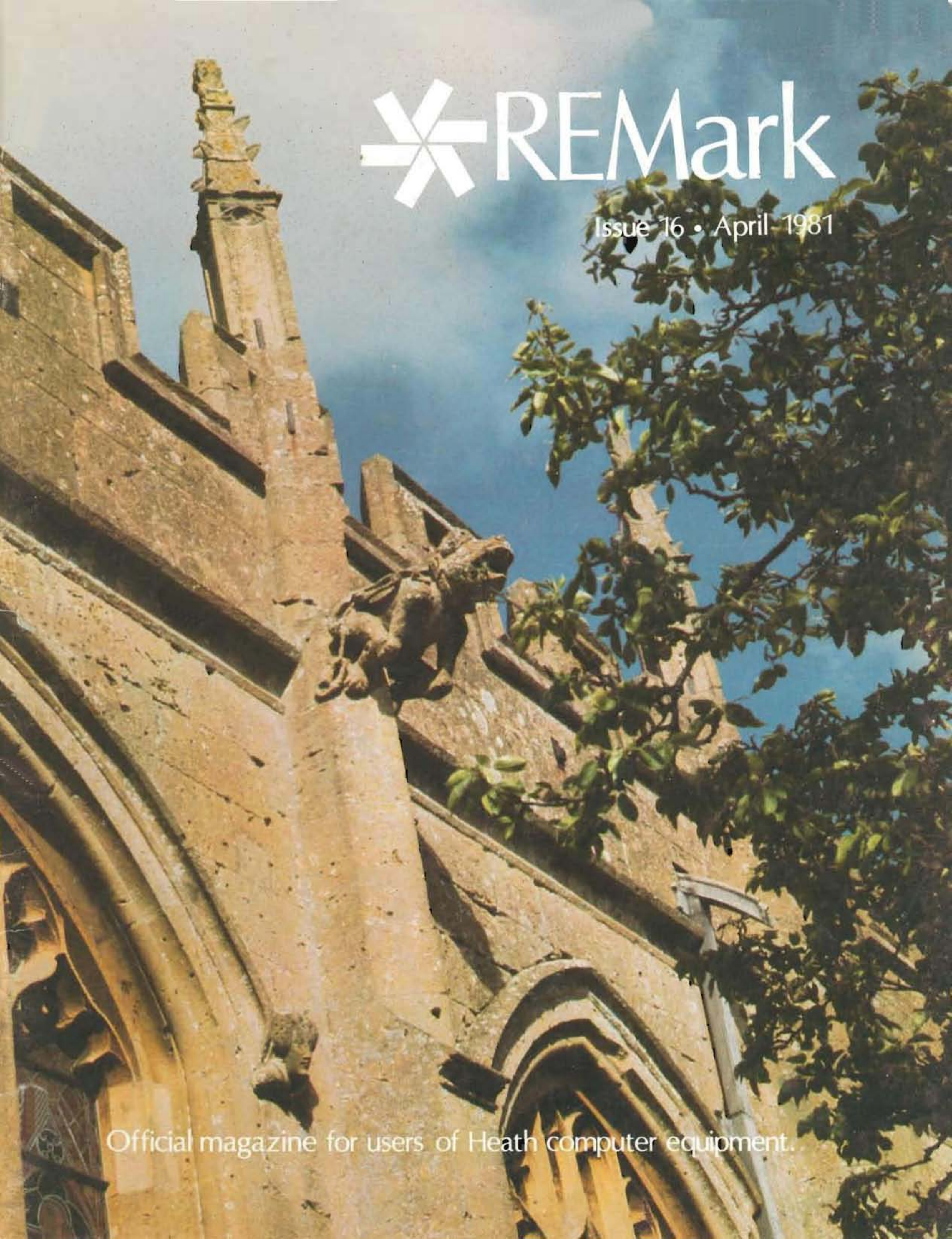
REMark

Issue 16 • April 1981



Official magazine for users of Heath computer equipment.

on the cover . . . .

Sudeley Castle near Cheltenham Spa, England
(See DND Page 16 for real adventure!)

Photo by Gerry Kabelman

on the stack

>CAT

"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

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

REMark

A KISS for Assembly Programming

A CONTINUING ENCOUNTER

FROM REMark ISSUE #15

In Issue #15 of REMark, we discussed a simple Assembly Language program that would enable the beginner to become familiar with some of the techniques used as building blocks for this particular method of programming a computer to do work. If you remember, we described this program as being runnable on both the H8 and the H89. Also, our program was to "ring" the "bell" used by both of these computers five times. You may find it helpful to review the "KISS" article presented in Issue #15 before proceeding with "the second edition" so that you may fully understand our intended goals.

As suggested earlier, we are going to "try" to learn something about the HDOS Editor (EDIT) and Assembler (ASM). The Editor will be used in this situation to create the "source" file. In turn, we will then use the source file (.ASM) to create the "listing" file (.LST) and the "runnable" file (.ABS) by making use of the HDOS Assembler (ASM).

THE HDOS EDITOR

EDIT, the Editor used by HDOS, has been implemented by most individualsto create text files i.e. letters, recipes, lists, etc. For those of you that have not explored the use of EDIT's features, it may be helpful to examine your HDOS Software Reference Manual. However, we will discuss some of EDIT's "commands" here including DELETE, INSERT, PRINT, REPLACE, NEWOUT, READ, NEWIN, and BYE. Also, to prevent future confusion, it must be remembered that the Editor uses a thing called "command completion". Command completion causes a completed spelling of a partial word. As an example: typing "I" after the EDIT prompt invokes INSERT or typing "D" invokes DELETE, both words (INSERT and DELETE) will be spelled for you. The point of this caution is aimed at proficient typists out there that will find "garbage" on the screen if their fingers get carried away. Further, some words such as "NEWOUT" require that a "NWO" be typed to finish the word. I will try to catch these special commands as we go.

GET IT IN....

NOTE: Before beginning to input our project, it would be best to have only the Assembler and Editor on disk. This will ensure plenty of disk space for "playtime" and later experiments you may wish to perform. You can INIT and SYSGEN a disk, then use PIP or ONECOPY to transfer EDIT and ASM to the finished disk. Please review your manual for operating procedures if necessary.

Beginning our programming task requires that we "load" the EDITor into our computer. Once we have obtained the HDOS prompt (>), we type EDIT followed by a carriage return (RETURN KEY). When loading is complete, the EDITor will respond with its' prompt (--) indicating we are ready to begin entry. Type "I". EDIT will respond with "INSERT" (--INSERT). Type RETURN. EDIT will then move the cursor to a blank line to begin the actual INSERTing of text.

Please examine the example given (fig. A). (This program, in text form, is exactly the same program shown in Issue #15 with the comments, stars, and extra "goodies" removed. Issue #15 fig. B). Note that there are four numbered "fields" as shown at the top of the text (1. 2. 3. 4.). Each of these fields is an important component in our completed project. EACH FIELD IS ACCESSED BY PUSHING THE TAB KEY.

TYPE:

```
START(TAB)ORG(TAB)100000A(TAB)COMMENTS
```

In this example we have entered the first line of the source listing beginning with the word "START". The TAB key is pushed to enter the next "field".

THE LABEL FIELD

The first field is known as the "LABEL" Field and is shown under the #1 at the top of our example (fig. A). The Label Field is used to indicate loops or GOTO line numbers when compared to BASIC.

Also, the Label Field is used to define words selected by the programmer (more on this later).

THE MNEMONIC FIELD

The second of the four fields is defined as the "MNEMONIC" Field. This field uses the predefined mnemonics or abbreviations that the that the Assembler translates for the CPU. Examples of defined mnemonics would include MOV, MVI, JMP, etc. These mnemonics could also be called command words or instructions for the CPU similiar to PRINT, GOTO, or READ in BASIC.

THE ARGUMENT FIELD

The third field is the "ARGUMENT" Field. This field supplies the "argument" for the mnemonic supplied by field #2. As an example: "MVI" of the mnemonic field is given the argument of B,FIVE which translates to "MOVE IMMEDIATE to the B register the number FIVE" or MVI B,FIVE.

THE COMMENT FIELD

The easiest field would most certainly be #4 or the "COMMENT" Field. This field is used to "keep notes" on the progress of your program. You would normally use this area to explain your "moves" similiar to a REM statement in BASIC. Examine the comments of fig. B in REMark Issue #15. The comments and extra B... were removed to show the "bare bones" program in this issue.

```
1.      2.      3.      4.
***************************

START   ORG     100000A COMMENTS
        MVI     B,FIVE
LOOP    CALL    HORNO
        MVI     D,TIME1
LOOP2   MVI     E,TIME2
LOOP3   DCR     E
        JNZ     LOOP3
        DCR     D
        JNZ     LOOP2
        DCR     B
        JNZ     LOOP
        JMP     HDOSP
*****
FIVE    EQU     005Q
HORNO   EQU     002136A
TIME1   EQU     377Q
TIME2   EQU     377Q
HDOSP   EQU     040100A
*****
        END     START
```

fig. A

SUMMARY ON FIELDS

There are FOUR individual FIELDS each of which is separated by the TAB function of EDIT. As we type the example shown in fig. A, we ACCESS each FIELD with a TAB after a completed entry. FIELD 1.=LABEL. FIELD 2.=MNEMONIC. FIELD 3.=ARGUMENT. FIELD 4.=COMMENTS.

```
1.      2.      3.      4.
START   ORG     100000A COMMENTS
```

For the most part, we have covered the essential "tools" required to type the program shown in fig. A. After you have typed the info contained in each field for each line, you will type a RETURN to complete the line. You may add comments if you wish (field #4) as shown in fig. B Issue #15 of REMark or supply your own "notes"....whatever!

WHAT ABOUT MISTAKES!!!

If some of you are like I am, you tend to fumble around on a keyboard causing characters to appear that you didn't necessarily want or need. NO PROBLEM! At this point, type a RETURN and retype the line "accurately"! Continue this process until you have completed the text listing shown in fig. A. ....
....
....
By now probably several hours have gone by and the wife is P.O.ed! You've probably noticed that I didn't tell you everything. But, what better way to learn than what a little experience can provide! You should now be at the last line waiting further instructions.

OUT OF "INSERT"

Type a Control-C or hold the CTRL key and the "C" key down together. This action should cause the EDITor to return to its' prompt (--). If we then wish to view the mess we have made of typing the program press the "space" bar followed by "P" for PRINT. Again EDIT will complete the word "PRINT" for us.

NOTE: If your listing exceeds one page on your terminal, CTRL-S is used to stop printing and CTRL-Q is used to resume.

CORRECTING MISTAKES

OK...Let's ASSUME that some of us were not perfect!!! (WHO? ME!) We find that it becomes necessary to correct some of the lines for various reasons (refer to fig. B). We want to correct line #4. It must be remembered that we have to set EDIT's line "pointer" to the appropriate line position. To accomplish this, we type "^+3P" (--^+3PRINT). EDIT will respond by printing the line. Now we have several options open to us. We can DELETE an extra line, we can REPLACE a "defective" line, or INSERT more lines as necessary.

DELETE A LINE

Type "D". EDIT will respond with DELETE. If you now type RETURN, the line magically vanishes! If you type CTRL-C, the DELETE function is disabled.

REPLACE A LINE

Type "RP" (two letters). EDIT will respond with "REPLACE". A RETURN will allow you to retype the line as necessary. Again, CTRL-C disables the REPLACE command.

INSERT MORE LINES

Type "I". EDIT responds as demonstrated earlier. A RETURN will allow us to enter more text and CTRL-C will return us to the command mode.

NOTE: Good practice dictates that we re-print our listing after using any of the commands shown above. This is done to ensure we have correctly modified the text. Accuracy is essential if our finished project is to operate. To print the text, press the "space" bar followed by the letter "P". If you fail to press a space before "P", only the first line will be printed.

```
START  ORG    100000A COMMENTS
       MVI    B,FIVE
LOOP   CALL   HORNO
ERROR JUNK ERROR JUNK ERROR JUNK
LOOP2  MVI    E,TIME2
LOOP3  DCR    E
       JNZ    LOOP3
       DCR    D
       JNZ    LOOP2
       DCR    B
       JNZ    LOOP
       JMP    HDOSP
*****
FIVE   EQU    005Q
HORNO  EQU    002136A
TIME1  EQU    377Q
TIME2  EQU    377Q
HDOSP  EQU    040100A
*****
       END    START
```

fig. B

USING THE ($) AND THE (^)

To invoke the commands of EDIT described above, we must know the position of the "line pointer" or we will obtain some very unexpected results (i.e. missing lines, misplaced lines, etc.). Accomplishing proper editing of our text requires that we use the ($) and (^) to "find" the correct line.

The ($) is used to indicate the bottom of our text or the last line of text. If we type "$P" the EDITor will respond with --$PRINT and a RETURN will cause the last line of text to be printed.

```
       END    START
```

Further, if we type "$-1P" and a RETURN the EDITor will print the next to the bottom line.

```
*****
```

Each line can be counted and every line should be counted to give a proper location of the line we wish to edit. The ($) operates from the bottom of our text using minus (-) numbers for location while the (^) works from the top down using positive (+) numbers.

```
--^+3PRINT (RETURN)
       MVI    D,TIME1
```

Remember the importance of the "line pointer" when editing your text. And, don't forget to use "PRINT" before any of the EDIT commands such as DELETE!

