REMark

Issue 10 • May 1980

Official magazine for users of Heath computer equipment.

On the cover... is the first photograph of a computer on the GREAT WALL. A Heath H 88.

on the stack

>CAT

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

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

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

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

Send payment to:

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

HUG Manager and Editor .....Jim Blake
HUG Secretary .....Susan Gilfoyle
Software Developer .....Gerry Kabelman
Software Developer .....Jon Falkner

REMark

THEY CALLED ME 'MR. BRAKE'

My watch says it's 2 o'clock in the morning but it's been daylight for over 20 hours. I'm not sure what day it is. I don't know whether to have breakfast , dinner or a three martini lunch. Just a few hours ago, I was having my favorite steak dinner in front of the fireplace. Plush carpeting, soft music, the TV going in the other room. A newspaper lying on the floor. Then BAM! All gone.

"On behalf of the Peoples Republic of China, we welcome you, Mr. Brake, to Peking!" It is -10 degrees outside. Jian Guo Men Wai street (eight lanes) is nearly black with workers (all dressed nearly alike) riding their bicycles to work. A two hour ride for most. Many must walk. All will be paid 80 Yaun for a months work. $56. The high humidity and low temperature seeps into your body within seconds like an anaesthetic.

In a few hours, I am to meet with 17 representatives from various government agencies including the Bank of China and the chief of the export/import bureau to demo the H 89. And... my thermal underwear and computer is in Tokyo!

But days later, as the Public Relations chief, an engineer, my interpreter and I ride through the country side headed North, I stare out the window while re-winding the many meetings and conversations. I see men and women dredging a lake a shovel full at a time. The streets and even the road out here in the countryside are spotlessly clean since hundreds of women are happily sweeping with a crude broom. We pass several men pushing a disabled one ton truck UP hill to some distant place for repair. (Who would you call? Triple A?) They are triming the trees along the roadside with what appears to be a six inch key-hole saw. And, it looks like they must have measured each limb to the centimeter before trimming. They carry the limbs away on a bicycle or donkey-pulled cart. And I remember my visit to the Forbidden City where every room of the Emporer's many quarters is laden with precious jewels, huge gold clocks, priceless Ming vases. All unguarded. There are no locks on the doors. And tipping is forbidden. The floor attendant at the hotel acted like he could hardly wait for me to get in at night so he could bring me ice and shine my shoes. Free. He liked his job, even though he worked twelve hours a day, six days, because it was a plush position. After all, the hotel is heated. And he enjoyed looking at the computer. Even feeling the foam in which it was packed. I insist he accept a stick of gum. It made his day! - - Fortunately, the sky is blue and it must be in the thirties and we have reached our destination 50 Kilometers Northeast of the city... Pataling Pass. The Great Wall. What a sight! Built a stone at a time, it spans 2000 miles following the contour of the mountains reaching heights of almost 4000 feet. From 'up here' I look out over the beautiful countryside. North to what used to be Mongolia. South over China. I think about it. Five. Maybe ten years from now, you won't recognize it. The dining room of the Peking Hotel looks like a businessmen's Las Vegas. Big businesses rolling for billions. All here to get a piece of the action. "Help China modernize."

But what beautiful people, these Chinese. Sensitive. Kind. Hospitable. Friendly. Trustworthy. Considerate. Thankful.

It lasted two long weeks. On the way to the airport, my host reaches into his bag and retrieves a small hand-carved wooden turtle. "Long life, Mr. Brake. Long life." Long life, Mr. Liang.

I think you will find this issue very informative. Doc Campbell's article should be welcomed by the novice as well as the experienced. And, we had a new problem with this issue. We have more information than we have room. But I think we have a solution. Beginning with the next issue (in late JULY) REMark will be out each month excluding January and July. You asked for it. It will cause the dues to raise four bucks, but (in the survey) you said you didn't care. If you have something to contribute, send it along on a disk or cassette. :JB:

GETTING STARTED WITH MBASIC AND HDOS WITH PARTICULAR REFERENCE TO RANDOM FILES

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

It is assumed that the reader has a fundamental knowledge of BASIC and an H89 or H8 with "floppy disk" storage, HDOS (Heath Disk Operating System) and MBASIC (Microsoft H8-21).

MBASIC offers many features not contained in Heath's Extended Benton Harbor BASIC. It also has some minor differences in syntax. CONVER.BAS (PROGRAM 1) is a conversion program which will convert many of the differences for you. It is not too important to understand the logic of this program, but casual review will reveal several syntax differences which the program corrects. No claim is made that CONVER.BAS is foolproof, but I found it extremely timesaving.

Certain MBASIC commands and statements should be noted here. (LP: references a "Line Printer".) You load MBASIC by typing `MBASIC'. There are also certain options that can be entered at this time. (Note that you must `LOAD LP:' before you load MBASIC, if you intend to "call" your LP: while in MBASIC.) You get an existing disk BASIC program into memory by command LOAD "OLDPGM.BAS". Almost all MBASIC commands can also be used as statements within a BASIC program. Thus, you can place on a line number RUN "SY1:PGMNAME". This is how you can chain from one program to another, variables will be lost when using RUN. You could even use LIST or DELETE in a program if you so desired (yes, it is possible to write a program that deletes itself!). Unless otherwise instructed, MBASIC saves a program onto disk as a "compiled" program. This saves disk space and also causes the program to load faster. The command is SAVE "PGMNAME" and MBASIC provides a "default" extension of `.BAS'. If you later attempt to TYPE PGMNAME.BAS from HDOS monitor, the compiled program lists as "garbage". However, you can SAVE a program in ASCII (as Heath Extended BASIC always does) with the command SAVE "PGMNAME",A. (This is also necessary if you intend to use the MERGE command to append the program to another currently in memory.) To get "hard copy" listing of a program currently in memory, you SAVE "LP:",A. (and don't forget the ,A for if you do you get garbage.) Note that MBASIC does not possess a REPLACE command; you use SAVE instead. To exit MBASIC you type `SYSTEM' and this takes you to HDOS monitor. The SYSTEM command can also be used as the last logical statement in a program so that after the program has finished it will automatically take you to HDOS.

MBASIC provides for "error-handling". MBASIC also allows RANDOM FILE access which makes for extremely potent file handling capabilities. `NEW' deletes all current memory (similar to SCRATCH in HBASIC.) There are numerous other commands/statements available to you such as PRINT USING and double precision features and all these are described in your MBASIC manual. The rest of this article will deal with the use of MBASIC in handling "data files".

I have kept my SEQUENTIAL DATA FILES in the following format.

```
LASTNAME\FIRST\STREET\CITYSTATEZIP
```

Such a file is easily generated by a program such as MAKSEQ.BAS (PROGRAM 2). Programs to alter, delete, append to and SORT (to put in order, e.g. to alphabetize, or put in ascending or descending numerical order) and provide appropriate printout are relatively easy to write and use. As can be seen, each record is divided into segments by a "delimiter" (a reverse slash is used here as a delimiter), and the records are separated in a New Line character by HDOS. I prefer to use the INSTR function of MBASIC (MATCH function in BHBASIC) to find the locations of each delimiter, and the MID$ and LEN functions to isolate and manipulate the various items in each record. A sample program to provide formatted printout is SEQOUT.BAS (PROGRAM 3). Lines 70 to 80 determine the positions of the delimiters and lines 90 to 110 `grab' the different segments of the records. Note that it is not necessary to put delimiters between the town and state, nor between state and zip code. The 2 letter state abbreviation and the 5 number zip code bear a constant relationship to LEN(R$). If you are not familiar with the INSTR and MID$ functions, an easy way to learn is to just enter lines 70-110, and then add as line 60:

```
R$="LAST\FIRST\MR\STREET\CITY\STATEZIP"
```

```
10 REM  CONVER.BAS  CONVERTS H8-17 BASIC PGMS TO MICROSOFT SYNTAX
15 REM  PROGRAM 1
20 REM
30 CLEAR (2000)
40 PRINT "ENTER NAME OF FILE FOR CONVERSION (EX.  SY1:OLD.BAS)....? ";
50 LINE INPUT C9$
70 I1$="SY1:NEW.BAS"
80 DIM T(200):I=1
90 OPEN "I",1,C9$
100 OPEN "O",2,I1$
110 PRINT #2,"2 CLEAR(1000):REM  THIS IS JUST ESTIMATE."
120 IF EOF(1) THEN 930
130 LINE INPUT #1,X$
140 IF K9=-1 THEN IF INSTR(1,X$,"CHR$")<>0 THEN K9=0:GOTO 130
150 IF INSTR(1,X$,"LINE INPUT")<>0 THEN GOSUB 230
160 IF INSTR(1,X$,"MATCH")<>0 THEN GOSUB 370
170 IF INSTR(1,X$,"PRINT#")<>0 OR INSTR(1,X$,"PRINT #")<>0 THEN GOSUB 520
180 IF INSTR(1,X$,"OPEN")<>0 THEN GOSUB 630
190 IF INSTR(1,X$,"CIN(")<>0 THEN GOSUB 800
200 PRINT #2,X$
210 PRINT X$
220 GOTO 120
230 REM ";" HANDLER for LINE INPUT - also handles LINE INPUT #0,
240 E=INSTR(1,X$,"LINE INPUT")
250 IF INSTR(1,X$,"#0,")<>0 THEN T=INSTR(1,X$,"#0,") ELSE 270
260 X$=MID$(X$,1,T-1)+MID$(X$,T+3,LEN(X$)-(T+3)+1)
270 IF INSTR(1,X$,CHR$(34)+";")<>0 THEN 360
280 IF INSTR(1,X$,CHR$(34)+" ;")<>0 THEN 360
290 IF MID$(X$,E+10,1)=";" THEN MID$(X$,E+10,1)=" "
300 IF MID$(X$,E+11,1)=";" THEN MID$(X$,E+11,1)=" "
310 IF MID$(X$,E+12,1)=";" THEN MID$(X$,E+12,1)=" "
320 X$=MID$(X$,1,E+9)+MID$(X$,E+11,LEN(X$))
330 IF INSTR(1,X$,"#")<>0 THEN A=INSTR(1,X$,"#") ELSE 360
340 IF INSTR(A,X$,";")<>0 THEN A=INSTR(A,X$,";") ELSE 360
350 X$=MID$(X$,1,A-1)+MID$(X$,A+1,LEN(X$))
360 RETURN
370 REM  MATCH handler
380 X=1
390 X1=INSTR(X,X$,"MATCH")
400 MID$(X$,X1,5)="INSTR"
410 X2=INSTR(X1+5,X$,",")
420 X3=INSTR(X2+1,X$,",")
430 X4=INSTR(X3+1,X$,")")
440 A$=MID$(X$,1,X1+5)
450 B$=MID$(X$,X1+6,(X2-1)-(X1+5))
460 C$=MID$(X$,X2+1,X3-X2-1)
470 D$=MID$(X$,X3+1,X4-X3-1)
480 X$=A$+D$+","+B$+","+C$+MID$(X$,X4,LEN(X$)-X4+1)
490 X=X1+1
500 IF INSTR(X,X$,"MATCH")=0 THEN RETURN
510 GOTO 390
520 REM ";" handler for PRINT # statements
530 X=1
540 X1=INSTR(1,X$,"#")
550 IF INSTR(X1,X$,";")=0 THEN RETURN
560 E=INSTR(X1,X$,";")
570 A$=MID$(X$,1,E-1)
580 B$=MID$(X$,E+1,LEN(X$)-E+1)
590 X$=A$+B$
600 X=X1+1
610 IF INSTR(X,X$,";")=0 THEN RETURN
620 GOTO 540
630 REM  handler for 'OPEN' statements
640 E=INSTR(1,X$,"OPEN")
650 F=INSTR(1,X$,"#")
660 G=INSTR(1,X$,"FOR")
670 A$=MID$(X$,1,E+4)
680 B$=MID$(X$,E+5,G-1-(E+5))
690 IF INSTR(1,X$,"READ")=0 THEN 710
700 X$=A$+CHR$(34)+"I"+CHR$(34)+","+MID$(X$,F+1,1)+","+B$:GOTO 720
710 X$=A$+CHR$(34)+"O"+CHR$(34)+","+MID$(X$,F+1,1)+","+B$
720 IF INSTR(1,X$,".")=0 THEN 740
730 RETURN
740 PRINT:PRINT"MICROSOFT DOES NOT HAVE DEFAULT EXTENSION FOR FILES TO BE"
750 PRINT"OPENED. MAKE SURE THAT ";B$;" HAS AN EXTENSION SUCH AS '.DAT'."
760 PRINT "IF NOT, ADD A LINE SUCH AS 000 ";B$;" = ";B$;"+";CHR$(34);".DAT";CHR$(34)
770 PRINT "JUST BEFORE THE OPEN STATEMENT IN QUESTION."
780 PRINT "HIT RETURN TO CONTINUE...";:INPUT K
790 GOTO 730
800 REM  CIN() eof handler
810 A=INSTR(1,X$," ")
820 A$=MID$(X$,1,A-1)
830 B=INSTR(1,X$,"(")
840 B$=MID$(X$,B+1,1)
850 LINE INPUT #1,Y$
860 FOR R=LEN(Y$) TO 1 STEP -1
870 IF MID$(Y$,R,1)=" " THEN 890
880 NEXT R
890 C$=MID$(Y$,R+1,LEN(Y$),-R)
900 X$=A$+" IF EOF("+B$+") THEN "+C$
910 K9=-1
920 RETURN
930 PRINT
940 PRINT "CONVERTED FILE ON DISC AND CALLED  SY1:NEW.BAS"
950 PRINT:PRINT "CHEERS"
960 CLOSE
970 END
```

Then, add a temporary line 120 PRINT A, B, C, D,:STOP . When you run this program you can check the values of A thru D and see that they do show each position of the reverse slash. Then you can check each MID$ and LEN function to see how each works. (To OUTPUT to LP: with this program, add line 45 OPEN "O",2,"LP:" and change "PRINT" to "PRINT 2," in lines 120 through 150.)

