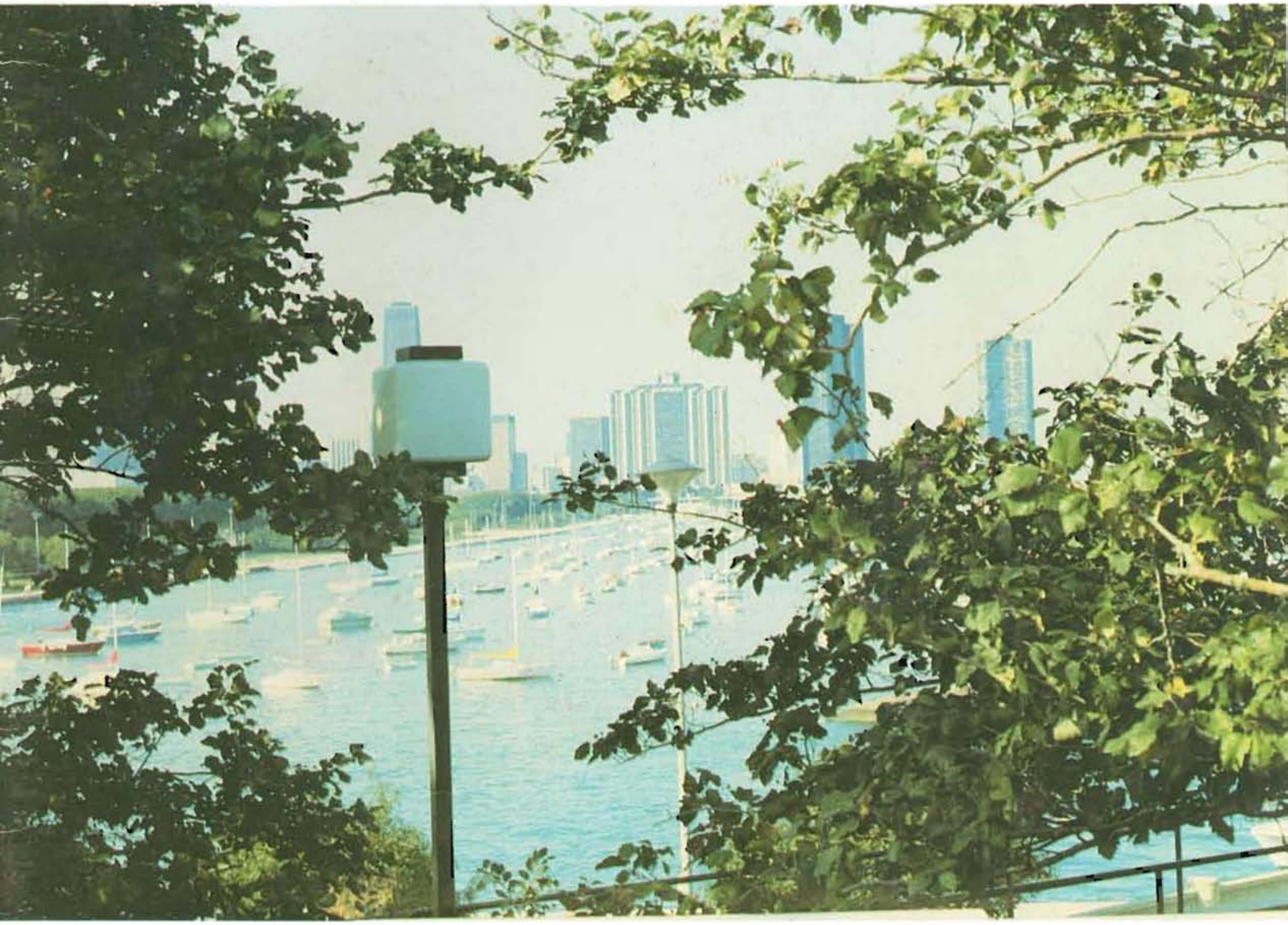
REMark

Issue 17 • May 1981



Official magazine for users of Heath computer equipment.

on the cover . . . .

Chicago Marina on Lake Michigan —

Photo by Gerry Kabelman, Again!

on the stack

>CAT

A Re-visit with "The Source" 3

A Kiss for Assembly Programming 4

Using the LSI-11/23 with the H11 7
Thomas G. Barnum

Super RAM Test for the H89 8
Herbert Drake

HUGBB via MicroNET 12

Buggin' HUG 14

New HUG Software 16

HUG Product List 17

Wither STAT? 18

Comments on the Pascal Language 21
Fred Pospeschil

Color Graphics in Tiny Pascal 24

Tiny BASIC Tricks 28

Who's on First? 29

Local HUG News 29

HUG Meets at the WCCF 30

"PRINT USING" for B H BASIC 30

H8 to H11 interface 31

Electronic Billboard 32
Roy S. Reichart

"REMark" is a HUG membership magazine published ten 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 available at $2.50 plus 10% handling and shipping. Requests for magazines mailed to foreign countries should specify mailing method and add the appropriate cost.

Send payment to:

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

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

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

HUG Manager and Editor .... Bob Ellerton
Assistant Editor and
Software Developer .... Patrick Swayne
HUG Secretary .... Nancy Strunk
Software Developer .... Gerry Kabelman
HUG BB .... Terry Jensen

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

REMark

A Re-Visit with the Source

It was a cold afternoon when two shadows appeared at my office door and introduced themselves as representatives of the "SOURCE". Being new to the Heath Users' Group, I invited these individuals for a little discussion (something I don't often do with a sales person, let alone two). A small bell went off somewhere in the back of my head and a hollow voice said "APPROACH WITH CAUTION"! As we began our discussion the normal name exchange occurred. I met Jim Clark, Vice President of Sales and Jim Rutt the Regional Marketing Representative for my area. JC: began the discussion with a brief intro to the "SOURCE" which I was to learn is a telecommunication system similiar to MicroNet. The bell in my head then rang clear. The "SOURCE" was the system with slow response time that Jim Blake had mentioned. JB: also suggested that the system was overloaded and that the personnel didn't quite have "their stuff together" yet. Soooo, I sat quiet and nervous as the VP presented his case. The first indication that changes had occurred and that the "SOURCE" was indeed alive and well came when Jim Clark indicated that there were some problems. I, for one, appreciate the straight forward approach and I cleaned my ears for a good listen.

Mr. Clark explained that the "SOURCE" had been acquired by "Readers Digest". He then handed me a couple of books that later would become a valuable and constant companion. He went on to describe that the acquisition by "Readers Digest" had allowed them to obtain four PRIME 750 computers, three of which were online at the time. JC: added that these computers were backed by the PDP-10 for periods of peak loads. He pointed out some of the more interesting features offered by the "SOURCE" and listed in the books for my review. As I perused the "SOURCE" manual. The subjects that were covered seemed too good to be true. United Press International, Consumer Information, Science and Engineering Library, New York Times News Summary are just a small portion of what I was to find in the following weeks. JC: then felt that I was in the mood for a demo (I was chomping at the bit actually) so he turned the show over to Jim Rutt who whips through the "SOURCE" like so much butter.

Jim immediatly "tuned-in" UPI for a "keyword" search of some 1,000 stories that had been written that day. He chose "Reagan" for the search and returned 43 stories between 8:00am and 1:00pm. Impressed? Indeed I was! He moved quickly through a thing called "CHAT" and into the AIRSCHED program for home users who wish to make their own flight arrangments! JR: demonstrated the "SOURCE" mail system that allows instant message transfers from one user to another (unlike the EMAIL of MNet, you don't have to wait five hours to get results). Jim continued with a couple of other demos and described in detail some more of the "neat" features. My mind was elsewhere, however. I started thinking of our membership and that brought me to the BIG question....COST? Well, the bomb dropped (I thought)! The sign-up fee for an individual was $100 bucks. I nearly fell on the floor! But, keeping a "cool" image, I asked a few more questions. Here is a brief comparison of what I was told and the existing charges for MNet.

1. PRIME TIME CHARGES	
MNET	22.50/HR.
SOURCE	15.00/HR.
2. NON-PRIME TIME CHARGES	
MNET	5.00/HR.
SOURCE	4.25/HR.
3. AFTER MIDNIGHT	
MNET	5.00/HR.
SOURCE	2.75/HR.
4. SURCHARGE FOR TYMNET (PRIME)	
MNET	10.00/HR.
SOURCE	0.00/HR.
5. SURCHARGE FOR TYMNET (NON-PRIME)	
MNET	2.00/HR.
SOURCE	0.00/HR.

NOTE: SOURCE pays communications charges if you do not have a direct number. This however, does not include a long distance phone call toll.

(vectored to page 11)

A Kiss for Assembly Programming

THE LAST IN A THREE PART SERIES

FROM REMark ISSUES #15 AND #16

In the previous issues of REMark we covered a simple Assembly Language program that would be runnable, when completed, on both the H8 and H89 computers. We discussed the basics of program layout and construction by first defining the program and following up with an examination of the "logical flow" using the Flow Chart. Next, we took a brief look at the necessary "tools" required to enter our program into the computer. In this section, we examined the operation of the HDOS EDITor and the commands we needed to enter the ASM file and save the file to the disk storage area and on a printer (optional).

>CAT ???

To ensure that everybody is still with me, let's do a catalog of our working disk. Boot your working disk. After you have received the HDOS prompt (>), type "CAT" followed by a carriage RETURN. If everything went well in our description of using the EDITor, you will find on your disk directory a file called BEEP.ASM. This file is the Assembly Language Source Code which we will use to create the runnable program we have been working to construct. We can take a look at the source code by typing the following:

```
>TYPE BEEP.ASM    (RETURN)
```

The computer will respond by displaying the program we typed in Issue #16 of REMark (fig. A). If everyboby is still with me, let's proceed!!

HDOS ASM........

HDOS, the Heath Disk Operating System, contains a program know as the Assembler or on our directory "ASM.ABS". ASM will be used to "build" the finished program we have named "BEEP". As suggested by the name (Assembler), ASM is used to "change" the Assembly Language Source Code (.ASM) into a new file known as the Absolute Binary File (.ABS). If you examine your HDOS disk, you will notice that some of the files such as EDIT, ASM, and BASIC are followed by the "extension" .ABS. This extension means that the file is directly executable from the prompt (>). If, for instance, you type BASIC after the prompt, BASIC is loaded into the computer and is ready to use. You must remember, however, that BASIC "IS" a running program. And, it is merely a larger version of our tiny "BEEP". You may ask yourself..."OK, if BASIC is similiar to "BEEP" where is the ASM file like the one we typed for "BEEP"!? The answer is a little difficult, but boils down to BIG $$$$$! Most software is supplied with the ABS (runnable) file only to prevent tampering and piracy. To obtain a copy of the ASM file or source can be very expensive. Further, many programmers wish to protect their work by keeping the "source file" secret. Anyway, what we will examine will be most useful with HUG software since almost all of the disks are supplied with both the source (ASM) and runnable (ABS) files. This means that you will be able to try the ASSEMBLY procedure on programs other than "BEEP".

MAGIC ... BEEP.ASM = BEEP.ABS

Well, it is time to FINALLY create our finished masterpiece. Follow me closely for a minute and we will see the result of our long hours of work. At the HDOS prompt (>) type:

```
>ASM    (RETURN)
```

The computer will respond by "loading" the HDOS Assembler or ASM. Keep in mind, that like BASIC, ASM is a running program. ASM has for a prompt a "*" just like B.H. BASIC. When you receive the prompt, type:

```
*BEEP.ABS=BEEP.ASM    (RETURN)
```

Your disk drive will "clunk" around a bit and shortly you should receive a message something like this:

```
00020 Statements Assembled
xxxxx Bytes Free
No Errors Detected
```

NOTE: xxxxx in the above message denotes the remaining memory in your computer. Also, "No Errors Detected" will occur only if the BEEP.ASM file was constructed properly from Issue #16......of course, we all typed the thing right.....RIGHT!?

If the above procedure went as described, we should now check the disk directory and see what "magic" our computers performed. At the HDOS prompt (>), type:

```
>CAT     (RETURN)
```

Somewhere on our directory we should find two files, one with the name BEEP.ASM, the other with the name BEEP.ABS. The computer has now constructed the required "runnable" file known as BEEP.ABS. There is one more file which we can obtain known as the ".LST" file. We will discuss this in a minute. But first, let's try "BEEP" and see what is does.

"BEEP", as described, should cause the "bell" in either the H8 or the H89 to ring five times. Further, our program should provide a "wait" so that we can count the five rings. Lastly, the program should return us to the HDOS prompt so that we may continue our work.

LET'S GO....

After the HDOS prompt, type:

```
>BEEP    (RETURN)
```

Neat! huh!? Our program went into action just by typing the program name. If you continue to type BEEP after the prompt, you will be able to repeat the identical program each time. Now that you are done driving the wife nuts with your "beeper", let's take a look at some of the other features more closely......