THE OUTPUT FILE AND FILE NAME

Some practice may be required to "build" the desired text file shown in fig. A. However, once you have completed the necessary typing, your text should look exactly like the text in fig. A. Compare your work against this example for accuracy. We have now reached the point of saving our text on disk.

```
--NEWOUT....
```

NEWOUT is one-half of the necessary command to save our project to disk. NEWOUT is invoked by typing "NWO" followed by our file name.

```
--NEWOUT/BEEP.ASM/
--
```

Above, I have used the file name "BEEP" followed by the extension ".ASM" to indicate that our "NEWOUT" will be a "source" file named "BEEP". EDIT will will respond by placing the name ONLY on the disk. You will then type "BY" to complete the transfer of the text to disk.

(vectored to page 30)

Create, Change, Display, and Print an Address File

(For Microsoft BASIC)

Need an address or a Christmas card list, but yet don't want a complex system that requires a PHD to operate. The sample program included with this article will allow you to create, add to, change, display and print a simple mailing list. Our purpose in presenting this program is to teach you how to handle string files.

The progam does have limitations. It will NOT sort the list. You can not select individual names to be printed. You must make changes according to the order that the names are on the disk and as the names increase it will become quite slow. However if you need a cheap and easy way to get a mailing list program, here it is.

The program is divided into seven basic sections:

1. Setup
2. Menu
3. Add Routine
4. Change Routine
5. Print Routine
6. Subroutines
7. Error Messages

A complete discription of the program is included for the beginner. If the desciption is not needed proceed directly to the STARTUP NOTES:.

SET UP

Line 10 -- Simply a title line complete with the name of the program, brief description, version number, which actually is the date of the last revision.

20 -- Clears 2000 bytes of string space, sets the terminal width to 255 (for graphic functions), and tells the program to go to line 20010 if there is an error.

30 -- Sets up the following string variables:

E$	= (CHR$(27) (the ESC key)
E1$	= clear the screen
P$	= enter reverse video
Q$	= return to normal video
Y$	= direct cursor addressing (+ address)
Y5$	= turn cursor on
X5$	= turn cursor off

40 -- Also sets up string variables:

J$ = Go to the location fourth row down from the top, first character from the right (more on direct cursor addressing later) and clear remaining screen. This is done to leave the title at the top of the screen.
WI$ = Set the cursor near the middle of the screen, clearing everything below and to the right of that point. Also print the words "What is the" at that location.

50 -- This line retrieves the date from memory which was entered when the system was booted up. The memory locations 8383 to 8391 contain the date after bootup.

60 -- A simple spacing line.

MENU

100 -- Prints E1$ to clear the entire screen, P$ to go into reverse video, Y$ to start the direct cursor addressing mode. The sequence of the ESC (escape) key followed by a capital "Y" and two characters will select a position on the screen of the terminal. The first character will determine the position down from the top. The second character will select the position to the right of the left margin. The actual ASCII value (ASCII is the codes that the computer understands for each character, see your computer manual for an ASCII code table) of a character such as the space is 32. The screen is set up so that the HOME position is 32-32 or space-space. As the ASCII value is increased the position on the screen changes. In this line (100) we are using Y$"!3", this will print on the second line from the top and the twentieth column (position) to the right, followed by the title of the program and the Q$ to exit reverse video. Note the spaces before and after the title to give a little fancier look to it. Try it both ways and I think you'll agree. The colon (:) and the apostrophe (') means the rest of the line is a comment (REMark).

110,120,130 -- Print the menu options on the nineth, tenth, and eleventh lines respectively in the sixteen column to the right of the left margin.

```
10 '    SAMPLE.BAS                        File Sample--ADD, EDIT & PRINT  @
                                          Version 02.27.81
20 CLEAR 2000:WIDTH 255:ON ERROR GOTO 20010
30 E$=CHR$(27):E1$=E$+"E":P$=E$+"p":Q$=E$+"q":Y$=E$+"Y":Y5$=E$+"y5":X5$=E$+"x5"
40 J$=Y$+"# "+E$+"J":WI$=Y$+"00"+E$+"J"+"What is the "
50 D$="":FOR D=8383 TO 8391:D$=D$+CHR$(PEEK(D)):NEXT D
60 :
100 PRINT E1$P$Y$"!3 Mail List Program "Q$:'            Menu
110 PRINT Y$"(01. ADD to list"
120 PRINT Y$")02. CHANGE information"
130 PRINT Y$"*03. PRINT list"
140 PRINT P$Y$"/0 Your choice? (1 TO 3) <END> ";:GOSUB 10030
150 ON VAL(A$) GOTO 1000,2000,3000:END:'          OR SYSTEM
160 :
1000 GOSUB 10000:'                                      Add Routine     #1
1010 IF EOF(2) THEN 1030
1020 GOSUB 10040:GOSUB 10050:GOTO 1010
1030 H=1:PRINT J$Y$"#6 Add Routine "Y$"% Hit the RETURN to <END> ";:A=1:GOSUB 10060
1040 IF A$(1)="" THEN H=0:CLOSE:GOSUB 10110:GOTO 100
1050 PRINT Y$"% "E$"J"Y$"%0"A$(1)Y$"&0";:A=2:GOSUB 10070:PRINT Y$"&0"A$(2)
1060 A=3:GOSUB 10080:PRINT Y$"'0"A$(3)
1070 A=4:GOSUB 10090:PRINT Y$"0$"A$(2)
1080 GOTO 2040
1090 :
2000 GOSUB 10000:'                                      Change Routine  #2
2010 IF H=1 THEN 1030
2020 IF EOF(2) THEN CLOSE:GOSUB 10110:GOTO 100
2030 GOSUB 10040
2040 PRINT J$Y$"#3 Change Routine "Y$"%01. "A$(1)Y$"&02. "A$(2)Y$"'03. "A$(3)
2050 PRINT Y$ "(04. "A$(4)Y$"20If not enter which number you wish to change. (1-4)"
2060 PRINT Y$"00Are these, OK? <YES> ";:GOSUB 10030
2070 IF A$=CHR$(13) OR A$="y" OR A$="Y" THEN GOSUB 10050:GOTO 2010
2080 A=VAL(A$):ON A GOSUB 10060,10070,10080,10090:GOTO 2040
2090 :
3000 PRINT J$Y$"#6 PRINT LIST ":'                       Print Routine   #3
3010 PRINT Y$"00 Do you want Hardcopy?  (Y or N) <Y> ";:GOSUB 10030
3020 GOSUB 10010:IF A$="N" OR A$="n" THEN LP$="TT:" ELSE LP$="LP:"
3030 OPEN "O",3,LP$:PRINT #3,TAB(10)"Name List as of "D$:PRINT #3,
3040 PRINT J$Y$"* ";:IF EOF(2) THEN CLOSE:GOTO 100
3050 GOSUB 10040:FOR I=1 TO 4:IF I=3 AND A$(3)="" THEN 3070
3060 PRINT #3,TAB(10)A$(I)
3070 NEXT I:IF A$(3)="" THEN PRINT #3,:
3080 PRINT #3,:IF LP$="TT:" THEN CLOSE #3:GOSUB 10020:GOTO 3030
3090 GOTO 3040
3100 :
10000 OPEN "O",1,"TESTO.DAT":'                          Subroutines
10010 OPEN "I",2,"TESTI.DAT":RETURN
10020 PRINT P$Y$"50 Hit any key to continue! ";
10030 PRINT Y5$Q$" ";:A$=INPUT$(1):PRINT X5$;:RETURN
10040 FOR I=1 TO 4:LINE INPUT #2,A$(I):NEXT I:RETURN
10050 FOR I=1 TO 4:PRINT #1,A$(I):NEXT I:RETURN
10060 PRINT WI$"name";:GOTO 10100
10070 PRINT WI$"1st street address";:GOTO 10100
10080 PRINT WI$"2nd street address";:GOTO 10100
10090 PRINT WI$"City, ST ZIP--CODE";
10100 PRINT Y5$"? ";:LINE INPUT A$(A):PRINT X5$;:RETURN
10110 KILL "TESTI.DAT":NAME "TESTO.DAT" AS "TESTI.DAT":RETURN
20000 '                                                 Error Messages
20010 IF ERL=10010 AND A$="2" THEN @
      PRINT Y$"3# You must start a file first!!":GOSUB 10020
20020 IF ERL=10010 THEN OPEN "O",2,"TESTI.DAT":CLOSE:RESUME 1000
20030 PRINT "ERROR #"ERR"IN LINE #"ERL
```

140 -- Prints in reverse video at location "/0" (line 16 column 16) the words "Your choice? (1 to 3) <END>". The "<END>" shows what will happen if the RETURN key is pressed. The semi-colon (;) is used to keep the cursor from dropping to the next line for the gosub statement. The GOSUB's will all be explained later.

150 -- This line will select which option you have chosen for the menu. The VAL(A$) is used since the input of a character was done in string form and we need a numerical value for the "ON GOTO" function.

160 -- Same as line 60.

ADD ROUTINE

1000 -- Is the beginning of the ADD ROUTINE. The first thing we do is GOSUB 10000 to open the input and the output files. Again we also show a REMark to help retrace our path at a later time.

1010 -- Checks to see if file #2 is at the END (EOF(2) is equal to The End Of File #2). If it is then we GOTO 1030.

1020 -- First we GOSUB 10040 to retrieve information from the present mailing list file (TESTI.DAT). Then we GOSUB 10050 to output to the new file (TESTO.DAT). After retrieving and outputting one set of name and address then we GOTO 1020 to check for another name and address. This process is repeated until the EOF is found then we go to line 1030.

1030 -- First we set an indicator (H=1) to indicate that we are in the add routine for use in the change routine. Then we print J$ which clears the screen below the title, using direct cursor addressing we print the title for this routine, leave a message telling us if we have any more names and addresses to be entered to hit the RETURN key to end. The A=1 is another indicator, this time it is to tell a subroutine that we are requesting the NAME of the individual to be entered to the list. We then proceed to the subroutine at line 10060.

1040 -- Checks to see if A$(1)="" and if it is, that means that the RETURN key was hit telling us that the end of the new entries has been reached. We then set the indicator "H" to zero, close all files, GOSUB 10110 to remove the input file and rename the output file to the new input file. We then return to menu at line 100.

1050 -- If A$(1) was not equal to "" in line 1040 then we proceed to this line where we clear the screen, below the subtitle by printing E$ followed by a capital "J". We then print the string A$(1) at location "%0" (line 6 column 17). We are now ready for the second item to be entered. The indicator is changed (A=2) and we go to the subroutine at line 10070. Again we print the results, this time at "&0".

1060,1070 -- Select items three and four printing them at selected locations.

1080 -- Go to the CHANGE routine before we return to line 1030 for another name and address. This allows checking and correcting the entry before the next is entered.

1090 -- Same as line 60.

CHANGE ROUTINE

2000 -- Same as line 1000.

2010 -- Checks to see if this is the ADD routine and if it is, returns to 1030.

2020 -- Checks for the end of file number two, if it is the end, then GOSUB 10110 and return to 100.

2030 -- Goes to the subroutine at 10040 to input the first name and address.

2040 -- Clears the screen except the master title, prints the subtitle and the first three pieces of the first name and address.

2050 -- Prints the fourth piece of information and prints information on what to do if one of the four items is incorrect.

2060 -- Prints the question "Are these, OK?" with a default of "YES" and goes to the subroutine at line 10030 for an input (A$).

2070 -- Checks "A$" for a carriage return or the letter "Y" either in upper or lower case. If the check proves true then we GOSUB 10050 to output the data and then return to 2010 for the next input.

2080 -- If 2070 was false the variable "A" is made equal to the value of the string variable "A$". Next, on the value of "A" we GOSUB to the appropriate line number, (A=1, GOSUB 10060; A=2, GOSUB 10070; etc.) to ask the question about the incorrect piece of information and change it. Then GOTO line 2040 for the next input.

2090 -- Same as 60.

PRINT ROUTINE

3000 -- Clears the screen except the main title, prints the subtitle and includes a REMark to help keep track of the program.

3010 -- Prints the question about hardcopy and asks for an input (A$) through the subroutine at line 10030.

3020 -- Goes to the subroutine at line 10010 for opening the input file for reading. If "A$" from the previous line was the letter "N" then the output file (LP$) becomes the terminal (TT:) otherwise the output file is the line printer (LP:).

3030 -- First we open the output file for file number three and then we output to the output file (#3) a tab of ten, the title, the date (D$) and a blank line for spacing.

3040 -- Clears the screen and positions the cursor on the line number eleven in the first space. A check is done for the end of file for file number two and if it is we close all files and goto 100.

3050 -- The first data to be printed is input by executing the subroutine at line 10040, and then a loop is started which will print the information. We also perform a check here to see if the third item is blank, and if it is, then we jump to 3070 for the next item.

3060 -- This line prints a tab of ten and the item identified as A$(I) where the "I" is a number between one and four from the loop started in line 3050.