You will probably have noted that I use no punctuation in my mailing list programs. This is a matter of personal preference. I don't, since it saves memory. And, although I have a fine matrix printer, I prefer all CAPS as shown. The last name first, followed by first name is for purposes of SORTING. It is also easy to "grab" the zip code and place it first if one desires to sort on, for example, zip codes, to put them in ascending numerical order.

```
10 REM MAKSEQ.BAS   CREATE A SEQUENTIAL DATA FILE
15 REM PROGRAM 2
20 REM
30 CLEAR(1000):INPUT "SEQUENTIAL FILE NAME (EX. SY1:SEQ.DAT)... ";T$
40 OPEN "O",1,T$
50 PRINT "PLEASE USE NO PUNCTUATION."
60 PRINT "IF DONE TYPE  DONE":PRINT
70 LINE INPUT "'SALUTATION'.... ";A$
80 IF A$="DONE" THEN 190
90 LINE INPUT "FIRST NAME & MIDDLE INITIAL.... ";B$
100 LINE INPUT "LAST NAME.... ";C$
110 LINE INPUT "STREET ADDRESS.... ";D$
120 LINE INPUT "TOWN OR CITY.... ";E$
130 LINE INPUT "STATE (2 LETTER ABBREVIATION PLEASE).... ";F$
140 LINE INPUT "ZIP CODE.... ";G$
150 X$=C$+"\"+B$+"\"+A$+"\"+D$+"\"+E$+F$+G$
160 PRINT #1,X$
170 PRINT
180 GOTO 50
190 CLOSE:END
```

A sample program for sorting sequential data files is given in SMM.BAS (PROGRAM 4). I have included this since I have never seen a Microsoft listing published using the "Shell-Metzner" sort for strings. (Shell and Metzner are the individuals who developed this sorting algorithm.) (Algorithm - 'a rule or set of rules to solve a mathematical problem'.) Lines 80-120 are the lines concerned with the actual sort. Note that in this

as in most of the other programs a CLEAR statement is prominent at the beginning of each program. MBASIC requires this. You must declare "string space" and this is what the CLEAR statement does in this context. Lines 60, 80, and 110 show the use of the ELSE statement. And line 110 also shows MBASIC's SWAP statement.

```
10 REM   SEQOUT.BAS    PRINTS OUT SEQUENTIAL FILE (AS FOR MAILING LABELS)
15 REM   PROGRAM 3
20 REM
30 INPUT "SEQUENTIAL DATA FILE DESIRED (EX. SY1:SEQ.DAT)... ";P$
40 OPEN "I",1,P$
50 IF EOF(1) THEN 170
60 LINE INPUT #1,R$
70 A=INSTR(1,R$,"\"):B=INSTR(A+1,R$,"\")
80 C=INSTR(B+1,R$,"\"):D=INSTR(C+1,R$,"\")
90 A$=MID$(R$,1,A-1):B$=MID$(R$,A+1,B-1-A):C$=MID$(R$,B+1,C-1-B)
100 D$=MID$(R$,C+1,D-1-C):E$=MID$(R$,D+1,LEN(R$)-D-7)
110 F$=MID$(R$,LEN(R$)-6,2):G$=MID$(R$,LEN(R$)-4,5)
120 PRINT C$+" "+B$+" "+A$
130 PRINT D$
140 PRINT E$+" "+F$+"   "+G$
150 PRINT:PRINT
160 GOTO 50
170 CLOSE:END
```

The problem with SEQUENTIAL file handling is simply that it can be very time consuming to access any given record, since a sequential examination of all records before the desired record must be made.

```
10 REM  SMM.BAS   SHELL-METZNER SORT FOR SEQUENTIAL STRINGS
15 REM  PROGRAM 4
20 REM
30 CLEAR (2500):DIM A$(200):I%=1
40 INPUT "SEQ FILE NAME TO BE SORTED.. ";P$:INPUT "OUTPUT FILE NAME... ";T$
50 OPEN "I",1,P$:OPEN "O",2,T$
60 IF EOF(1) THEN 70 ELSE LINE INPUT#1,A$(I%):I%=I%+1:GOTO 60
70 I%=I%-1:C%=I%:B%=I%
80 C%=INT(C%/2):PRINT C%:IF C%=0 THEN 130 ELSE D%=1:E%=B%-C%
90 F%=D%
100 G%=F%+C%:IF A$(F%)<=A$(G%) THEN 120
110 SWAP A$(F%),A$(G%):F%=F%-C%:IF F%<1 THEN 120 ELSE 100
120 D%=D%+1:IF D%>E% THEN 80 ELSE 90
130 FOR X%=1 TO I%:PRINT #2,A$(X%):NEXT X%:CLOSE:END
```

RANDOM ACCESS FILES - If data files are created and manipulated using RANDOM ACCESS, any desired record can be accessed almost instantaneously, regardless of the size of the data file! This makes for quick and easy "updating" of any desired record. Since MBASIC finds the desired record by "counting" this implies RECORDS OF FIXED LENGTH. The user MUST follow the necessary MBASIC protocol in creation and manipulation of data files in order to make the random access possible!

Input from, and output to disc is done by moving a 256 byte (256 character) aggregate of data at a time. (The discerning reader notes that there are 256 bytes in one disc sector.) Whenever a file is opened, HDOS and MBASIC create (in computer memory) a 256 character "buffer" and movement of data to and from disc is always done through this buffer. One can visualize, therefore, that when inputting from disc, 1 sector of data is placed in the buffer, and then the desired data is taken from buffer and displayed or manipulated as desired. When outputting to disc, conversely, the data is placed in buffer and, when desired, the data is written to disc, again as a 256 byte aggregate. (This happens also with sequential data files, but here the process is usually transparent to the user.)

Listed below are summaries of procedures for random file manipulation: (Note that all data records must be of same length.)

To write data to disc -
1. OPEN disc file (this opens a given file and creates a 256 byte buffer.)
2. FIELD the buffer (creates "discrete" divisions within buffer, if desired.)
3. Enter the data.
4. LSET the data ("pads" data to desired length with spaces and puts in buffer.)
5. PUT the data (writes the 256 character buffer to disc.)
6. Repeat 3 to 5 as necessary.
7. CLOSE the disc file.

To access data from disc -
1. OPEN disc file (and create a 256 byte buffer associated with this file.)
2. FIELD the buffer. (This step is sometimes done AFTER step 3.)
3. FIND desired record using a formula.
4. GET the correct sector and its data from disc into memory buffer.
5. MANIPULATE data (display, update, etc.) after removing any padded spaces.
6. CLOSE the file.

It should be noted that after each GET or PUT statement, unless a record number is specified, MBASIC automatically increments to next higher sector, unless the buffer is reFIELDed.

The newcomer to Microsoft Random files frequently has trouble with "FIELDing", and the use of the formula mentioned previously.

FIELDS - You could use one 256 byte sector for each record. But, this is usually very wasteful of disc space. Therefore, it is customary to divide the buffer into 2,3,4 or more subdivisions, each of the same size. (There may be some bytes left over but this doesn't matter.) For all of our sample random programs, we will divide the buffer into 4 equal subdivisions of 64 bytes each. (64 times 4 = 256.) A convenient way of doing this in a program is:

```
FOR I=0 TO 3
FIELD n,(I*64) AS D$,64 AS R$(I)
NEXT I
```

Where n= the number of the disc file OPENed for "R", that we are fielding. This creates 4 fields in the 256 byte buffer, each 64 bytes long. D$ is used as a "dummy" field. For I=0, I*64=0 and R$(0) is therefore fielded as the first 64 bytes of the sector (buffer). For I=1, I*64=64 and D$ is fielded as 64 and R$(1) is therefore fielded as the second 64 bytes of the sector. And, so on. It should be pointed out that you may field a buffer any number of times in a program. We will not use D$, but only R$(I). Another way of doing exactly the same thing takes longer but may help explain fields:

```
  FIELD #n, 64 AS R$(0),192 AS D$
  FIELD #n, 64 AS D$,64 AS R$(1),128 AS D$
  FIELD #n, 128 AS D$,64 AS R$(2),64 AS D$
  FIELD #n, 192 AS D$,64 AS R$(3)
```

Or, another way:

```
  FIELD #n, 64 as R$(0),64 as R$(1),@
        64 as R$(2),64 as R$(3)
```

NOTE: The '@' = continued on next line.

All 3 methods divide our buffer into 4 units, each 64 bytes long.

Important Note: VARIABLES THAT ARE USED IN FIELD STATEMENTS (such as R$(I) above) should NOT be used in LET (to left of =), READ, or INPUT statements elsewhere in your program. This is because "field variables" are pointing into the 256 character buffer. Use in the 3 statements mentioned causes them to point to a different area of memory. If you DO use them then you MUST refield the buffer before LSETing, PUTting, or GETting.

FORMULAS TO ACCESS RECORDS - To find any given record in a random data file one usually accesses it by a 'KEY' record number. This 'key' number is the absolute record number of the record in a sequential listing of all the records.

For example, assume we had 10 records in a file which had been created by 'FIELDing' the buffer into four 64 byte records. Our first disc sector would contain records 1 through 4, the second disc sector would contain records 5 through 8, and the third sector would contain records 9 and 10. Further assume we wanted to access record number 7. 7 is our key number and one of our sample programs (RNUM.BAS) will provide us with a sequential listing of all records in a random file, and each record is preceeded by its 'key' number.

Here is how to find record 7 (for example):

```
S=INT((K-1)/4)+1
```

'S' is the sector that will contain the record and 'K' is our 'key' number (7). INT of (7-1)/4=1 and 1+1=2. So, S=2 and 2 is the sector that contains record 7.

```
SS=K-(4*(S-1))
```

'SS' is the desired subdivision (or subsector) (or 'FIELD') of the desired sector (we just determined that record 7 is in sector 2) so 2-1=1 and 4*1=4 and 7-4=3. So, SS=3 and record 7 therefore is the third record in sector 2.

With such a few records as used as illustration, the above is probably obvious. But, the formulas prove true regardless of the key number of the record.

The above formulas use '4' since '4' was how the sector was originally fielded. If you 'fielded' your buffer as 3 equal records of 85 bytes each, then '3' would be used in the formulas instead of '4'.

You should now realize why it is necessary to usually USE THE SAME FIELDS IN ACCESSING RECORDS AS WERE USED IN CREATING THE FILES!!

3 PITFALLS - There are 3 possible problems that may come up when dealing with random files. They are mentioned now so that you may see how the various programs which follow deal with them.

First, the "padding" spaces that MBASIC fills out your records with (to bring each record to the desired length) are NOT automatically removed when you access a record from a random disc file. The user's programs must remove the spaces! Second, when you PUT, you are always "putting" 256 bytes of data. You may unknowingly be "putting" some "leftover" material. For example, assume 64 byte long records and that you are creating a file. You are only going to PUT 6 records total. You have just PUT the first 4 records onto your disc file. You next enter records 5 and 6, LSET, then PUT them. What you actually PUT were records 5 and 6 AND records 3 and 4 which were "leftover" in the buffer from the first PUT! The third pitfall can occur when you are creating a new random data file, but unknowingly there may be a pre-existing file of the same name on disc. Assume again that you have 6 records to PUT to a file fielded as 4*64. Assume also that there is a pre-existing file on disc with the same name and that it contains 100 records. If you OPEN the 'new' file you are actually opening a pre-existing file. You PUT your 6 records and CLOSE. What you then have is a file containing these 6 records PLUS 94 records that were in the old pre-existing file!

These 3 problems are very easily solved when writing random access programs, but you must be aware of their potential dangers!

EXPLANATION OF RANDOM PROGRAMS - Since I already had my data in various sequential data files, I wanted to put the data into random files to utilize the fast access time of random files, and the rest of these programs show my approach.

First, I had to check to make sure none of the records were longer than 64 characters. LENFIX.BAS (PROGRAM 5) took care of that.

```
10 REM  LENFIX.BAS  FLAGS SEQ RECS WHOSE LEN IN CHARS EXCEEDS DESIRED LEN
15 REM  PROGRAM 5
20 REM
30 CLEAR (1000):INPUT "MAXIMUM RECORD LENGTH DESIRED..... ";L%
40 INPUT "SEQ INPUT FILE NAME... ";A$:INPUT "OUTPUT FILE NAME... ";B$
50 OPEN "I",1,A$:OPEN "O",2,B$:PRINT
60 IF EOF(1) THEN 130 ELSE LINE INPUT #1,X$
70 IF LEN(X$)>L% THEN 80 ELSE PRINT #2,X$:GOTO 60
80 T%=5:PRINT X$
90 PRINT:PRINT "ABOVE IS TOO LONG BY";LEN(X$)-L%;"CHARACTERS. ABBREV NOW... "
100 PRINT:LINE INPUT C$:IF LEN(C$)=0 THEN 80
110 PRINT:IF LEN(C$)>L% THEN X$=C$:GOTO 80
120 X$=C$:GOTO 70
130 CLOSE:IF T%=5 THEN PRINT"REVISED FILE ON DISC CALLED ";B$:GOTO 150
140 KILL B$:PRINT "ALL RECORDS WITHIN REQUIRED LENGTH! ORIGINAL FILE OK!"
150 END
```

LENFIX.BAS is a relatively straightforward program. Line 50 shows MBASIC's syntax for opening sequential files for input, and also for output. Line 60 shows how MBASIC's "end of file" routine is used, and also uses the ELSE statement. The T%=5 in line 80 is simply used as a marker. The '5' could just as well be '-1' as long as the '5' in line 130 was changed to '-1'. The point here is that if, when file is closed in line 130, and the marker IS present (T%=5), then one or more records of file was longer than L%, and abbreviation was necessary. The marker would only be set if line 80 was executed at least once. Line 80 is never accessed if LEN(X%) is never | L%. Line 140 is only executed if the marker is NOT set (T%=0) and it demonstrates how to delete a file, from an MBASIC program. The '%' that is appended to the 2 numeric variables in the program declare them as integers. (Note that when a program has a lot of numeric variables that can be declared as integers, it pays to do so since the program will run approximately 30 percent faster!) It is suggested that the reader run a short "test" data file through this program, and enter a value for L% that is known to be shorter than some of the records in the "test" file. This will demonstrate how the program works.

SEQTORAN.BAS (PROGRAM 6) is designed to be a demonstration program.