MAKING A LISTING

All of us have probably talked of our "listings" in BASIC etc. But, you have also wondered how to read some of the listings provided by PAM-8 for the H8 owners or the MTR-88 for owners of the H89. Well, let's make our own and see if we can't pick up a little additional knowledge!

To begin, type "DELETE BEEP.ABS" at the HDOS prompt......YES, get rid of the work we tried so hard to finish. You should now have only one "BEEP" file and that should be BEEP.ASM. Next, we re-enter the Assembler by typing ASM at the HDOS prompt. We will now attempt to create not only the .ABS file, but the .LST file or listing file as well. Type:

```
>BEEP.ABS,BEEP.LST=BEEP.ASM     (RETURN)
```

Again your drive will "clunk" around and you will receive another message. But, if you now examine the directory, you will find three "BEEP" files. BEEP.ASM, BEEP.ABS, and BEEP.LST will have been created by the Assembler. You can "type" these files to your printer or terminal by using the procedures outlined previously. (Don't try to send the ABS file to your printer unless you are prepared to pick up a lot of paper!!!) Take a look at fig. A as this is an example of the listing file.

```
                                                       HEATH ASM #104.06.00
                                                        No-Date   Page    1

100.000                 00001   START   ORG     100000A
100.000  006 005        00002           MVI     B,FIVE
100.002  315 136 002    00003   LOOP    CALL    HORNO
100.005  026 377        00004           MVI     D,TIME1
100.007  036 377        00005   LOOP2   MVI     E,TIME2
100.011  035            00006   LOOP3   DCR     E
100.012  302 011 100    00007           JNZ     LOOP3
100.015  025            00008           DCR     D
100.016  302 007 100    00009           JNZ     LOOP2
100.021  005            00010           DCR     B
100.022  302 002 100    00011           JNZ     LOOP
100.025  303 100 040    00012           JMP     HDOSP
                        00013   *****
000.005                 00014   FIVE    EQU     005Q
002.136                 00015   HORNO   EQU     002136A
000.377                 00016   TIME1   EQU     377Q
000.377                 00017   TIME2   EQU     377Q
040.100                 00018   HDOSP   EQU     040100A
                        00019   *****
100.030  000            00020           END     START


  00020 Statements Assembled
  32537 Bytes Free
   No Errors Detected
```

WHAT DOES THIS MEAN? ......

You will notice that the listing or .LST is very similiar to the .ASM file we created using the HDOS EDITor. The major difference would be the columns of funny looking numbers to the left. The first vertical column contains the ADDRESSES of the instructions in the computer's memory. On the H8, we could run the program once. Then follow the "run" with a "hard" reset (0 key and alter key pushed together). If we then examine the memory location 100000A we would find the number 006 in the data register. This same procedure can be performed on the H89 by using the "SHIFT and RESET" keys. However, to examine the memory location we would use the <S>ubstitute 100000 followed by a carriage RETURN. This procedure would cause the address (100000) to be printed followed by the contents of that address or memory location (i.e. 006). Incrementing the memory locations on your computer would result in the three digit OCTAL numbers of the second, third, and fourth columns in the listing to appear in order of the memory addresses (+ key on the H8 or the spacebar on the H89).

```
100000 = 006
100001 = 005
100002 = 315
100003 = 136
etc.
```

Now we can see that the ASM (Assembler) changed our symbolic .ASM file to a series of numbers that the computer understands as instructions and data. Although the computer only understands 0's and 1's, the "MONITOR" converted these numbers to OCTAL for easier reading for us! Without going into great detail, the bottom portion of the listing is as described earlier, the "EQUATE TABLE" that gives us an OCTAL number in place of the words used to define numeric values (i.e. FIVE EQU 005 or 000.005 in the left column).

A COUPLE MORE THINGS......

If you have been relating to all of the information contained in this and previous articles you should now be able to experiment with our program to:

1. change the number of "rings" that the bell sounds by changing the "FIVE" in the "B" register.

2. change the delay or "WAIT" by changing the value of either the "D" or "E" register (maybe both).

3. edit an .ASM file using the HDOS EDITor (remember, this applies to any .ASM file .... even those mysterious things called "patches" that appear in REMark occasionally.)

4. use the ASM (Assembler) to create the .ABS or runnable file and the .LST or listing file.

5. use the ASM (Assembler) to make a .ABS file or .LST file from almost any .ASM file.

I hope that some of you have benefited from this series. For some of you this may have been your first exposure to Assembly Language Programming. I hope you have as much fun with it as I have had trying to describe it. Doc Campbell is doing some similiar work that will take you a little further down the road with this fascinating and very powerful area of computer usage. If you would like to see more articles like this mini-series please let us know. Further, if you now have interest in learning more you may consider the course produced by Heath known as the EC-1108 which is much more detailed. One last thought, the EDITor and the Assembler can be used to "patch" programs as described in REMark when the "source" is given. On a scratch disk, play around a bit....this is the only way to become confident. Also, try assembling some of the .ASM files on any disk in your library and see what happens!! You may surprise yourself!!

TNX BE:

DOC'S COURSE

Doc Campbell just sent me a copy of his new Assembly Language Programming Mini-Course. Doc indicates that he is offering his manuscript for $15.00 for those interested. He calls the course "GETTING STARTED WITH HDOS AND ASSEMBLY LANGUAGE PROGRAMMING". His text is designed as a primer for the beginner and relieves much of the "pain" when getting acquainted with this area of computer programming. Doc indicates that he will ship this information (36 pages) by first class mail. This is a great little primer that can be obtained by writing to Doc at the following address:

William (Doc) Campbell, M.D.
855 Smithbridge Road
Glen Mills, PA 19342

Thanks Doc! For your continued and valuable support of our users.

BE:

Using the LSI-11/23 in the H11

by Thomas G. Barnum
Vice President,
Corporate Research and Developement
Bradley Corporation
P. O. Box 309
Menomonee Falls, WI 53051

EDITORS NOTE: Mr. Barnum makes reference to MicroNOTE in his letter. MicroNOTE is a publication of DIGITAL COMPONENTS GROUP of DIGITAL EQUIPMENT CORPORATION (DEC). References are made to:

#047 Incompatibility Between the REV11 and the LSI-11/23
Page 99 & 100

#053 PDP-11 Family Differences
Page 125 & 126

#055 TABLE 1: LSI-11 vs. LSI-11/23 BUS TIMING DIFFERENCES
Page 142

#078 LSI-11/23 Processor Differences
Page 231 & 232.

Dear HUG,

Here's the letter I promised concerning the use of the 11/23 in the H-11 system.

We first brought the system up in April, 1978. It then consisted of an H-11 computer, H-10 paper tape and H-9 CRT terminal. We have since added the H-27 floppy disc unit, an H-14 printer and an H-19 terminal. All I/O boards are either H-11-5 serial or H-11-2 parallel. We are using the Monolithic RAM Board (32K word).

This system is operating 8 hours per day, 6 days per week and has exhibited almost flawless performance throughout. Only 2 I.C. problems the whole time, although I do mechanical lube and cleaning from time to time.

Our programs grew too lengthy for this machine to handle, so we started looking at the 11/23.

Preliminary studies showed that all necessary lines were in the H11 backplane to service the 11/23 including memory management. The main hazards seemed to be that the Heath (and earlier DEC) bootstrap/diagnostic ROMs used some instructions which are treated differently between the 11/03 and the 11/23. In the Heath bootstrap the offending instructions are:

```
165162/10242
165336/100414
173714/10272
```

(See MicroNote 047,053, and 078.)

Bus timing differences did not appear to be a major element as reasonably fast I.C.'s are used in the Heath interface circuits.

We removed the old processor board and installed the 11/23 CPU with Memory Management and floating point processor. The 11/23 was patched to come up in microODT and a dialog with the terminal indicated that all was well (character echo, memory management using manually entered code). We then added 64K words of DEC memory (MSV11-DD) to the 32K Monolithic memory. In microODT this memory, too was addressable and seemed to function properly.

The H-27 and H-14 cards were installed, and the following boot typed in:

```
1000/5000        1022/100405
1002/12701       1024/105711
1004/177170      1026/100004
1006/105711      1030/116120
1010/1776        1032/2
1012/12711       1034/770
1014/3           1036/0
1016/5711        1040/5000
1020/1776        1042/110
```

this is the DEC RX11/V11 Boot. Typing in 1000G brought the normal HT-11 bootup sequence.

The increase in speed was immediately apparent and resulted in markedly improved performance for the H-27 as well. We must be saving revolutions on the service routines. The DO-loop (or FOR-NEXT loop) routine seems about 2.5 times as fast.

I have noted that I cannot yet use the BASIC.FIS package although 11/03 GIS/FIS commands are supposed to be standard on the 11/23 even without the floating point set.

It is also obvious that I cannot yet use

(vectored to page 28)

Super RAM Test for the H89

by Herbert Drake, Jr.
40 Pikes Peak Drive
San Rafael, CA 94903

This program arose from a stubborn and intermittent dynamic memory chip in the H89. Running the memory diagnostic for hours never revealed a failure, but I was getting daily system crashes. This program moves itself down into the H17 static RAM area, and uses many of the dynamic memory test routines in MTR-88 (those routines didn't move around in the new H88-7 ROM set). This test differs from the Heath memory test in that it "walks" 1 bit through each memory chip one cell at a time, checking all cells for correct values at each step. As a result, it runs much slower than the routine in the ROM, but it gives a much more thorough test. I was able to locate and replace the bad chip, and things have been running fine ever since.