3070 -- We continue the loop with the next statement, and after the loop is finished a check is done on "A$(3)" to see if it is a blank line. If it is then we output a blank line. The extra blank line is to keep the spacing from the first line of a name and address the same to the next.

3080 -- Another blank is outputted and a check is done to determine if the output device is the terminal (TT:). If it is, then we close the output file and GOSUB 10020 to ask you to hit a key after having read the information. Then we go to line number 3030.

The output file is closed if we are using the terminal because all data is sent to an output device in blocks of 256 characters except when a file is closed; then all remaining characters are sent. In this case most of the information being sent to the output device will be much less than the 256 characters, so we close the file to allow you to see it all.

3090 -- Is used to return to the next item when the output device is a line printer (LP:).

3100 -- Same as 60

SUBROUTINES

10000 -- This line starts the series of subroutines used by other portions of this program. This line is to open the file "TESTO.DAT" for output as file number one.

10010 -- Opens the input file "TESTI.DAT" for input as file number two. We then return to the line of the GOSUB.

10020 -- This line goes into reverse video and asks you to hit any key to continue. The semicolon (;) is used to allow the next line to continue after this line without dropping to the next line.

10030 -- First prints the "Y5$" to turn the cursor back on then a "Q$" to make sure the display is not in reverse video. Now "A$" is input from the terminal as only one character. Next the "X5$" is printed to turn the cursor off and we return to where we came from.

10040 -- This line is a loop to input four items from the input file (#2) and then to return.

10050 -- This line is a loop to output four items to the output file (#1) and then return.

10060 to 10090 -- Will print the question "What is the", the item to be changed or added and then goes to 10100.

10100 -- Turns on the cursor, prints the question mark (?), inputs a new value for the "A$(I)" where "I" was determined by the routine using this subroutine. The cursor is shut off and we return to the routine.

10110 -- Caution: This line can destroy files, so be careful to type it correctly and use it only when needed. The first thing we do is kill the input file (TESTI.DAT) and then rename the output file (TESTO.DAT) to the input file. We then return to where we came from.

ERROR MESSAGES

20000 -- This line is used to separate the error messages from the rest of the program.

20010 -- This line checks for an error in line number 10010 and that A$="2". If this is true then we print a message telling the user that a file must be started before it can be changed. This line continues onto line 20020.

20020 -- Checks for an error in line 10010 and then opens an output file as number two and immediately closes the file. The RESUME takes you back to the ADD routine to start a file.

(vectored to page 25)

Safe Disk Reset

On my home system I have two 5-inch disk drives, and with them I work on some fairly large files (for example the source for FOCAL-8), so I use stand-alone quite a bit and do a lot of disk RESETting. The first time I did a RESET, I wondered if it was safe, so I looked into the drive and saw that the head load arm is pulled away from the disk when the door is opened even though the drive solenoid is engaged. I thought that all was OK until I took a closer look into the drive and saw that a little foam pad that holds the disk against part of the drive frame does not disengage when the door is opened with the solenoid engaged. This pad is below and behind the spindle. Each time you remove and replace a disk in RESET, it rubs against this pad.

It seemed to me that the pad might wear out eventually, so I wrote the program in the listing below. This little program uses the .MOUNT and .DMOUN SCALL's to accomplish a reset so that the drive solenoid is not engaged when you open the door. If the source is named R.ASM, and the resulting .ABS file (R.ABS) is placed on your system disk, then giving the command R or R 0 will reset SY0:, the command R 1 will reset SY1:, and the command R 2 will reset SY2:. If you enter another number or character, the program will just return control to HDOS. The only difference from a normal reset is that you must press RETURN after you insert the new disk. The program prompts you to do this by printing "Replace disk in SYn:, hit RETURN" on the console (n in "SYn:" is the drive number). If no disk was already mounted in the drive, the prompt is "Insert disk in SYn:, hit RETURN".

You must SET HDOS STAND-ALONE to use this program to reset SY0:. You only need to do it once for each system disk, because when you set STAND-ALONE, a flag is set and recorded on the disk indicating the STAND-ALONE condition. The next time you turn on your computer and boot that disk, the flag will still be there, and will remain set until you SET HDOS NOSTAND-ALONE.

If you never use STAND-ALONE, but would like to use this program to reset drives SY1: and SY2:, you can make the following changes. Change the line JZ RZERO to JZ BAD, and the line CPI '0' to CPI '1'. After these changes, the program will not reset drive SY0:.

Another change you can make to this program is to use the .MONMS and .DMNMS SCALL's instead of .MOUNT and .DMOUN. If you do this, the only message printed on the screen will be the prompt to hit RETURN.

PS:

```
        TITLE   'R.ASM -- SAFE DISK RESET'
        STL     'by Patrick Swayne 27-FEB-81'

* THIS PROGRAM RESETS DISK DRIVES BY DISMOUNTING
* AND MOUNTING INSTEAD OF RESETTING.  THIS METHOD
* IS SAFER FOR BOTH DISKS AND DRIVES.

.SCIN   EQU     1               SINGLE CHARACTER INPUT
.LOADO  EQU     10Q             LOAD OVERLAYS
.CLEAR  EQU     55Q             CLEAR CHANNEL
.MOUNT  EQU     200Q            MOUNT DISK ON DRIVE
.DMOUN  EQU     201Q            DISMOUNT DISK
$TYPTX  EQU     31136A          TYPE TEXT ON CONSOLE
WBOOT   EQU     40100A          WARM BOOT ENTRY

        ORG     42200A          NORMAL USER ENTRY
START   LXI     H,0
        DAD     SP              LOCATE STACK POINTER
        MOV     A,L
        CPI     80H             ANY ARGUMENT ENTERED?
        JZ      RZERO           IF NOT, RESET SY0:
GETNUM  MOV     A,M             GET ARGUMENT CHARACTER
        CPI     40Q             IS IT A SPACE?
        INX     H               INCREMENT POINTER
        JZ      GETNUM          IGNORE SPACES
        CPI     '0'             LESS THAN ZERO?
        JC      BAD             IF SO, EXIT
```

(listing continues on page 12)

Attention H11 Owners (Hard Disk Group Purchase)

The purpose of this information is is to see if you are interested in a possible group purchase of a hard disk subsystem for your machine. I would be willing to coordinate such a purchase if there is sufficient interest.

I currently know of at least three controllers that are available at the board level and work with the new eight-inch hard disks. The controller that I am proposing, however, is built by Andromeda Systems, a company in Los Angeles that builds accessories for the LSI-11 and which I have had very good experience. Their controller handles both eight-inch hard disks with the Shugart SA1000-compatible interface, and floppies (four drives of each type). The hard disk portion of the controller emulates the DEC RK05 drive and the floppy portion emulates the RX02 (double density). This means that operating systems like RT-11, UCSD Pascal, UNIX, etc. can be run directly without having to mess with the handlers. (You can patch the RK05 handler to vary the amount of space on each volume--it comes from DEC set for 2.5 megabytes per volume, but you can make it a lot bigger if you want to, depending on the capacity of the hard disk itself.) The reason this controller is attractive is that it would permit those of us who already have floppy systems to sell our current controllers to offset the price of the new one. (Also this would save slots in the backplane. For those of you with H-11s, it fits on one dual-width card.)

Andromeda's single-unit price for this controller is $2,000. In quantities of 10-24 this price drops to $1,800; 25-49 is $1,700; and 50-99 is $1,550. I didn't ask about quantities above 99 because I can't imagine getting that many participants, but, if we do it will probably drop another couple of hundred dollars.

The other part of the group purchase would be for the drives themselves, and possibly a box and powersupply for the drives. As I mentioned, the Andromeda controller interfaces with the Shugart SA1000 drive, which is becomming a de facto standard for the eight-inch disks. This means the controller also talks to drives like the Seagate five-inch mini-Winchester, the Quantum drive (which offers up to 40-megabytes, I think), as well as the 10-megabyte SA1000. There are undoubtably other drives that would be useful as well and I am investigating these--please let me know of any that you have seen actually working and that have the correct interface. I don't think we will want to go with the mini-Winchesters because they cost about as much as the eight-inch drives, are slower, and have less capacity. Prices for the drives are still unclear, depending on where we buy them and whether we buy bare drives (you provide your own powersupply and box) or whether we get boxed and powered units. I would guess that bare SA1000 drives in quantities of 50-99 would cost about $1,100, and boxed units would be about $2,000 in the same quantities.

If you are potentially interested in such a group purchase, please write and let me know. If there is enough response, I will put out another notice in about a month with more details on exactly what we will be buying, how much it will be, and how the money is to be handled. On this latter point, I am going to try to set up some kind of escrow account at a bank that will protect you, but at the same time will guarantee that the money will actually be there when needed. (This may be somewhat complicated. If any of you have done something similiar, let me know how you did it.) I am fairly sure that by offering cash, we can reduce the above quoted prices a little further, but this is still up in the air.

Assuming there is enough interest, I am tentatively planning to adhere to the following schedule:

```
Cut off for expression of interest................May 30
Formal proposal mailed to interested parties.......June 15
Cutoff for checks in escrow account................June 30
Purchase...........................................July 15
```

It is likely that the actual shipments will be spread out over a month or two, depending on how many we order. Shipping priorities will be determined by the date your check reaches the escrow account.

Please let me know if you are interested in this idea. If you don't hear from me by June 25, 1981, you can assume that there wasn't enough interest and that I have dropped the attempt. Please, also, pass along the word to any friends or associates who may be interested, including companies, universities, etc. A hard disk system enhances the utility of the LSI-11, and I hope we can put together a large, supportive group to make it financially attractive as well.

For further information contact

Jim McCord
330 Vereda Leyenda
Goleta, CA 93117
Telephone: (805) 963-6589 (days)
(805) 968-6681 (evenings)

```
***************************************
* NOTE: THIS INFORMATION IS SUPPLIED *
* AS A SERVICE TO H-11 OWNERS.   THE *
* PURCHASE PLAN IS NOT SUPPORTED  BY *
* HUG OR HEATH.  THIS PLAN IS MERELY *
* REPRODUCED HERE FOR YOU REVIEW.... *
***************************************
```

(If you want to call, I'd prefer that you do so during the day. I'm usually there from about 8:30 am to 6:00 pm Pacific time.)

EOF

(listing continued from page 10)

```
         CPI     '3'               MORE THAN TWO?
         JNC     BAD               IF SO, EXIT
         STA     DNAME+2           CORRECT MESSAGE
         STA     DRIVE+2           SET UP DRIVE NUMBER
RZERO    MVI     A,0FFH
         SCALL   .CLEAR            CLEAR CURRENT CHANNEL
         XRA     A
         SCALL   .LOADO            LOAD FIRST OVERLAY
         JC      BAD
         MVI     A,1
         SCALL   .LOADO            LOAD SECOND OVERLAY
         JC      BAD
         LXI     H,DRIVE           POINT TO DRIVE
         SCALL   .DMOUN            TRY TO DISMOUNT DISK
         JNC     REPLACE           GOOD DISMOUNT, TYPE "REPLACE"
         CALL    $TYPTX            NO DISK, TYPE "INSERT"
         DB      12Q,'Insert',240Q
         JMP     TDISK
REPLACE  CALL    $TYPTX
         DB      12Q,'Replace',240Q
TDISK    CALL    $TYPTX
         DB      'disk in '
DNAME    DB      'SY0:, hit RETURN',212Q
         SCALL   .SCIN             WAIT FOR RETURN
         JC      *-2
         LXI     H,DRIVE
         SCALL   .MOUNT            MOUNT DISK
BAD      JMP     WBOOT             RETURN TO HDOS
DRIVE    DB      'SY0:',0
         END     START
```

EOF

SOFT BUGS
SF-9005 INVENTORY PACKAGE

The following line numbers should be changed under the programs listed to correct or improve the performance of the respective components of the Inventory package.

START.SFS

```
20 CLEAR 2000:ON ERROR GOTO 360
210 IF D<15 THEN D=D+12
```

NEWMONTH.SFS

```
20 CLEAR 1000:ON ERROR GOTO 460
190 IF D<15 THEN D=D+12
```

ADD.SFS

```
20 CLEAR 4000:ON ERROR GOTO 420
600 FOR G1=1 TO G:IF P1$=G$(G1) THEN 630
    ELSE NEXT G1
```

CHANGE.SFS

```
20 CLEAR 2000:WIDTH 255:ON ERROR GOTO
    830
```

Manipulation of String Data with MBASIC

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

Abstract

MBASIC's MID$, LEN, and INSTR functions are used to isolate certain items in records and then strings are rearranged. A practical use of same is demonstrated. Of most value to newcomers to MBASIC.

Note: Reference is made to data files, records and fields. These terms are defined in my DBMS article in REMark, issue 14. Reference is also made to certain programs which were in Random File article in REMark, issue 10 (corrections to this article in REMark, issue 11.) Note also that the correct text of both those articles, along with all 21 programs are available on 5 inch diskette postpaid for $10.00. Write Mr. Jerry Leon, Heathkit Electronic Center, 630 Lancaster Pike, Frazer, PA 19355 and enclose check for same if you desire the diskette (HDOS format).

