleted. I provided instructions for installing the patch with PATCH.ABS, a program supplied with HDOS. If you're like me, you probably started experimenting with PATCH as soon as you started using HDOS, and wondered why it asked for a "Patch ID" if you tried to patch certain files, and what the ID was. And now you are probably wondering how I got the codes for that PIP patch, and why you were able to make the patch even though PIP.ABS was write protected.

The PATCH Codes

If you try to patch an HDOS system file with PATCH, it will ask you to enter some codes before it allows you to make the patch. It does this because system files have an extra sector appended to them which is called the Patch History Table (PHT). The Patch program detects the presence of the PHT sector, and enters the system patch mode. The codes you must enter are actually encrypted numbers. The first code (Patch ID) is simply a one byte number, which is the patch number. The six letters that you must enter for it represent the number itself and a two-byte checksum, making three bytes, or two letters for each byte. The Prerequisite Code is next, which consists of 14 letters. They represent 5 bytes and a two-byte checksum. The 5 bytes are actually a field of 40 bits, and each bit represents a patch number. The least significant bit of the first byte represents patch no. 0 and the most significant bit represents patch no. 7. The least significant bit of the next byte represents patch no. 8, etc. The numbers decoded from the letters you enter are compared with a 5-byte field in the PHT, and if any bits you set are not also set in the PHT, PATCH informs you that the patches whose numbers correspond to the unmatched bits are missing, and returns you to the File Name prompt. At the end of the patch session, the prerequisite field in the PHT is updated by setting a bit corresponding to the Patch ID no. that you entered first.

The Patch ID and Prerequisite Code are used for the user's own bookkeeping more than for protection, because PATCH will allow you to enter the ID and Prerequisite codes that represent zero regardless of any previous patches. The codes that I used in the PIP patch represent zero, and can be used when you patch any system file. The code that you enter <u>after</u> the patch, the Patch Check Code, is the one that provides the most protection. This code consists of 8 letters, representing 4 bytes. The first two bytes are a checksum of all the patches entered, and the last two are a checksum of the checksum. This code will be different for each patch, so you can't "fake" it. If you are entering a patch such as the one presented in the last REMark, and you mistype just one character in either the patch or the Check Code, PATCH will not accept the entry, and you must start over (Control-C) if the error is in the patch. The PHT is not affected by the Check Code.

The PHT contains a two byte header at the beginning (376Q,001Q) that tells PATCH that it is a PHT. This is followed by the ASCII characters "PHT/HSG", then by the date on which the PHT sector was created in HDOS ASCII format (DD-MMM-YY). Based on the comments in the source listing, the original intent was to update the date each time a patch is made, but that is not done in the current version of PATCH. Following the date is a byte intended to be a patch counter, but it is also not updated. Next is the prerequisite field, and this is the only part of the PHT that is updated when you do a patch. The rest of the PHT sector is just filler.

The System User's PATCH

If you examine the source listing for PATCH, which is in volume 3 of the HDOS 1.6 or 2.0 source listings, you will see at the very beginning a line that reads

```
.SYS.   EQU     1
```

If you set this label equal to 0 instead of 1 and re-assemble the source, you get the System User's version of PATCH. To explain what this does, first let me explain that there is a switch that you can specify when you enter a file name with PATCH. The switch is /DISP followed by a colon and a number, which is a sector offset to be added to the sectors PATCH looks for. In other words, if the offset is 1, PATCH looks for the header information (to determine if the file is binary) in the second sector of the file, instead of the first, where it is. I am not sure what good that is, but it lets you know that PATCH has the ability to process switches. When you assemble the System User's version, you get two more switches, /PHT and /CHECK. The first, /PHT, is used to write a PHT sector onto a file that does not already have one. The second switch, /CHECK, gives you the codes that are required when you patch a file with a PHT. Here is how it works.

After you enter the file name (followed by /CHECK), it asks you to enter an octal number. You must enter a single octal byte consisting of 3 digits, which is the Patch ID. It will then encode this number and its checksum into 6 letters and print them out. You must then enter the 6 letters yourself as you would with standard PATCH. Then it asks for more octal numbers. This time you enter 5 three digit octal numbers, and you may separate them with spaces if you wish. These 5 numbers are the prerequisite field. After you enter them, PATCH will print out the code letters as before, and you must again enter the letters. Then you are prompted for an address, and you can make your patch. When you are finished, and you type Control-D in response to the Address prompt, PATCH will print out the checksum of the patches in octal. You must enter the checksum number as printed, and PATCH will encode it into letters, which you must also enter. The letter codes that are printed out during this process can be used by someone with standard PATCH to make the same patch.