This test tests a 16k block of your memory. It signs on with a menu that allows you to select which block you wish to test, then it moves the actual test into the H17 static RAM and starts the test. You should remove your disks when the test starts, and you will have to re-boot when you are through testing.

```
Walking-one Z-80 Dynamic Ram Test                        HEATH ASM #104.06.00
                                                         01-Apr-81  Page    1

                     * This performs a "walking one" test in H89 dynamic RAM. It
                     * will test any specified 16K segment. The program runs
                     * in the static RAM used normally for the Floppy Disk
                     * Controller.
                     *
                     * Program will continuously display the address of the walk-
                     * ing "one" (actually a 377Q). If an error is encountered,
                     * it will also reveal the location of the error and the
                     * defective memory data.

                     * (by H Drake, June 1980.  Modified by P. Swayne, March 1981)

                     * ROM routines we'll need

007.306              DYMSG    EQU    7306A
003.143              DYASC    EQU    3143A
007.324              MSG.RAM  EQU    7324A
003.160              DYBYT    EQU    3160A
000.307              DYMEM9   EQU    307A

                     * Some Z-80 Instructions

000.030              JR       EQU    030Q
000.040              JR.NZ    EQU    040Q
000.050              JR.Z     EQU    050Q
041.335              LD.IX    EQU    041335A
041.375              LD.IY    EQU    041375A
000.010              EX.AF    EQU    010Q
000.331              EXX      EQU    331Q

                     * MTR88 Definitions

000.362              H88.CTL  EQU    362Q
000.010              BKSP     EQU    10Q
000.177              OP.DC    EQU    177Q           Disc Control OutPort

                     * HDOS ROM Code and SCALL's

030.072              $DADA    EQU    30072A
030.252              $MOVE    EQU    30252A
031.136              $TYPTX   EQU    31136A
000.001              .SCIN    EQU    1
000.002              .SCOUT   EQU    2
```

```
000.006              .CONSL  EQU     6

                     * Absolute addresses after movement

024.027              $WKON02 EQU     24027A
024.111              $WKON10 EQU     24111A
024.125              $WKON11 EQU     24125A
024.135              $WKON12 EQU     24135A

                     * Assembly parameters

040.000              DSTART  EQU     40000A          Dynamic RAM starts here
040.100              WSTART  EQU     40100A          HDOS Warm start
042.200              USERFWA EQU     42200A          Program start

                     * Get starting block from user
                     * Move test into static RAM, and start it

042.200                      ORG     USERFWA

042.200 257          BEGIN   XRA     A
042.201 001 201 201          LXI     B,8181H
042.204 377 006              SCALL   .CONSL          Set console for char input
042.206 315 136 031  REGET   CALL    $TYPTX
042.211 033 105 000          DB      27,'E',0,0,0,0,0,0
042.221 033 131 044          DB      27,'Y$0Super H89 Dynamic RAM Test'
042.257 033 131 046          DB      27,'Y&0Choose block starting address:'
042.321 033 131 050          DB      27,'Y(0040000.......................1'
042.363 033 131 052          DB      27,'Y*0140000.......................2'
043.025 033 131 054          DB      27,'Y,0240000.......................3'
043.067 033 131 056          DB      27,'Y.0300000 (56K machine only)....4'
043.131 033 131 060          DB      27,'Y00Exit to HDOS.................0'
043.173 033 131 062          DB      27,'Y20Enter your choice: |*|',8,8+200Q
043.227 377 001              SCALL   .SCIN           Get response
043.231 332 227 043          JC      *-2
043.234 377 002              SCALL   .SCOUT          Echo response
043.236 365                  PUSH    PSW             Save it
043.237 076 010              MVI     A,8
043.241 377 002              SCALL   .SCOUT          Back up cursor
043.243 361                  POP     PSW
043.244 376 060              CPI     '0'             Zero or less?
043.246 332 206 042          JC      REGET           Bad response
043.251 312 100 040          JZ      WSTART          Zero, return to HDOS
043.254 376 065              CPI     '5'             More than 4?
043.256 322 206 042          JNC     REGET           Bad response
043.261 346 017              ANI     0FH             Remove ASCII bias
043.263 075                  DCR     A               Subtract 1
043.264 041 321 043          LXI     H,BLKTBL        Point to block table
043.267 315 072 030          CALL    $DADA           Find place in table
043.272 176                  MOV     A,M             Get block start
043.273 062 355 043          STA     START           Store starting point
043.276 076 200              MVI     A,200Q
043.300 323 177              OUT     OP.DC           Disable static write protect
043.302 021 325 043          LXI     D,WKON00        Where code starts
043.305 041 000 024          LXI     H,24000A        Where to put it
043.310 001 157 000          LXI     B,LEND-WKON00   No. of bytes to move
043.313 315 252 030          CALL    $MOVE           Move it down
043.316 303 000 024          JMP     24000A          Start the test

                     * Table of starting points

043.321 040          BLKTBL  DB      040Q            Start at 040000A
043.322 140                  DB      140Q            Start at 140000A
043.323 240                  DB      240Q            Start at 240000A
043.324 300                  DB      300Q            Start at 300000Q

                     * Enter dynamic test here

                             LON     G               List all Z80 code
043.325 257          WKON00  XRA     A
043.326 323 362              OUT     H88.CTL         Clear GP Port
```

```

                      * Clear all Dynamic RAM and display header.

043.330  041 377 037          LXI   H,DSTART-1
043.333  043          WKON01  INX   H                 Advance RAM pointer
043.334  175                  MOV   A,L
043.335  264                  ORA   H
043.336  066 000              MVI   M,0               Zero the RAM
043.340  040 371              DB    JR.NZ,WKON01-*-1  Loop until 65K
043.342  041 324 007          LXI   H,MSG.RAM
043.345  335 041 027          DW    LD.IX,$WKON02     Addr of next routine
         024
043.351  303 306 007          JMP   DYMSG

                      * Initialize LWA regiester (address of the walking "one"),
                      * locate the first "one", and initialize the TOP register.

043.354  076 040      WKON02  MVI   A,40000A/256      DEFINE RAM RANGE HERE!
043.355               START   EQU   *-1
043.356  107                  MOV   B,A
043.357  306 100              ADI   100Q
043.361  127                  MOV   D,A               Store TOP
043.362  257                  XRA   A
043.363  117                  MOV   C,A               Do the lsb's here
043.364  137                  MOV   E,A
043.365  075                  DCR   A
043.366  002                  STAX  B                 Store the "one"

                      * Now test the RAM. Initialize (HL)= TOP+1, then check for
                      * (HL) = (BC)

043.367  142          WKON03  MOV   H,D               Reset starting address
043.370  153                  MOV   L,E
043.371  053          WKON04  DCX   H                 Decrement pointer
043.372  175                  MOV   A,L
043.373  271                  CMP   C
043.374  040 076              DB    JR.NZ,WKON07-*-1  Check for RAM zero
043.376  174                  MOV   A,H
043.377  220                  SUB   B
000.072               TEMP    SET   WKON07-*-2
044.000  040 072              DB    JR.NZ,#TEMP

                      * We are where the one's should be.

044.002  003                  INX   B
044.003  075                  DCR   A                 (A) = 377Q
044.004  276                  CMP   M
044.005  040 071              DB    JR.NZ,WKON06-*-1  One's are not there
044.007  064                  INR   M                 Clear
044.010  002                  STAX  B                 and relocate the "one"

                      * Check to see if we are at the end of the RAM.

044.011  257          WKON05  XRA   A
044.012  275                  CMP   L
044.013  040 354              DB    JR.NZ,WKON04-*-1  Loop for next byte
044.015  172                  MOV   A,D
044.016  326 100              SUI   100Q
044.020  274                  CMP   H
044.021  040 346              DB    JR.NZ,WKON04-*-1

                      * A pass was just completed. Update LWA display.

044.023  046 006      WKON08  MVI   H,6
044.025  076 010              MVI   A,BKSP
044.027  375 041 111  WKON09  DW    LD.IY,$WKON10
         024
044.033  303 143 003          JMP   DYASC
044.036  045          WKON10  DCR   H
044.037  040 366              DB    JR.NZ,WKON09-*-1  Loop for 6 backspaces
044.041  170                  MOV   A,B
```

```
044.042  335 041 125 00185          DW     LD.IX,$WKON11
         024
044.046  331         00186          DB     EXX
044.047  303 160 003 00187  DYBYT.  JMP    DYBYT            Display Pass msb's
044.052  331         00188  WKON11  DB     EXX
044.053  171         00189          MOV    A,C
044.054  335 041 135 00190          DW     LD.IX,$WKON12
         024
044.060  030 365     00191          DB     JR,DYBYT.-*-1    and lsb's
                     00192
                     00193  * Check to see if LWA = TOP.
                     00194
044.062  173         00195  WKON12  MOV    A,E
044.063  271         00196          CMP    C
044.064  040 301     00197          DB     JR.NZ,WKON03-*-1  Start at the TOP
044.066  172         00198          MOV    A,D
044.067  270         00199          CMP    B
044.070  040 275     00200          DB     JR.NZ,WKON03-*-1
044.072  030 260     00201          DB     JR,WKON02-*-1    Do whole test again
                     00202
                     00203  * Test for zero at all HL <> BC
                     00204
044.074  257         00205  WKON07  XRA    A
044.075  276         00206          CMP    M
044.076  050 311     00207          DB     JR.Z,WKON05-*-1  It was zero
                     00208
                     00209  * Arrange to display memory content in the error mssg.
                     00210
044.100  126         00211  WKON06  MOV    D,M
044.101  303 307 000 00212          JMP    DYMEM9           Go to error routine.
044.104              00213  LEND    EQU    *
                     00214
044.104  000         00215          END    BEGIN

00215 Statements Assembled
31631 Bytes Free
 No Errors Detected
```

(vectored from page 3)

With this additional information, I asked that I may be signed up. Jim Clark placed the call to his home office and returned a thing called an ID which was and is TCW600 along with a password that would allow me to "get on" the system. Armed with this new "toy" I headed for home to compare the "SOURCE" to MNET.

At first I felt that the "SOURCE" was much slower. But, as I collected the necessary data to make things fly, I realized the power that I had available. Then it happened! My screen jumped and a message appeared.....

TCW112 user 12
HELLO....DO YOU WANT TO CHAT?????

I was taken off guard completely!! Was my computer talking to me? I responded (incorrectly)......YES!.......Nothing happened, so I pulled out the manual JC: had given me earlier and looked frantically for the needed info. I remembered that Jim Rutt had mentioned "CHAT" and quickly looked for this info. I found the correct data which started a conversation and a lasting computerized friendship with Roger, Chris, Julie, and Tom in Connecticut. I have used the "CHAT" mode to directly communicate with other SOURCE people including Ruthie at the "SOURCE" who has made the transition a lot less painful.

The "NEW SOURCE" is a real plus with its' powerful database, good concerned people, and improved response. The initial fee includes a subscription to "SOURCEWORLD" which gives details on the "inside" developments and the future directions of this particular telecommunications system. If you can, I would suggest investigating the "SOURCE" in the near future. Look for further details in REMark as we learn more about the system.

BE:

HUGBB Via MicroNET

Once again the response and activity of the HUG Bulletin Board has increased tremendously. Through the month of March, Bob and I added about 70 new members to the HUGBB. This is fantastic!

There is still no official date when the completion of the new Bulletin Board will be up and running. The first of May is the new projected time of completion. Due to this, once again I am in a bind as what to write about the HUG Bulletin Board. So I will take a different angle and explain how to use MicroNET by using the HUG modem package P/N 885-1043, MCS (Modem Communications System).

In issue 15 of REMark, I said that it is necessary to have a software interface as well as a hardware interface to talk to CompuServe. MCS is probably the most widely used modem package, however, it is not the easiest to learn to operate.

Before we begin, we better clarify the hardware connections on the H8 and the H89 as both of these are slightly different: on the H89 the modem is connected at the DTE connector which is located on the back of the H89. The H8 connection is made on the DTE connector on any free channel on the H8-4 serial I/O board. In either case the interrupt on each machine must be jumpered to level 5. The port assignment to the channel used on the H8-4 serial I/O board must also be jumpered to 330Q.

MCS is such a widely used modem communications package that there are now more patches for it than I can keep a record of. With this in mind, I am forced to explain only the released version from HUG to keep confusion to a minimum. I will also explain the two most popular problems associated between MicroNET and MCS.

To understand MCS as any other piece of software you must play with the package to get a "feel" of what it is intended to do. Many of the commands of MCS may seem a little confusing but with use they will begin to fall into place. Assuming you have all the necessary hardware in its proper order, we will begin our "encounter" with MicroNET.

When you obtain a MicroNET User ID and password you will also receive a telephone number that you can call to access the CompuServe system. This telephone number may be a CompuServe number or a TYMNET number. In issue 15 of REMark, I used TYMNET as the example for getting into MicroNET. This time I will use a CompuServe number. (PLEASE NOTE: TYMNET has changed their login procedure since issue 15 of REMark. When using a TYMNET number enter at login "CIS02" and there is no password.)

Before you dial, bring up MCS on your computer. It will be in "COMMAND" mode. Enter CONversation mode, and dial the CompuServe telephone number. When you hear the continuous high-pitched tone, place the telephone receiver into the modem. You should receive a CNTRL-C prompt from CompuServe and it will then ask for your "User ID" and password. You are now on CompuServe.

As a new user, you will be at the CompuServe menu, page CIS-01. From here you can "play" with the available options from the menu or you may enter "GO 28" or "MIC", which will take you to the MicroNET prompt "OK". (To return to the CompuServe menu enter "R DISPLA".) At this time you may enter "R HUG" which will take you to the HUG Bulletin Board. (PLEASE NOTE: This is also a changed feature since issue 15.)

Now it is time to explain a confusing COMMAND option of MCS. While on the HUGBB (or MNET), you will undoubtably want to COPY parts of the HUGBB or MNET e.g. a file from your DIRectory (more on that later). To do this you must do a CNTRL-B, which puts you into COMMAND mode. This means you are no longer sending signals to MNET. Enter COPy. MCS will ask you if you want a new buffer. You should enter "YES" to clear anything that might be in the buffer you don't want. MCS then asks to "ENTER FUNCTION" before you input anything you must understand that you are back into MNET without any prompt or indication from MCS or MNET . . . this is the confusing part. At this statement you are no longer in MCS but back in MNET, therefore, remember where you leave MNET when you do the CNTRL-B. What you input now will be what you want to COPy from MNET or the HUGBB.

At the conclusion of what you are COPying, do a CNTRL-B to get you back to the COMMAND mode. OPEN the disk filename.ext, WRITE what is in the buffer to disk, then CLOSE the file you have just saved on disk.

This function of COPying from MNET is a very important part of being on MNET and the HUGBB. Much of the information exchange is done through the personal DIRectories that each member of MicroNET receives upon membership. (I bet many MNET users did not know that.) From the MNET prompt "OK" you are able to access your own DIRectory by entering "DIR" and your fellow members' DIRectory by entering "DIR<XXXXX,XXX>" where the "X's" indicate his User ID. (See your users manual for protection of your DIR and your files.) Exchanging files on MicroNET is not difficult to learn after studying your users guide. The hard part comes when you try to upload to or download from the PDP-10 (CompuServe). We have just explained in the previous paragraphs how to "download" from the PDP-10 by COPying from MNET.

SENDing files is also possible through MCS. CPS (Computerized Phone System) has an EXECUTIVE protocol which allows the operator using CPS to upload directly from disk (H8/H89) to disk (PDP-10) without using any other media. With MCS this is not possible. MCS will send a file from your H8/H89 but you will need to send the file into MicroNET's FILGE editor.

To upload to your DIRectory on MNET using MCS; first, get to the MNET prompt "OK", then type "R FILGE". MNET will give the FILGE prompt "*", which at this time you will enter the "filename.ext" you want the file to be called on your MNET DIRectory. MNET's FILGE will print a line or two of info then you are ready to upload from your disk. Do a CNTRL-B, which will put you into MCS's COMMAND mode. Enter "SEND". MCS will ask for your HDOS "filename.ext" you want to SEND. After your< CR>, you will be transferring your file to your MNET DIRectory.

When the transfer is complete you will still be in MCS's "COMMAND" mode. Enter "CON"versation mode and you will return to MNET. PLEASE NOTE: No prompt will be showing as it is still in the FILGE editor waiting for more input!!! At this time enter "/EX" which will exit from FILGE and you will have just completed the transfer. You will now be back at the MNET prompt "OK". (You may now want to look at your DIRectory by typing "DIR" to see that the transfer was indeed complete.)