It is frequently necessary to dissect string data and rearrange the string. For example, assume you have a sequential mailing list file with each record formatted as follows (the reverse slash is used as a delimiter):

```
LAST NAME\FIRST NAME\MRS\STREET
ADDRESS\CITYSTATEZIP
```

The state in the above is the 2 letter abbreviation of the state. At present the ZIP consists of 5 digits. Hence, the last 5 characters of each record are the 5 digit zip code and the 2 characters immediately preceding the zip are the 2 letter abbreviation of the state. The other portions of each record are separated from each other by a "delimiter", a "reverse slash". The location of the zip and 2 letter abbreviation of state bear a constant relationship to the end of each record. We can isolate either of these with the LEN and MID$ statement. We can isolate other items in each record by finding the position of the delimiter immediately preceding and immediately after each item using the INSTR function, then the MID$ and LEN functions.

Programs for creation of such a file, the sorting of such a sequential file, and a program for creating labels from such a file will be found in my Random File article in REMark, issue 10 (programs 2, 4, and 3 respectively).

However, frequently one desires to have the labels sorted in ascending order of zip codes, with names alphabetized within each group of different zip codes. Since the zip is always last in our records we need to "grab" the zip from each record and place it first in each record. After doing this to each record within the file, we could THEN sort the file, then reformat back to original format and obtain all our objectives.

The program "GRAB.BAS", (listed at the end of this article) is a demonstration program that will show you how to do this, among other things. We use the LEN, MID$, and INSTR functions of MBASIC in this program. When you enter this program, just enter the program in logical units, then RUN and check and study the results until you feel comfortable with the different functions. For example, enter lines 10 to 130 and RUN. Then add lines 140 to 200 and RUN. And so on. If we are using the INSTR function you might wish to add a line at 445 -- PRINT A,B,C,D. Similarly, add this line at 485. With this program and your MBASIC manual you should be able to fathom these string handling functions.

Assuming you already have a sequential file, formatted as above, you can use program ZIPFIRST.BAS to prepare your file for sorting. After sorting, use program ZIPLAST.BAS to format your file back to original format. Program ZIPFIRST.BAS simply line inputs your file, record by record and line 120 reformats each record with zip code at beginning of record, line 130 puts each reformatted record in TEMP.DAT. This continues until EOF when file is closed. Now, sort the output file TEMP.DAT. The resulting file (we will call it SORT.DAT) is now ordered by zip code, and all records within each zip group are alphabetized. Next, run program ZIPLAST.BAS which simply inputs, record by record, SORT.DAT, then reformats back to original format and outputs records to TEMP2.DAT. Now, use TEMP2.DAT as the input file to SEQOUT.BAS (in issue 10 of REMark). If you want to use your line printer for the labels, simply open a file for output as "LP:" and substitute PRINT #n, for each PRINT in program as now listed. (n = the number of file that was opened; example -- PRINT #2,). Remember to close the files.

Note that the SORT program appeared in

issue 10 of REMark. If your data is in a random file to begin with, just convert it to a sequential file before reformatting; again, program to do this is in issue 10 of REMark.

You may wonder why you would be interested in isolating the "street address" from a "name and address" file as is done in lines 370-540 in program "GRAB.BAS". Frankly, I can't think of a good reason BUT suppose the data file was not a mailing list, but rather an inventory data list, and suppose the 4th field of each record was the part number of an item in inventory. Then, you might well wish to "grab" the 4th field, place it first in each record, and then sort the file.

EOF

```
10 ' GRAB.BAS       using MID$, LEN, INSTR functions in MBASIC
20 '
30 CLEAR 1000
40 '   ASSUME OUR FILE CONTAINS RECORDS FORMATTED AS IN X$:
50 X$="JONES\JACK\MR\1 FIRST AVE\FIRSTVILLEPA12345"
60 PRINT X$
70 '   ISOLATE ZIP AND PUT IT FIRST IN RECORD
80 A$=MID$(X$,LEN(X$)-4,5):'   grab the zip and put in A$
90 B$=MID$(X$,1,LEN(X$)-5):'   everything except zip goes into B$
100 X$=A$+B$:'   now X$ has zip first
110 PRINT A$:'  print for inspection
120 PRINT B$
130 PRINT X$
140 '   NOW WE PUT BACK TO ORIGINAL FORMAT
150 A$=MID$(X$,1,5):'   put the zip (1st 5 char) in A$
160 B$=MID$(X$,6,LEN(X$)-5):'  put everything else in B$
170 X$=B$+A$:'    reformat back to original format
180 PRINT B$:'  print for inspection
190 PRINT A$
200 PRINT X$
210 ' LETS ISOLATE THE 2 LTR ABBREV OF STATE AND PUT IT FIRST
220 A$=MID$(X$,LEN(X$)-6,2):'   isolate state and put in A$
230 B$=MID$(X$,1,LEN(X$)-7):'   put all before state into B$
240 C$=MID$(X$,LEN(X$)-4,5):'   put zip in C$
250 X$=A$+B$+C$:   'reformat all with 2 ltr state abbrev first
260 PRINT A$:'  print for inspection
270 PRINT B$
280 PRINT C$
290 PRINT X$
300 '   AND PUT BACK INTO ORIGINAL FORMAT
310 A$=MID$(X$,1,2):'   put state abbrev into A$
320 B$=MID$(X$,LEN(X$)-4,5):'   put zip in B$
330 X$=MID$(X$,3,LEN(X$)-7)+A$+B$:'  concatenate back to original format
340 PRINT A$:'  print for inspection
350 PRINT B$
360 PRINT X$
370 '  LETS GRAB THE FOURTH FIELD (STREET ADDRESS) AND PUT IT FIRST
380 '  next 2 lines finds position 1st 4 delimiters & puts in A,B,C,D
390 A=INSTR(1,X$,"\"):B=INSTR(A+1,X$,"\")
400 C=INSTR(B+1,X$,"\"):D=INSTR(C+1,X$,"\")
410 A$=MID$(X$,1,C-1):'  grab first three fields and put in A$
420 B$=MID$(X$,C+1,(D-1)-(C)):'  grab 4th field and put in B$
430 C$=MID$(X$,D,LEN(X$)):'  put rest in C$
440 X$=B$+"\"+A$+C$:'  reformat all with 4th field first
450 PRINT A$:PRINT B$:PRINT C$:PRINT X$:'  print for inspection
460 '  AND PUT BACK INTO ORIGINAL FORMAT
470 A=INSTR(1,X$,"\"):B=INSTR(A+1,X$,"\")
480 C=INSTR(B+1,X$,"\"):D=INSTR(C+1,X$,"\")
490 A$=MID$(X$,1,A):' grab first field and put in A$
500 B$=MID$(X$,A+1,D-(LEN(A$))):'  grab next 3  fields and put in B$
510 C$=MID$(X$,D+1,LEN(X$)):'  rest of record into C$
520 X$=B$+A$+C$:'  and concatenate back to original
530 PRINT A$:PRINT B$:PRINT C$:PRINT X$:' print out for inspection
540 END
```

```
10 '   ZIPFIRST.BAS   pgm for reformatting to sort on zips.
20 '            After program is run, TEMP.DAT contains reformatted records
30 '         with zip first, followed by last name.  After sorting TEMP.DAT,
40 '            use ZIPLAST.BAS to place in original format, then use
50 '            TEMP2.DAT as input file when printing out labels.
60 '
70 CLEAR 1000
80 OPEN "I",1,"LIST.DAT":'  quoted file should be name of your seq file
90 OPEN "O",2,"TEMP.DAT"
100 IF EOF(1) THEN 150
110 LINE INPUT #1,X$
120 A$=MID$(X$,LEN(X$)-4,5):B$=MID$(X$,1,LEN(X$)-5):X$=A$+B$
130 PRINT #2,X$
140 GOTO 100
150 CLOSE:END
160 '  TEMP.DAT now contains file with records formatted with zip first,
170 '          ready to be sorted.
```

```
10 '  ZIPLAST.BAS  after sorting TEMP.DAT to SORT.DAT, use this pgm which
20 '               reformats back to original format; TEMP2.DAT is now
30 '               ordered by ascending order of zips, ready for label printout
40 '
50 CLEAR 1000
60 OPEN "I",1,"SORT.DAT":'  quoted file here should be name of the sorted file
70 OPEN "O",2,"TEMP2.DAT"
80 IF EOF(1) THEN 130
90 LINE INPUT #1,X$
100 A$=MID$(X$,1,5):B$=MID$(X$,6,LEN(X$)-5):X$=B$+A$
110 PRINT #2,X$
120 GOTO 80
130 CLOSE:END
```

EOF

Changes for HDOS 2.0 Boot-Up

The patches presented in this article are the version 2.0 (5 inch disks only!) equivalent of some of the patches in the article "Changes for HDOS Boot-up" in REMark #11. These patches will accomplish the following:

1. Eliminate the need to type a CR or the letter B to initiate BOOT.

2. Eliminate the need to enter a system date (without having to set NO-DATE).

3. Remove the printing of "BOOT" at start-up.

To make these patches, you will need DUMP.ABS from HUG disk VIII (885-1062). Other DUMP-type programs will work, but some of them use a different sector numbering scheme than DUMP, so be sure to check that the old values are as listed before you make changes.

1. Eliminate CR after <BOOT>

This patch causes BOOT to execute automatically. You loose the ability to do CHECKSUMS or to use the IGNORE command.

TRACK 0 SECTOR 2

LOCATION	OLD VALUE	NEW VALUE
17	CD	C3
18	17	3E
19	27	25

2. Eliminate CR after date

This patch eliminates the requirement to enter a system date or a CR after existing date. If the system has been on, and a date is in memory, that date is used. If the system has been just turned on, the last date stored on the disk is used. You can enter a new date with the DATE command. Note: If the disk on which you make this patch has never been used, or has been used with NO-DATE set, it will come up with "DD-MMM-YY" as the system date (literally), but you can change it with the DATE command. You should SET HDOS DATE if NO-DATE was set. Setting NO-DATE eliminates the need to enter a date or type CR or enter a date, as this patch does, but with NO-DATE set you get the

(vectored to page 31)

New HUG Software

885-1093 DND HDOS DISK $20.00

DND.BAS is the HUG version of the popular game "DUNGEONS AND DRAGONS". The object of DND is to find the lord master of the 50 level Heathkit Dungeon by exploring it. You will begin your search on level 1 where there are taverns in which you may cash any gold you find for experience points. Accumulated experience points will allow you to become a higher level character.

During your quest you will encounter many obstacles and find objects that may at sometime help you in your search. The deeper you go into the dungeon the harder it will be to survive as a low level character. The lord of the dungeon will be found in a HEATHKIT VAULT. As the game progresses you will be given the combination. The lord may or may not be there -- other mysterious things may be in a vault -- so beware!

The minimum system requirement is a 56K machine, H19 or H89, HDOS Microsoft BASIC, and two drives. You will use ALL available memory so you will not be able to load any device drivers.

The instuctions are very brief. There are many aspects of the game to explore. Be adventurous, be careful and have fun!!!

Adventure using HDOS 2.0 and a Single Drive

Now that HDOS 2.0 is out, the ADVENTURE Disk (HUG P/N 885-1010) requires a couple of special notes. HDOS has grown again in its minimum configuration by several sectors. With this increase in size of of HDOS, Adventure becomes a little tight on a single five and a quarter inch disk. If all files that are <u>NOT</u> needed for Adventure are deleted then the three files ADVENT.ABS, ADVENTUR.DTB and NEWGAME.CAV will fit. You must delete all system files without a 'W' flag except SY.DVD. After copying the above three files onto the disk using ONECOPY.ABS there are only eight (8) sectors free, less than half required to save a game. That's fine if you always want to start a NEWCAVE everytime you play.

As a suggestion we recommend making a complete copy of the Adventure Disk without the documentation files. This should be done by using INIT.ABS to create a blank disk and then copying the three Adventure files, shown above, using ONECOPY.ABS. To check the spellings of the file names use <u>/L</u> when you're in ONECOPY and the file names will be listed for you.

Once the disk is created then make another disk this time using INIT.ABS and SYSGEN.ABS. This disk will be known as the 'SYSTEM' disk.

Make sure that SET.ABS is on the system disk (if not use ONECOPY and get it from your HDOS disk). Boot up the system disk and proceed to what is known as HDOS STAND-ALONE. To do this we must type the following:

```
SET HDOS STAND-ALONE <CR>
```

Where <CR> equals the Carriage Return.

A message will appear warning you that you could be in trouble if you proceed, pick your heart up off the floor and do exactly what I tell you and everything will be fine. By the way the worse thing that could happen is that we could destroy one of the two disks we are working with. But let's think positive and follow these simple instructions and we will be playing Adventure before you know it.

```
Type:     RESET SY0: <CR>
```

When told to replace the disk take the system disk out of the drive and replace it with the <u>COPY</u> of the Adventure disk.

Wait for the HDOS prompt (>) and then type:

```
ADVENT <CR>
```

When asked which cave, simply hit the RETURN key to get a NEW cave. An old cave can be called by typing in the name of the cave that has been previously played and saved using the "SAVE" command during the Adventure game. NOTE: When using the SAVE command remember the name of the CAVE that you have saved so that it may be recalled later.