Patching Write Protected Files

When you work on a file with PATCH, it opens the file for Read (.OPENR) and looks for a PHT sector. If it does not find one, it closes the file and re-opens it for Update (.OPENU) so the patch can be made. If the file has the Write Protect flag set, HDOS will not allow it to be opened for Update, and an error message appears. However, if the file you are patching has a PHT sector on it, PATCH alters the HDOS Channel Table (more on that later) so that the file appears to have been opened for Update instead of for Read. This allows the file to be written to even if it has the Write Protect flag set. The code that does this magic is a little subroutine called CFU, and it looks like this:

```
**      CFU - KLUDGE FILE TO UPDATE STATUS
*
*       FILE MUST BE OPEN ON CHANNEL 0

CFU     LHLD    S.CFWA  GET CHANNEL FWA
        CALL    $HLIHL  GET CHANNEL 0 ADDR
        LXI     D,IOC.FLG  IO FLAG OFFSET
        DAD     D       POINT TO FLAGS
        MVI     A,FT.OW+FT.OU  NEW FLAGS
        ORA     M       ADD IN OLD FLAGS
        MOV     M,A     CHANGE FILE STATUS
        RET
```

The label S.CFWA is found in ESINT.ACM, and the labels IOC.FLG, FT.OW, and FT.OU are in IOCDEF.ACM. These .ACM files are provided with HDOS 2.0. In previous issues of REMark, we have presented patches to PATCH that cause it to ignore the PHT so you can patch system files without entering the codes. Unfortunately, these patches also removed the ability to patch write protected files, because CFU was never called. Now we can rectify that situation with a patch that causes PATCH to ignore the PHT while causing it to use CFU for every file. Start with a fresh copy of PATCH.ABS from your HDOS distribution disk, and enter the following patch (what you type is in bold print).

```
PATCH Issue #50.06.00

File Name? PATCH
Patch ID? IFOJIC
Prerequisite Code? IFBEIADPGEFFCF

Address? 42231
042231 = 312/303
042232 = 244/^D      (Control-D typed)
Address? 42263
042263 = 247/257
042264 = 304/^D
Address? 44055
044055 = 076/303
044056 = 000/354
044057 = 377/047
044060 = 046/^D
Address? ^D
Patch Check Code? DLMIAGPD

PATCH Issue #50.06.00.

File Name? ^D
```

The Channel Table

The Channel Table is an area in HDOS memory where information about I/O channels is stored. Any time you open a disk file for Read, Write, or Update, whether at the assembly level or in a high level language, you use an I/O channel. Each I/O channel has space alloted to it in the Channel Table where information about the file you are working with is stored. One of the items of information stored is how the file was opened, and we have seen how it is possible to alter that information and change a file from Read status to Update status. The file IOCDEF.ACM shows the other information that is stored in the Channel Table for each channel. If you look at this file, you will notice that the directory entry of the file you are working with is stored there. This is the entire directory entry as it appears in DIRECT.SYS. HUG member Bill Parrott told me that when you open a file for Write or Update and close it again, HDOS writes the information in the Channel Table back into DIRECT.SYS. This opens up a world of possibilities, such as a FLAGS program that can clear Lock flags under unmodified HDOS. I wrote such a program, and it is listed following this article (bet you didn't think that an article on PATCH would end up with a FLAGS program). Operation of this program is similar to the regular FLAGS program except that you hit RETURN at the File Name prompt to get back to HDOS instead of Control-D. If you