Line 60 deletes any pre-existing file named "T$" from the disk, IF a file named T$ is present on the disk. However, consider that if NO file named T$ is present on disk, MBASIC would stop and print message "FILE NOT FOUND". To prevent this happening an 'error routine' is used. Line 30 tells MBASIC to goto line 300 if ANY error happens during program run. Line 300 says IF error is "FILE NOT FOUND" (53 is code for this - see manual), AND, IF the error occurred on line 60 (ERL=60) then RESUME NEXT (keep right on with program resuming at next line after error and don't stop!) However, if the conditions of line 300 are NOT met, the program executes line 310 which is the error routine's syntax for saying "STOP and print the error message". If you are unfamiliar with MBASIC's error handling routines check manual now. Line 80 opens file T$ for random access. Line 90 "fields" the buffer as we have discussed. Line 130 inputs a record from our pre-existing sequential data file (P$). Lines 140 and 150 use the formulas previously discussed and are present in this program only because of the program's "demonstration" nature. LINE 160 appends CHR$(0) to our record. CHR$(0) is a "null" and is a binary zero. Reasons for this will be given in discussion of PROGRAM 7 (RNUM.BAS). Line 170 shows how LSET is used to 'pad' the record with spaces if necessary. The LSET also puts the record into the buffer in its correct element R$(I). Line 180 PUTS the whole 256 character buffer on disk in sector S. Lines 190-210 are for informational purposes. Line 220 increments I but keeps it in the range of 0 to 3. K is incremented in line 230 (for our edification) and the program cycles back to line 120 and next record is input. This keeps up until EOF(1) and then program branches to line 240. Here, if I=0 or I=4, our records have come out 'even' (no 'leftovers') and we are done. If we did NOT come out even, we want to "fix" the buffer so that any subsectors beyond our desired records are not filled with "leftovers" when we PUT the 'odd' records onto disk from buffer. Line 250 puts nulls in the next segment of the buffer, and incidentally demonstrates how MBASIC's STRING$ function operates (see your manual). Line 250 then PUTs the nulls onto disk, increments I and branches to 240, until I=4, when we are done.

```
10 REM  SEQTORAN.BAS  CONVERTS SEQ DATA FILE TO RANDOM ACCESS DATA FILE.
15 REM  PROGRAM 6         A DEMONSTRATION PROGRAM
20 REM
30 CLEAR (1000):ON ERROR GOTO 300
40 LINE INPUT "TYPE SEQ DATA FILE NAME (EX. SY1:EXAMPLE.DAT)... ";P$
50 LINE INPUT "TYPE OUTPUT FILE NAME (EX. SY1:REX.DAT)... ";T$
60 KILL T$
70 OPEN "I",1,P$
80 OPEN "R",2,T$
90 FOR I=0 TO 3:FIELD 2,(I*64) AS D$,64 AS R$(I):NEXT I
100 I=0
110 K=1
120 IF EOF(1) THEN 240
130 LINE INPUT #1,X$
140 S=INT((K-1)/4)+1
150 SS=K-(4*(S-1))
160 X$=X$+CHR$(0)
170 LSET R$(I)=X$
180 PUT 2,S
190 PRINT S,SS
200 PRINT R$(I)
210 PRINT "LEN(R$(I)) = ";LEN(R$(I))
220 I=I+1:IF I=4 THEN I=0
230 K=K+1:GOTO 120
240 IF I=0 OR I=4 THEN 260
250 LSET R$(I)=STRING$(64,CHR$(0)):PUT 2,S:I=I+1:GOTO 240
260 CLOSE
270 PRINT:PRINT
280 PRINT "THERE WERE";K-1;"RECORDS PUT INTO RANDOM ACCESS FILE."
290 END
300 IF ERR=53 AND ERL=60 THEN RESUME NEXT
310 ON ERROR GOTO 0
```

```
10 REM  RNUM.BAS    PRINTS OUT RANDOM FILE CONTENTS WITH KEY NUMBERS
15 REM  PROGRAM 7
20 REM
30 CLEAR (1000):INPUT "READY LINE PRINTER, HIT RETURN WHEN READY..";Z9$
40 INPUT "RANDOM ACCESS FILE NAME (EX. SY1:EXAMPLE.DAT)... ";P$
50 OPEN "R",1,P$:OPEN "O",2,"LP:":K%=1
60 GOSUB 140
70 FOR I%=0 TO 3:FIELD #1,(I%*64) AS D$,64 AS R$(I%):NEXT I%
80 GET #1
90 FOR I%=0 TO 3
100 IF ASC(LEFT$(R$(I%),1))=0 THEN 260
110 IF INSTR(1,R$(I%),CHR$(0))=0 THEN R$=R$(I%):GOTO 130
120 Z%=INSTR(1,R$(I%),CHR$(0)):R$=LEFT$(R$(I%),Z%-1)
130 PRINT #2,K%;"  ";R$:K%=K%+1:NEXT I%:GOTO 80
140 DA$="":FOR I=8383 TO 8391:DA$=DA$+CHR$(PEEK(I)):NEXT I
150 IF MID$(DA$,1,1)="0" THEN DA$=" "+MID$(DA$,2,9)
160 A$="January.February.March.April.May.June.July.August."
170 A$=A$+"September.October.November.December."
180 X$=MID$(DA$,4,3):X=INSTR(1,A$,X$):Y=INSTR(X,A$,".")
190 DA$=MID$(A$,X,Y-X)+" "+MID$(DA$,1,2)+", 19"+MID$(DA$,8,2)
200 PRINT #2,DA$:PRINT #2,:PRINT #2,
210 PRINT #2,"THE KEY NUMBER TO LEFT OF EACH RECORD IN THIS PRINTOUT OF RANDOM"
220 PRINT #2,"FILE ";P$;" IS THE KEY ACCESS NUMBER OF EACH RECORD AND"
230 PRINT #2,"INDICATES THE ABSOLUTE CURRENT POSITION OF EACH RECORD IN FILE."
240 PRINT #2,:PRINT #2,TAB(20)P$:PRINT #2,:PRINT #2,
250 RETURN
260 CLOSE:END
```

Our next task is to provide "hard copy" of our newly created random file and furnish "KEY" record numbers for each record. This is done with - RNUM.BAS (PROGRAM 7). (If you do not have LP:, change the reference in line 50 from LP: to TT:.) Line 60 GOSUBs to line 140 and line 140 furnishes the present system date. DA$ on line 140 contains date in format that we used when entering it. (Note that you can use more than one alphabetic character when naming variables in MBASIC. But only the first 2 are significant to MBASIC.) Wanting the date more neatly displayed, lines 150 through 190 furnish a snappy DA$ for printout. The gosub prints the desired message and then returns at line 250. Line 70 FIELDs the buffer and line 80 GETs the first sector from disk file 1 and places its contents into the 256 character buffer associated with file opened as 1. Remember the "nulls" we inserted in the last program? Line 100 checks for the presence of a null (or a binary zero) in the FIRST character position of R$(I). (Note that all sectors above the ones in our file will be filled with nulls by MBASIC). When a "FIRST character null" is found our file input is finished! We had either put the null there or MBASIC had done so. That is how to bypass the first "pitfall"! We took care of the third "pitfall" with the 'KILL statement' in the previous program. Line 120 takes care of the other pitfall. Line 110 looks for the first occurrence of a binary zero (other than a "FIRST character null" which line 100 screened out) in the record R$(I) using the INSTR function. The first occurrence of a binary zero will be the one we appended to our record when we setup the random file. (See CHR$(0), line 160, of previous PROGRAM 6.) If there is NOT a null present, that means that our actual record was, and is, exactly 64 characters in length, and the null was truncated (means 'removed') when we LSET that record in the previous program 6. The program then branches to line 130 where K% (K% is a 'counter') is printed

and then the record is printed; K% is incremented and then we go to NEXT I%, until I%=4 when line 130 branches back to 80 and we GET the next sector into buffer memory. Consider what happens if a null IS present (usually it IS) when tested for in line 110. If present (INSTR[|0), program would fall to line 120. Line 120 strips away the null and any padded spaces from the record! Then K% and this record are printed. (In this particular program it really doesn't matter whether or not we strip them away. But, I feel it is good practice to include the two statements in line 120 in all appropriate random file programs.) Now that we have hard copy listing of the records with preceding KEY record numbers, a program is needed to access the records by key number to check everything out.

RTEST.BAS (PROGRAM 8) - The various parts of this test program have been previously discussed. Briefly, line 50 OPENs file P$ for random access. You enter the key record number of desired record in answer to line 60's prompt (the key record number is obtained from the printout of PROGRAM 7, RNUM.BAS.) The 2 formulas in line 70 determine the desired sector (S%) and subsector (SS%). Line 80 FIELDs the buffer, and GETs the desired sector into the buffer. Line 90 tells us if record number entered was too high by finding a "FIRST character null" (a first character null means that we have passed the last valid record in this random file). Line 100 says that if there are no "nulls" in valid record then there is no need to strip away the null and any padded spaces (which line 110 does) so program skips line 110 and branches to 120. If there IS a "null" with or without spaces, the INSTR function of line 100 returns a number other than "0" and program falls to line 110 which strips the offenders away. (Z% gives the position of the "null" we appended to the record when it was created, and R$ is made to contain just the record itself with the LEFT$ statement.) Lines 120 through 160 are exactly the same as those in PROGRAM 3 (SEQOUT.BAS). Lines 170 to 190 provide a formatted output, then the program cycles back to line 60. IMPORTANT NOTE: Each time a record is accessed in this program, the buffer is refielded! This is because we have used the buffer variable (R$) in one or more LET statements!

STOR.BAS (PROGRAM 9) - This is the program actually used to convert a sequential data file to a random access data file. It is similar to Program 6, except that the 'non-essentials' have been left out. One important difference is that this program (9) is more efficient. The demonstration program (6) PUT each record to disk immediately after the record was placed in the buffer. This program waits until 4 records have been placed in their proper 'subsectors' in the buffer, and THEN PUTs the whole buffer onto disk (line 110 - after the "FOR - NEXT loop" has been completed.) Note once again the 3 solutions to the potential "PITFALLS". Line 50 KILLs any pre-existing output file (and the error routines prevent the program from stopping in the event there was no prior file); line 100 appends a "null" to our record for later ease in stripping any 'padded spaces'; and, line 130 fills the rest of the buffer with "nulls" if our input records did not come out "even". Also, please note line 120. Assume that line 120 had been branched to, and that I% = 2. Following the ELSE, the statement says 'FOR 2=2 TO 3'. This IS a legal use of the FOR - NEXT statement.

RTOS.BAS (PROGRAM 10) - This program converts a random file to a sequential file. The statements have all been explained in previous programs, except for line 105 which is explained in RPACK.BAS (PROGRAM 16). Line 80 tests for a "FIRST character null" and, if present, branches to 120. Note again that a "FIRST character null" means the end of the file. Note that the fielded variable (R$(I)) is never used (to the left of =) in a "LET" statement so that the buffer does NOT have to be 'refielded' in this program.

COMPARE.BAS (PROGRAM 11) - This is a handy program to have available. After creating a random file from sequential file 'X', and then creating a 'new' sequential file 'Y' from the random file just created, it seemed desirable to have a way of checking to make sure that the original and the last sequential file just made were identical. This program compares 2 sequential ASCII files for "sameness". Again, we use the INSTR function in line 70 to do the comparison. This program is quite straightforward.

RALTER.BAS (PROGRAM 12) - To update a random file I use a program similar to this one. Most parts of this program have been covered previously except for the "updating" sections. After the KEY record number is entered in response to the prompt in line 60, line 70 checks to make sure that a numerical value was probably entered; if not, or if just a [cr| was entered, line 60 is branched to. Otherwise, line 80 converts the string K$ to a numerical integer, K%. Line 90 uses the 2 formulas to "find" the record, and line 100 FIELDs the buffer, and GETs the desired sector into the buffer. Lines 200-220 display the contents of the record and assign a number to each item in the record. In answer to line 230's prompt, you type the number associated with item to be altered. Assume that you wanted to alter item 2. Line 240 uses INPUT$ to place the '2' in variable T$. Since T$ contains '2' line 280 is brought into

```
10 REM   RTEST.BAS     TEST PGM TO CHECK INPUT METHOD FOR RANDOM FILES
15 REM   PROGRAM 8
20 REM
30 CLEAR (1000)
40 INPUT "RANDOM ACCESS FILE NAME (EX. SY1:REX.DAT)... ";P$
50 OPEN "R",1,P$
60 INPUT "TYPE REC # WANTED (IF DONE, TYPE 0)... ";K%:IF K%=0 THEN 210
70 S%=INT((K%-1)/4)+1:SS%=K%-4*(S%-1)
80 FIELD #1,((SS%-1)*64) AS D$,64 AS R$:GET #1,S%:PRINT
90 IF ASC(LEFT$(R$,1))=0 THEN PRINT "NUMBER ENTERED TOO HIGH":PRINT:GOTO 60
100 IF INSTR(1,R$,CHR$(0))=0 THEN 120
110 Z%=INSTR(1,R$,CHR$(0)):R$=LEFT$(R$,Z%-1)
120 A=INSTR(1,R$,"\"):B=INSTR(A+1,R$,"\")
130 C=INSTR(B+1,R$,"\"):D=INSTR(C+1,R$,"\")
140 A$=MID$(R$,1,A-1):B$=MID$(R$,A+1,B-1-A):C$=MID$(R$,B+1,C-1-B)
150 D$=MID$(R$,C+1,D-1-C):E$=MID$(R$,D+1,LEN(R$)-D-7)
160 F$=MID$(R$,LEN(R$)-6,2):G$=MID$(R$,LEN(R$)-4,5)
170 PRINT C$+" "+B$+" "+A$
180 PRINT D$
190 PRINT E$+" "+F$+"   "+G$
200 PRINT:GOTO 60
210 CLOSE:END
```

```
10 REM  STOR.BAS  CONVERTS SEQ DATA FILE TO RANDOM ACCESS DATA FILE.
15 REM  PROGRAM 9
20 REM
30 CLEAR (1000):ON ERROR GOTO 150
40 INPUT "SEQ INPUT FILE NAME... ";P$:INPUT "RAN OUTPUT FILE NAME... ";T$
50 KILL T$
60 OPEN "I",1,P$
70 OPEN "R",2,T$
80 FOR I%=0 TO 3:FIELD 2,(I%*64) AS D$,64 AS R$(I%):NEXT I%
90 FOR I%=0 TO 3:IF EOF(1) THEN 120
100 LINE INPUT #1,X$:X$=X$+CHR$(0):LSET R$(I%)=X$
110 NEXT I%:PUT 2:GOTO 90
120 IF I%=0 THEN 140 ELSE FOR I%=I% TO 3
130 LSET R$(I%)=STRING$(64,CHR$(0)):NEXT I%:PUT 2
140 CLOSE:END
150 IF ERR=53 AND ERL=50 THEN RESUME NEXT
160 ON ERROR GOTO 0
```

```
10 REM   RTOS.BAS   CONVERTS RANDOM TO SEQUENTIAL FILE (ALSO 'PACKS' FILE)
15 REM   PROGRAM 10 (ALSO PROGRAM 16)
20 REM
30 CLEAR (1000)
40 INPUT "RAN INPUT FILE NAME... ";P$:INPUT "SEQ OUTPUT FILE NAME... ";T$
50 OPEN "R",1,P$:OPEN "O",2,T$
60 FOR I%=0 TO 3:FIELD #1,(I%*64)AS D$,64 AS R$(I%):NEXT I%
70 GET #1:FOR I%=0 TO 3
80 IF ASC(LEFT$(R$(I%),1))=0 THEN 120
90 IF INSTR(1,R$(I%),CHR$(0)) =0 THEN R$=R$(I%):GOTO 105
100 Z%=INSTR(1,R$(I%),CHR$(0)):R$=LEFT$(R$(I%),Z%-1)
105 IF LEFT$(R$,1)="*" THEN NEXT I%:GOTO 70
110 PRINT #2,R$:NEXT I%:GOTO 70
120 CLOSE:END
```

```
10 REM COMPARE.BAS    COMPARES 2 SEQ ASCII DISC FILES FOR IDENTICALNESS
15 REM PROGRAM 11
20 REM
30 INPUT "SEQ FILE #1 NAME... ";A$:INPUT "SEQ FILE #2 NAME... ";B$
40 OPEN "I",1,A$:OPEN "I",2,B$
45 IF EOF(1) THEN 90
50 IF EOF(2) THEN 90
60 LINE INPUT #1,A$:LINE INPUT #2,B$:PRINT A$:PRINT B$:PRINT
70 IF A$<>B$ THEN PRINT "DIFFERENT!":T=5:INPUT "CR TO CONTINUE";M$
80 GOTO 45
90 IF T=5 THEN PRINT: PRINT  "FILES WERE DIFFERENT.":GOTO 110
100 PRINT:PRINT "FILES ARE SAME!"
110 CLOSE:END
```

```
10 REM   RALTER.BAS   PGM TO ALTER RECORDS IN A RANDOM FILE (1 S = 4X64 SS)
15 REM   PROGRAM 12
20 REM
30 CLEAR (1000)
40 INPUT "RANDOM ACCESS FILE NAME (EX. SY1:REX.DAT)... ";P$
50 OPEN "R",1,P$
60 PRINT:INPUT "TYPE REC # WANTED (IF DONE, TYPE 0)... ";K$
70 IF LEFT$(K$,1)>"9"OR LEFT$(K$,1)<"0" THEN PRINT:PRINT "INVALID ENTRY!":GOTO60
80 K%=VAL(K$):IF K%=0 THEN 420
90 S%=INT((K%-1)/4)+1:SS%=K%-4*(S%-1)
100 FIELD #1,((SS%-1)*64) AS D$,64 AS R$:GET #1,S%:PRINT:X$=R$
110 IF ASC(LEFT$(X$,1))=0 THEN PRINT "NUMBER ENTERED TOO HIGH":GOTO 60
120 IF LEFT$(X$,1)="*" THEN GOSUB 400
130 IF INSTR(1,X$,CHR$(0))=0 THEN 150
140 Z%=INSTR(1,X$,CHR$(0)):X$=LEFT$(X$,Z%-1)
150 A=INSTR(1,X$,"\"):B=INSTR(A+1,X$,"\")
160 C=INSTR(B+1,X$,"\"):D=INSTR(C+1,X$,"\")
170 A$=MID$(X$,1,A-1):B$=MID$(X$,A+1,B-1-A):C$=MID$(X$,B+1,C-1-B)
180 D$=MID$(X$,C+1,D-1-C):E$=MID$(X$,D+1,LEN(X$)-D-7)
190 F$=MID$(X$,LEN(X$)-6,2):G$=MID$(X$,LEN(X$)-4,5)
```

```
200 PRINT "HERE IS RECORD NUMBER ";K$;" ...."
210 PRINT:PRINT "1  ";C$:PRINT "2  ";R$:PRINT "3  ";A$:PRINT "4  ";D$
220 PRINT "5  ";E$:PRINT "6  ";F$:PRINT "7  ";G$:PRINT
230 PRINT "TYPE NUMBER OF ITEM TO BE ALTERED (IF ALL OK TYPE 0)... ";
240 T$=INPUT$(1):PRINT T$
250 PRINT
260 IF T$>"7" OR T$<"0"  THEN PRINT "INVALID ENTRY!":PRINT:GOTO 230
270 IF T$="1" THEN N$=C$:GOSUB380:C$=N$:GOTO 210
280 IF T$="2" THEN N$=B$:GOSUB380:B$=N$:GOTO 210
290 IF T$="3" THEN N$=A$:GOSUB380:A$=N$:GOTO 210
300 IF T$="4" THEN N$=D$:GOSUB380:D$=N$:GOTO 210
310 IF T$="5" THEN N$=E$:GOSUB380:E$=N$:GOTO 210
320 IF T$="6" THEN N$=F$:GOSUB380:F$=N$:GOTO 210
330 IF T$="7" THEN N$=G$:GOSUB380:G$=N$:GOTO 210
340 IF T$="0" THEN X$=A$+"\"+B$+"\"+C$+"\"+D$+"\"+E$+F$+G$+CHR$(0)
350 IF LEN(X$)>65 THEN PRINT:PRINT "ENTRY TOO LONG! ABBREVIATE!":GOTO 210
360 LSET R$=X$:PUT #1,S%
370 GOTO 60
380 PRINT "OLD ITEM.... ";:PRINT N$:PRINT
390 LINE INPUT "ENTER NEW ITEM... ";N$:RETURN
400 PRINT "** NOTE THAT THIS RECORD HAS PREVIOUSLY BEEN MARKED FOR DELETION! **"
410 RETURN
420 CLOSE:END
```