Answer the question about instructions with a 'NO'. NOTE: Adventure looks for upper case only, so push the <u>CAPS LOCK</u> down.

<u>You are standing at the end of a road before a small brick building. Around you is a forest. A small stream follows out of the building and down a gully.</u>

Try going inside and see what you can find and remember the magic word <u>XYZZY</u>.

That's the end of my help gang, but I will suggest that you feed the bear the food and not yourself.

Adventure is a very addictive game, so if you're short of time try a less adventurous type game such as tic-tac-toe.

GK:

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	$ 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
885-1054	SmBusPkg II 3 Disks H8/H19/H89	$ 60.00
885-1055	MBASIC Inventory Disk H8/H89	$ 30.00
885-1056	MBASIC Mail List H8/H89 Disk	$ 30.00
885-1070	Disk XIV Home Finance H8/H89	$ 18.00
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.

More on Home Control with the ET3400

Michael C. Frieders
9318 Clanbrook Ct.
Fairfax, VA 22031

Minor modifications to the hardware and software presented in REMark Issue 13 are required if you have the ETA-3400 Accessory. The changes are summarized below:

1) The machine code program has been moved up one page (256 bytes), since the first page is reserved by the accessory. A number of lines of code have changed due to the fact that above the first page you can't take advantage of certain direct addressing instructions. The new program is two bytes longer than the old program described is Issue 13.

2) The command stack now starts at address 01A0 (HEX).

3) The delay between commands has been increased to one second. I have found that the additional delay improves the reliability of the system.

4) An error (of no consequence) at address 017C (HEX) has been corrected.

5) The wire between IC 74LS30 Pin 8 and the RE connector block is no longer necessary. In fact, IC 74LS30 is no longer necessary at all. This is due to the method by which the ETA-3400 accesses the data lines in the ET-3400.

I have dumped the program to tape followed by a dump of my command stack beginning at address 01A0 (HEX). This makes it very easy to regenerate the system by doing two consecutive tape reads. Then all you have to do is set address 00F7 (HEX) to 3B (HEX), set the time, and you're back in business. I also have the program on disk, and dump it from my H8 to the ETA-3400 using the CPS modem software offered by Softstuff. Using tape and disk makes it easy to have a number of different command stacks readily available for use on weekends, vacations, etc.

If you have trouble implementing this system, make sure that the BSR command console is operating properly. After several months, my command console broke down. Sears replaced it with a new one, and it has worked perfectly since. Also, be sure you have a good battery in the cordless controller. If you have specific questions, I can be reached at the above address, or you can leave a message via MicroNET (ID No. 70140,371).

The following is a listing of the new program for the ETA-3400. Note that the addresses start at 0100 (HEX). GOOD LUCK !

```
ADDRESS    OPCODE     MNEMONIC         COMMENTS
*******    ******     ********         ****************************

0100       00                          HOURS
0101       00                          MINUTES
0102       00                          SECONDS
0103       CE FF04    LDX   #FF04      SET UP THE PIA
0106       FF 8002    STX   #8002      WITH THE B SIDE OUT
0109       86 FF      LDA   #FF        LOAD BSR NULL COMMAND
010B       B7 8002    STAA #8002       SEND NULL COMMAND
010E       FE 015F    LDX   #015F      GET ADDRESS OF NEXT COMMAND
0111       B6 0100    LDAA 0100        LOAD HOUR FROM CLOCK
0114       A1 00      CMPA,X 00        COMPARE CLOCK HOUR WITH NEXT HOUR
0116       26 09      BNE  09          NO MATCH - CONTINUE CLOCK
0118       B6 0101    LDAA #0101       HOURS MATCH, LOAD MINUTES FROM CLOCK
011B       A1 01      CMPA,X 01        COMPARE CLOCK MINUTE WITH NEXT MINUTE
011D       26 02      BNE  02          NO MATCH - CONTINUE CLOCK
011F       8D 40      BSR  40          TIMES MATCH! SEND COMMANDS
0121       CE 00FD    LDX   #00FD
```

```
0124        A6 03      LDAA,X 03       LOAD AND DISPLAY
0126        BD FE20    JSR  #FE20      HOURS,MINUTES
0129        08         INX             AND SECONDS
012A        8C 0100    CPX  #0100
012D        26 F5      BNE  F5
012F        BD FCBC    JSR  #FCBC      RESET DISPLAY COUNTERS
0132        C6 3D      LDAB #3D
0134        5A         DECB
0135        27 04      BEQ  04         ONE SECOND DELAY ROUTINE
0137        0E         CLI             ("LINE" MUST BE CONNECTED TO "IRQ")
0138        3E         WAI
0139        20 F9      BRA  F9
013B        8B 01      ADDA #01        INCREMENT SECONDS
013D        19         DAA             DECIMAL ADJUST SECONDS
013E        A7 02      STAA,X 02       STORE SECONDS
0140        81 60      CMPA #60        60 SECONDS YET?
0142        26 DD      BNE  DD         IF NOT, GO BACK FOR NEXT SECOND
0144        8D 0F      BSR  0F         ELSE CLEAR SECONDS AND UPDATE MINUTES
0146        81 60      CMPA #60        60 MINUTES YET?
0148        26 C4      BNE  C4         IF NOT, GO BACK AND CHECK NEXT TIME
014A        09         DEX             ELSE CLEAR MINUTES
014B        8D 08      BSR  08         - AND UPDATE HOURS
014D        81 24      CMPA #24        24 HOURS YET?
014F        26 02      BNE  02         IF NOT, DON'T CLEAR HOURS YET
0151        6F 01      CLR,X  01       ELSE CLEAR HOURS
0153        20 B9      BRA  B9         GO BACK AND CHECK TIME
0155        6F 02      CLR,X  02       CLEAR SECONDS OR MINUTES
0157        A6 01      LDAA,X 01       LOAD MINUTES OR HOURS         LOAD
0159        8B 01      ADDA #01        INCREMENT MINUTES OR HOURS    AND
015B        19         DAA             DECIMAL ADJUST                UPDATE
015C        A7 01      STAA,X 01       STORE MINUTES OR HOURS        ROUTINE
015E        39         RTS             RETURN
015F        01A0                       ADDRESS OF NEXT TIME; SCRATCH AREA
0161        A6 02      LDAA,X 02       LOAD NEXT COMMAND TO BE SENT
0163        27 12      BEQ  12         IF END OF STRING SET UP FOR NEXT TIME
0165        8D 22      BSR  22         ELSE GOTO SEND-AND-DELAY ROUTINE
0167        86 FF      LDAA #FF        LOAD BSR NULL COMMAND
0169        8D 1E      BSR  1E         GOTO SEND-AND-DELAY ROUTINE
016B        B6 0102    LDAA #0102      UPDATE SECONDS TO
016E        8B 02      ADDA #02        COMPENSATE FOR
0170        19         DAA             SENDING COMMANDS,
0171        B7 0102    STAA 0102       AND
0174        08         INX             SET INDEX REGISTER FOR NEXT COMMAND
0175        20 EA      BRA  EA         GO BACK AND REPEAT FOR NEXT COMMAND
0177        A6 03      LDAA,X 03       LOAD HOUR FOR NEXT TIME
0179        81 99      CMPA #99        IS THIS THE END OF -
017B        27 05      BEQ  05         THE COMMAND STACK?
017D        08         INX
017E        08         INX             IF NOT, SET THE INDEX REGISTER
017F        08         INX             FOR THE NEXT TIME
0180        20 03      BRA  03
0182        CE 01A0    LDX  #010A      ELSE RESET TO THE TOP OF COMMAND STACK
0185        FF 015F    STX  #015F      STORE ADDRESS OF NEXT TIME
0188        39         RTS             RETURN TO CLOCK
0189        B7 8002    STAA #8002      SEND COMMAND TO PIA
018C        C6 3D      LDAB #3D
018E        5A         DECB
018F        27 04      BEQ  04         ONE-SECOND DELAY
0191        0E         CLI
0192        3E         WAI
0193        20 F9      BRA  F9
0195        39         RTS             RETURN TO COMMAND PROCESSOR

00F7        3B         RTI             RETURN FROM INTERRUPT

01A0        >BEGIN COMMAND STACK HERE<
```

What's a FOCAL?

Those of you who read the "New HUG Products" section of REMark #13 know that HUG is selling a FOCAL interpreter (885-1059) that I wrote for Heath computers (HDOS system). If you "grew up" on PDP-8's, or used the H11 back in the paper tape days, you may be familiar with FOCAL, but most of you probably don't have the foggiest notion of what it is all about.

The purpose of this article is to introduce you to FOCAL-8 (HUG FOCAL), not so that you will all be converted to it, but to let you know that there are other ways of doing things besides BASIC. Back in High School, I took Latin, and it helped me understand English better. In the same way, an exposure to FOCAL or another language will help you understand BASIC better. So let's take a look at FOCAL and see what it is like, and how it compares to BASIC.

In the introductory material in Heath's software for BASIC, it says that BASIC is a conversional programming language. This means that BASIC can interact with the user, allowing him or her to type in problems and get answers right away. BASIC is also a general purpose language, which means that it can be used for business programs, engineering programs, games, etc. You may have heard the expression "structured language". BASIC is not a structured language. You can arrange the parts of a program in just about any order as long as you GOTO the right place at the right time.

Most of these attributes apply to FOCAL as well as to BASIC. FOCAL is conversational like BASIC. It is not a structured language, but the way it works causes you to arrange your programs into a main body and subroutines more than you might do in BASIC. FOCAL differs from BASIC mainly in that it is not a general purpose language. It was designed primarily as a "number cruncher", for scientific and engineering use. While most BASIC's allow three data types (numeric, logic, and string), FOCAL allows only numeric data. The name FOCAL suggests its intended use: it means FOrmula CALculator.

Now, let's make some direct comparisons between FOCAL-8 and BASIC. In both languages, there is a command mode and a program mode. That is, you can type in a one line problem, hit RETURN, and it will execute immediately (command mode). Or, you can enter a number of lines preceded by line numbers (a program), and then RUN the program. The line numbers in FOCAL are a little different from those in BASIC. They consist of a one or two digit number, a period, and another one or two digit number. A typical line number would be 10.45, and it might be followed by 10.50 (which you could enter as 10.5). The whole number is called the line number, as in BASIC, but the first part is called the <u>group</u> number. Every line number that started with 10 would be in group 10. The reason for this is that some of the commands in FOCAL can operate on either a single line or a whole group. For example, the WRITE command, which is like LIST in BASIC, can be used to list line number 10.45 on your terminal, or it can be used to list all of group 10. But it doesn't stop with simple commands like that. The DO command, which is the FOCAL version of GOSUB, can be used to call line 10.45 as a subroutine in one part of a program, then call all of group 10 as a subroutine an another part.

As you can see, the commands in FOCAL have different names from their BASIC counterparts. This was not intentionally done to confuse you, but is the result of a restriction placed on FOCAL commands that does not exist in BASIC. Each command in FOCAL can be abbreviated by its first letter, so each command must start with a unique first letter. There is a GOTO command in FOCAL, so there cannot be a GOSUB command. Writing a program is faster when all you have to type is the first letter of each command. You might think you would be restricted to 26 commands, but it is possible to have subcommands under a command. In FOCAL-8, the LIBRARY command, which is used for disk I/O, has 6 subcommands, all of which start with letters used by other main commands.

Figure 2 (at the end of this article) is a sample program in FOCAL-8 ("Electronic Checkbook"). This program is not only an example of the language, but it demonstrates how I overcame some limitations of the language. More on that later. Here is a list of the commands used in that program and their BASIC equivalents.

FOCAL	BASIC	USE
ASK	INPUT	Interractive input
COMMENT	REM	Remarks
DO	GOSUB	Subroutine call
ERASE	CLEAR	Clear variables
FOR	FOR	Loops

GOTO	GOTO	Control transfer
HARDCOPY	various	Toggle printer
IF	IF	Conditional control
JUMP	ON GOTO	Computed transfer
LIBRARY	various	Disk I/O
POKE	POKE	Change memory
RETURN	RETURN	Subroutine exit
SET	LET	Assignment
TYPE	PRINT	Print values, etc.
X	DEF USR	Set user address

In addition to these commands, FOCAL-8 has several pre-defined functions. For the most part, they are like their BASIC counterparts, except that the name starts with a capital F. For example, the BASIC ABS function is FABS in FOCAL.

Some of the commands differ more from BASIC than the command list would indicate. The IF command, for instance, evaluates an expression following it in parentheses, and jumps to one of three following line numbers (see line no. 03.64 in the program). It jumps to the first number if the expression is negative, to the second if it is zero, and to the third if it is positive. FORTRAN programmers will be familiar with this technique. There are no comparison operators in FOCAL, so comparison is done with subtraction (see line no. 05.12). Note that you can leave off one or two of the line numbers and add an executable statement instead.

The HARDCOPY command takes as its argument ON or OFF (or OF). When HARCOPY is ON, everything that goes to the screen also goes to the printer. When you turn HARDCOPY OFF, printer operations stop. This makes it easy to send program output to a printer without having to worry about opening files, closing files, PRINT #n, etc.