This has been a basic overview of using MCS on MicroNET and the HUGBB. Many patches are floating around that make some of the functions of MCS a bit easier and less complicated. I sincerely hope this will be an aid to old-timers as well as the new users of MCS.

In issue 16 of REMark, I began to explain the "contract" between CompuServe and one of our competitors. Well, the name can no long remain nameless, as you must be informed because it will affect any of you who want to get on MicroNET.

CompuServe Inc. has signed this "contract" with Radio Shack which allows them to "sell" the CompuServe request applications to MicroNET. This package they stock in some of their stores, allows a person to buy a package today and be on MNET tonight. Other hardware and software support services or sales are unable to have this same "privilege". Heath or HUG are not able to sell this package (as of this time) due to the limitations specified in the "contract". CompuServe has informed us that they will honor any application form which you may have sent or will send, however they do not guarantee any turn-around time. Most of what is said in issue 15 is still pertinent. One exception is that you can no longer write or call CompuServe and ask them to send you a request application form. Eventually, you will be able to purchase MicroNET membership only through Radio Shack or other distributors who agree to the stipulations of the agreement with CompuServe.

(vectored to page 27)

885-1091 Grading and Score Keeping BUG

Both EDIT programs need line 840 changed to:

```
840 W(I)=S(I,J):IF S(I,J)<>-999 THEN W=W+W(I):NEXT I ELSE NEXT I
```

Also the MENU program needs line 1410 changed to:

```
1410 N=N-1:GOTO 1470
```

A special thanks to Dr. M.J. Mansfield for this update.

BUGGIN' HUG

Dear HUG,

What's that you say fella'? Life is getting you down? You say your car is broken, the dog is sick, and now your mother-in-law is coming for a six month visit? To top it all off, you tried to boot up your disks the other day and the ones that used to run at 20mS now won't even chatter at 40mS? Is that it? Is that what's bothering you??

Well, CHEER UP! Look at the bright side! You probably can't do much about the car, the dog, or your mother-in-law; but, maybe you can do something about those disks!

If you take the top off the the H-17 cabinet, you will see that each drive has a printed circuit board held on by four phillips screws. Remove these screws and gently lift the board. You will have to unplug the wires on the right hand side of the board. After lifting the board out of the way, take a look at the mechanism of the drive. You will see the READ/WRITE head on the bottom of the drive, facing up. The drive head is held in a metal/plastic piece of the mechanism that travels along the worm gear that runs from the back of the drive frame to the front. On the right side of the drive frame, you will see a metal shaft that a stirrup shaped arm rides on. What we want to do is gently clean the accumulation of dust and dirt that has worked its' way into the worm gear and the shaft. This accumulation may be slowing down the drive. Using a Q-tip, apply a small amount of head cleaner to clean the READ/WRITE head. Clean the head by carefully brushing across it. Then, use the same Q-tip to clean as much of the worm gear as you can reach. Take care NOT to brush so hard that pieces of the cotton on the Q-tip come loose and find their way into the worm gear!!! This is not what we want! After completing this step, clean the shaft. If you gently apply pressure to the mechanism that the READ/WRITE head is on, you can move it back and forth along the worm gear to clean additional areas as needed. After cleaning, use another Q-tip and apply VERY LIGHT amount of light grade oil. I use a product called "BREAK-FREE" as it leaves a coat of teflon when it dries. Lubricate the shaft and the gear, then replace the circuit board. Test your drives. You should find them operating at their original speed if the cause of the deteriorzation was dirt. If they don't improve, then your difficulties could be isolated to the stepper-motor. I have used this technique with success on three-year old drives and they both operate as well as when they were new. I clean the drives about once every two months to keep them in tip-top shape!!

That is it for now. Gotta' take my mother-in-law to the Vet after I get the dog out of the muffler shop and the pound releases my car!!!!

Greg Green
207-885 Craigflower Rd.
Victoria B.C. Canada
V9A 2X4

Dear HUG,

Bob Ellerton's "THE MAGIC EGG" and "DISK CARE - OR ELSE" in Issues 13 and 14 of REMark on the effects of humidity on floppy diskettes brought to mind a related problem we had when the humidity in our lab got too low. During Minnesota winters, when the outside temperatures routinely drop below zero, indoor relative humidities are often less than 10% and the static charge buildup can spell instant death for a floppy. One winter both of our H-11 computer systems were experiencing mysterious, unexplained disk "crashes". By running a humidifier continually during the winters since, we have not had a single failure due to static electricity. I would therefore qualify your comment that humidity is probably your greatest enemy to read "too little or too much."

We have also had an occasional "crash" apparently from dirt particles on the heads scratching the oxide coating. But, since we started using 3M head cleaning diskettes we've had none of this problem, and I can't recommend them highly enough for users of a floppy system.

Chuck Knox
Physiology Department
University of Minnesota
Minneapolis, MN 55455

Dear HUG,

I have a hint that may be of value to H-14 owners. In order to keep the output of the H-14 dark enough to permit reproduction I spray the ribbon with WD-40, a silicone spray lubricant. I have found that if the ribbon is sprayed when the type gets a little light the result will be a marked darkening of the type. This permits extended use of the ribbon. I spray the WD-40 on the ribbon by spraying on the outside of the ribbon and through the holes in the ribbon reels. I get the best results if the ribbon is almost completely wound on one spool and the sprayed ribbon is permitted to stand overnight before it is again used. No need to remove it from the H-14, just spray before shutting down for the night. It will be ready for use the next morning. This can be repeated as often as desired until the ribbon wears out.

LTC Albert Guenzburger
USA TRADOC LIAISON OFFICE
Box 115
APO New York 09080

EDITORS NOTE: We do not know the long term effects, if any, on the head.

Msg#- 4824
Date- MAR. 14, 1981 22:07
From- ROBERT PEARCE 70140,356
To- SYSOP
Subject- STATIC DISCHARGE

HERE'S ONE FOR REMARK.
IF YOU'RE FORTUNATE OR NOT SO FORTUNATE TO HAVE A RUG UNDER YOUR COMPUTER, HERE IS A TRIED AND TRUE FIX FOR STATIC. "STATIC GUARD" FOUND IN GROCERY STORES AT ABOUT $2.50 A 6 OUNCE CAN, IS A SURE SHOT FOR GETTING RID OF STATIC DISCHARGE. I HAVE FOUND THAT IT ONLY TAKES A LIGHT SPRAYING UNDER YOUR CHAIR AND ABOUT 3 FEET BEYOND THAT TO CLEAR THE PESTY ELECTRON BUILD-UP. "STATIC GUARD" IS A PRODUCT OF THE ALBERTO-CULVER CO. MELROSE PARK, IL. 60160
BOB

EDITORS NOTE: We have found this to be true. It seems to last about a week and then needs another "shot" depending on weather and other relative conditions.

Dear HUG,

While contructing the ET3400 Morse Code Transmitter described in Issue 14 of REMark, I found several bugs. There are three typographical errors. The machine code intructions at addresses 0C7F, 0C87, and 0CA1 should be F7 C3 0E, as noted in the remarks at the right of the instructions. In addition, the circuit diagram contains an error. Pins 11 and 12 of IC-2 should not be connected to ground. The 7400 IC is an eight input NAND, and all eight inputs must be 1 (high) in order to generate an output pulse. I connected them to pin 1 of the same IC with satisfactory results.

I very much appreciate articles of this kind for the ET-3400 with the ETA-3400 accessory. It has many advantages over any other small computer in the accessibility of its' circuitry and its' facility with machine language. My thanks to Mr. Wolach for this interesting article.

Lee Aamodt
Route 5, Box 251
Santa Fe, NM 87501

Dear HUG,

Some newcomers may want to try the following:

If you have a dual disk system, include the program "MOUNTALL" from issue 12 of REMark, on each sysgened disk and rename it "PROLOGUE.SYS". Now, when you BOOT-UP, SY1: will automatically be mounted without further typing. NOTE: Be sure that you change the name of "OBJECT CODE" version (MOUNTALL.ABS) and not the "SOURCE CODE" version (MOUNTALL.ASM).

Another help - include the "OBJECT CODE" version (RESET.ABS) of the program "HDOS DISK RESET PROGRAM", given in issue 9 of REMark on each sysgened disk for ease of changing disks. With this program, it is not necessary to type "DISMOUNT/MOUNT" each time you change disks.

Keep up the nice work and I find the REMark more and more interesting.

Regards,

Lawrence R. Lankston
3475 St. Catherine St.
Florissant, Missouri 63033

REMa-BUG....

On page 15 in Issue #13 of REMark there is a serious error. The part numbers listed as 885-1057 and 885-1058 are interchanged. Please make this correction to ensure that you order the right item. We apologize for this oversight!

New HUG Software

Small Business Package III
Using HDOS and MBASIC

The NEW 885-1071 Small Business Package III (SBP III) is now available running under the Heath Disk Operating System (HDOS) and Microsoft BASIC (MBASIC). This new SBP III package sells for $75.00. (Owners of 885-1054 see notice below.)

The SBP III consists of three 5-1/4 inch disks requiring use of ONLY one drive. The package however runs much faster with two drives and even faster with three drives. HUG recommends three drives for ideal operation. A minimum of 48k of memory is required and the SBP III will run with either the H8 or the H89 (Z89). If using an H8 the H17 and H19 are required. A printer is required for all hardcopy reports.

The SBP III is set to handle 60 customer accounts.

The following features are included for the ACCOUNTS RECEIVABLES:

1. Print mailing labels for all accounts.
2. Print invoices, credit memos and statements.
3. Print accounts receivable balance only, aging report and total receivable.
4. Print a bargraph of sales.
5. Sales and invoice summary.

The SBP III contains the following features for the ACCOUNTS PAYABLE:

1. Print general expense ledger.
2. Print profit-loss statement for the month or year to date.
3. Print checks.

This package has been in development over the last year and been used by several businesses from coast to coast. The users' feel that this package is ideal for the SMALL business as it provides the needed daily, monthly and yearly reports.

Also included in this package is conversion routine to allow the owners of the Small Business Package II (885-1054) to convert to the SBP III without having to reenter each transaction.

OWNERS: Of 885-1054 Small Business Package II

This is a limited time offer for the owners of the Small Business Package II (885-1054). Due to the many improvements of the new SBP III (desribed above), HUG feels that they are important enough that HUG is offering a special deal for purchasers of the 885-1054 package. If the original THREE disks from Small Business Package II are returned along with a check for $40.00 you will receive the new 885-1071 Small Business Package III.

The THREE disks and the check for $40.00 (which includes postage and handling) and a request for the SMALL BUSINESS PACKAGE III (885-1071) must be returned to:

Heath Users' Group
ATTENTION: Nancy Strunk (SBP III)
Hilltop Road
St. Joseph, MI 49085

DO NOT include any other orders or requests with the three original disks, NO COD's, NO Credit Cards and NO Charges of any kind. THIS OFFER EXPIRES ON SEPTEMBER 30, 1981, NO EXCEPTIONS.