Continued from page 10

play. First, the 'old item' (B$) is placed in N$. GOSUB 380 prints the old item for review and line 390 prompts us to enter the corrected item into N$ (replacing the 'old item' in N$.) RETURNing to line 280, N$ is now put back into B$. (At this point note that the corrected item is now in B$.) Continuing, the program branches to line 210, and again the various items are displayed, only the corrected item is now displayed opposite 2. You can recorrect the same item, or correct other items as desired. When display is finally suitable (updated) you enter 0 (zero) and all 'IF' statements fail until line 340. Since T$ does contain 0 (zero) line 340 concatenates the various items and appends a "NULL". Line 350 makes sure the record length is not over 65 characters (including the 1 appended null), and line 360 LSETs the record into the correct buffer variable (R$) (note that as soon as we "got" R$ in line 100, it was immediately placed in X$, so that R$ which is a buffer variable is still pointing to the same location in the buffer). The next statement on line 360 PUTs the sector in buffer back on disk. Program then cycles to line 60 where another record KEY number can be entered, or, when all updating is concluded, user types 0 (zero) and line 80 branches to 420 to terminate. Line 120 and it's GOSUB are explained in DELRAN.BAS (PROGRAM 15).

RTORSM.BAS (PROGRAM 13) - This is the same Shell-Metzner SORT program as Program 4, except that this program takes a random disk file, puts it in an array in memory, sorts it, then returns the sorted data to disk as a random file.

MAKRAN.BAS (PROGRAM 14) - Here is a program to make a random data file from scratch or append records to a pre-existing random data file. Note that any appended records are added to the bottom of the data file. They can be sorted into the file at a later time if desired. A new MBASIC statement is introduced in line 70. LOF(n) is the keyword which gives the number of the last 'sector' in a random data file n. If each record occupies one sector, then LOF(n) gives the KEY of last record in file. In files under discussion, however, we get the last sector number of the file. If a file is nonexistent (or has been named but never used), LOF(n) returns a zero (0) and this is utilized in line 70 to GOTO 300 where we are so informed. After user interaction, program goes to line 100 where a new file is started and data information is solicited from user with the GOSUB 140. After the information is entered, formatted, and 'LSETed' with an appended CHR$(0) (line 280), line 290 RETURNS to NEXT I% on line 100. When 4 records have been entered, line 100 PUTs them on disk, and recycles to beginning of line. However, IF there was a pre-existing file, then LOF(1) places the last used sector number in S% and we fall to line 80 which GETs that sector. At this point we know that there is one (or more) record(s) in S% but we do not know how many. Line 90 finds out for us. It looks for a "FIRST character null". Since we know there is at least ONE record we use 1 TO 3 (rather than 0 TO 3) in FOR...NEXT loop. If loop does NOT find a "FIRST character null" then that means that this last file sector has its full complement of 4 records; so, N% is set to 0, S% is incremented by one, and we fall to line 100, and begin to append records. Consider, however, if line 90's FOR..NEXT loop finds a "FIRST character null". If it DOES, then ASC function = 0 and we immediately branch to 100, with N% containing the number (1, 2, or 3) of first unused subdivision of this sector. Line 100's FOR...NEXT loop then functions as described above. Note that the loop is for N% TO 3. This insures that as we add our appended records they are put in

appropriate subdivision(s). Line 230 helps force user to enter only a 2 letter abbreviation for state, and line 250 helps ensure a 5 digit zip code entry. Line 270 checks to make sure record is not too long. The 'PRINT CHR$(7)' in line 390 rings the bell on our terminal.

DELRAN.BAS (PROGRAM 15) - To delete one or more records in random data files, we usually simply tag the record(s) for later actual physical deletion. This program replaces the first character of a record with an asterisk (*). (Any suitable character unused for other purposes in the data file could alternatively be used.) Line 260 does the task. The other portions of this program have been previously explained. Note that line 120 of RALTER.BAS (PROGRAM 12) informed us if a selected record in that program had been marked for deletion. The actual physical deletion of record(s) is explained below in "PROGRAM 16"

```
10 REM  RTORSM.BAS  RANDOM FILE TO RANDOM FILE SHELL-METZNER SORT
15 REM  PROGRAM 13
20 REM
30 CLEAR (9500):DIM A$(250):L%=1:ON ERROR GOTO 290
40 INPUT "RAN FILE NAME TO BE SORTED.. ";P$:INPUT "RAN OUTPUT FILE NAME... ";T$
50 KILL T$
60 K%=1
70 OPEN "R",1,P$
80 OPEN "R",2,T$
90 FOR I%=0 TO 3:FIELD #1,(I%*64)AS D$,64 AS R$(I%):NEXT I%:GET #1,K%
100 FOR I%=0 TO 3
110 IF ASC(LEFT$(R$(I%),1))=0 THEN 160
120 IF INSTR(1,R$(I%),CHR$(0))=0 THEN 140
130 Z%=INSTR(1,R$(I%),CHR$(0)):R$(I%)=LEFT$(R$(I%),Z%-1)
140 A$(L%)=R$(I%):L%=L%+1
150 NEXT I%:K%=K%+1:GOTO 90
160 L%=L%-1:C%=L%:B%=L%
170 C%=INT(C%/2):PRINT C%:IF C%=0 THEN 220 ELSE D%=1:E%=B%-C%
180 F%=D%
190 G%=F%+C%:IF A$(F%)<=A$(G%) THEN 210
200 SWAP A$(F%),A$(G%):F%=F%-C%:IF F%<1 THEN 210 ELSE 190
210 D%=D%+1:IF D%>E% THEN 170 ELSE 180
220 X%=1
230 FOR I%=0 TO 3:FIELD #2,(I%*64) AS D$,64 AS R$(I%):NEXT I%
240 FOR I%=0 TO 3:IF X%=L%+1 THEN 260 ELSE LSET R$(I%)=A$(X%)+CHR$(0):X%=X%+1
250 NEXT I%:PUT 2:GOTO 240
260 IF I%=0 THEN 280 ELSE FOR I%=I% TO 3
270 LSET R$(I%)=STRING$(64,CHR$(0)):NEXT I%:PUT 2
280 CLOSE:END
290 IF ERR=53 AND ERL=50 THEN RESUME NEXT
300 ON ERROR GOTO 0
```

RPACK.BAS (PROGRAM 16) - There is NO listing for this program as it is the SAME program as RTOS.BAS (PROGRAM 10). Line 105 of program 10 simply says that IF the first character of any record is an asterisk (*), then GOTO NEXT I%. This prevents any "asterisk headed record" from being output to disk in line 110, and so those records are physically removed from the sequential file being created!

MERGE.BAS (PROGRAM 17) - Again, this program is included simply because I have never seen a published listing of a "merge" for Microsoft Basic. The program will merge into one sequential sorted data file, any two previously sorted sequential data files. Note the use of -1 as a 'marker' in lines 120 and 150. This is necessary for proper operation. The logic of the program is not difficult to ascertain.

Continued on the next page

```
10 REM MAKRAN.BAS   CREATES RANDOM FILE FROM SCRATCH OR APPENDS TO EXISTING ONE
15 REM PROGRAM 14
20 REM
30 CLEAR (1000)
40 INPUT "RANDOM  FILE NAME (EX. SY1:OLD.DAT)... ";P$
50 OPEN "R",1,P$
60 FOR I%=0 TO 3:FIELD 1,(I%*64) AS D$,64 AS R$(I%):NEXT I%
70 S%=LOF(1):IF S%=0 THEN 300
80 GET #1,S%
90 FOR N%=1 TO 3:IF ASC(LEFT$(R$(N%),1))=0 THEN 100 ELSE NEXT N%:S%=S%+1:N%=0
100 FOR I%=N% TO 3:GOSUB 140:NEXT I%:PUT 1,S%:N%=0:S%=S%+1:GOTO 100
110 IF I%=0 THEN 130 ELSE FOR I%=I% TO 3
120 LSET R$(I%)=STRING$(64,CHR$(0)):NEXT I%:PUT 1,S%
130 CLOSE:END
140 PRINT:PRINT "(PLEASE USE NO PUNCTUATION.)"
150 PRINT:PRINT "IF FINISHED, TYPE    DONE..":PRINT
160 LINE INPUT "'SALUTATION'.... ";A$
170 IF A$="DONE" THEN 110
180 LINE INPUT "FIRST NAME & MIDDLE INITIAL.... ";B$
190 LINE INPUT "LAST NAME.... ";C$
200 LINE INPUT "STREET ADDRESS.... ";D$
210 LINE INPUT "TOWN OR CITY.... ";E$
220 LINE INPUT "STATE (2 LETTER ABBREVIATION PLEASE....) ";F$
230 IF LEN(F$)<>2 THEN PRINT:PRINT "INVALID ENTRY!":PRINT:GOTO 220
240 LINE INPUT "ZIP CODE.... ";G$
250 IF LEN(G$)<>5 THEN PRINT:PRINT "INVALID ENTRY!":PRINT:GOTO 240
260 X$=C$+"\"+B$+"\"+A$+"\"+D$+"\"+E$+F$+G$
270 IF LEN(X$)>64 THEN GOSUB 380:GOTO 140
280 LSET R$(I%)=X$+CHR$(0)
290 RETURN
300 PRINT:PRINT TAB(10)"THERE IS NO FILE BY THAT NAME!"
310 PRINT TAB(10)"IF YOU DESIRE TO START A NEW FILE,"
320 PRINT TAB(10)"TYPE   YES   AND <cr>. TO ENTER A"
330 PRINT TAB(10)"DIFFERENT FILE NAME, JUST <cr>... ";
340 INPUT B$
350 IF LEN(B$)=0 THEN CLOSE:KILL P$:PRINT:GOTO 40
360 IF LEFT$(B$,1)="Y" OR LEFT$(B$,1)="y" THEN S%=1:PRINT"NEW FILE....":GOTO 100
370 GOTO 300
380 PRINT:PRINT "ENTRY TOO LONG BY ";LEN(X$)-64;" CHARACTERS!!"
390 PRINT CHR$(7):PRINT TAB(10)"PLEASE ABBREVIATE AND ENTER AGAIN...":PRINT
400 RETURN
```

FINAL SUGGESTIONS AND REMARKS - Your MBASIC manual shows other ways of FIELDing random files. For data files such as we have discussed, the entire record has been fielded as one string into one subdivision of a sector. Of course, depending on the nature of your data, numbers can be converted to strings (MKI$, etc.) before LSETing the data, and reconverted later for manipulation (CVI, etc.). And there are many other ways of manipulating random files. Many of our programs can be combined. For example, it would be easy to add a few statements to STOR.BAS so that it would provide hard copy output of each record (preceded by its absolute KEY record number) as each record was being accessed by the program and put into a random file.