The six subcommands under the LIBRARY command are LOAD, RUN, SAVE, DELETE, PUT, and GET. LOAD, RUN, and SAVE are like the same commands in MBASIC, and DELETE is like KILL. PUT saves all of a program's variables in a file, and GET loads variables from a file. Line 01.30 in the Electronic Checkbook program saves your checkbook in a file called CHECK.FD (.FD is the default extension for FOCAL Data) on SY1:, and line 03.14 loads it back in the next time you use it. As with HARDCOPY, the programmer is freed from concern with opening and closing files, etc. Sometimes you need the file flexability afforded by BASIC, but when all you want to do is save the data you have entered into a program on disk, the FOCAL-8 approach is adequate.

It is not my intention to present a complete FOCAL manual in this article, but there is one more thing I should mention before I discuss some the tricks I used in the checkbook program. In FOCAL, arrays (dimensioned variables) are handled differently than in BASIC. Let's say that you have written a program with an array that could have as many as 1000 elements in it. If the program is in BASIC, you would have a statement near the beginning like this:

```
10 DIM A(1000)
```

You have reserved some memory in your computer for the array with the above statement. Let's say that each time you actually run the program, you use different elements in the array, but not all of them. For example, one time you run it you would use A(9), A(55), and A(998), and the next time you use A(44), A(322), A(444), and A(1000). The first time you used 3 elements, and the second time you used 4 elements, and all of the other elements you reserved were wasted space in the computer's memory. In FOCAL-8, there would be no wasted space. You do not have to reserve space for arrays (there is no DIM statement) because FOCAL-8 only uses space as it is needed. In the above example, if you needed A(2000), the program in BASIC would have to be re-written to allow the larger array. In FOCAL-8, no re-writing would be necessary. Some FOCAL interpreters allow only single dimensioned arrays, but FOCAL-8 allows two dimensioned arrays also. In theory, a single dimensioned array can have 65535 elements, and a two dimensioned array can be 255 by 255. The actual limit is the amount of memory in the computer.

Now we will look at the "Electronic Checkbook" program. Figure 1 is a partial sample run. As you can see, there is a description of each item in the checkbook. How did we do this in a language that does not have strings? It was done by using POKE to put in a little machine language program to get a character from the console (see lines 01.04, 01.06, and 01.08). This routine is listed below:

```
.SCIN   EQU     1
.SCOUT  EQU     2
        SCALL   .SCIN
        JC      *-2
        CPI     177Q      DELETE KEY?
        JNZ     NOT
        MVI     A,8       REPLACE WITH BS
NOT     SCALL   .SCOUT
        MVI     B,0
        MOV     C,A
        RET
```

The subroutine at group 8 determines where to put this little program. FOCAL-8 loads in the HDOS overlays when it starts up and stores the highest user address under them at locations 8835 and 8835 decimal (low byte first). The program gets the high byte, multiplies it by 256 to make it an even page address, then subtracts 1k (1024) to get an address for the machine code routine. Group 30 calls the routine to get characters from the keyboard and put them in an array N, with the number of the character within the string as one dimension, and the entry number in the checkbook as the other. This is not a very memory efficient way of storing strings, but it works.

Group 40 is called when the checkbook entries are printed. It retrieves the characters from array N and prints them, and fills in with spaces if there are less than the maximum number of characters allowed, which is 15.

In the right column of numbers in the sample run you will notice right justified numbers in dollars and cents format. The routine that does this trick is group 50. I included several comments to help you figure it out. The $27's in this routine are equivalent to CHR$(27) in BASIC, which is the code for ESCAPE. The purpose is to cause the terminal (H19) or printer to ignore the next character being printed, which is the leading space that FOCAL-8 (or BASIC, for that matter) prints before a number. It will work on any printer or terminal that is looking for escape sequences. If the character following the escape is not part of the printer or terminal's codes, (for example a space) it is ignored. If you have a printer that does not use escape sequences, but does allow backspacing, replace $27 with $8.

It should be obvious by now that FOCAL is definitely not a business language. The sample program that comes with FOCAL-8, which plots data to a curve and draws a graph, is more indicative of what FOCAL can do. But, we have demonstrated that the deficiencies of a language can be overcome to some extent if a little effort is applied.

PS:

Figure 1. "Electronic Checkbook" Sample Run

```
ELECTRONIC CHECKBOOK


Functions available:

1.  Start a new checkbook
2.  Update an existing checkbook
3.  Print last N entries on screen
4.  Print last N entries on printer
5.  Save data on disk
6.  Exit to operating system

Enter your option:   3


How many entries do you wish to list?  8


 Entry no. Check no.  Date                 Description           Amount    Balance

 24        206        February  18 1981  J C WHITNEY             2.21     258.99
 25        207        February  18 1981  BOOKS                  20.85     238.14
 26        208   Void February  18 1981                          0.00     238.14
 27        209        February  18 1981  I AND M                59.85     178.29
 28        210        February  18 1981  W W MAG                 8.95     169.34
 29        211        February  18 1981  MERVYN'S               50.00     119.34
 30        212        February  18 1981  PHONE                  38.86      80.48
 31        213        February  18 1981  J C PENNY              65.00      15.48
```

Figure 2. "Electronic Checkbook" Program in FOCAL-8

```
01.02 C     Main Menu and machine code setup
01.04 D 8;P X,255;P X+1,1;P X+2,218;P X+3,0;P X+4,X1;P X+5,254
01.06 P X+6,127;P X+7,194;P X+8,12;P X+9,X1;P X+10,62;P X+11,8;P X+12,255
01.08 P X+13,2;P X+14,79;P X+15,6;P X+16,0;P X+17,201;X X
01.10 T "ELECTRONIC CHECKBOOK
01.12 T !!!"Functions available:
```

```
01.14 T !!"1.  Start a new checkbook
01.16 T !"2.  Update an existing checkbook
01.18 T !"3.  Print last N entries on screen
01.20 T !"4.  Print last N entries on printer
01.22 T !"5.  Save data on disk
01.24 T !"6.  Exit to operating system
01.26 A !!"Enter your option:  ",A,!
01.28 J (A)2.1,3.1,4.1,5.1,1.3,1.32;G 1.12
01.30 L P SY1:CHECK;G 1.12
01.32 X 8256;S Y=1:N=2;A !!"Are you sure? (Y/N) <N>",A,!
01.34 J (A)1.36,1.12;G 1.12
01.36 S A=FUSR(A);C            Exit to HDOS

02.02 C      Start a new checkbook
02.10 E;A !!"Enter the year for this checkbook: ",YEAR,!
02.12 I (YEAR-1900)2.14;G 2.16
02.14 S YEAR=YEAR+1900
02.16 A !"Enter the number of the first check in your checkbook: ",CKNUM,!
02.18 A !"Enter your current balance: ",BALANCE,!;S ENTRY=1;G 3.22

03.02 C      Second Menu -- Deposits and withdrawals
03.10 S Y=1:N=2;A !!"Want to get data from disk? (Y/N) <Y>"A,!
03.12 J (A)3.14,3.16;G 3.14
03.14 L G SY1:CHECK;C         Get data from disk
03.16 T !"The last check written was number",%2,CKNUM-1," dated "
03.18 S M1=M2(CKNUM-1);D 20;T %2,D2(CKNUM-1),!
03.20 T !!"Your current balance is $";S NUM=BALANCE;D 50
03.22 T !!"Transactions available:
03.24 T !!"1.  Make a deposit
03.26 T !"2.  Write a check
03.28 T !"3.  Make a withdrawal (no check used)
03.30 T !"4.  Change the amount on a previous item
03.32 T !"5.  Return to main menu
03.34 A !!!"Enter your choice: ",A,!;J (A)3.4,3.6,3.8,3.88,1.12;G 3.22
03.40 A !!"Enter the month (1-12)",MONTH(ENTRY),"  The day (1-31)",DAY(ENTRY)
03.42 A "  Amount of deposit: ",MONEY,!;D 30;S BALANCE=BALANCE+MONEY:TRANS(ENTRY)=MO
NEY
03.44 S OLDBAL(ENTRY)=BALANCE:CH(ENTRY)=1:ENTRY=ENTRY+1;G 3.2
03.60 A !!"Enter the month (1-12)",MONTH(ENTRY),"  The day (1-31)",DAY(ENTRY),!
03.62 T %2,!"Enter amount for check no.",CKNUM;A " (0 to void, -1 to exit)",MONEY
03.64 I (MONEY)3.2,3.66,3.66
03.66 D 30;S CH(ENTRY)=CKNUM:BALANCE=BALANCE-MONEY:OLDBAL(ENTRY)=BALANCE
03.68 S TRANS(ENTRY)=MONEY:ENTRY=ENTRY+1:MONTH(ENTRY)=MONTH(ENTRY-1)
03.70 S DAY(ENTRY)=DAY(ENTRY-1):M2(CKNUM)=MONTH(ENTRY-1):D2(CKNUM)=DAY(ENTRY-1)
03.72 S CKNUM=CKNUM+1;G 3.62
03.80 A !!"Enter the month (1-12)",MONTH(ENTRY),"  The day (1-13)",DAY(ENTRY)
03.82 A "  Amount of withdrawal: ",MONEY,!;S MONEY=FABS(MONEY);D 30
03.84 S BALANCE=BALANCE-MONEY:TRANS(ENTRY)=MONEY:CH(ENTRY)=2
03.86 S OLDBAL(ENTRY)=BALANCE:ENTRY=ENTRY+1;G 3.2
03.88 A !!"Enter item no. to change: ",ITEM,"  Enter new amount: ",MONEY,!
03.90 S DIFF=TRANS(ITEM)-MONEY:TRANS(ITEM)=MONEY
03.92 J (CH(ITEM))3.96;F I=ITEM,ENTRY-1;S OLDBAL(I)=OLDBAL(I)+DIFF
03.94 S BALANCE=BALANCE+DIFF;G 3.2
03.96 F I=ITEM,ENTRY-1;S OLDBAL(I)=OLDBAL(I)-DIFF
03.98 S BALANCE=BALANCE-DIFF;G 3.2

04.02 C      List checkbook on screen
04.10 A !!"How many entries do you wish to list? ",A,!
04.12 I (ENTRY-A)4.14,4.14;S A=ENTRY-A;G 4.16
04.14 S A=1
04.16 T !!" Entry no. Check no.  Date                Description        Amount   Bala
nce"!!
04.18 F I=A,ENTRY-1;D 10
04.20 A !"Hit RETURN to continue",A,!;G 1.12

05.02 C      List checkbook on printer
05.10 A !!"How many entries do you wish to list? ",A,!
05.12 I (ENTRY-A)5.14,5.14;S A=ENTRY-A;G 5.16
05.14 S A=1
05.16 H ON;T !" Entry no. Check no.  Date                Description        Amount
Balance"!!
```

```
05.18 F I=A,ENTRY-1;D 10
05.20 H OF;G 1.12

08.02 C      Subroutine to locate machine program
08.10 S X=FPEK(8836)*256-1024:X1=FINT(X/256)

10.02 C      Subroutine to make checkbook listing
10.10 T %10,I;J (CH(I))10.12,10.14;G 10.16
10.12 T " Deposit    ";G 10.2
10.14 T " Withdrawal ";G 10.2
10.16 I (TRANS(I)) 10.18,10.18;T %12,CH(I);G 10.2
10.18 T %6,CH(I),"Void  "
10.20 S M1=MONTH(I);D 20;T %3,DAY(I);T %7,YEAR
10.22 D 40;S NUM=TRANS(I);D 50;T " ";S NUM=OLDBAL(I);D 50;T !

20.02 C      Subroutine to print month name from number
20.10 J (M1)20.12,20.14,20.16,20.18,20.2,20.22,20.24,20.26,20.28,20.3,20.32,20.34
20.12 T "January  ";R
20.14 T "February ";R
20.16 T "March    ";R
20.18 T "April    ";R
20.20 T "May      ";R
20.22 T "June     ";R
20.24 T "July     ";R
20.26 T "August   ";R
20.28 T "September";R
20.30 T "October  ";R
20.32 T "November ";R
20.34 T "December ";R

30.02 C      Subroutine to put string characters in array N
30.10 T !"Enter description of item: (15 characters max)"
30.12 T " ";F X=1,15;S N(X,EN)=FUSR(X);I (N(X,EN)-13)30.16
30.14 S X=15;R
30.16 I (N(X,EN)-8)30.14,30.18,30.14
30.18 I (X-2)30.12;T " ",$8;S X=X-2

40.02 C      Subroutine to print string characters from array N
40.10 F X=1,15;I (N(X,I)-13)40.12;T $N(X,I)
40.12 S X1=X:X=15
40.14 F X=X1,16;T " "
40.16 R

50.01 C      Subroutine to print right justified dollars and cents
50.02 I (NUM) 50.04;S MIN=32;G 50.06;C       Test if number is negative
50.04 S MIN=45;C                             "MIN" = ASCII minus if so
50.06 S NUM=FABS(NUM);C                      Get absolute value of number
50.08 I (NUM-10)50.24;C                      Test number for number of digits
50.10 I (NUM-100)50.26
50.12 I (NUM-1000)50.28
50.14 I (NUM-10000)50.3
50.16 T %2,$MIN,$27,FINT(NUM),".",$27;C      Print dollars and decimal
50.18 S CENTS=FINT(NUM-FINT(NUM)*100+.5);C   Isolate cents
50.20 I (CENTS-10)50.22;T %2,CENTS;R;C       If cents >= 10 then print as is
50.22 T %2," 0",$27,CENTS;R;C                If cents < 10 then add leading 0
50.24 T "    ";G 50.16;C                     Print spaces for right
50.26 T "   ";G 50.16;C                           justification
50.28 T "  ";G 50.16
50.30 T " ";G 50.16
*
                                                     EOF
```

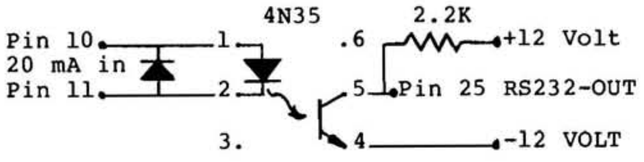
NON-HUG BUG: The RDT debugging tool introduced in REMark issue #14 will not work with the DG-FP8 monitor sold by DG Electronics for their Z80 CPU board. The reasons were mentioned in BUSS Issue #28, which stated that DBUG also would not work under that monitor. The first reason is that the DG-FP8 monitor saves the Z80 IX and IY registers on the stack in addition to the 8080 registers when a RST 2 is executed. (RST 2 is used for breakpoints). The second reason is that the single step interrupt must be armed immediately prior to jumping to a user program with DG-FP8. Both DBUG and RDT arm the interrupt several instructions prior to the jump.