want to leave a directory entry unaltered
after you get to the New Flags prompt,
type Control-C.

```
                                                    HEATH ASM #104.06.00
                                                    14-Apr-82  Page    1
                                PS:

                        00001   *       SFLAGS.ASM
                        00002   *
                        00003   *       SUPER FLAGS PROGRAM -- CHANGES S, W, AND L
                        00004   *       FLAGS, AND DOES NOT USE THE .CHFLG SCALL
                        00005   *
                        00006   *       BY P. SWAYNE, HUG  12-APR-82
                        00007
042.200                 00008           XTEXT   HOSEQU          THESE .ACM FILES ARE
042.200                 00036           XTEXT   HOSDEF          STANDARD ONES SUPPLIED
000.207                 00102           XTEXT   DIRDEF          WITH HDOS 2.0
000.027                 00129           XTEXT   ESINT
041.123                 00244           XTEXT   IOCDEF
000.200                 00282   DIF.SYS EQU     10000000B       SYSTEM FLAG
000.100                 00283   DIF.LOC EQU     01000000B       LOCK FLAG
000.040                 00284   DIF.WP  EQU     00100000B       WRITE PROTECT FLAG
030.211                 00285   $HLIHL  EQU     30211A          LOAD HL THROUGH HL
031.136                 00286   $TYPTX  EQU     31136A          TYPE TEXT (ADDR IN STACK)
                        00287
042.200                 00288           ORG     USERFWA
                        00289
042.200  076 377       00290   SFLAGS  MVI     A,-1
042.202  377 055       00291           SCALL   .CLEAR          ALLOW WORK ON THIS FILE
042.204  315 136 031   00292           CALL    $TYPTX          SIGN ON
042.207  012 104 151   00293           DB      12Q,'Directory Flag Altaration Program,'
042.252  040 126 145   00294           DB      ' Version 1.0',12Q,212Q
042.270  076 003       00295   ENTER   MVI     A,3
042.272  041 000 000   00296           LXI     H,0
042.275  377 041       00297           SCALL   .CTLC           CLEAR CONTROL-C
042.277  315 136 031   00298           CALL    $TYPTX
042.302  105 156 164   00299           DB      'Enter File Name (or RETURN to exit):',240Q
042.347  041 027 044   00300           LXI     H,FNAME         PUT FILE NAME HERE
042.352  377 001       00301           SCALL   .SCIN           GET FIRST CHARACTER
042.354  332 352 042   00302           JC      *-2
042.357  376 012       00303           CPI     12Q             RETURN?
042.361  302 001 043   00304           JNZ     GNAMEO          IF NOT, GET REST OF NAME
042.364  257           00305   EXIT    XRA     A
042.365  377 000       00306           SCALL   .EXIT           ELSE, EXIT
042.367  377 001       00307   GNAME   SCALL   .SCIN           GET A CHARACTER
042.371  332 367 042   00308           JC      *-2
042.374  376 012       00309           CPI     12Q             END OF NAME?
042.376  312 020 043   00310           JZ      GOTNAM          YES
043.001  376 004       00311   GNAMEO  CPI     4               CONTROL-D?
043.003  312 364 042   00312           JZ      EXIT            IF SO, LEAVE
043.006  376 041       00313           CPI     ' '+1           SPACE OR LESS?
043.010  332 367 042   00314           JC      GNAME           IF SO, SKIP
043.013  167           00315           MOV     M,A             ELSE, STORE CHARACTER
043.014  043           00316           INX     H               MOVE TO NEXT LOCATION
043.015  303 367 042   00317           JMP     GNAME           GET MORE CHARACTERS
043.020  066 000       00318   GOTNAM  MVI     M,0             TERMINATE NAME WITH 0
043.022  257           00319           XRA     A               USE CHANNEL 0
043.023  062 014 044   00320           STA     OPFLAG          CLEAR OPEN FILE FLAG
043.026  021 021 044   00321           LXI     D,DEFALT
043.031  041 027 044   00322           LXI     H,FNAME
043.034  377 042       00323           SCALL   .OPENR          TRY TO OPEN FILE
043.036  332 360 043   00324           JC      ERR             COULDN'T DO IT
043.041  076 003       00325           MVI     A,3
043.043  062 014 044   00326           STA     OPFLAG          FLAG FILE OPENED