Remember, if you change the number of sector subdivisions (we used 4) to another number, then the FIELD statements must be changed, and also the "FOR...NEXT" loops must be changed from 0 TO 3 to 0 TO n-1 where n equals the number of subdivisions in each sector. Also, you must change "4" in each formula to "n".

In several programs we used the "KILL" statement. This is always a dangerous statement and you might wish to substitute the "NAME...AS...." statement (example - NAME "FILNAME.DAT" AS "FILNAME.BAK").

In general, since MBASIC itself takes up a lot of memory space, sorting is frequently done with a machine language sort. An excellent machine language sort is available from HUG (Part 885-1044) and is called "SORTER". I use it frequently as it is extremely fast and based on the Shell-Metzner routine. (This disk also has some other useful utility programs.) SORTER will sort any ASCII sequential disk data file such as we have discussed, is MUCH faster than any BASIC sort, and is very easy to use!

```
10 REM  MERGE.BAS    MERGES 2 SORTED SEQ FILES INTO 1 SORTED SEQ FILE
15 REM  PROGRAM 17
20 REM
30 CLEAR (1000)
40 INPUT "FIRST SEQ SORTED FILE NAME (EX. SY1:A.DAT)... ";A$
50 INPUT "SECOND SEQ SORTED FILE NAME (EX. SY1:B.DAT)... ";B$
60 INPUT "OUTPUT MERGED SEQ FILE NAME (EX. SY1:C.DAT)... ";C$
70 OPEN "I",1,A$
80 OPEN "I",2,B$
90 OPEN "O",3,C$
100 IF EOF(1) THEN 170
110 LINE INPUT#1,X$
120 IF Z=-1 THEN Z=0:GOTO 150
130 IF EOF(2) THEN 220
140 LINE INPUT#2,Y$
150 IF X$<Y$ THEN PRINT #3,X$: Z=-1: GOTO 100
160 PRINT #3,Y$: GOTO 130
170 PRINT #3,Y$
180 IF EOF(2) THEN 270
190 LINE INPUT#2,Y$
200 PRINT #3,Y$
210 GOTO 180
220 PRINT #3,X$
230 IF EOF(1) THEN 270
240 LINE INPUT#1,X$
250 PRINT #3,X$
260 GOTO 230
270 PRINT
280 PRINT "MERGED DATA FILES ARE NOW IN FILE CALLED ";C$;"."
290 PRINT "HAVE A GOOD DAY!"
300 CLOSE #1: CLOSE #2: CLOSE #3
310 END
```

MBASIC has a "built-in" EDIT function which is extremely helpful and timesaving in writing and revising programs. A renumbering command (RENUM) is another important feature.

Although our programs have used a "mailing list" as illustration, the same programs could be used, with minimal modification, for other string data.

*** EOF ***

```
10 REM   DELRAN.BAS  FUNCTIONALLY DELETES RECORD(S) IN RANDOM FILE
15 REM   PROGRAM 15
20 REM
30 CLEAR (1000)
40 PRINT
50 INPUT "RANDOM INPUT FILE NAME (EX. SY1:FNAME.DAT)... ";P$
60 OPEN "R",1,P$
70 PRINT
80 IF LOF(1)=0 THEN PRINT "NO FILE BY THAT NAME EXISTS!":CLOSE:KILL P$:GOTO 40
90 INPUT "TYPE RECORD # OF RECORD TO BE DELETED. (IF DONE TYPE 0)... ";K$
100 IF LEFT$(K$,1)>"9"OR LEFT$(K$,1)<"0" THEN PRINT:PRINT "INVALID ENTRY!":GOTO70
110 K%=VAL(K$):IF K%=0 THEN 210
120 S%=INT((K%-1)/4)+1:SS%=K%-4*(S%-1)
130 FIELD #1,((SS%-1)*64) AS D$,64 AS R$:GET #1,S%:PRINT:X$=R$
140 IF ASC(LEFT$(X$,1))=0 THEN PRINT "NUMBER ENTERED TOO HIGH":GOTO 70
145 IF LEFT$(X$,1)="*" THEN PRINT "PREVIOUSLY DELETED!":GOTO 70
150 PRINT X$
160 PRINT:PRINT "DELETE ABOVE RECORD ? (Y/N).. ";
170 A$=INPUT$(1):PRINT
180 IF A$="Y" OR A$="y" THEN 260
190 IF A$="N" OR A$="n" THEN 70
200 GOTO 150
210 CLOSE
220 PRINT
230 PRINT "'DELETED' RECORDS NOW HAVE 1ST CHARACTER REPLACED WITH '*' AND"
240 PRINT "ARE 'FUNCTIONALLY' DELETED. IF YOU DESIRE TO PHYSICALLY REMOVE"
250 PRINT "RECORDS, RUN 'RPACK.BAS'.":GOTO 290
260 X$="*"+RIGHT$(X$,LEN(X$)-1)
270 LSET R$=X$:PUT #1,S%
280 PRINT:PRINT "RECORD DELETED!":GOTO 70
290 END
```

Improvements to FOCAL-8

By Patrick Swayne
290 Springdale
Sebastopol, CA 95472

Since sending FOCAL-8 to HUG (P/N 885-1045), I have not left it alone, but have worked on some improvements. I have the following patches to offer. These patches, among other things, will improve the accuracy of the floating point package and some of the pre-defined functions.

This first patch improves the routines that save the floating point accumulator on the stack and retrieve it. The old routines introduced errors in the least significant digits of numbers. The package still is not perfect (no binary floating point package is), but you will be able to see the improvement. If you run the sample Curve Fitting Program with and without the patch, you will notice a difference in the numbers printed after "RMSERROR:" in the first example. Before the patch, the value printed was 1.854160E-03. After, it is 2.908642E-06, a much lower number. A perfect floating point package would give a value of zero. (This applies to example number 1 only, not number 2). A BASIC version of this program gives 5.0278E-06 with B H BASIC 5.01.02, and 4.04609E-06 with 10.05.01. Improved FOCAL-8 is the winner is this case. (Note: These patches are for vers. 1.1).

Save accumulator to stack:

ADDRESS	NEW DATA
040325	301,136,043,126,325,043,136,043
	126,325,043,303,252,074
074252	136,043,126,325,043,305,303,147
	047

Retrieve accumulator from stack:

ADDRESS	NEW DATA
040343	301,043,043,043,043,043,321,162
	053,163,321,303,263,074
074263	053,162,053,163,321,053,162,053
	163,305,311

This patch fixes the problem of the error message "INPUT BUFFER FULL" being printed when the input buffer is not full.

ADDRESS	NEW DATA
050031	072,030,075,315,276,074
074276	052,000,075,167,043,042,000,075
	311

This patch fixes the code that executes DO commands. In the original version, if you supply a non-existant line or group number as the argument to DO, the wrong error message is printed, and if in the program mode, the wrong line number (indicating where the error is) is printed.

ADDRESS	NEW DATA
050232	303,234,074
074234	041,027,075,315,343,040,341,341
	042,210,074,303,201,044

The code that executes the FOR command is effected by this patch. It changes the way the test pointer is moved to and from the stack. The only effect is a slight increase in execution speed.

ADDRESS	NEW DATA
052275	052
052300	345,000,000
052311	341,042,002,075,000,000

In the original version, if you supply a number greater than 19 as the argument to the width control character (%) in TYPE statments, the number is interpreted incorrectly. This patch fixes that problem, allowing numbers up to 99 (but do not exceed the line width of your terminal).

ADDRESS	NEW DATA
054052	207,365,207,207,301,200,137,072
	030,075,346,017,203,137

This patch changes the sign-on title so that it reads "FOCAL-8 V 1.1A" instead of "FOCL-8, VERS. 1.1"

ADDRESS	NEW DATA
045047	101,114,055,070,054,040,126,040
	061,056,061,101,377

Other FOCAL-8 Information

Here is some information not covered in the original documentation. If you want a program to run as fast as possible, you should define the most frequently used variables early in the program. You can even use a dummy assignment statment at the beginning of the program that assigns a constant to the variables.

The trace function can be invoked in the command mode. If you type ?RUN, the entire program will be executed in trace.

The FSUB function uses the DO code to execute the specified group or line. If you supply a non-existant line or group number, you will get a DO error message.

EOF

The Ins and Outs of MODEMs

By D. C. Shoemaker
2000 A Foxridge
Blacksburg, VA 24060

MODEM. We've all come across that term at one time or another in the pursuit of our hobby, and most are aware it means MODulator-DEModulator. Up to a year or so ago, if we bothered to look into the subject very deeply we found that it referred to a rather expensive piece of equipment used for communicating between data devices, usually over the phone lines. If you're like me, with a more or less well-developed system with adequate memory (does anyone ever really have enough memory?), a good terminal and a disk drive or two, you may be ready to take another look.

Without going into great detail, a MODEM will allow you to access any source of digital data that meets some fairly common definitions. First, for the less expensive varity of MODEM, the data transfer rate must be 300 baud or less. The audio signal on the phone line must be in accordance with Bell Telephone standards (that is, compatible with their Bell 103A MODEM.) And finally, the MODEMs we'll talk about are serial devices that fall into the class of Data Terminal Equipment (DTE), the same as a computer but different from a terminal or a printer, which are Data Communication Equipment (DCE.) This last point will be important to keep in mind when it comes time to interface the equipment.

For the purposes of this article, we'll confine ourselves to just one type of MODEM, Novation's CAT. This is possibly the most frequently seen small system MODEM for a number of reasons. First of all, it works, and quite well. The price is also quite reasonable, from about $160.00 on special sales to normal retail of about $200.00. Interfacing is relatively easy, due to the EIA 25-pin DCE female socket built-in. The CAT has its own power supply, isolated from the 120 volt AC line for safety (UL approved) and finally, it is switch-selectable for originate or answer mode. More about that later.

The CAT is one of a family of MODEMs that use accoustical couplers to connect the MODEM to the phone line. With this type, the receiver is placed into the rubber cup-like recepticles on the MODEM, and no other link is required. There are other types, known as hard-wired dial-up MODEMs, that are connected directly to the phone line, usually by plugging into the wall phone recepticle. These are generally more expensive, and are subject to local phone company tariffs and regulations. They may not be quite so portable, but often transmit at up to 1200 baud due to their ability to avoid the accoustical coupling. Many such MODEMs are available that plug directly into S-100 and other busses, but we'll leave it at that.

Because the CAT is so common, many computer manufacturers have chosen it as their standard. Heath and Radio Shack are just two examples. This also increases the number of sources you can check, since a MODEM with Radio Shack's label will work just as well with a Heath.

Regardless of the brand chosen, all MODEMs of this type work on the same principle. The serial digital signal from your computer's I/O port is converted into an audio signal by the same technique used by most cassette I/O boards, Frequency Shift Keying. This is just a fancy way of saying that a "mark" or "1" will be translated into a 1,270 cycle tone, while a "space" or "0" will become a 1,070 cycle tone. Actually, there's a whole set of frequencies involved in the CAT's operation, due to the fact that it can be used either to originate a message or to receive a message, and these standard frequencies are shown for reference in Table 1. This isn't quite what it may seem, though, because from a practical standpoint it won't usually make much difference whether you're in one mode or the other. The important thing is that your MODEM and the one you're talking to must be in different modes. Otherwise, no communication takes place. For instance, most of the large time-sharing systems that allow you to "dial-up" the computer from a remote site are operating in the answer mode, and your "dial-up" link will be set for the originate mode. As a

matter of fact, the communication link would work as well if the modes were reversed. It's the difference that's important. The one time this mode factor becomes important is when you want to talk to a non-standard system, such as the one owned by your friend across town. If you both bought originate-only MODEMs, you're out of luck. Such MODEMs are available, and they're usually cheaper, so keep in mind what you want to do with it before you buy it. Keep in mind, too, that some MODEMs advertized as capable of originate and answer modes can actually be altered only by re-jumpering the circuit board. Look for a MODEM that's switch-selectable.

The reason for the limitation on transmission speed is primarily one of design limitations, partly for equipment but mostly for the phone lines. Voice-grade lines such as we use aren't the best possible. A better circuit can be provided but the cost is astronomical. However, if time is money (as it is for the Big Boys) you can do what they do and order a data-grade line. But at several kilobucks a month you may not really want a 9,600 baud circuit. There's an inevitable trade-off between speed and cost. Keep in mind that at 300 baud, the reliability of your data transfer will be limited primarily by the quality of the phone line.

After this brief look at the theory and operation of MODEMs in general, we're ready to consider interfacing. This is the point where you'll need to dig out your system specifications and see what you have to work with. We'll assume to begin with that you will be interfacing a computer to the MODEM. Since the MODEM is a serial device that speaks in RS232C, so that's the standard to which we must work. If you own a TRS-80, Radio Shack's systems approach to their computer makes it easy. All you need to do is buy an expansion interface and plug the MODEM into the RS232C socket. If you have a buss-oriented system like the Heath H8 or most SS-50 or S-100 machines, it will ba a matter of what sort of serial I/O ports you have available. With the H-8, the four-port H8-4 board provides four RS232C ports to choose from. As an alternative, the Heath Users' Group can provide information on how to convert an H8-5 one-port serial board or an H8-2 three-port parallel I/O board. (see issue 5 of REMark, page 21) Most other buss-oriented systems will have similar provisions. No-buss machines like the Heath H-88 and H-89 have slightly different provisions, but in general the idea is to plug the MODEM into an RS232C port. The pins used in the CAT MODEM are shown in Table 2. These are the only ones you need to connect.

Transmitter Frequencies		
Originate Mode		
	Mark	1,270 HERTZ
	Space	1,070 HERTZ
Answer Mode		
	Mark	2,225 HERTZ
	Space	2,025 HERTZ

Receiver Frequencies		
Originate Mode		
	Mark	2,225 HERTZ
	Space	2,025 HERTZ
Answer Mode		
	Mark	1,270 HERTZ
	Space	1,070 HERTZ

Table 1. These are the Bell 103A audio frequencies used by voice grade MODEMs. Note how the frequencies are assigned depending on originate/answer mode and whether the MODEM is transmitting or receiving.

MODEM PIN CONNECTIONS

1	Protective ground: Used to link the chassis grounds of the modem and the data terminal.
2	Transmitted data: Used to transfer data from the data terminal to the modem.
3	Received data: Used to transfer data from the modem to the data terminal.
4	Request to send: Indicates the data terminal is ready to transmit data.
5	Clear to send: Indicates the modem is ready to receive data.
6	Data set ready: Indicates the modem is on-line to the phone and is in the data mode.
7	Signal Ground: Provides a common reference for the signals.
8	Carrier detect: Identical to clear to send.
9	Ring indicator: Indicates the modem is receiving a ring signal from the phone. Not normally used in simple interfaces.

Decription of the pins and functions on the standard 103A compatible MODEM. Note the duplication of some pins, and the fact that all MODEM pins are not always required.