HUG Product List

Part Number	Description	Selling Price

CASSETTE SOFTWARE

MISCELLANEOUS COLLECTIONS

885-1008	Volume I Documentation	$ 9.00
885-1009	Tape I Cassette	$ 7.00
885-1012	Tape II BASIC Cassette	$ 9.00
885-1013	Volume II Documentation	$ 12.00
885-1014	Tape II ASM Cassette H8 Only	$ 9.00
885-1015	Volume III Documentation	$ 12.00
885-1026	Tape III Cassette	$ 9.00
885-1036	Tape IV Cassette	$ 9.00
885-1037	Volume IV Documentation	$ 12.00
885-1057	Tape V Cassette	$ 9.00
885-1058	Volume V Documentation	$ 12.00

UTILITIES

885-1034	Character Ed Cassette H8 Only	$ 11.00
885-1035	ED/ASM/DEBUG Cassette H8 Only	$ 11.00

PROGRAMMING LANGUAGES

885-1039	WISE on Cassette H8 Only	$ 9.00
885-1040	PILOT on Cassette H8 Only	$ 11.00
885-1045	FOCAL Cassette H8 Only	$ 11.00
885-1085	PILOT Documentation	$ 9.00

AMATEUR RADIO

885-1027	Morse8 Cassette H8 Only	$ 14.00
885-1028	RTTY Cassette H8 Only	$ 11.00

HDOS SOFTWARE

MISCELLANEOUS COLLECTIONS

885-1024	Disk I H8/H89	$ 18.00
885-1032	Disk V H8/H89	$ 18.00
885-1044	Disk VI H8/H89	$ 18.00
885-1060	Disk VII H8/H89	$ 18.00
885-1062	Disk VIII H8/H89 (2 Disks)	$ 25.00
885-1064	Disk IX H8/H89	$ 18.00
885-1066	Disk X H8/H89	$ 18.00
885-1069	Disk XIII Misc H8/H89	$ 18.00
885-1083	Disk XVI Misc H8/H89	$ 20.00

GAMES

885-1010	Adventure Disk H8/H89	$ 10.00
885-1029	Disk II Games 1 H8/H89	$ 18.00
885-1030	Disk III Games 2 H8/H89	$ 18.00
885-1031	Disk IV Music H8 Only	$ 23.00
885-1067	Disk XI H8/H19/H89 Games	$ 18.00
885-1068	Disk XII MBASIC Graphic Games	$ 18.00
885-1088	MBASIC Games Disk	$ 20.00
885-1093	DND Game for HDOS and MBASIC	$ 20.00

UTILITIES

885-1019	Device Drivers (HDOS 1.6)	$ 10.00
885-1022	HUG Editor (ED) Disk H8/H89	$ 15.00
885-1025	Runoff Disk H8/H89	$ 35.00
885-1043	MODEM Heath to Heath H8/H89	$ 21.00
885-1050	M.C.S. Modem for H8/H89	$ 18.00
885-1061	TMI Load H8 Only	$ 18.00
885-1063	Floating Point Disk H8/H89	$ 18.00
885-1065	Fix Point Package H8/H89 Disk	$ 18.00
885-1075	HDOS Support Package H8/H89	$ 60.00
885-1077	TXTCON/BASCON H8/H89 Disk	$ 18.00
885-1079	HDOS Page Editor	$ 25.00
885-1080	EDITX H8/H19/H89	$ 20.00
885-1082	Programs for Printers H8/H89	$ 20.00
885-1092	RDT Debugging Tool H8/H89 Disk	$ 30.00

PROGRAMMING LANGUAGES

885-1038	WISE on Disk H8/H89	$ 18.00
885-1042	PILOT on Disk H8/H89	$ 19.00
885-1059	FOCAL-8 on Disk H8/H89	$ 25.00
885-1078	HDOS Z80 Assembler	$ 25.00
885-1085	PILOT Documentation	$ 9.00
885-1086	Tiny Pascal Disk	$ 20.00

BUSINESS AND FINANCE

885-1047	Stocks H8/H89 Disk	$ 18.00
885-1048	Personal Account H8/H89 Disk	$ 18.00
885-1049	Income Tax Records H8/H89 Disk	$ 18.00
885-1051	Payroll H8/H89 Disk	$ 50.00
885-1055	MBASIC Inventory Disk H8/H89	$ 30.00
885-1056	MBASIC Mail List H8/H89 Disk	$ 30.00
885-1070	Disk XIV Home Finance H8/H89	$ 18.00
885-1071	SmBusPkg III 3 Disks H8/H19/H89	$ 75.00
885-1091	Grade and Score Keeping	$ 30.00

AMATEUR RADIO

885-1023	RTTY Disk H8 Only	$ 22.00
885-1052	Morse8 Disk H8 Only	$ 18.00

H11 SOFTWARE

885-1008	Volume I Documentation	$ 9.00
885-1033	HT-11 Disk I	$ 19.00

CP/M SOFTWARE (version 1.43 -- ORG 4200H)

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
885-1206	CP/M Games Disk	$ 21.00

CP/M SOFTWARE (version 2.2 -- ORG 0)

885-1207	TERM and H8COPY	$ 20.00

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

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

Whither STAT?

Those of you who switched from HDOS 1.6 to HDOS 2.0 know that the STAT command has changed completely. If you are not familiar with both HDOS's, here is a rundown of what STAT does in each. In HDOS 1.6, giving the command STAT results in a report of the number of disk reads and writes since you booted up. It also reports the number of hard (irrecoverable) and soft (recoverable) disk errors. If a hard error has occurred since you last used STAT, the sector containing the error is reported.

In HDOS 2.0, the STAT command gives you none of the above information. Instead, it shows where user memory ends (in split octal), where HDOS begins, and where the overlays will begin if they are in place. It also reports on the current status of the overlays, and on the status of all device drivers. As it prints out all of this information, it uses too many blank lines, and as a result some of the information scrolls off the top of the screen before the printout is finished.

Since both versions of STAT give us useful information, it would be nice if we could carry the HDOS 1.6 stat over into 2.0, and also take out some of those blank lines. Well, in this article I will tell you how you can "have your cake and eat it too."

The first thing we are going to do is to take out some of the blank lines in the HDOS 2.0 STAT command so that the printout will all fit on one video page. We are going to use PATCH to make the patches. PATCH will not ordinarily work on system files, so we will have to patch it first. In REMark #14 such a patch to PATCH was presented, but it required an H8, leaving H89 users out. Here is a patch that you can do with DUMP (from HUG part no. 885-1062). Locate the first sector of PATCH.ABS and make the following changes:

ADDRESS	OLD DATA	NEW DATA
21	CA	C3
3B	A7	AF

If you have RDT (HUG part no. 885-1092) and do not have DUMP, you can make the patch with RDT. Remove the write protect flag from PATCH, and make sure it is on a disk with at least 10 free sectors. Then run RDT and enter the following command sequence:

```
]LOAD SYn:PATCH  (n = drive number)
]CHANGE 42231,303
]CHANGE 42263,257
]SAVE SYn:PATCH,42200,53371,42200
```

We are dealing with HDOS 2.0 in this article, but the above patches will also work on the PATCH program in HDOS 1.6. If you do the patch with RDT, you may notice that PATCH now uses 10 sectors of disk space instead of 11. This is because RDT did not save the File History record that was on the end of PATCH, and is on the end of all system files. It is the File History record that causes unmodified PATCH to ask you for ID codes.

Now that you have modified PATCH, use it to patch SYSCMD.SYS on your system disk as follows:

ADDRESS	OLD DATA	NEW DATA
047145	012	000
047163	012	000
047317	012	000
050022	012	000
050056	012	000
050102	012	000
050250	012	000

After you make the above patches, the STAT command will produce a display that fits on one video page. But what about the information provided by the old HDOS 1.6 STAT command? Below is the source for a program that duplicates the old STAT command. If you give it the name OLDSTAT.ASM, and put the assembled object on your system disk, you can see disk reads, writes and errors by giving the command OLDSTAT. NOTE: The 8-inch disk driver (DK.DVD) does not update the read, write and error counts, so this program only provides information on 5-inch disks.

PS:

```
OLDSTAT -- OLD STYLE STATUS PROGRAM                     HEATH ASM #104.06.00
by Patrick Swayne 5-Mar-81                              11-Mar-81  Page    1

                          * OLDSTAT.ASM
                          * THIS PROGRAM DUPLICATES THE STAT COMMAND AS IT
                          * WAS IMPLEMENTED ON HDOS 1.6.  IT ALLOWS HDOS 2.0
                          * USERS TO SEE DISK READS, WRITES, AND ERRORS.

                          * EXTERNALS

```

```
000.002              00010  .SCOUT  EQU    2            SINGLE CHARACTER OUTPUT
000.003              00011  .PRINT  EQU    3            CHARACTER STRING PRINTER
000.057              00012  .ERROR  EQU    57Q          HDOS ERROR PRINTER
030.072              00013  $DADA   EQU    30072A       HL = HL + A
030.106              00014  $DU66   EQU    30106A       HL = BC / DE
030.324              00015  $MU10   EQU    30324A       HL = DE * 10
040.100              00016  WBOOT   EQU    40100A       HDOS WARM BOOT
040.126              00017  D.ERTS  EQU    40126A       LAST HARD ERROR TRACK, SECTOR
040.261              00018  D.HECNT EQU    40261A       NO. OF HARD ERRORS
040.262              00019  D.SECNT EQU    40262A       NO. OF SOFT ERRORS
040.273              00020  D.OPR   EQU    40273A       NO. OF DISK READS
040.275              00021  D.OPW   EQU    40275A       NO. OF DISK WRITES
040.277              00022  S.DATE  EQU    40277A       SYSTEM DATE
                     00023
042.200              00024          ORG    42200A       NORMAL STARTING POINT
                     00025
042.200  076 012     00026  OLDSTAT MVI    A,12Q
042.202  377 002     00027          SCALL  .SCOUT       PRINT A CRLF
042.204  257         00028          XRA    A            CLEAR A
042.205  046 011     00029          MVI    H,11Q        ASCII TAB
042.207  377 057     00030          SCALL  .ERROR       PRINT HDOS SIGN-ON
042.211  041 277 040 00031          LXI    H,S.DATE     POINT TO SYSTEM DATE
042.214  076 011     00032          MVI    A,9          NO. OF CHARACTERS
042.216  315 371 042 00033          CALL   PRTMSG       PRINT THE CURRENT DATE
042.221  052 273 040 00034          LHLD   D.OPR        GET NO. OF DISK READS
042.224  104         00035          MOV    B,H
042.225  115         00036          MOV    C,L          PUT IT IN BC
042.226  041 067 043 00037          LXI    H,READNO     WHERE TO PUT NUMBER
042.231  076 005     00038          MVI    A,5          MAX NO. OF DIGITS
042.233  315 005 043 00039          CALL   PUTNUM       PUT NUMBER OF READS IN
042.236  052 275 040 00040          LHLD   D.OPW        GET NO. OF WRITES
042.241  104         00041          MOV    B,H
042.242  115         00042          MOV    C,L          IN BC
042.243  041 104 043 00043          LXI    H,WRITENO    WHERE TO PUT IT
042.246  076 005     00044          MVI    A,5          NO. OF DIGITS
042.250  315 005 043 00045          CALL   PUTNUM       PUT WRITES IN
042.253  072 261 040 00046          LDA    D.HECNT      GET NO. OF HARD ERRORS
042.256  117         00047          MOV    C,A
042.257  006 000     00048          MVI    B,0          IN BC
042.261  041 144 043 00049          LXI    H,HARDNO     WHERE TO PUT IT
042.264  076 003     00050          MVI    A,3          NO. OF DIGITS
042.266  315 005 043 00051          CALL   PUTNUM       PUT HARD ERRORS IN
042.271  052 262 040 00052          LHLD   D.SECNT      GET SOFT ERRORS
042.274  174         00053          MOV    A,H          CORRECT TO BINARY
042.275  247         00054          ANA    A
042.276  037         00055          RAR
042.277  107         00056          MOV    B,A
042.300  175         00057          MOV    A,L
042.301  037         00058          RAR
042.302  117         00059          MOV    C,A          IN BC
042.303  041 165 043 00060          LXI    H,SOFTNO     WHERE TO PUT IT
042.306  076 005     00061          MVI    A,5          NO. OF DIGITS
042.310  315 005 043 00062          CALL   PUTNUM       PUT IT IN
042.313  041 054 043 00063          LXI    H,MSG1       POINT TO STATUS MESSAGE
042.316  377 003     00064          SCALL  .PRINT       PRINT IT
042.320  072 126 040 00065          LDA    D.ERTS       GET LAST HARD ERROR
042.323  247         00066          ANA    A            GOT ONE TO REPORT?
042.324  312 100 040 00067          JZ     WBOOT        IF NOT, EXIT TO HDOS
042.327  137         00068          MOV    E,A
042.330  026 000     00069          MVI    D,0          TRACK NO. IN DE
042.332  315 324 030 00070          CALL   $MU10        HL = TRACK NO. * 10
042.335  072 127 040 00071          LDA    D.ERTS+1     GET SECTOR NUMBER
042.340  315 072 030 00072          CALL   $DADA        HL = ABSOLUTE SECTOR NO.
042.343  104         00073          MOV    B,H
042.344  115         00074          MOV    C,L          BC = SECTOR NO.
042.345  041 261 043 00075          LXI    H,SECTNO     WHERE TO PUT IT
042.350  076 003     00076          MVI    A,3          NO. OF DIGITS
042.352  315 005 043 00077          CALL   PUTNUM       PUT NUMBER IN
042.355  041 216 043 00078          LXI    H,MSG2       PRINT HARD ERROR MESSAGE
042.360  377 003     00079          SCALL  .PRINT
```

```
042.362  257           00080              XRA      A
042.363  062 126 040   00081              STA      D.ERTS               CLEAR LAST HARD ERROR
042.366  303 100 040   00082              JMP      WBOOT
                       00083
                       00084    * PRINT A MESSAGE AT (HL)
                       00085    * NUMBER OF CHARACTERS IN A
                       00086
042.371  247           00087    PRTMSG    ANA      A                    DONE?
042.372  310           00088              RZ                            IF SO, EXIT
042.373  365           00089              PUSH     PSW                  SAVE COUNT
042.374  176           00090              MOV      A,M                  GET CHARACTER
042.375  043           00091              INX      H                    INCREMENT POINTER
042.376  377 002       00092              SCALL    .SCOUT               PRINT CHARACTER
043.000  361           00093              POP      PSW                  RESTORE COUNT
043.001  075           00094              DCR      A                    DECREMENT COUNT
043.002  303 371 042   00095              JMP      PRTMSG               LOOP UNTIL FINISHED
                       00096
                       00097    * CONVERT A BINARY NUMBER TO ASCII DECIMAL
                       00098    * AND PUT IT AT (HL).  NULL OUT LEADING ZEROS
                       00099    * NUMBER IN BC, DIGIT COUNT IN A
                       00100
043.005  315 072 030   00101    PUTNUM    CALL     $DADA                FIND END OF AREA
043.010  365           00102    PUTNUO    PUSH     PSW                  SAVE COUNT
043.011  345           00103              PUSH     H                    SAVE AREA AGAIN
043.012  021 012 000   00104              LXI      D,10
043.015  315 106 030   00105              CALL     $DU66                DIVIDE BY 10
043.020  104           00106              MOV      B,H                  RESULT BACK IN BC
043.021  115           00107              MOV      C,L
043.022  341           00108              POP      H                    RESTORE AREA FOR NUMBER
043.023  076 060       00109              MVI      A,60Q                ASCII BIAS
043.025  203           00110              ADD      E                    ADD TO BINARY
043.026  053           00111              DCX      H                    DECREMENT POINTER
043.027  167           00112              MOV      M,A                  STORE DIGIT
043.030  170           00113              MOV      A,B
043.031  261           00114              ORA      C                    DONE CONVERTING?
043.032  312 043 043   00115              JZ       PUTNU1               IF SO, CHECK COUNT
043.035  361           00116              POP      PSW                  GET COUNT
043.036  075           00117              DCR      A                    DONE WITH NUMBER?
043.037  302 010 043   00118              JNZ      PUTNUO               IF NOT, CONTINUE
043.042  311           00119              RET
043.043  361           00120    PUTNU1    POP      PSW                  GET COUNT
043.044  075           00121              DCR      A                    DONE WITH NUMBER?
043.045  310           00122              RZ                            IF SO, EXIT
043.046  053           00123              DCX      H                    DECREMENT POINTER
043.047  066 000       00124              MVI      M,0                  PAD WITH NULLS
043.051  303 044 043   00125              JMP      PUTNU1+1             LOOP UNTIL FINISHED
                       00126
                       00127    * MESSAGES
                       00128
043.054  012 104 151   00129    MSG1      DB       12Q,'Disk I/O:',11Q
043.067  116 116 116   00130    READNO    DB       'NNNNN Reads, '
043.104  116 116 116   00131    WRITENO   DB       'NNNNN Writes Performed',12Q
043.133  105 162 162   00132              DB       'Errors:',11Q,11Q
043.144  116 116 116   00133    HARDNO    DB       'NNN Hard Errors ('
043.165  116 116 116   00134    SOFTNO    DB       'NNNNN Recovered Errors)',12Q,212Q
043.216  114 141 163   00135    MSG2      DB       'Last Hard Error Ocurred on Sector #'
043.261  116 116 116   00136    SECTNO    DB       'NNN',12Q,212Q
043.266  000           00137              END      OLDSTAT
                                                                   EOF
```