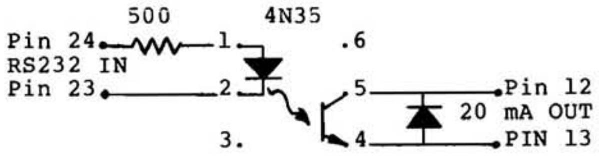
PS:

RS-232 to 20 mA Current Loop Converter for the H19 or H89

William A. Deutschman
and
Tim V. Schwibbe
Rose-Hulman Institute of Technology
Terre Haute, Indiana 47805

Many time-sharing computer systems use a 20 milliampere current loops to transmit data between the computer and the terminal; hence it is necessary to convert the RS-232 from the H-89 serial interface card to 20 ma current signals. The circuit shown below is a simple, inexpensive circuit to do the conversion without changing any of the existing functions of the computer.





The circuit is wired on a 2-inch square of perf-board. Use a single 16-pin socket and place one opto-isolater in the top six pins and the other in the bottom six. D1 and D2 are general purpose diodes. Mount the board in the upper right corner of the terminal board using the two tapped holes in the heat sink. Be sure to insert a sufficient number of washers and a sheet of insulating material to prevent the circuit from touching the conductors on the terminal board. Failure to do this may severely damage your computer if the converter touches the computer terminal board.

The power for the circuit is obtained by soldering wires to the foil side of the terminal circuit board. Connect the +12 volts to U403 (make the connection between U403 and C406) and the -12 volts to U405 (between U405, U404 and C408). Check the diagram for the terminal logic board on page 19 of the illustration booklet to make certain that you have the correct connection points. You may also wish to measure the voltages to insure that you have not made a mistake.

The signal leads are connected to plug, P1, on the back mounting plate of the computer. You will have to insert additional pins in the plug or replace the existing plug. In either case you should not change the existing connections to pins 1 through 8 and pin 20. Connect the converter to pins 10 through 13 and 23 through 25 as shown on the circuit diagram. The modifications to your computer/terminal are now complete.

The final step is to make an adapter plug to connect the H-89 to the 20 ma computer loop. Connect pins 10 & 11 to the current loop sending data from the H-89 to the external device (pin 10 to the +20 ma; pin 11 to the -20 ma). Connect pins 12 & 13 to the loop receiving data from the external source (pin 12 to the +20 ma; pin 13 to the -20 ma). Finally connect pin 3 to pin 25, pin 7 to pin 24 and pin 2 to pin 23 within the plug.

The advantage of this connection is that P2 retains its original function if a standard RS-232 plug is connected to the terminal. When the adapter plug is connected the RS-232 signals are sent to the converter and 20 ma signals to the external device. Note however that this converter does not contain a current source, hence the external device must supply the current and MUST LIMIT IT TO 20 MILLIAMPERES as the converter does not contain a current limiter.

EOF

(vectored from page 9)

20030 -- Is a catch-all in that if there is an error for some unknown reason the error number (ERR) and the line (ERL) that the error occurred in will be displayed on the screen for easier trouble shooting.

STARTUP NOTES:

Before attempting to type in the SAMPLE program and while you are still in HDOS (Heath Disk Operating System) be sure to load your printer device driver.

Example: >LOAD LP:

Load in Microsoft BASIC and have fun typing the SAMPLE program.

GK:

BUGGIN' HUG

Dear HUG,

I would like to share this idea with other members who use HDOS MICROSOFT BASIC. When I received my MICROSOFT early in 1980, and worked through the software manual, I discovered that there was no multi-statement (line) command. Each time you needed a new line similiar to a line you had just entered, you were required to type that new line with a new line number. One day, when I read and worked through chapter 4,4 (EDIT COMMANDS), I worked especially with the command CTRL/A and discovered what I needed:

```
User types: LIST 100
Computer prints: REM FOR I=1 TO N
User types: CTRL/A (A AND CTRL KEYS)
Computer prints: !
User types: I(for insert) 1500 (RETURN)
User types: LIST 1500
Computer prints: REM FOR I=1 TO N
```

You still have the old line number and you can proceed with the multi-statement because the last number is always in the memory buffer. (Just use the CTRL/A and I(nsert) before the new line you wish to create.

Knud Hinrichsen
Oernevang 54
3450 Alleroed
DK Denmark

Dear HUG,

I just spent hours looking through back issues of REMark to find a way of getting at the date one puts in at boot-up. Well, I couldn't find anything. So, I thought up a little one-liner in Microsoft BASIC which may not be anything new to the old hands, but may help some of the others.

```
 10 DA$="":FOR J%=8383 TO 8391:
    DA$=DA$+CHR$(PEEK(J%)):NEXT J%:END
```

The Variable DA$ now contains the boot-up date>

Best regards

Lex Reinkeluers
RR 5 London, Ontario
Canada N6A 4B9

NOTE: a similiar technique was used in the Screen Format article presented in Issue 12 of REMark. You may wish to examine this program to see how the "date" function was implemented.

Dear HUG,

The "MOUNTALL" program described in Issue 12 of REMark will no longer function under HDOS 2.0. The following corrections will allow proper operation using the newest version.

A. In the definitions the .ERROR EQU should now be 057Q.

B. After the line 042.237 with the instruction STA DNUM you should insert an additional instruction - STA DNUM2

C. Line 042.255, with the instruction LXI H,DRIVE, should be changed to indicate LXI H,DRIVE2.

D. After line 043.003, the following lines should be inserted:

```
DRIVE2 DB 'SY'
DNUM2  DB 'X:',0
```

The reason the program failed under version 2.0 was the end message character at line 043.003. When HDOS was rewritten, this character caused the Mount Scal to exit presumably under an undefined file error.

Yours truly,
Clifford C. Lundberg
310 King Avenue
Elk River, Minn. 55330

8k Memory Blues

ARCKLE, SPARKLE, LITTLE SNAP
HOW I WONDER WHAT WENT ZAP?
UP ABOVE MEMORY HIGH
I PLUG YOU IN AND NOW I CRY.

IT MUST HAVE BEEN THOSE CHIPS THEY TUBED
NOW THEY'RE GONE, ZERO TO "F" CUBED.
WHY'D YOU DO THIS THING TO US?
WHOOPS! LOOK OUT! THERE GOES THE BUS!

Kurt Schultz
115-1 Roxanne Ct.
Walnut Creek, CA 94596

HUGBB on MicroNET

Wow, the response and activity of the HUG Bulletin Board is fantastic. I have been adding at least two or three users to the list of HUGBB members every two days. With each new member comes more monitoring and maintenance to be done by yours truly. Even with all this new, additional work load the HUGBB has been running great.

The credit for the smooth running Bulletin Board goes to Richard Taylor, the original creator of the BB, and Russel Renshaw of MicroNET. Gentlemen, we thank you for your support over the past weeks when we needed your help and expertise.

All this "smooth sailing" is just in time for a new rumor . . . the HUGBB may be moving. Russel Renshaw (the MNET Wizard) is writing a new Bulletin Board system that we may move to. From the outline he submitted the projected BB looks really nice . . . much more user oriented.

At this time however, we cannot project that we will or will not change to the new BB system. Therefore, it makes it extremely difficult to write an article about the HUGBB when by the time this REMark magazine "hits" the street what we print may not be pertinent.

Also of importance, and which we have discovered is a most discouraging and disheartening fact . . . that is that MicroNET as of around the first of February has been selling and promoting their membership through one of our "friendly neighborhood" competitors of which will remain nameless at this time.

MicroNET and our competitor have signed a contract that limits all other hardware manufacturers if they choose to sell MicroNET memberships. MicroNET did indicate to us that they will honor all mailed membership forms but did not indicate any turn-around time. So for the time being if you plan on becoming a member of the MicroNET system and the HUG Bulletin Board you might have to visit our competitor to "sign on".

As operators of the finest microprocessors available on the market today and as a Users' Group second to none, I do not like to sit and have our BB on a system that caters to another group. (That was a little pride showing through . . . I hope they take the "hint"!!) Meanwhile, we must sit tight and wait to see what developes in the near future.

Related to this is my next thought . . QUESTION . . . How many of you HUGBB users are familiar with "SOURCE"? We just want to put out some "feelers" and see what kind of response you give. The SOURCE people stopped by to pay us a visit again . . I guess in the past there were pretty bad feelings about SOURCE. At the present time they have much improved there services as I understand. This was my first introduction to SOURCE and I was impressed to say the least.

Bob will be writing an article for REMark 17 which will explain SOURCE and their revisit here to HUG. So until we have a chance to view SOURCE, we will "stay low" so as not to cause any "uproar" . . whatever that may include.

Well, there you have it . . . good news, bad news, and informative news all in one nice "swoop". Until we know more, that is the extent that we can cover in detail about the MicroNET system and the HUG Bulletin Board.

For your reference I am including an outline of tenative facilities of the new bulletin board that MNET is designing for the HUG Bulletin Board and other Special Interest Groups (SIGs). (The numbers in parentheses are relative priorities in implementation.)

NOTE: This outline is dated the 10th of February 1981 with a transistion completion date of after March 1st. (This is why we may be into a new Bulletin Board before you get this issue.)

MNET-80 type of message system
- (1) a. Public messages
- (1) b. Member Only (MO) messages

```
                    i.  For everyone
                   ii.  For specific member
(1) Variable line length output, keyed to user's terminal
        parameters
(2) Direct feedback to SYSOP
    MO and Public Databases
        (1)  a.  Direct typeout
        (2)  b.  FILTRN access for downloading
        (3)  c.  Send to user's filespace
(3) Host-resident Program Library
        For example, programs which exploit specific features
        of manufacturer's equipment, such as sound or special
        graphics.
(1) Choice of Menu or Command mode
(2) Real-time Facilities
             a.  USTAT - show other current users of the SIG (BB)
             b.  "CB-like" communication with:
                  i.  Public channel
                 ii.  Member Only channel
             c.  "TALK" for private user-to-user communication
    SYSOP functions (keyed on SYSOP's User ID)
        (1)  a.  Edit MO access list
        (1)  b.  Edit databases
        (3)  c.  Edit program library
        (1)  d.  Delete any public or MO message
        (2)  e.  Read/delete SYSOP feedback
```

This will give you a general picture of what the new HUGBB could conceivably consist of . . . We may or may not be on the new Bulletin Board at the time of this issues' release.