TABLE 2

At this point it may be necessary to determine whether or not your chosen port is configured for DTE or DTC. In most systems it will probably not matter, but in some like the H8 when equipped with the H8-4 board having sophisticated programmable I/O features, it will be

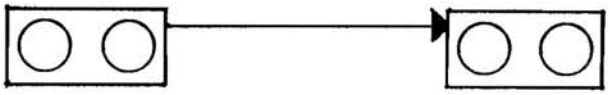
necessary to insure that the proper configuration is used. If this is a consideration, your system documentation will make that clear.

This is a good place to point out the fact that it isn't necessary to have a computer to make use of a MODEM. As noted, the MODEM translates the audio signals to RS232C-compatible digital signals. Your stand-alone terminal is capable of printing the data out, on the CRT or in hard copy (DECwriter, etc.) In fact, this is one of the most common uses of our class of MODEM. If you have access to a large time-sharing system at work or at school, you can interface a MODEM to your terminal and operate in the same manner as you could at work or at the school's computer center or terminal room. For many people, this is entirely sufficient; after all, if you have a CDC 7600 at work, a TRS-80 may lose some of its appeal. You might also note that the CAT-type MODEMs provide a convenient test mode for use with just a stand-alone terminal. This allows you to talk through the MODEM from the keyboard to the CRT or printer, in both full and half duplex modes. Any problems uncovered here can be corrected before you undergo the frustration of trying to communicate through a defective MODEM. In reality, the MODEM itself is highly reliable, but some of the inter-connections may have come adrift or you may uncover a power supply problem.

The final obstacle to your operations will be the MODEM device driver. For most systems, based as they are on a Disk Operating System (DOS) of some sort, the device driver is a program residing on the system disk that can be called up for data transfer when desired. The precise form of the MODEM device driver will depend on the DOS; there is a new version available from Lifeboat Associates called "BSTAM," intended for all CP/M-bases systems. The Heath Users' Group provides two for the H-8/H-89, one especially tailored for Heath-to-Heath communications (H8COMM) and one intended for use by a Heath owner communicating with any other type machine (MCS.) The point here is that without the device driver the MODEM can't be used with the computer. It can, of course, be used with the terminal alone. In some cases, the device driver is an extra-cost item, so be sure to check that it's available. The alternative is to write your own, a tedious course of action.

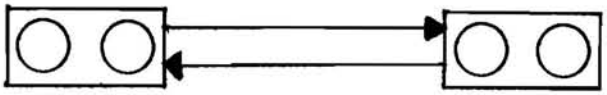
Once the interfacing and device driver questions are answered, you're ready to go on the wire. Well, almost. First is the little matter of protocol. This term means different things to different individuals; the engineer thinks of the way the signals are transmitted, received and processed, but what we mean is the manner in which you and your partner on the other end are going to conduct communications. Remember, one of the MODEMs must be set to originate, the other to answer. You will also need to decide whether to use half duplex or full duplex modes, and you may need to configure the device driver for a particular I/O port number and baud rate. In most cases, the device driver specifications will tell you what it requires, and in some instances you may have the option to reassemble the driver to your own specifications using the source code provided. The Heath Users' Group drivers fall into this category. It is, however, possible that you'll have to experiment. Half duplex means that in effect there is only one signal carrying wire and a ground connecting the data terminal and the MODEM (and hence the two MODEMs.) This means that signals travel in one direction at a time. Full duplex means that there are two paths for the signals, one each way, so signals can travel in both directions at once. Under normal circumstances the mode will be invisible to the user, and it won't make any difference in system performance. Figure 1 illustrates the connections.

Once you've decided how to communicate with the other terminal, you're set. If you're dealing with a time-sharing system, there will be certain formalities to observe. You will be in the originate mode and the other system will be in the answer mode. This isn't a bad rule of thumb to follow even if you have the option of reversing the modes. That way, you'll always be sure who's in what mode, because it will depend on who calls whom.



Data tranfer one way at a time.

HALF-DUPLEX



Data transfer both ways at once.

FULL-DUPLEX

Figure 1

This illustrates the difference between half and full duplex.

Keep in mind the fact that the data transfer reliability will be somewhat degraded by any noise on the phone lines, especially on long distance. When you first pick up the receiver, dial the first number and listen for the background noise. If there's much at all, hang up and re-dial. Few things are more frustrating than to lose the carrier signal on the last sector of a 50 sector file you're transferring. Some MODEM drivers allow partial recovery, but its a hassle any way you look at it.

One final note. Keep track of how long you're on the line and how far you're calling. At 300 baud, a 4K program will take about 2 1/4 minutes to transfer, and those long-distance toll charges can creep up on you. You might stop by the local phone office for a rate calculator, a circular card device that gives the rates for different times and distances.

Once you try data transfer over the phone, you'll wonder why you didn't start sooner. There's no easier or faster way to exchange data. Remember that in addition to being able to communicate with others who have the same type computer, you can also download programs and data from computers vastly different in type and size. Once the MODEM drivers put the data in RS232C format, it doesn't matter whether your remote site is a TRS-80, an Apple II or a Heath H-8, or the source. You can talk to them all.

(For those interested in further reading on MODEMs in particular or data communications in general, a good place to start is the Blacksburg Group's edition of NCR's "Basic Concepts" book titled "Data Communications Concepts," available as "Bugbook BRS-6" for $6.95 from Group Technology, Ltd., P.O. Box 87, Check, VA 24072.)

EOF

Editor Note: We have had many inquiries from users of MCS for a modification which would allow use with a HP 2000 or similiar host that requires Control S and Control Q. Changing the code in the C.SEND routine as shown below will accomplish the desired results. CTL.S EQU 21Q CTL.Q EQU 23Q :JB:

```
C.SEND5 EQU     *
        CPI     A.NUL
        JE      C.SEND6
        CALL    PUTM
SEND5A  EQU     *
        IN      $MPORT+5
        ANI     1
        JZ      SEND5A
        IN      $MPORT
        CPI     CTL.S
        JNE     C.SEND6
SEND5B  EQU     *
        IN      $MPORT+5
        ANI     1
        JZ      SEND5B
        IN      $MPORT
        CPI     CTL.Q
        JNE     SEND5B
  IF $MH84
        IN      $MPORT
  ELSE
        CALL    GETM
  ENDIF
C.SEND6 EQU     *
        POP     PSW
        INR     A
        ANA     A
        JNZ     C.SEND4
        JMP     C.SEND2
        SPACE   4,10
```

COMMSOFT, a software company located in Palo Alto, California, has introduced RTTY89 for the Heath H89 or H8/H17/H19 computer. By taking advantage of the disk and dynamic video graphics capabilities of either computer, this unique program adds a new dimension to amateur radio communications. Version 3.0 of the W6LLO program provides exclusive 3-level split screen to allow pretyping messages while copying incoming data. Other features include: disk based autostart (record incoming/outgoing data on disk); disk load into pretype buffer; sophisticated on-screen graphics displaying complete system status including time; automatic CW identification; and ASCII or Baudot operation. These and many other features are described in a free brochure.

Disk plus comprehensive user's manual $100 PPD. California residents add required sales tax.

COMMSOFT
665 Maybell Avenue
Palo Alto California 94306
(415) 493-2184

A printed circuit board for the CW transmitt/receive program HUG P/N 885-1027 is available from D. C. French W7IWL at 9509 Rolling Hills Drive Sun City Arizona 85351 Circuit boards for both circuits are available.

INIT and SYSGEN "how to"

Before you can use a disk for saving any programs (to be called files in this text) you must prepare the disk for use. A disk that has never been written on (you could say recorded on), needs to have a number of operations preformed on it so that it will accept data from the computer. INIT will initialize a disk, that is, prepare it to receive data, give it a name and volume number. A disk that is only INITialized can be used to save files on, but can not be used to run the system since there is no system files (system files are what makes up HDOS). SYSGEN (system generation program) will produce a disk that can run the system and store files. However you can only SYSGEN a disk that has been initialized (INIT).

To INIT a disk, Boot up on a distribution disk that has the file INIT.ABS (always use the latest version of HDOS). Type INIT⟨CR⟩ (where ⟨CR⟩ stands for RETURN EG: type INIT and hit the RETRUN key once) the program will sign on and dismount the disk you were using which means that the operating system is no longer in charge, INIT is.

It will ask you if you wish to proceed. You do, so type YES ⟨CR⟩, by now the program has told you to remove your disk and install the disk you want to INIT in the drive. Remember that INIT will destroy everything on the disk, so do not forget to remove your system disk at this point. INIT will try to read the disk to see if it has been used by HDOS before and if so what its volume number and title are. Since you could have installed the wrong disk by mistake INIT will give you a chance to abort the disk and start over. Again type a YES ⟨CR⟩ to move on.
INIT will ask you for a volume number enter one (must be between 1 and 255) and hit the return. It is a good idea not to number disks the same volume number as this makes it easy to mix them up. Another good idea is to create a system under which you number the disks EG: games 50 to 100, math programs 100 to 150, electronics programs 150 to 200, and general from 200 and up. INIT will now ask you for a volume label. A volume label is what you want to call this disk and is helpful in keeping track of your disk. A volume label may be 60 characters long, if you do not want to label it just hit a return and the label will be blank. Type your label and hit the RETRUN key.

Now INIT is doing its job and will ask you if there are any bad sectors on the disk. The reason for this is that HDOS can use a disk that has bad sections on it. By using the TEST program you could find these if any and tell INIT this and save a otherwise useless disk. You should for this time through just hit the RETURN and ignore the option until later. This should do it, the disk you have should now be initialized. The program will ask if you have more disk to INIT type NO ⟨CR⟩ to go back to the BOOTSTRAP (start up procedure). Your done with INIT.

Now for you to use this disk by itself you will have to put the HDOS operating system on it. This is done with the program called SYSGEN.ABS. SYSGEN is again on the master disk and you will need to call it up (type SYSGEN ⟨CR⟩).

The program will sign on and dismount the disk in the drive, and ask for the master distribution disk (the reason for this is that not all files that SYSGEN will need may not be on the the first disk, so SYSGEN wants to use the master as a source), install it and follow the instructions on the console terminal.
SYSGEN uses only one , so you will be moving disks in and out of the drive alot. SYSGEN will tell you which disk it wants next EG: install the source (that is the master disk you had) install the destination (that is the disk we just made with INIT). SYSGEN will go back and forth until all system files are copied.

SYSGEN will go back to the BOOT up procedure when done, you can now install your new disk here (the one we just made) and go through the BOOT up procedure, in other words we just made a ready to go disk for your computer.

Now you will need to copy on to that disk the files (programs) that you will want to use, remember all we have done up to this point is to make the disk, now we will put it to use. There are two common ways to copy from disk to disk, one way is for a system with one drive the other is for a system with at least two drives. ONECOPY is for a ONE drive system where PIP is used if you have at least two drives.

Before we start to copy files lets take a look at what is on the disk. Type CAT/S ⟨CR⟩ (why the /S , that is called the show

flag and will force HDOS to give us a complete list of files on the disk). Now you will see a program called SYSHELP.DOC, that program is taking up room on the disk and is not required to run the system so lets get it off the disk so we have more room for our programs. If you notice in the catalog to right side there is a space for a list called FLAGS, and to the side of SYSHELP.DOC there are letters. The S is for show and will cause a file with this flag not to be shown on a CAT, the W is a write protect flag and keeps you from writing to that file. The L is a LOCK flag which is the same as the W but cannot be removed. The Program that sets and removes the flags is FLAGS.ABS.

What we want to do is remove the SYSHELP.DOC file from the disk, but we need to remove the flags first so type FLAGS ‹CR› it then will ask if you want instructions, type YES ‹CR›. When you type the file name FLAGS will give you the present flag status, you then may change them. To do this type the flags you want and hit return. EG: SW ‹CR› to set a program for the S (show) and W (write protect) flags. If you do not want any flags on the file then just hit the RETURN key. Now remove the flags on SYSHELP.DOC. To get out of FLAGS you will need to type a CNTRL-D (hold the control key down and hit the D key). Now we can delete the file.

Type DELETE SYSHELP.DOC ‹CR› and it will be deleted. (Note once a file is DELETEd there is no way of getting it back.)

This seems like a lot of work for a simple task but once you are use to HDOS it will take a very short time. Now remove the files you do not want on the disk the same way as we did SYSHELP.DOC. Remember you can not remove a L flag and that the W flag will have to be removed before you can DELETE the file. Do not remove the file FLAGS.ABS unless you do not want to change anymore flags. It is a good idea to leave FLAGS.ABS and SET.ABS on the disk (you never know when you will need them). Note the file HELP. will need removing also as this help file is different from SYSHELP.DOC.

ONECOPY can be used now to copy the files you want to copy. (If you have a dual drive system go to PIP). ONECOPY is a lot like SYSGEN in that it will have you move the disks in and out of the drive. ONECOPY prompts you with a :OC: prompt. Type the file name you want copied (making sure it is on the disk you have in the drive) and ONECOPY will take over. Do this for all the files you want copied. Later on you can learn how to use wildcards, which let you transfer more than one file without typing all the filenames. If you try to copy a file from a disk to a disk with the same volume number ONECOPY will have problems knowing which disk is which, so be sure you keep the numbers different. Once you have copied all the files you want on disk you will have a working disk to use.

Having two disk drives will make it simpler to copy, by using the HDOS program called PIP.ABS. There are two common ways to use PIP, one is to type PIP ‹CR› and the other is to type COPY xxxxxxxxx, where the x,s are copy from what to what. We will work on that first.

Type PIP ‹CR› , this will call the program. Lets assume that we are going to copy FROM SY1: TO SY0:, and the program we want to copy is DEMO.BAS. Here is how it would look.

```
:P:SY0:DEMO.BAS=SY1:DEMO.BAS
```

Or we could have typed:

```
:P:*.*=SY1:DEMO.BAS
```

The reason there is no SY0: in this command is that PIP will default to the SY0: device when no device is given. The :P: in the above line is the PIP.ABS prompt and is typed by PIP, not the user. Also the *.* are wild cards which, in this case says let the name of this file be the same as the source. The wild cards can be used in many different ways. Here is another one:

```
:P:*.*=SY1:*.*
```

This simple command will copy all files on SY1: to SY0: except those with the S flag set. If you wanted to input this command from the command mode of HDOS it would look like this:

```
COPY *.*=SY1:*.*
```

As you can see there are alot of ways to perform the same function in HDOS. Be careful with wildcards they will destroy files on the destination device, that is if you copy a file to a disk and that file name is already on that disk it will write over the old without any notice to the user.

We have used INIT and SYSGEN to create a bootable disk, now use Onecopy or PIP to transfer BASIC or any other file you want onto it and you ready to run.

:JF:

EOF

BUGGIN' HUG

Dear Jim:

I recently reviewed my growing stack of carefully collected literature distributed or published by alternate sources of Heath related products, and it occurred that newer members of HUG's ranks might like an indication of what's available these days. All too well can I recall the early days, when the only source for hardware items was Heath (there was a Users Group in name only) and the only applications software was Heath's game tapes (biorhythms, hangman, and after a long pause, Space War.) I've dealt with some of the following, but all information is offered strictly as a guide, with no endorsements of any kind implied. I have also omitted reference to any HUG-produced or sponsored items.