HUG BUG: The instructions for modifying SUBMIT for HDOS 2.0 operation (Editor's Note on page 26) in REMark #13 contain an error. In addition to the modifications to Jay Gold's program given in the second paragraph on page 26, add the following: Remove three lines starting with the label BUFUL and ending with JMP DONE. If this change is not made, the result is that the STUFF.ACM that you make is more than the 256 bytes that get saved on disk as your .ABS file by SUBMIT, and some of your commands (entered when you ran SUBMIT) may get chopped off. Another area that might have caused confusion is in the first paragraph on page 26. The phrase that reads "from the label PROGl to the line SCALL .EXIT" should read "from the label PROGl to and including the line SCALL .EXIT".

PS:

Comments on the Pascal Language

by Fred Pospeschil
3108 Jackson Street
Bellevue, NE 68005

Editor's Note: The following is an update of an article that originally appeared in the OMAHUG newsletter (Omaha HUG). It is a follow-up to an earlier article in that newsletter on C-80, so there is some comparison in this article between C-80 and Pascal. -- PS:

I would like to make a few comments on a language which has just become available under HDOS thanks to Jim Teixeira. This language is Pascal (HUG part no. 885-1086). Although it has been available for some time under CPM this implementation of "Tiny Pascal" does not require CPM and appears to be in the public domain for non-commercial purposes. As with C80, Tiny Pascal implements only a part of the full language but it is sufficient to begin learning the language. C80 implements more of C than Tiny Pascal does of Pascal. I would not recommend using either of the subsets for making a comparison or evaluation of the full languages.

The last page of each PASCAL NEWS, which is published by the Pascal User's Group (PUG), states:

FACTS ABOUT Pascal, THE PROGRAMMING LANGUAGE:

"Pascal is a small, practical, and general purpose (but not all-purpose) programming language possessing algorithmic and data structures to aid systematic programming. Pascal was intended to be easy to learn and read by humans, and efficient to translate by computers.

"Pascal has met these design goals and is being used quite widely and successfully for:

*teaching programming concepts
*developing reliable "production" software
*implementing software efficiently on
 today's machines
*writing portable software

"Pascal is a leading language in computer science today and is being used increasingly in the world's computing industry to save energy and resources and increase productivity.

"Pascal implementations exist for more than 62 different computer systems, and the number increases every month.

"Pascal User's Group is each individual member's group. We currently have more than 3357 active members in more than 41 countries. This year Pascal News is averaging more than 120 pages per issue."

Although there are only three to five issues per year it only costs $6.00 and it usually take a long time to get through each issue. In addition some of the issues contain extensive Pascal source code. One issue contained over 100 pages of code which can be used to test your compiler to see if it meets all of the language capabilities and requirements. To get as much as possible into each issue PUG prints two regular pages on each page of the newsletter.

When the University of Wisconsin announced their implementation of Pascal they provided the following informal description:

"In 1971 Niklaus Wirth announced a new programming language, a successor to ALGOL W in many respects, which he named Pascal.

"Pascal has the following advantages over ALGOL W. It supports file I/O; it's dynamically allocated structure mechanism is more comprehensive; it makes the case statement safe by introducing case labels chosen from a declared, finite set; it provides strong type checking to help inforce the integrity of user-defined abstractions; and it introduces the notion of a finite set (of data, or of alternatives) in a way that will make it a useful data type for every programmer. Pascal also expands somewhat the choice of iterative control structures available to a programmer.

"It also has significant advantages over PL/1. Its strong type checking protects the novice (and experienced) programmer from errors of misunderstanding that the default mode conversions of PL/1 tend to invite. I/O in Pascal is much simpler to specify than in PL/1, yet it provides comparably powerful capabilities. The bounded iteration for and case statements of Pascal are safe, whereas the do for statement of PL/1 can accidentally produce infinite loops, and PL/1 has no case statement. Because of the typing of finite sets in Pascal, many instances of array reference do not require runtime bounds checking in order to guard against out-of-range indexing. The use of pointers can be rendered safe so that a user can not inadvertently create a wild reference in a list-processing application by missing a pointer variable.

And perhaps most important, the basic structure of Pascal is simple and consistant, so that a student can master the whole language in a period of a few weeks, a feat achieved by relatively few PL/1 programmers in an entire career."

As I see it, Pascal is clearly a high level language - much more so than C, BASIC, COBOL, or FORTRAN. Although I find it easier to move between C and Pascal than between any other two languages, Pascal is a higher level language than C in that it allows the programmer to express his/her ideas in terms which more closely resemble the human thought process.

The language includes a wide variety of features and capabilities which make it fairly easy for the programmer to tell the computer what to do and at the same time make the code reasonably easy for humans to read and understand. Some of the features or characteristics of the language include:

Data Types:
integer(*), real(*), character (string)(*), boolean, defined scaler, and user defined

Structuring Methods:
array(*), record, set, file, pointer

Operators:
mixed arithmetic(*), integer division, modulus, exponentiation(*), relational operators(*), set operators, and logical operators

Control Structures:
if(*), case, for(*), while, repeat, goto(*)

Subprograms:
recursion, local variables, parameters
For comparison purposes those items which are common to most BASICs are marked with an (*). Some of the more extended BASICs have more of Pascal's features so as to provide better tools to work with. (Note that these features apply to fully implemented Pascal, and many are missing from Tiny Pascal.)

Pascal's built-in file handling procedures include: put(f), get(f), reset(f), rewrite(f), read, write, readln, writeln, and page. Pascal also includes four procedures which are used for dynamic allocation and de-allocation of storage space. These include: new(p), new(p,Tl,...Tn), dispose(p), and dispose(p,Tl,...Tn). Pascal supports mathmatical and other data manipulation tasks with the the following built-in functions:

ABS(X)	SQR(X)	SIN(X)
COS(X)	EXP(X)	LN(X)
SQRT(X)	ARCTAN(X)	TRUNC(X)
ROUND(X)	ORD(X)	CHR(X)
SUCC(X)	PRED(X)	ODD(X)
EOF(F)	EOLN(F)	

Pascal's boolean operatons include logical AND, OR, and NOT and its set operators are UNION, DIFFERENCE, and INTERSECTION.

The above comments provide but a brief introduction to some of the language's capabilities and in many ways do it injustice. Additional insights into Pascal can be gained by reading the many items found mostly in Interface Age and BYTE magazines. Of particular note are the Pascal/M description in the September Interface Age (pp 96-102) and the rather lengthy series in the June 1979 thru April 1980 issues. If you are going to use the Tiny Pascal compiler mentioned earlier you will need a copy of the three part series "A Tiny Pascal Compiler" in the September - November issues of BYTE. One of the most widely acclaimed paperback texts is Peter Grogono's Programming in Pascal (Addison Wesley, about $10.40). Of course the original definition of the language is Jensen and Wirth's Pascal User Manual and Report (Springer-Verlag). Kenneth L. Bowles's Problem Solving Using PASCAL (Springer-Verlag) is the text used as the introductory programming text at the University of California San Diego (UCSD). This text does not go into the language as deeply as Grogono's but it covers most of the extensions added by UCSD and is therefore a good starting point for the UCSD Pascal package now offered by Heath. (NOTE: UCSD Pascal is not compatible with either HDOS or CPM. It is its own operating system.) Another excellent text is Algorithms + Data Structures = Programs by Nicklaus Wirth (Prentice-Hall series in Automatic Computation). This is not only an often cited text but it, as one should expect, uses Pascal as the language to show how various situations can be handled. Its chapter three provides about the best coverage of sorting I have seen.

Given the rapidly expanding use of the language around the world, in colleges, and the Department of Defense's new high-order language ADA, which is based on Pascal, I expect to see the language take a dominate place in the programming field. Commercial firms are also moving towards Pascal. For example, Texas Instruments has made it the company language (see Electronics, June 1979, pp 109-121), and if I remember correctly the Australian Atomic Energy Commission has re-written the operating system for their IBM 370 in Pascal and Pascal was the first, and may still be the only, high level language used on the Cray 1 super computer.

There are a lot of languages around, each with its own strengths and weaknesses. Pascal is not all-purpose and other languages, such as assembler, will probably always be needed for certain functions. For many applications BASIC can be a good language as can FORTRAN, COBOL, and C. However, I go along with Carol Anne Ogdin, technical director of Software Technique, Inc. (see "The many choices in development languages", MINI-MICRO SYSTEMS, August 1980, pp 81-84) in thinking that in a few years BASIC will be used by the casual users and the serious programmers will migrate to Pascal and then to ADA when it becomes available.

To show that Pascal is not at all hard to work with, three short programs are provided which all do the same thing -- define an array of 5000 numbers and then fill the array with ascending values. The programs all display the same messages on the terminal. Being compiled languages, the C80 (example 2) and Tiny Pascal (example 3) programs do the job twenty times faster than the BASIC program (example 1).

```
<EXAMPLE 1:  B. H. BASIC>
00010 PRINT "START FILLING ARRAY"
00020 S=5000
00030 DIM A(S)
00040 FOR J = 0 TO S-1
00050 A(J) = J
00060 NEXT J
00070 PRINT "ARRAY FILLED"
00080 END
```

```
<EXAMPLE 2:  C80
#define SIZE 5000
main()
{
int arr1[SIZE];
int j;
char *p;
     j = 0;
     for(p = "Start filling array\n";
      *p; putchar(*p++));
     while (j < SIZE)
           {
           arr1[j] = j;
           j++;
           }
     for(p = "Array filled\n";
      *p; putchar(*p++));
}
```

```
<EXAMPLE 3:  Tiny Pascal>
CONST SIZE=5000;
VAR   ARR1: ARRAY[SIZE] OF INTEGER;
      J: INTEGER;
BEGIN
     J:=0;
     WRITE('START FILLING ARRAY',10);
     WHILE J < SIZE DO
          BEGIN
               ARR1[J] := J;
               J := J+1;
          END;
     WRITE('ARRAY FILLED',10);
END.
```

Both Tiny Pascal and C80 will help you to learn better programming practices and produce programs which are both easier to read and execute faster than BASIC. Although C80 and Tiny Pascal programs execute a little slower than an ASM language program the same application will normally take far less time to write and will normally be much easier to read. I found C80 provided an excellent tool for writing a multikey sort program and Tiny Pascal seems to be up to the task of driving the HA-8-3 color graphics board. These are excellent learning tools and stepping stones to more complete implementations.