The following messages have been left on the HUG Bulletin Board and I thought you might find them informative . . .

```
Msg#-   3805
Date-  FEB. 5, 1981   23:38
From- Tom Jorgenson 70120,153
To- JERRY ZUCKERMAN
Subject- HDOS/CPM

JERRY - NEVER WOULD'VE BELIEVED THAT SUCH A CONTROVERSY COULD
ARISE OVER SUCH THINGS AS THESE TWO OPERATING SYSTEMS.
AS WITH ANYTHING ELSE, NEITHER SYSTEM IS ACTUALLY 'BETTER' OR
MORE PROFESSIONAL THAN THE OTHER! (BESIDES, HAVING WORKED ON
'PROFESSIONAL MINIS AND LARGE-FRAMES FOR THE LAST DECADE' I
CAN TELL YOU THAT BOTH ARE SUPERIOR TO MANY OF THE PROS).
CP/M IS A NON-PROTECTIVE SYSTEM INTENTIONALLY, IT'S INTENDED
MAINLY FOR THE ASSEMBLY PROGRAMMER (WHO GETS PERTURBED WHENEVER
THE SYSTEM FIGHTS HIM WITH NON-RETRIEVABLE ERRORS). HDOS IS
INTENDED FOR MAXIMUM PROTECTION OF THE SYSTEM FROM THE USER,
AND TO BE EASY TO LEARN (WHICH CP/M ISN'T). BOTH SYSTEMS HAVE
BOTH FLIGHTS OF GENIUS AND STRAINS OF FOOLISHNESS IN THEM (WHICH
THEY KNOW AND ARE STUCK WITH)...SO, WHAT THE HEY! I USE BOTH AS
EACH IS INTENDED...BUT I SURE HATE TO SEE EITHER OF THEM KNOCKED!!
BEST REGARDS AND ALL!! TOM JORGENSON
```

```
Msg#-   3891
Date-  FEB. 8, 1981   18:14
From- ROB RALSTON 70040,767
Subject- MICRONET COMMUNICATIONS

JIM, I HAD ANOTHER PROBLEM WHEN GETTING ACQUAINTED WITH MICRONET THAT MAY
BE USEFUL TO PASS ON TO OTHER HUG MEMBERS WHO HAVEN'T BECOME USERS OF THE
MICRONET SYSTEM YET. WHEN I FIRST TRIED TO LOG-ON, I GOT A REQUEST FOR
"U3%2 ID :". AT FIRST I THOUGHT THERE WAS A NOISE PROBLEM ON THE LINE, BUT
NO MATTER HOW MANY TIMES I HIT CTL-C I GOT THE SAME MESSAGE. I KNEW THEN THAT
IT WASN'T WITH THE SYSTEM BUT WAS PROBABLY SOMETHING 'SIMPLE' ON MY END.
WELL, A COUPLE OF PHONE CALLS LATER I WAS TALKING TO MR. JIM DAVENPORT AT
COMPUSERVE. WHEN I EXPLAINED MY PROBLEM HE SAID, "YOU MUST HAVE A HEATHKIT
```

TERMINAL". I KNEW THEN THAT I WAS SAVED. WELL, AS I THOUGHT, IT WAS SIMPLE. I WAS RECEIVING LOWER CASE ASCII WHICH MY H9 COULDN'T SWALLOW. HE INFORMED ME THAT I COULD TEMPORARILY RECONFIGURE OUTPUT TO MY TERMINAL BY TYPING: "MIC<RETURN>" THEN:"TER VTD 232<RETURN>". NOW I WAS RECEIVING ALL UPPER CASE ASCII AND COULD RUN THE 'R DEFALT' PROGRAM TO PERMANENTLY RECONFIGURE OUTPUT TO MY TERMINAL. OF COURSE, AT LOGIN I STILL RECEIVE THE GARBLED INFO UNTIL IT RECOGNIZES ME AFTER MY PASSWORD.

Msg#- 3890
Date- FEB. 8, 1981 18:14
From- ROB RALSTON 70040,767
To- JIM BLAKE
Subject- UDS MODEM

JIM, I WOULD LIKE TO PASS ON SOME INFORMATION FOR POSSIBLE PUBLICATION IN A FUTURE ISSUE OF REMARK. I RECENTLY PURCHASED THE 'UDS' DIRECT CONNECT MODEM THRU HEATH. THE UNIT APPEARED NOT TO FUNCTION WHEN CONNECTED UP TO MY H9 TERMINAL. UPON INVESTIGATING WITH A SCOPE, I FOUND THAT AFTER ISSUING THE INITIAL CTL-C TO ACCESS THE SYSTEM, I WAS INDEED GETTING A RESPONSE BUT MY TERMINAL WASN'T PRINTING ANYTHING. THE MODEM WAS APPARENTLY NOT DRIVING THE EIA INPUT HARD ENOUGH, AS THE VOLTAGE LEVELS WERE BORDERLINE. WITH A RESISTANCE SUBSTITUTION BOX I FOUND THAT ADDING 10K ACCROSS R618 ON THE I/O BOARD MADE EVERYTHING WORK JUST FINE. I CERTAINLY DON'T KNOW IF THIS IS UNIQUE TO MY H9, OR POSSIBLY DUE TO LOW VOLTAGE LEVELS ON MY PHONE LINES, BUT IT WORKED AND I HOPE THAT PASSING THIS ON MAY HELP SOMEONE ELSE.

Msg#- 4141
Date- FEB. 17, 1981 20:43
From- Mike Cogswell 70140,363
To- SYSOP: 70000,21 (X)
Subject- GENDW.DW

There is a copy of 'GENDW.DW' available on my directory for your use, (or anyone else's). The program can be downloaded from MNET and assembled using the standard HDOS 'ASM.ABS' assembler. Once assembled, GENDW will take any '.ABS' program as an input file and use it to create a source code that consists of nothing but an 'ORG' and a lot of 'DW' statements. The resulting '.DW' file can then be uploaded to MNET and made available for others to use since the "source" is in ASCII. These '.DW' files can be downloaded and reassembled using the Heath assembler back into the original '.ABS' program. A slick way of transferring '.ABS' files over the phone using programs that will not transmit an absolute file. There is one small catch however, the 'GENDW.DW' program WILL NOT WORK with a relocatable program (i.e., one originally assembled using the CODE PIC psuedo-op). In other words, it won't work with the system overlays or any of the Device Drivers. Other than that, it's great!
Hope you find this helpful -Mike
P.S. Please delete message #4118 for me. TNX

Tom Jorgenson, I hope you don't mind me printing your expose' of HDOS verses CP/M. There was a big "discussion" going on over the Bulletin Board that weekend and I felt that Tom summed up the most objective view . . . and may I say, I agree whole heartedly with him.

Well, that about "raps it up" for the "HUGBB on MicroNET" for this issue of REMark. I thank all of you HUGBB members who have been so very encouraging during the "ups and downs" of the Bulletin Board. You are the "gang" that makes the HUGBB an interesting and informative Board . . . Keep up the excellent input and "good work".

SYSOP <TLJ>

(vectored from page 5)

```
--NEWOUT/BEEP.ASM/
--BYE
```

When we typed "BY" the EDITor completed its' task and returned us to the HDOS prompt (>). Under HDOS, we can now produce a "hardcopy" if a printer is available (optional) by typing the following:

```
>COPY LP:=BEEP.ASM (RETURN)
```

You should now have a listing which looks exactly like fig. A. Further, if you catalog your disk you will find the file named BEEP.ASM under the directory.

WHAT NEXT?

We have now DESIGNED a program. We have examined the FLOW. We have (hopefully) used the HDOS EDIT to enter the program. We have discussed FIELDS used by the Assembler. Next, we will use the ASM (Assembler) to obtain a listing and a "runnable" program. But, right now it's time for my coffee-break and you'll have to wait 'till the next issue of REMark to complete this thing.

Local HUG News

NOHUG (New Orleans Heath Users' Group) is currently meeting the first Wednesday of every month at 7:30 pm. This group meets at the Kenner Heath Electronics Center located at 1900 Veterans Memorial Highway; Kenner, LA 70062. For additional information, contact the Heath Center by calling (504) 467-6321.

Richard Senecal is interested in forming a new club for Heath users' in his area. Richard can be contacted at his home by writing him at RFD#3 Box 283 A.; Columbia, SC 29206 or by calling (803) 736-0510 after 6:00 pm. Rich says he's a little remote and hopes there are other users in his area to help out!

H-11 OWNERS.....

A Special Interest Group (SIG) for H-11 users is forming within the Capital Heath Users' Group known as CHUG in the Washington, DC area. For specific information, contact J. Bramlage c/o CHUG; P.O. Box 341; Fairfax, VA 22030.

WANTING TO START NEW GROUP

Mr. Robert Sloat of Tice Florida is interested in contacting owners of Heath equipment with hopes of forming a local HUG. Bob teaches novice classes sponsored by the Fort Myers Amateur Radio Club. He informs us that he is equipped with an H89 along with an H8 and H9. Bob can be contacted by writing to P.O. Box 05-37; Tice, FL 33905.

From Canada...The Greater Vancouver Heath Users' Group meets on the final Wednesday of each month at their local Heath Electronics Center located at 3058 Kingsway, Vancouver B.C. at 7:30 pm.

The Pomona Heath Users' Group has been formed in Pomona, California. Meetings are scheduled for the first and third Saturday of each month at 3:30 pm. They will be held at the Heath Electronics Center located on 1555 North Orange Grove Ave.; Pomona, CA 91767. Contact Doug at (714) 623-3543 for additional information.

DENHUG INCORPORATES
AS A NON-PROFIT ORGANIZATION

The Denver Heath Users' Group general membership meetings are held on the second Monday of each month at the local Heath Electronics Center located on 5940 West 38th Ave.; Denver, CO 80212. Further Details can be obtained from Alfred K. Carr, Treasure and Registered Agent for DENHUG by writing P.O. Box 20422; Denver, CO 80220; or calling (303) 320-7552 (voice and modem). Al informs us that their group is currently exchanging newsletters with CHUG (Capital Heath Users' Group) to increase the exchange of information for all members of both groups!

NEW GROUP IN LITTLEROCK

Richard Allen is interested in forming a local HUG in or around Little Rock. If you wish to contact Dick you may write him at 4300 East 43rd Street; North Little Rock AR 72117.

Jack Leach and Richard Crawford are attempting to contact local users in the Portland area in an effort to form a new HUG group. If you are interested in joining their campaign, you can contact them at the following addresses:

Jack Leach (206) 693-2485
5008 N.E. 18th Ave.
Vancouver, Wash 98663

Richard Crawford (503) 642-9307
3220 S.W. 173rd
Aloha, OR 97006

CLUB HAPPENINGS

Remember, if you're interested in forming a new Heath Users' Group in your area or if your club is planning special activities, PLEASE let us know as quickly as possible so that we can pass on this valuable information to others!

(vectored from page 15)

expression "No-Date" in the date column of your directory for all files created under NO-DATE. In my opinion, having a date looks nicer, even if it is an old one.

TRACK 2 SECTOR 0

LOCATION	OLD VALUE	NEW VALUE
53	20	3A
54	A8	A0
76	29	20
77	3F	20
7C	CD	C9

3. Remove "BOOT"

This patch is a cosmetic improvement for the first patch. It eliminates the printing of the word "BOOT" by HDOS, which is still done after the first patch even though it is not needed.

TRACK 0 SECTOR 2

LOCATION	OLD VALUE	NEW VALUE
C6	42	20
C7	4F	20
C8	4F	20
C9	54	20

These patches, when combined with an appropriate PROLOGUE.SYS and the proper switch settings on an H89 with the new ROMs or an H8 with the extended configuration option, can give you true turn-key operation. You can turn the power on, insert your disk(s), and be in a user program with no further operations necessary.

PS:

HUG BUG: The part number for the CP/M programs TERM and H8COPY was listed incorrectly in the REMARK #15 HUG Products List (page 17) as 885-1027. The correct part number is 885-1207.

Hole to Fill

Your Heath Users' Group team has been working at "break-neck" speed to bring REMark back on schedule! I am happy to report that we are almost there. In the future we hope to bring you some interesting and "powerful" data base systems submitted by your fellow users. Things are beginning to shape up rapidly with many new and exciting programs for all of us to look at. We're still knee deep in work and hope that each of you will offer the continued support that has been noted in the last couple of months. THANX TO ALL.

BE:

P.S. This is all the room the "gang" would give me for comments!!!

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

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

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

H11/H19 Video Editor

The H-19 Video Editor is a powerful screen oriented editor which uses the features of the H-19 terminal to allow the H-19 screen to serve as a "window" on the Text Buffer. All changes to the Text Buffer are displayed immediately, making the editing process very accurate. The cursor is used as the Text Pointer thus keeping you informed of the current buffer position.

The main editor of the H-19 Video Editor is the DEC editor TECO Version 28. TECO is an extremely powerful and complicated editor. Over 40 TECO commands have been implemented on the H-19 keypad and Special Function keys so that most editing tasks may be completed without ever typing an editor command sequence. For example, 'HOME' takes the text pointer to the start of the Text Buffer and the 'ERASE' deletes text from the Text Pointer position to the end of the buffer.

ORDERING INFORMATION

The H-19 Video Editor is available on an 8" floppy disk with user manual for $75.00 from:

David L. O'Conner
370 Eden Street
Buffalo, NY 14220

NOTE: A version of this program for RT-11 Version 4 and TECO Version 36 is available. This will not run on older versions of TECO or HT-11. The price is the same. Specify VIDEO EDITOR FOR TECO 36 when ordering.

That Football in Tiny BASIC

```
10 LET W=-5
20 LET A=-5
30 LET B=A*A
40 LET C=0
50 IF B=C THEN PRINT "*";
60 IF B>C THEN PRINT " ";
70 LET C=C+1
80 IF C<=B THEN GOTO 50
90 LET A=A+1
100 LET X=W*W
110 LET Y=58-X
120 LET Z=C
130 IF Y=Z THEN PRINT "*"
140 IF Y>Z THEN PRINT " ";
150 LET Z=Z+1
160 IF Y>=Z THEN 130
170 LET W=W+1
180 IF W<>6 THEN 30
190 END
```

Heath Users' Group
Hilltop Road
St. Joseph MI 49085

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

POSTMASTER: If undeliverable, please do not return.

885-2016