First, in the realm of hardware:

Henry E. Fale produces an extensive line of detailed plans and directions for building useful interfaces between the H8 and the outside world. These applications include a Selectric interface, a sound generator, an 8-channel analog-to-digital-to-analog converter, phone circuits (dialers, autoanswer, ring detector, etc.), 16-channel on/off controller and a jogger computer. He also has plans for a 64K dynamic RAM board based on a conversion of an S-100 board, and TED-8 source code for an assembly language H-8 clock routine that can be called as a subroutine. Henry's address is 2918 S. 7th St., Sheboygan, WI 53081, phone 414 452-4172.

Northwest Computer Services, 8503 N.E. 30th Ave., Vancouver, WA 98665, phone 202 573-8381, advertises GRAFIX, a hardware modification to the H-9 CRT terminal that enables it to produce denser graphics, of a simple type similar to the TRS-80. This mod is described as being compatible with the 24-line modification described in Interface Age for June 1978 (as amended by August 78 issue) in one brochure, but as being incompatible in another report, so check first. This seems to be a clever approach to getting some use out of an other wise limited terminal, and might be especially useful for an application using the H-9 c/GRAFIX mod as an alternate terminal under HDOS.

Godbout Electronics hardly needs an introduction. As the first major manufacturer to second-source Heath memory, their 12K static RAM board was a very popular item, although now out of production. Replacing that are two newer designs, a 16K and a 32K static board, at quite attractive prices. The address is Bldg 725, Oakland Airport, CA 94614.

Ace Technical Services, Inc., 78-40 16TH ST., Flushing, NY 11366, 212-591-0632, offers a combination 4K RAM-3K ROM board, a 2708 EPROM burner board with software, a cable-continuity/short tester board with software, an 8-bit parallel printer interface, a 64-channel temperature monitor board with software, and modified PAM-8 ROM chips (the nature of the modification is not described in my literature.)

A 64K dynamic RAM board with on-board refresh is available from Tryonix Electronics, Box 5131, Santa Ana, CA 92704, 704 830-2092. It may be set up for 16K to 64K in 16K blocks, and uses TMS 4116's (but note that of the 64K, only 56K is addressable, but the board can be mixed with an old Heath 8K static RAM board.) Tryonix sells both kits and assembled and tested boards.

Another dynamic RAM board, this time in an assembled 32K version only, is available from D-G Electronics Development Co., Box 1124, 1827 South Armstrong, Denison, TX, 75020, 214 465-7805. This board may be addressed as four independent 8K blocks.

CCM, Inc., Box 2308, Reston, VA 22091, offers an eight-channel 8-bit analog/digital converter board that comes fully assembled and tested.

SciTronics, Inc., can provide information on interfacing the BSR remote AC line carrier switches as sold by Sears, Radio Shack and others. Their RC-80 controller includes a power supply and case. Their address is Box 5344, Bethlehem, PA 18015, 215 868-7220.

The Keyboard Studio, 1726 Mansfield, Birmingham, MI 48008, has a set of plans and a prepared printed circuit board that allows one serial port (either the H8-5 or one of the H8-4 ports) to operate a phone dialer. Called "Tele-pulse," it can be operated with an optional software package that manages the system, providing

extensive functions including re-dial, search and dial, alphabetizing, adding and deleting names and numbers. They also have a line of heavy vinyl covers for the H-8, H14, H-17 and H-19.

Digital Dynamics also has dust covers. At last check, they had covers for the H-8 and the H-9 (as well as the H-11). Their address is Box 27243, San Antonio, TX 78227, 512 231-2012.

Mullen Computer Products, Box 6214, Hayward, CA 94545, has one of the most useful of H-8 accessories, an extender board for the H-8 buss. A great aid to experimenters and troubleshooters.

Next, the software. Here I'll be listing only those vendors that advertise and send out brochures. This does not include individuals who list their products in the want-ad sections of computer publications, not because they're no good, but only because I don't know enough about them to list them. Here's the information I have:

John Eggert of Eggert Engineering, 95 Adams Drive, Stow MA 01775, has a 32K version of "Adventure" available on tape. Virtually identical with Heath's disk version, it runs a bit faster as it resides entirely in memory. To do this, some of the instructions are given on paper instead of in the program, and there's an even briefer version for 24K machines. Both require the H8-5 interface. A 16K disk version is available, too.

The Keyboard Studio offers several BASIC programs on disk. Mailing List requires the H-19 and 32K, and stores about 1000 entries. Multiple sorts are featured, and label printing can be readily configured. Datalog.BAS is a data base that maintains up to 150 single-record entries on one disk. Quizmaster allows the creation and administration of individually tailored tutorials on the H-19. A graphic subroutine package illustrates the software creation of graphics subroutines that can be incorporated in user programs. Electronic Billboard types messages "stock exchange" fashion across the CRT, and Dice Roll draws dice on the screen for games. Score Display allows the display of large-format numbers on the screen for games and similar applications. They will also provide source code for most programs, at a reduced rate.

Tru-Data, 19 Sands Point Drive, Toms River, NJ 08753, has a very good Burroughs ribbon for the H-14 printer. It prints blacker, longer. They also provide a vast amount of softare, mostly in binary format, that are in the nature of utilities. Too many to list, they run from ZAP, that duplicates disks quickly and easily using two drives, to UNLOCK, which removes ALL flags from a disk. They also have a payroll program and a series of tape-based programs, plus the inevitable mailing list program. Send for a catalog.

Clark Systems Corporation has programs that run under Microsoft BASIC, including a general ledger (versions I and II), mailing list programs (3) and two personal use programs that do things like budgeting, interest calculations and shopping lists. Their address is Box 490156, Atlanta, GA 30349, 404 964-1262. (Note - this is the same address used by Ed-Pro, Inc., that used to advertise cassette-based programs and recently was advertising the personal use programs. Possibly a reorganization or a name change.)

J.D. Hill, 6400 Gila Court, Plano, TX 75023, provides a series of programs on both tape and disk. They include a data base management system, a personal accounts payable program, a BASIC development system (including a whole series of utilities) and a computer assisted instruction system.

Lifeboat Associates, of course, provides CP/M in a Heath-compatible format, and will provide many applications packages in a similar format. The list is far too long to reproduce; check Byte or Microcomputing for the most recent.

Another CP/M-based applications package is available from Micro-Ap, 9807 Davona Drive, San Ramon, CA, 94583, 415 828-6697. Called Selector III-C2, this is a full-service information management program, to which may be added a general ledger program. They configure the software to the computer system, so it should be optimized for the H-8/H-89.

A.E. Dessler, 5126 Loch Lomond, Houston TX 77096, lists four game-type disk programs for use with the H-8, at least some of which will require the latter's front panel and would be unsuitable for the H-89. He also has an H-8 implementation of PILOT, and a set of assembly language subroutines with wide application.

Walt Bilofsky has a set of programs available, including a "c" compiler, a Z-80 assembler, an H-89 screen editor and a program called "Reach" that seems to sound like a MODEM routine. His address is 14478 Glorietta Drive, Sherman Oaks CA 91423.

Dr. Jim Gillogly, 2520 Chard Ave., Topanga CA 90290, has taken Walt's "c" compiler and written two application programs, a text formatter and a Huffman coding routine called "Pack".

A firm I haven't heard from for a long time, but used to produce some first-rate software, called Multi-Micro Media, Box 1025, Arvade, CO 80001. The only program I have any knowledge of is an 8080 instruction set educator, written in assembly language, with extensive documentation.

A very interesting tape-based monitor program is available from Chesney E. Twombly, 15 Stoner St., Kennebunk ME 04043. Published in four irregularly spaced parts in Microcomputing, it is available on tape, assembled to originate at the memory location of your choice.

Finally, there's Microcomputing's Instant Software. They currently have two rather trivial-looking game tapes, but this may be just the beginning. Look for a tape monitor program by Patrick Swayne that will reside in the H-8's RAM. And that may be just the beginning.

By no means do I claim this listing is complete, but as an illustration of what's become available within the past year or so for the Heath H-8, I think it might be of some benefit. I have not mentioned any of the local Users' Groups, as none yet publish a catalog, but this is only a matter of time.

Sincerely,

D.C. Shoemaker
2000 A Foxridge
Blacksburg, Virginia 24060

Dear HUG:

I would like to thank you for the excellent work you are doing. You are providing a major service for all Heath owners in keeping them up-to-date on what others are doing with their computers. In each issue of REMark, I find at least one item that I wish I had thought of myself.

I would like to submit a few minor, but nonetheless important, ideas that I have found useful. I own a H8/H9/H17 system and do a lot of work in BASIC. Although this language has some severe limitations (at least the B H version), I have found a few small ways to make it more useful.

1) I constantly see listings of other programs which print 12 blank lines in order to clear the screen. I have been amazed that I have not seen anyone using PRINT CHR$(12) to accomplish the same thing.

2) Another character string I use is CHR$(34) which is a double quote. I use this to print fields which contain commas imbedded in the data to files, as follows: PRINT #1,;CHR$(34);A$;CHR$(34) which would produce "HEATH" if A$="HEATH". This allows you to use INPUT as opposed to LINE INPUT when reading the file back in. This would be necessary if there were multiple fields on one input line.

3) It is possible to update files in BASIC without having to use an editor or create a temporary version of the file and then renaming it. This can be accomplished as follows. Read the file into memory, using arrays if necessary. Close the input file, then make any changes to the data that are necessary. Open the same file for output, and write the data back to the file. Here is an example.

```
DIM A$(100),B(100)
OPEN "TEST" FOR READ AS FILE #1
INPUT #1,;I
FOR J=1 TO I:INPUT #1,;A$(J),B(J):NEXT J
CLOSE #1
INPUT "WHAT NAME DO YOU WANT TO UPDATE?";N$
FOR J=1 TO I
IF N$=A$(J) THEN INPUT "NEW DATA? ";B(J)
NEXT J
OPEN "TEST" FOR WRITE AS FILE #1
PRINT #1,;I:
FOR J=1 TO I:PRINT #1,;A$(J);",";B(J)
NEXT J:CLOSE #1:END
```

4) I have a number of large files in which I wanted to save as much space as possible. This became annoying whenever I wanted to print a variable to a file because BASIC insisted on printing a blank space before and after the number. That made two unnecessary characters for each variable I printed. Now I use a subroutine to bypass those blanks when printing.

```
---- text ----
Z=50:GOSUB 10000
---- text ----
10000 Z$=STR$(Z)
10010 PRINT #1,;MID$(Z$,2,LEN(Z$)-2)
10020 RETURN
```

5) Regarding HDOS, I have found that I can run both my drives at a seek time of eight, even though one only passed at 14 using TEST17. You might want to try this for awhile if you have this problem. If when you check STATUS you have gotten no errors then there is no reason why you can't continue to run at the faster speed. I suspect that TEST17 may be a bit more strict than it needs to be.

6) Lastly, I have a question regarding the terminal driver which is part of

HDOS. I like to run my terminal at 9600 baud because it is much faster and I get more done. The only problem with this is listing files. You have to constantly skip between CNTRL S and CNTRL Q to hold the screen. I think it should be possible to alter the terminal driver to pause after every 12 lines and wait for a carriage return or CNTRL Q before continuing. If anyone knows how this may be done, or has any suggestions, I would appreciate the information.

Sincerely,

Brian Polk
86-02 Little Neck Parkway
Floral Park New York 11001

PARALLEL I/O FOR THE H89

the reserved area above address 340.000

BY: Brent S. Simmons
501 Forest Ave.
Palo Alto Ca. 94301

Parallel I/O is available for the H89 via a small interface card that inserts into any of the three unused connections on the upper left side of the CPU board.

The interface card uses the +5 volts from the H89 and provides decoding and buffering of the address and data read/write lines. The area from address location 340.000 to 340.377 is decoded and may be used for parallel I/O, however, in practical terms, only a few ports would be necessary for most needs and are easy to decode using SN74LS138 decoders. The remaining area could be decoded and used for EPROM utilization in special applications.

Having parallel I/O opens up many new uses for the H89 in areas such as high speed data transfer. Data may now be written to a peripheral via a STA or read by a LDA. Instructions including MOV M, MVI M, LHLD, SHLD, ADD M, ANA M and others can be used directly with a peripheral device.

The I/O card uses 7 chips and is presently hard wired, but if there is enough interest a printed circuit will be designed. A schematic diagram can be obtained from the author.

Due to the manner in which the memory map ROM is coded in the H89, the system must be configured for 32K.

Dear HUG,

Menu Operating System

Here's an operating system for the non-programmer/user. In my case, my wife. Two goals are achieved in this process.

1) Ease of operation.
2) Disk independence.

Ease of Operation

The SY0: disk contains all of the system modules with an exception of BASIC.ABS. Two files are added. 1) MENU.ABS and 2) MENU.BAS. This reduces the execution commands to: BOOT, DD-MMM-YY, MENU, and RUN. Program executions and BYE. The "program executions" are conversational and prompt the user for the necessary input.

Disk Independence

SY1: contains a file called "files.dat" which contains a list of all the executable BASIC programs under "MENU". These programs are also stored on SY1:. In this manner, SY1: disks may be catagorized as to their content, e.i. games, business, experimental, etc.

MENU.ABS

MENU.ABS is a concatination of BASIC, the HDOS file handler (cntrl 4,1), and MENU.BAS saved using the FREEZE "MENU.ABS" command and can be executed directly from HDOS.

MENU.BAS

SEE LISTING ON PAGE 31

Note that menu read SY1:FILES.DAT to get its Index for display. The program number and a 'return' is entered to chain to the desired program. The END(0) executes an UNFREEZE "SYSCMD.SYS" to return the operator to HDOS and all that remains is "BYE". Each program under MENU ends by chaining back to MENU.BAS.

Options

If you flag all the .BAS programs with "S" or "SW" then the CAT SY1: command will display only the data files. We operate primarily under SCELBIE'S PIMS and have many PIMS data files.

Robert Pearce
504 McCoys Fork Rd
Walton Kentucky 41094

Dear Hug:

Regarding Issue 5 of REMark, I believe that there is an error in the program CONFIG.SAV, page 29. This program is by H.G.Miller, of the Heath Company. I entered this program on an H11 and found that the output would not display beyond the test for the EIS-FIS chip. My H11 does have the chip.

Tracing this down, I found the error in the EISTST routine

```
eistst:   moy       sp,R1
          cmp       -(R1),-(R1)
          fadd      R1
          rts       pc
```

The problem is in the second statement: cmp -(R1),-(R1). The first instruction causes R1 to point at the stack. The second instruction moves R1 two words down the stack; that is two words below the return address for the routine. However, the EIS-FIS chip requires four words for its operation. Thus, when the EIS-FIS chip is present, it performs an add operation and changes the return address for the rts pc.