The more full implementations include the UCSD package mentioned above. Although the UCSD system is widely used on other machines you can share software only with others who are using the UCSD operating system. For those who want to stay with HDOS there are several choices. One is Tiny Pascal which produces binary load modules for speed and is quite easy to use. Compile and translate time is much faster than C80. Though it is not a full implementation it is quite powerful for many applications.

If you are willing to pay $95 ($65 in five or more copies) the Lucidata Pascal V2.8 under HDOS looks like a substantial step up in capabilities. This one is a P-code interpreter, like UCSD Pascal, which makes it 2-8 times faster than BASIC but slower than binary load modules. I have not seen the product in action but have a good report on its implementer and the language summary sheets sent by Larry Reeve indicate that the package will be able to handle a much wider range of applications than Tiny Pascal.

Another package which one might want to keep in mind is Pascal MT+ from MT Microsystems (this is not Pascal MT). Although it is still under conversion to HDOS it should be full ISO Standard Pascal with numerous extensions. The user's manual, which I purchased for $30, is quite good and indicates that it should be the most powerful and flexible of the compilers. With 18 digit BCD and both software and hardware (AMD9511A) floating point math it should do well in both business and scientific applications. It produces binary load modules via a linking loader and includes many compile and link control options. It is somewhat more expensive, about $250, but will probably be worth the price -- only time will tell.

These are the four packages that I know of and at prices which range from $20 (Tiny Pascal) to about $300 (UCSD Pascal) which gives you a pretty good set of options to select from.

EOF

ps from PS: To further illustrate the flexibility of Pascal, here is another way of doing the sample program.

```
CONST SIZE=5000;
VAR   ARR1: ARRAY[SIZE] OF INTEGER;
      J: INTEGER;
BEGIN
     WRITE('START FILLING ARRAY',10);
     FOR J:=0 TO SIZE DO ARR1[J] := J;
     WRITE('ARRAY FILLED',10);
END.
```

And I just coundn't miss this opportunity to include my own baby, FOCAL-8 (HUG part no. 885-1059), in this comparison, so here is the program in FOCAL.

```
01.10 T "START FILLING ARRAY"!
01.20 S SIZE=5000
01.30 F J=0,SIZE;S ARR1(J)=J
01.40 T "ARRAY FILLED"!
```

PS:

Color Graphics in Tiny Pascal

The following program is an example of software for the HA-8-3 color graphics board in Tiny Pascal (HUG part no. 885-1086). This program was one of the demonstration programs used at the West Coast Computer Faire in San Fransisco. It draws a three sided box (the top is open) in blue on a light gray background. Then a red ball is thrown into the box. It bounces off one side, bounces on the bottom in smaller and smaller bounces, then rolls to a stop. The throwing is done 5 times, then the program stops and returns to HDOS.

If you plan to write your own color software in Tiny Pascal, it might be a good idea to save the constant definitions, the procedures WVD, WVA, and WVR, and the function RVD from this program in a separate file as a "common deck" for other color programs. Each time you write a new program, you can load the "common deck" file with your editor and start from there. REMark will be featuring more color software in future issues.

```
{ Bounce -- MIT bounce for the HA-8-3 color board }
{ by J W Tittsler and P Swayne }

{ PROGRAM BOUNCE }

{ Constants }

CONST   TRANS=0;
        BLACK=1;
        MDGRN=2;
        LTGRN=3;
        DKBLU=4;
        LTBLU=5;
        DKRED=6;
        CYAN=7;
        MDRED=8;
        LTRED=9;
        DKYEL=10;
        LTYEL=11;
        DKGRN=12;
        MAGEN=13;
        GRAY=14;
        WHITE=15;
        VDPDAT=%270 {VDP DATA PORT};
        VDPCTL=%271 {VDP CONTROL PORT};

        MAXBALLS=200;
        PNT=0;          { Pattern mode deffinitions }
        PNTL=768;
        PGT=%10000;
        PGTL=2048;
        PGTE=%20000;
        PCT=%3000;
        PCTL=32;
        PCTE=%3040;
        SAT=%3200;
        SNT=%3200;
        SGT=0;

{ Variables }

VAR K: INTEGER;

{ Functions and Procedures }

{ Write data into the video processor's RAM }

PROC WVD(ADR,VAL);
BEGIN
        PORT[VDPCTL]:=ADR;
        PORT[VDPCTL]:=(((ADR SHR 8) AND %77) OR %100);
        PORT[VDPDAT]:=VAL;
END; {WVD}

{ Write a RAM address to the video processor }

PROC WVA(ADR);
BEGIN
        PORT[VDPCTL]:=ADR;
        PORT[VDPCTL]:=(((ADR SHR 8) AND %77) OR %100);
END; {WVA}

{ Read data from the video processor RAM }

FUNC RVD(ADR);
VAR I: INTEGER;
BEGIN
        PORT[VDPCTL]:=ADR AND %377;
        PORT[VDPCTL]:=(ADR SHR 8) AND %77;
        RVD:=PORT[VDPDAT] AND %377;
END; {RVD}

{ Write to a video processor register }

PROC WVR(REG,VAL);
BEGIN
        PORT[VDPCTL]:=VAL;
        PORT[VDPCTL]:=((REG AND 7) OR %200);
END; {WVR}

{ Draw a box }

PROC DRAWBOX;
VAR X,Y: INTEGER;
BEGIN
        FOR Y:=1 TO 23 DO BEGIN
                WVD(PNT+(Y SHL 5)+5, DKBLU SHL 3);
                WVD(PNT+(Y SHL 5)+27, DKBLU SHL 3);
                END;
        FOR X:=6 TO 26 DO WVD(PNT+(23 SHL 5)+X, DKBLU SHL 3);
END; {DRAWBOX}

{ Throw a ball }

PROC THROW;
VAR VX, VY, X, Y, I: INTEGER;
```

```
BEGIN
        X:=50; Y:=10; VX:=15; VY:= 0;
        WHILE ((VX <> 0) OR (VY <> 0) OR (Y < 173)) DO BEGIN
                IF (Y < 175) THEN VY:=VY+2;
                X:=X+VX; Y:=Y+VY;
                IF X > 208 THEN BEGIN X:=416-X; VX:=-VX DIV 4; END;
                IF X < 48 THEN BEGIN X:=96-X; VX:=-VX DIV 4; END;
                IF Y > 176 THEN BEGIN Y:=352-Y; VY:=-VY DIV 2; END;
        WVA(SAT);
        PORT[VDPDAT]:=Y; PORT[VDPDAT]:=X;
        FOR I:=0 TO 100 DO;
        END; {WHILE}
END; {THROW}

{ Intiialize color board }

PROC IVDP;
VAR     I, J, K, ROWST: INTEGER;
        SUR: ARRAY[8] OF INTEGER;
        BALL: ARRAY[8] OF INTEGER;

BEGIN
        SUR[0]:=0; SUR[1]:=%200;
        SUR[2]:=PNT DIV %4000; SUR[3]:=PCT DIV %100;
        SUR[4]:=PGT DIV %10000; SUR[5]:=SAT DIV %200;
        SUR[6]:=SGT DIV %10000; SUR[7]:=WHITE;

        BALL[0]:=%74;   BALL[7]:=%74;    { Define a ball in binary }
        BALL[1]:=%176;  BALL[6]:=%176;
        BALL[2]:=%377;  BALL[5]:=%377;
        BALL[3]:=%377;  BALL[4]:=%377;

        FOR I:=0 TO 7 DO WVR(I,SUR[I]);

        WVA(PGT);
        FOR I:=0 TO PGTL-1 DO PORT[VDPDAT]:=0;

        WVA(PNT);
        FOR I:=0 TO PNTL-1 DO PORT[VDPDAT]:=0;

        WVA(PCT);
        FOR I:=0 TO PCTL-1 DO PORT[VDPDAT]:=I MOD 16;

        WVA(SGT+%4000);
        FOR I:=0 TO 7 DO PORT[VDPDAT]:=BALL[I];

        WVA(SAT);
        PORT[VDPDAT]:=100;
        PORT[VDPDAT]:=100;
        PORT[VDPDAT]:=%200;
        PORT[VDPDAT]:=MDRED;
        PORT[VDPDAT]:=%320;

        WVR(1,SUR[1] OR %100);          { Enable video }
END; {IVDP}

{ Main program }

BEGIN {MAIN}

        IVDP;    { Initialize the color board }

        DRAWBOX;

        FOR K:=0 TO 4 DO BEGIN
                THROW;
                PAUSE(250);
                END; {FOR}
END. {MAIN}
```

EOF

(vectored from page 13)

The following two messages were left from the MNET "Wizard" concerning this subject. I am printing the two messages in part because they are informative and about explain it all:

Msg#- 4548
Date- Mar. 3, 1981 17:23
From- The Wizard 70000,1
To- All
Subject- CIS Packages from RS

Radio Shack has a package, part number 26-2224, selling for $19.95, which includes a SnaPak (prevalidated ID & password) and (very minimal) documentation. We are working on improved documentation, and when it becomes available will have it distributed in RS packages and sent to all existing customers free of charge. If your local RS store does not have this "dumb terminal" package, or says they can't or won't order it, call; (800) 433-1679 and give the store number. . .

Msg#- 4559
Date- MAR. 3, 1981 21:53
From- The Wizard 70000,1
To- ALL
Subject- RS Packages, contd.

I forgot to mention in my previous message that if you do encounter a RS store that refuses to deal with you, Tandy will deal rather harshly with them. Please contact us via FEEDBK immediately, or call our 800 voice line. Our Tandy interface will pass the word directly to John Roach's secretary, and SHE will take personal care of the problem.
I can't tell you how most of us feel about the limitations imposed on us by the contract. However, that is personal opinion. We will do as much as possible for you. . .
I understand the reluctance some Heath (and other) users might have at entering a Radio Shack store and attempting to deal with them . . .
All I can ask is that you hold in there; things will improve eventually.

We really appreciate the support given to us from the CompuServe staff whom we have dealt with since this incident took place.

One point I would like to emphasize; we have had many calls where users have gone into a Radio Shack store and found that they "had" to purchase their "special deal package" for $29.95, which included Radio Shack's software modem package. Those stores are not to do that as "The Wizard" said. They must sell the SnaPak for $19.95.

Since this issue has started I have called Barry Thompson the Product Line Manager for VIDEOTEX. He is responsible for the Radio Shack Part #26-2224. He relayed to me that many of the RS store managers are obviously reluctant to sell these parts and that it might take awhile for them to become more responsive to you the user. They may demand that you put down a downpayment for this part but they must order it for you. If you have any problems, I would like to know about them. If you choose to call RS's (800) number mentioned above, ask for Helen Martin. If you don't get the results you expect, I would like to know that also. Maybe in time this will not be such an awkward subject.

I do not intend for this to be a big deal concerning CompuServe and Radio Shack. It definately is a let down to any users of Heath equipment. The CompuServe staff has been very kind to us and at the present is making significant changes to the existing HUG Bulletin Board. We are grateful for their support and help.

Well, maybe next month we will be able to report on the new HUG Bulletin Board. If not I will continue to explain more of the basics that relate to newcomers on the BB. (I promise the next article will be much shorter!)

SYSOP <TLJ>

Tiny BASIC Tricks

HEX-DECIMAL CONVERSION

This trick was submitted by Don Colner, 2202 Sherbrooke Way, Rockville, MD 20850.

In order to use the USR function in Tiny BASIC, you must provide it with the starting address of your machine code routine in decimal. Since the monitor (which you will probably use to enter your machine code program) uses the hex base, you will have to convert when you go to BASIC, and could make errors in the process.

Here is a way to use the ETA-3400 to do the converting for you, and a way to use use Tiny BASIC's math capabilities on hex numbers. The variables A-Z used in Tiny BASIC are at a fixed place in memory starting at 82 hex. Since they are not cleared when BASIC starts up, it is possible to insert values while in the monitor (in hex notation), and print the values in decimal while in BASIC. You can also go the other way.