043.046  041 310 043   00327           LXI     H,CTRLC
043.051  377 041       00328           SCALL   .CTLC           SET UP CONTROL-C EXIT
043.053  052 352 040   00329           LHLD    S.CFWA          GET FWA CHANNEL TABLE
043.056  315 211 030   00330           CALL    $HLIHL          GET CHANNEL 0 ADDRESS
043.061  345           00331           PUSH    H               SAVE IT
043.062  021 004 000   00332           LXI     D,IOC.FLG       OFFSET TO CHANNEL FLAGS
043.065  031           00333           DAD     D
043.066  076 014       00334           MVI     A,FT.OW+FT.OU   FLAGS FOR UPDATE, WRITE
```

```
043.070  266          00335          ORA     M               ADD TO EXISTING FLAGS
043.071  167          00336          MOV     M,A             UPDATE CHANNEL STATUS
043.072  341          00337          POP     H               RESTORE CHANNEL ADDRESS
043.073  021 041 000  00338          LXI     D,IOC.DIR+DIR.FLG
043.076  031          00339          DAD     D               MOVE TO FILE FLAGS
043.077  345          00340          PUSH    H               SAVE THIS ADDRESS
043.100  315 136 031  00341          CALL    $TYPTX
043.103  103 165 162  00342          DB      'Current Flags =',240Q
043.123  176          00343          MOV     A,M             GET FLAGS
043.124  041 015 044  00344  TFF     LXI     H,TFFA          POINT TO FLAG STRING
043.127  207          00345  TFF0    ADD     A               MOVE MSB INTO CARRY
043.130  365          00346          PUSH    PSW             SAVE FLAGS
043.131  176          00347          MOV     A,M             GET ASCII FLAG CHAR
043.132  322 137 043  00348          JNC     NOFLAG          THIS FLAG NOT SET
043.135  377 002      00349          SCALL   .SCOUT          PRINT FLAG
043.137  043          00350  NOFLAG  INX     H               MOVE TO NEXT CHAR
043.140  361          00351          POP     PSW             RESTORE FLAGS
043.141  267          00352          ORA     A               MORE FLAGS?
043.142  302 127 043  00353          JNZ     TFF0            IF SO, PRINT THEM
043.145  076 012      00354          MVI     A,12Q
043.147  377 002      00355          SCALL   .SCOUT          PRINT NEW LINE
043.151  315 136 031  00356  GFLAGS  CALL    $TYPTX
043.154  116 145 167  00357          DB      'New Flags =',240Q
043.170  006 000      00358          MVI     B,0             CLEAR FLAG ACCUMULATOR
043.172  377 001      00359  GFLAG0  SCALL   .SCIN           GET A CHARACTER
043.174  332 172 043  00360          JC      *-2
043.177  376 012      00361          CPI     12Q             END OF ENTRY?
043.201  312 271 043  00362          JZ      SETFLGS         IF SO, SET FLAGS
043.204  346 137      00363          ANI     5FH             CAPITALIZE ENTRY
043.206  016 200      00364          MVI     C,DIF.SYS       ASSUME S FLAG
043.210  376 123      00365          CPI     'S'
043.212  312 263 043  00366          JZ      ADDFLGS
043.215  016 100      00367          MVI     C,DIF.LOC       ASSUME L FLAG
043.217  376 114      00368          CPI     'L'
043.221  312 263 043  00369          JZ      ADDFLGS
043.224  016 040      00370          MVI     C,DIF.WP        ASSUME W FLAG
043.226  376 127      00371          CPI     'W'
043.230  312 263 043  00372          JZ      ADDFLGS
043.233  315 136 031  00373          CALL    $TYPTX
043.236  007 111 154  00374          DB      7,'Illegal Flag.',212Q
043.255  315 352 043  00375          CALL    CLRBUF          CLEAR INPUT BUFFER
043.260  303 151 043  00376          JMP     GFLAGS          GET MORE FLAGS
043.263  171          00377  ADDFLGS MOV     A,C             GET NEW FLAG
043.264  260          00378          ORA     B               MOVE IT
043.265  107          00379          MOV     B,A             INTO FLAG ACCUMULATOR
043.266  303 172 043  00380          JMP     GFLAG0          GET MORE FLAGS
043.271  341          00381  SETFLGS POP     H               GET ADDRESS OF FLAGS
043.272  176          00382          MOV     A,M             GET CURRENT FLAGS
043.273  346 037      00383          ANI     -1-DIF.SYS-DIF.LOC-DIF.WP  REMOVE OLD ONES
043.275  260          00384          ORA     B               ADD NEW ONES
043.276  167          00385          MOV     M,A             UPDATE FLAGS
043.277  257          00386          XRA     A
043.300  377 046      00387          SCALL   .CLOSE          CLOSE FILE
043.302  332 360 043  00388          JC      ERR             CAN'T CLOSE
043.305  303 270 042  00389          JMP     ENTER
                      00390
                      00391  *       PROCESS CONTROL-C
043.310  315 136 031  00392  CTRLC   CALL    $TYPTX
043.313  136 103 212  00393          DB      ' C',212Q
043.316  061 200 042  00394          LXI     SP,USERFWA      FIX STACK
043.321  315 352 043  00395          CALL    CLRBUF          CLEAR INPUT BUFFER
043.324  377 007      00396          SCALL   .CLRCO          CLEAR CONSOLE
043.326  052 352 040  00397  CREAD   LHLD    S.CFWA          GET FWA CHANNEL TABLE
043.331  315 211 030  00398          CALL    $HLIHL          GET CHANNEL 0 ADDRESS
043.334  021 004 000  00399          LXI     D,IOC.FLG       OFFSET TO CHANNEL FLAGS
043.337  031          00400          DAD     D
043.340  076 363      00401          MVI     A,-1-FT.OW-FT.OU
043.342  246          00402          ANA     M               REMOVE .UPDATE STATUS
043.343  167          00403          MOV     M,A             SET NEW STATUS
043.344  257          00404          XRA     A
```

```
043.345  377 046      00405            SCALL   .CLOSE          CLOSE FILE
043.347  303 270 042  00406            JMP     ENTER           TRY AGAIN
                      00407
                      00408  *         CLEAR INPUT BUFFER
043.352  377 001      00409  CLRBUF    SCALL   .SCIN           GET A CHARACTER
043.354  322 352 043  00410            JNC     CLRBUF          CONTINUE UNTIL EMPTY
043.357  311          00411            RET
                      00412
                      00413  *         PROCESS ERRORS
043.360  365          00414  ERR       PUSH    PSW             SAVE ERROR CODE
043.361  315 136 031  00415            CALL    $TYPTX
043.364  007 105 122  00416            DB      7,'ERROR -',240Q
043.375  361          00417            POP     PSW             RESTORE ERROR
043.376  046 012      00418            MVI     H,12Q           END MESSAGE WITH NEWLINE
044.000  377 057      00419            SCALL   .ERROR          PRINT ERROR
044.002  072 014 044  00420            LDA     OPFLAG
044.005  267          00421            ORA     A               FILE STILL OPEN?
044.006  302 326 043  00422            JNZ     CREAD           YES, CHANGE TO READ, CLOSE
044.011  303 270 042  00423            JMP     ENTER           TRY AGAIN
                      00424
                      00425  *         CONSTANTS AND STORAGE
                      00426
044.014  000          00427  OPFLAG    DB      0               OPEN FILE FLAG
044.015  123 114 127  00428  TFFA      DB      'SLWC'          FLAG STRING
044.021  123 131 060  00429  DEFALT    DB      'SYOABS'        FILE DEFAULTS
044.027               00430  FNAME     DS      20              FILE NAME GOES HERE
                      00431
044.053  000          00432            END     SFLAGS

                                                                        EOF
Vectored from 5
```

Well, I see the bottom of the page coming up. Perhaps this has been of some help. One sure way of testing your program for **BULLET PROOF** testing is to give it to my neighbor. She can break it in record time. Better yet, I'll send you my neighbor and you can **BULLET PROOF** her! Cheers! :JB:

```
10000 'Return upper case for lower of Q$
10010 N%=INSTR("abcdefghijklmnopqrstuvwxyz",Q$)+96
10020 IF N%>96 THEN Q$=CHR$(N%-32)
10030 RETURN
```

EOF

NATIONAL HUG CONFERENCE
Registration Card

Name _____

Address _____

City _____ State ____ Zip _____

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

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

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

|__| Send me Hotel reservation information.

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

You may copy this page.

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

Heath
Users'
Group
Hilltop Road
St. Joseph MI 49085

POSTMASTER: If undeliverable,
please do not return.

885-2028