Here are some examples. Suppose you wanted to call a machine code routine from BASIC that starts at 300 hex, so you need to know that number in decimal. The following sequence of commands will perform the conversion (what you type is underlined -- terminate each command with CR):

```
MON> M 82     Variable A is at 82 hex.
0082 nn 3     Put in the most significant
              byte.  The "nn" is the
              number already there.
0083 nn 00    Put in low byte.
0084 nn       Type BREAK or ESC here.
MON> G 1C00   Start Tiny BASIC.
:PR A         Tell BASIC to print A.
768           This is 300 hex in decimal.
:BYE          Return to monitor.
```

If you want to go the other way, you can use this sequence:

```
MON> G 1C00   Start Tiny BASIC.
:A=5642       5642 is the number to
              convert.
:BYE          Return to monitor.
MON> M 82     Variable A address.
0082 16       Type CR.
0083 0A       Type BREAK or ESC.
MON>
```

The above example shows that the number 5642 decimal is 160A hex. Now, suppose we want to add 123 hex to AA hex. The following sequence will do the job:

```
MON> M 82     Point to variable A.
0082 nn 1     Put in the high byte.
0083 nn 23    Put in the low byte.
0084 nn 0     The variable B starts
              at 84 hex.  The high byte
              of AA hex is zero.
0085 nn AA    Put in low byte of AA.
0086 nn       Type BREAK or ESC.
MON> G 1C00   Start Tiny BASIC.
:A=A+B        Add the two numbers.
:BYE          Return to monitor.
MON> M82      Look at A.
0082 1        Type CR.
0083 CD       Type BREAK or ESC.
MON>
```

The answer to 123 hex + AA hex is 1CD hex. The above procedure can also be used for multiplication, division, and subtraction. You can use more variables if you need them. They start at 82 hex and are in alphabetical order with each one using 2 bytes.

PS:

(vectored from page 7)

the extended memory capacity. This is because the original HT-11 system was SYSGEN'd for only the 56K bytes of memory. I am therefore purchasing the RT-11 V4 and MUBASIC (multi-user) V2 to use the system completely.

For the present we will continue the manual bootup. Some day I may have a proper replacement ROM burned-in to replace those in the H-27 interface card. They are different from the normal DEC configuration.

So far we have checked out a good number of basic programs using all the disk file types. Editor and PIP are alive and well, and the H-19 graphics function just fine.

Here it is in a "nutshell". We're all very pleased here. Wish us luck on our first SYSGEN. Hope to hear from you and from others who try this trick.

Sincerely,
Thomas G. Barnum

ADDITIONAL NOTE: Mr. Barnum called to verify that the diagnostic tests all proved successful with no complications. He also indicated he has purchased the MUBASIC and is using it without difficulty and intends to expand the number of terminals being used.

Who's on First?

SY2: - SY1: - SY0:

If you have installed an H-77 drive on your H-89 and BOOT without having the H-77 power on you can erase the directory on the floppy disk. It looks like the cause is the loss of voltage on the resistor pack that is installed in the "last drive" on your system. Without the voltage, the drive thinks it's writing to the disk and the H-89 thinks it is reading the disk. The result? You now have a disk with no directory! Once the directory is gone, you can no longer read the disk. The H-77 instructions have a warning on just about every other page about this problem. The warning does little good after you have built the unit and someone else not familiar with the H-77 powers up your system only to destroy a few disks before asking why the H-89 will not BOOT.

There is a simple solution to this situation and it works! Actually, there are a couple of measures you can take to prevent this "disaster". First, if you only have two drives, both can be installed in the H-77. You can then place the storage box in your H-89 or place the front panel "plug" back on the H-89 so that it would appear to be an H-88. A second method would be the addition of a universal power strip with one power switch that would turn on all of your equipment at one time. The switch should be mounted in an out-of-the-way location so that you won't accidentally bump the thing when you are doing work on the system. The best method, however, is to simply interchange the SY0: plug located on the drive itself. This particular method would place your SY0: drive on the far right of a two or three drive system. Further, if you attempted to BOOT the system, the H-89 would now look to the H-77 for drive "0". If you did not have the power applied, the system simply would not boot. Therfore, you would never have to worry about loss of power to the H-77 when working on those valuable "goodies".

(These ideas were presented in a meeting of the San Jose HUG group on April 1, 1981, by Don Fiehmann and Dave Hildebrand. No! It was not an April Fools JOKE!) TNX to Don and Dave.

Local HUG News

CANAL-HUG located at the Panama Canal, Panama meets the first Tuesday of the month at 7:30 PM. The meeting place is currently Quarters 110A, Howard Air Force Base. For additional information you should contact Mike Gulick or Don Fricks at 84-4094 or 84-3031 (use overseas operator) or write these individuals at P.O. Box 1112, APO Miami 34001. They are looking for new members to join them in the expansion of their club.

Mark Hunt is interested in forming a new Heath Users' Group around the Bakersfield area. If you wish to contact Mark, call (805) 397-1509 or write 2135 South Union Avenue; Bakersfield, California 93307.

Fresno area users should contact Harlen Collins who is attempting to form a new group in his area. Harlen can be reached by calling (209) 291-6258 or by writing 4833 East Santa Ana; Fresno, California 93726.

Two California groups announce regular meeting schedules for their clubs. ANAHUG is currently meeting on the third Thursday of each month at the local Heath Electronics Center in Anaheim, California. Also, LUVAHUG is meeting on the second Thursday of each month at the Woodlawn Hills Heath Electronics Center; Woodlawn Hills, California. Both groups meet at 7:00 PM. For additional information please contact the Center nearest your location.

The Indiana Heath Users' Group or IHUG meets on the second Wednesday of each month at the Indianapolis Heath Electronics Center. Butch "Bent Hooks" Howard, the Manager of the Center (terrible fisherman), has requested that members bring chairs to the meetings or be prepared to sit on the floor. For further information please contact Jon Herbold; 2129 Braeburn Drive East; Indianapolis, Indiana 46219.

The LA-34 users guide contains an error on page 51. When using computer generated escape sequences to control the vertical pitch of the printer the coding called for in the guide should be a lower case "z" instead if a lower case "x".

Thanks to George S. Roth of Livonia, Michigan for this info!

HUG Meets at the WCCF

The HUG meeting scheduled for Saturday April 4, 1981, at the West Coast Computer Faire was much larger than we had ever anticipated. During our two week warm-up for the show, we guesstimated the attendance for the yearly meeting would be approximately 200 persons. Much to our surprise, there was standing room only and an overflow onto the main corridor of the fourth floor hall at the San Francisco Civic Center. The general response was positive. We received many good questions about the direction of the Heath Users' Group. I was fortunate enough to meet with and discuss many of our future goals with active members from all over the country. After a brief discussion about HUG, the floor was turned over to Barry Watzman, the Product Line Manager for Heath. For the next hour and a half, Barry answered questions for the members in attendance. He indicated that a new HDOS (3.0) was on the way along with several exciting products. Along with the new HDOS, he let us know that a new version of CP/M was heading our way. Barry mentioned that the new version of HDOS however, would be capable of supporting all 64K instead of the current 56K. He suggested that it would require the addition of the ORG. 0 modification. Further, comments that the new HDOS and CP/M would support both an "octal" density 5 1/4" drive system and a Winchester harddisk were aired. (octal density would equate to approximately 1.6 meg on twin 5's) Very interesting indeed!! A time frame of late fall/early winter was all that Mr. Watzman would commit to.

In response to a question about 16-bit systems, Barry mentioned a "NEW" mystery machine with some powerful advantages. He indicated though, that we would probably not see the "thing" until late '82 or early in 1983. He did, however, mention color, hard disk, etc. HMMMMM? Interesting huh?

Barry then fielded questions from the audience regarding topics of interest on HDOS and support from the Heath factory. He stated after the meeting "These Heath users sure have come a long way.....I was amazed at the questions I received and the positive response of the entire group this afternoon." He then walked away clutching the new H-8 Z80 board which he indicated should be released in July or August.

All in all, the WCCF was a fine show of the new products we can expect to see in the future. HUG presented the Color-Graphics demo as displayed at the Heath Booth during the entire show. These graphics programs are to be released shortly with all files included as a starting point for those of you just getting acquainted with this type of programming.

BE:

Print Using for B H BASIC

While it may not be too original, I wrote this simple program for use with Extended Benton Harbor BASIC because the language does not include the "PRINT USING" function. I thought others might find it as useful as I have. It simply sets the number of decimal places printed and eliminates the problem of precision that frequently causes an amount such as $10.54 to come out as $10.5399. This particular situation can get very annoying when trying to format in columns or when attempting to run an accounting program!

Edward Davie
21 Juliand Street
Bainbridge, NY
13733

```
00010 REM * Input decimal places
00020 INPUT "PLACES DESIRED ? ";P
00030 REM * Input desired number
00040 INPUT "ROUNDED NUMBER ? ";X
00050 PRINT "BEFORE ROUNDING X=";X
00060 GOSUB 1000
00070 PRINT "AFTER ROUNDING  X=";X
00080 REM * Do again
00090 GOTO 10


01000 REM * Rounding Routine
01010 X8=X-INT(X)
01020 X8=X8*10^P
01030 IF X8-INT(X8)>.5 THEN X8=X8+1
01040 X8=X8/10^P
01050 X=X8+INT(X)


01060 REM * Fix Decimal
01070 B9=INT(X)
01080 C9=X-B9
01090 C9=C9*10^P
01100 C9=INT(C9)/10^P
01110 X=B9+C9
01120 RETURN
```

(vectored to page 31)

H8 to H11 Interface

In issue 15 of REMark, I made a comment in "MOVIN' ON" that said I had "an H-11A, H-27 system interfaced to the H-8, H-19." Well, this is true but it is not what you might think "interfaced" means.

The interface comes from using the SOFTSTUFF(tm) modem package Computerized Phone System (CPS). This allows me to use the H-19, H-8 as the terminal while operating the H-11 system. To the MicroNET users this is nothing new. Instead of going through the telephone line, the two systems are connected direct.

Why do I have this interface between the H-8 and H-11? Well, I am able to store all that I do while operating the H-11 in the memory buffer of the H-8, and then able to copy it to the H-17 5 1/4 inch disk. I am also able to transfer a program from the H-8, H-17 to the H-11, H-27 by using the SEND feature of CPS into the EDIT program of the HT-11. (Any other modem packages will work in a similiar manner.)

Any transfer of files from one system to the other must be done in ASCII format. It is obvious that the H-11 processor is different from the 8080 microprocessor used by the H-8, therefore, this will prohibit us from copying object code from one machine to the other. Even though the two systems are "interfaced" this does not give you the liberty to transfer, copy, and store files interchangably from one system to the other. Actually the advantages of having this "interface" are extremely limited. From a programmers view there are some situations that may be of benefit but from the operations end there is really little advantage.

<TLJ>

(vectored from page 30)

SAMPLE RUN:

```
PLACES DESIRED ? 2
ROUNDED NUMBER ? 10.5399
BEFORE ROUNDING X=10.5399
AFTER ROUNDING  X=10.54
```

The variables in this easy program can be modified to fit your needs. Further, the value of "P" can vary between 0 and 6 depending on the numbers selected.

Editors Note: The program listed above can be replaced from lines 1000 to 1110 with this simple line:

```
1000 PRINT INT((10^P)*X+.5)/10^P
```

However, for learning purposes, the program was presented as written for its easy to follow "flow".

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

CUT ALONG THIS LINE

HUG MEMBERSHIP RENEWAL FORM

When was the last time you renewed?

Check your ID card for your expiration date.

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

Name ______________________________

Address ______________________________

City-State ______________________________

Zip ______________________________

REMEMBER — ENCLOSE CHECK OR MONEY ORDER

CHECK THE APPROPRIATE BOX AND RETURN TO HUG

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

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

Electronic Billboard

BY
Roy S. Reichert

Here is a little BASIC routine which can be fun. The routine will run under both B.H. BASIC and Microsoft BASIC. For B.H. BASIC, line 1000 is not necessay but can be left in if you desire. If the user wishes to change the number of lines of text that can be handled, simply change the value of "D" in line 1010. The program is most effective when each line of text ends with a short row of dots (Sample Line......) or spaces to separate one line from the next.

```
1000 CLEAR 2000
1010 D=20:DIM A$(D)
1020 N=1:E$=CHR$(27)
1030 PRINT E$;"E"
1040 PRINT "TICKER-TAPE or ELECTRONIC-BILLBOARD Program - By Roy S. Reichert"
1050 PRINT:PRINT "This program simulates a ticker-tape machine, or"
1060 PRINT "Times-Square-Building sign by displaying text in motion."
1070 PRINT "Enter up to";D;"lines of text......."
1080 PRINT:PRINT "Enter Line #";N;" (* to QUIT)":LINE INPUT ":";A$(N)
1090 IF A$(N)="*" THEN N=N-1:GOTO 1110
1100 IF N<D THEN N=N+1:GOTO 1080
1110 INPUT "Length of Banner (1 to 77) ";S
1120 IF S<1 OR S>77 GOTO 1110
1130 J$=CHR$(INT((80-S)/2)+32)
1140 PRINT E$;"E";E$;"x5"
1150 C$="":K=2
1160 FOR J=1 TO S:C$=C$+" ":NEXT J
1170 C$=C$+A$(1)
1180 IF K=N+1 THEN K=1
1190 L=LEN(C$)
1200 C$=RIGHT$(C$,L):B$=LEFT$(C$,S)
1210 PRINT E$;"Y+";J$;B$
1220 L=L-1:IF L=S THEN C$=C$+A$(K):K=K+1:GOTO 1180
1230 FOR J=1 TO 50:NEXT J
1240 GOTO 1200
```

Heath
Users'
Group
Hilltop Road
St. Joseph MI 49085

885-2017