If the second statement is changed to: sub#10,R1, then the routine works properly.

```
eistst:   mov       sp,R1
          sub       #10,R1
          fadd      R1
          rts       pc
```

The program now displays all system components, including the H27 mode and RX01 mode on the H11.

Yours very truly,

Warren Weaver
1789 Cherrylawn Dr.
Toledo Ohio 43614

Dear Jim:

Here is hard copy confirmation of corrections and additions to H11 "Dick" Focal Routine in REMark, issue #9. These are the result of telephone consultation with Dick Huntsinger.

Change line #6 of Absolute Loader Routine to;

```
.D 37620= 16703,152,105213,105713,100376,116303,0,60300
```

After loading either the Non-EIS or EIS version of 8K Focal;

```
Break
@25110/nnnnn    12700     CR  or    LF
 25112/nnnnn    0         "         "
 25114/nnnnn    12701     "         "
 25116/nnnnn    25140     "         "
 25120/nnnnn    12702     "         "
 25122/nnnnn    25210     "         "
 25124/nnnnn    12022     "         "
 25126/nnnnn    20001     "         "
 25130/nnnnn    1401      "         "
 25132/nnnnn    774       "         "
 25134/nnnnn    0         "
 25110G         Wait
 R4/ xxxxx      165264    CR
 165242G        Wait for KMON

.D 25110= 12700,0,12701,25050,12702,25210,12220,20001
.D 25130= 1401,774,137,4254
.D 40= 25110

SAV SY:FOCAL.SAV (ORFIS) 0-52300
```

The difficulty with the previous routine (aside from the deletion of the input at locations 24420-24424) was that the relocation of the language program was starting in the top part of the language. If trouble with the revision occurs, the user should check locations beginning with 24700 to find where Focal stops(177777 or 000000) and begin his routine above this point. Some of the addresses in the relocation routine will have to be changed to keep the relative addresses (in octal) the same.

Yours truly,

John B. Heffelfinger
9233 Ward Parkway, Suite 285
Kansas City Missdouri 64114

Dear Mr. Blake:

Here are some thoughts about large file manipulation on the HT11 system for inclusion/consideration in REMark.

Assume (as I have had to do) that you have updates of sequential files to be made using a BASIC program on your H11 and that these are LARGE files. HT11 assigns as a default one-half of the largest available file space currently left on the disk. If you had two files of about 100 blocks each to add into a space of 300 blocks available, the first would be assigned a default space of 300/2 or 150 blocks and the second (300-150)/2 or 75 blocks and the proverbial FTS (File-Too-Short) message is seen again.

In the best of all possible worlds (are you listening software writers?) the CURRENT size of a file could be read and the DESIRED size of the file could be specified indirectly. In HT11, however, the current size is invisible to BASIC and only the file name is able to be set indirectly.

I enclose for your consideration a 2-step method that would permit the reading and specification of file size using the OVERLAY feature and some other tricks.

Steps for execution:

1. .R PIP cr
   *DIRECT.DAT=DK:/L cr
   C

2. .R XBASIC cr
   cr

   READY

   CHAIN .filenm.

including the following code:

SEE LISTING ON PAGE 31

MEETINGS and CLUB NOTICES

On our return from the most recent visit to China, Mr. Nurse and I interrupted a 23 hour journey home to stop in Hawaii for R and R and an interesting visit with Gerry Cramm et al of the local users' group there. We thank you for your hospitality and many kind comments. Perhaps again soon?

And Jon Hodge of Buffalo area users' group, thank you for inviting me back again. Anyone in that area that is interested in learning, should contact the Heathkit Electronic Center there and join this aggressive organization.

This month I look forward to visiting the Louisville Ky. area, Indianpolis and a new group, AnaHUG (Anaheim Ca.) while at the NCC. They (AnaHUG) meet the third Thursday of each month.

And the PACIFIC Northwest HUG is still meeting the first Monday of each month (7:00PM) at the Heathkit store. For more info, contact Jan Johnson in Seattle at 206-464-5666.

In the Virginia Beach area contact THUG c/o Steven Kopp at 5549 Princess Anne Rd. They are looking forward to exchanging ideas, software and newsletters with anyone. They meet the 2nd and 4th Tuesday of each month at the HEC.

In the Detroit area, there is a group headed by Chuck Dattalo (928-7423). They publish the "DUMP" newsletter and meet monthly.

CHUG Capital Heath Users' Group in Fairfax Virginia can be contacted through PO Box 341, Zip 22015. They are disassembling HDOS and have an active modem group.

The I/O PORT is the name of the newsletter from the Woodlawn Ohio area. Contact the HEC or Pres. Ray Pelzel at 471-1711.

And if you need a grin, subscribe to :PIP: "a Data Interchange for the Computer User and Hobbists.". This newsletter not only has some very useful information, but the writers must be on something. Send $12 to :PIP: at P.O. Box 1126 Neptune N.J. 07753.

TIME SHARING SYSTEMS

If you are on MicroNET, our ID is 70003,207 and you can get a complete list of other Heath users by... COPY HUG.DAT [70003,207] TO YOU.DAT. From time to time we will leave messages under NOTICES using the keyword "HUG". The SOURCE is reported to be in some financial trouble and the system is, most of the time, painfully slow so our communications there should be limited to the MAIL. Our ID is TCD137.

Keep It Simple, Stupid
KISS

OK class, in our last KISS column we wrote a Mailing list program, but we do not have any way of correcting a record or making changes. Starting with line 1200 and thru line 1410 we have added a section that will allow us to EDIT our mailing list. We also had to add the new command to line 200 to ask if we want to EDIT. The remarks have been removed to save on space, (refer to issue 9 of REMark for them). Line 1230 and 1240 set up our FOR NEXT loops, where line 1250 will print the record. Line 1260 is again part of our FOR NEXT loop for J. Notice how we have a FOR NEXT loop inside another, this is called NESTING. Now we come to line 1270, here is where we find out if we want to make any changes. If we answer YES or just Y (notice the use of the OR statment in the IF THEN statment) the program will jump to line 1300. If we answer with any other string the program will flow into line 1290, which, once complete with its FOR NEXT loop will GOTO line 200. Now back to what happens when we get to line 1300. Line 1300 could be said to be reseting our FOR NEXT loop for J so we can rerun the last record starting with the NAME. We advance J one notch at a time, each time line 1320 stuffing the NEW DATA in, in place of the old. Again line 1330 and 1340 advance our loops and line. Line 1400 moves our I pointer back one and then goes to our next record. Line 1410 takes us back to our menu (line 200). Now we could shorten this program some but that would make it hard to see what is going on. In our last KISS column we said we would tell you how to convert this to disk BASIC, keep listening, we will. By the way, how about a sort. Do not give the list of names to your wife and telling her to sort them. We found this causes a hardware error--(you and your hardware end up in the middle of the street).

```
110 DIM A$(100,6),T$(6)
120 FOR I=1 TO 6:READ T$(I):NEXT I
130 DATA "NAME","ADDRESS","CITY STATE ZIP","PHONE","BUS PHONE","NOTES"
200 PRINT :LINE INPUT "LIST ALL NAMES, ADD NAMES, EDIT A RECORD, OR STOP ";A$
210 IF A$="LIST" THEN 1000
220 IF A$="ADD" THEN 2000
230 IF A$="STOP" THEN 3000
240 IF A$="EDIT" THEN 1200
300 PRINT :PRINT "TRY AGIN ":GOTO 200
1000 REM
1020 FOR I=1 TO N
1030 FOR J=1 TO 6
1040 PRINT T$(J);":";TAB(14);A$(I,J)
1050 NEXT J:NEXT I
1060 GOTO 200
1200 REM  **** NEW AREA ****
1230 FOR I=1 TO N
1240 FOR J=1 TO  6
1250 PRINT T$(J);":";TAB(14);A$(I,J)
1260 NEXT J
1270 LINE INPUT "DO YOU WANT TO EDIT THIS RECORD (NO) ";Z$
1280 IF  Z$="YES" OR  Z$="Y" THEN GOTO 1300
1290 NEXT I:GOTO 200
1300 FOR J=1 TO  6
1310 PRINT T$(J);":";TAB(14);A$(I,J);
1320 LINE INPUT "NEW DATA ";A$(I,J)
1330 NEXT J
1340 NEXT I
1400 I=I-1:NEXT I
1410 GOTO 200
2000 N=N+1:PRINT :REM  SET UP N
2010 PRINT T$(1);:LINE INPUT ":  ";A$(N,1):REM PRINT NAME
2030 IF LEN(A$(N,1))=0 THEN N=N-1:GOTO 200:REM IF DONE
2040 FOR I=2 TO 6:REM NOW PRINT ADD AND ST,STATE
2060 PRINT T$(I);:LINE INPUT ":  ";A$(N,I):REM FILE IN RAM
2070 NEXT I:REM KEEP GOING
2080 GOTO 2000:REM NEXT NAME
3000 REM  QUIT
3010 PRINT "DO A FDUMP"
3020 STOP
*
```

What about things like PUT, GET, FLOAD and FDUMP? Well, there are two ways we can save our mailing list program and the data. One is to save the program and data in one file on the tape. The command to do this is FDUMP. To retrive this as one item we would use FLOAD. Now if we wanted to store the data by itself and retrive it this way (you may have different mail lists) we would use PUT (to save it) and GET (to retrive it).

Consultant's CORNER

CONSULTANT'S CORNER

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

Q: How can I use the Special Function keys on my H-19 (H-89)?

A: Just write a program that looks for them specifically. For example, this Microsoft BASIC program will print out the name of the color keys when they are typed.

```
10 A$=INPUT$(1)
20 IF A$<>CHR$(27) THEN 10
30 A$=INPUT$(1)
40 IF A$="P" THEN PRINT "BLUE KEY"
50 IF A$="Q" THEN PRINT "RED KEY"
60 IF A$="R" THEN PRINT "GREY KEY"
70 GOTO 10
```

Q: Is it normal for a lightning bolt to flash across the screen of my H-19 (H-89) when I turn the power off?

A: Yes, if thunder does not accompany it! When power is removed, the sweep circuits decay as the protective discharge circuit increases CRT current. The result is a bright flash of light. See page 35 of your operation manual for more information.

Q: My H-14 printer prints about 6 lines and stops, sometimes with the ribbon motor running. Pushing Off-line and then On-line allows it to start again, but it stops after 6 more lines. What could cause this?

A: The H-14 must sense a paper hole / no-hole change every 6 lines. If no change is seen, the paper is assumed to be stuck or exhausted. The components involved are LED3, Q2, U109, and U106. A piece of paper scrap may cover the LED or transistor, blocking the light path; or the paper may not be positioned to allow the holes to pass through the light path. Such mechanical problems are much more common than actual part failures.

Q: My H-14 top cover seems a little loose. What is supposed to hold it down on my chassis?

A: Gravity. For those who are using the printer in a non-gravity situation, the following modification may be employed. Obtain two standard screw-on fuse clips (Heath p/n 260-92). Remove the positioning posts from the chassis, insert the threaded end through the hole in the clip from the inside, and remount the post-clip combination using the existing flat washer. Slight re-forming of the clip may be necessary to insure a tight grip on the cover standoff.

EOF

Magnolia Microsystems CP/M
By: Barry Watzman

Magnolia Microsystems (MMS) of Seattle Washington, a consulting and contract programming firm which also operates a retail computer store, has recently announced the availability of Standard (e.g. Org 0 as opposed to Org 4200H) CP/M 1.4 for the (W) H-89. For $249.95, you get a small printed circuit card which plugs into several IC sockets on the H-89 processor board, and a 5" diskette containing 1.4 CP/M with MMS's BIOS.

MMS has supplied one of their systems to Heath for evaluation, although they have not given us source code for their BIOS.

With regard to the hardware used to achieve the re-mapping of the CPU's address space, we find their product to be very satisfactory. On bootup the system runs in a normal mode, and it will execute both HDOS and Lifeboat Org 4200H CP/M "exactly the same manner as an unmodified system."

Under software control; however, memory is remapped to place all 48K of read-write RAM contiguously at address 0, thus permitting the use of Org 0 CP/M.

continued on next page

Just in... Another new User Group. Contact
SPOHUG c/o Charles Ballinger (Chuck)
RFD 1 Box 676
Spokane Wa. 99204

Early implementations of the MMS software represented a somewhat "minimum" system in that the IOBYTE was not implemented and NO SOURCE code was given for the BIOS (either the disk or the non-disk portion). However, their latest release (1.44 by their designation) does implement the IOBYTE and segment the BIOS into a "system" portion containing the disk drivers, as well as a "user" portion containing all non-disk device drivers. Source code for the user portion of the BIOS is included, as is the case with the Lifeboat version of CP/M. Regretfully, neither MMS or Lifeboat is willing to give out full source code for the BIOS (including disk drivers) at this time.

MMS's disk format is fully compatable with Lifeboat's, and diskettes made for either system can be read and written by the other.

EOF

32K Memory for the H 8

By: Brian Grams and Bruce Denton
DG Electronic Developments Co.

D-G Electronic Developments Company has developed the DG-32D dynamic RAM memory board for the HEATH* H8 computer.

Several features make the DG-32D an excellent memory source for the H8. Among these features are low power consumption, simple memory addressing by use of two eight position DIP switches, memory arrangement as four independently addressable 8K byte blocks, and compatibility with all current H8 hardware and software.

On the DG-32D it was decided to use a transparent refresh scheme in order to avoid slowing down of the CPU. Design characteristics were also incorporated to allow the DG-32D to run using external refresh from a Z80 CPU.

The DG-32D comes fully assembled and tested. Included with each board is an eight page operations manual which will prove helpful to the user. A one year guarantee is provided by D-G Electronic Developments Company covering parts and workmanship.

The DG-32D is available from D-G Electronic Developments Company at a cost of $479.00, freight pre-paid.

D-G Electronic Developments Company
P.O. Box 1124
1827 S. Armstrong
Denison, Texas 75020
Phone: (214) 465-7805

SOFTWARE

PAGED.ABS H19/H89 PAGE EDITOR ABS ONLY

A page editor for the H19/H8/H17 or H89. PAGED icludes commands like OUTPUT for outputing to a printer, JOIN and UNJOIN, also PAGED and append text. Order part number 885-1079 at 25.00 each.

ZSM.ABS Z80 ASSEMBLER ABS ONLY

A Z80 assembler for HDOS. Operates the same as ASM.ABS but uses and accepts Z80 code. ZSM will output your LST listing in either HEX or octal format. For use on H89 only. Order part number 885-1078 at 25.00 each.

HUG P/N 885-1065	FIXED POINT PACKAGE
H8/H89/ASM/ABS/16K	$18.00

A sample program produces a loan payment schedule utilizing the fix point package. The output has payment date, interest, principal, and balance for each period. It also prints out the year end and a final total.
FXDPT.ACM is the common deck fixed point routine used for the LOAN program.
The disk also has 60 sectors of Doc.

HUG P/N 885-1066	DISK X MISCELLANEOUS
H8/H89/ASM/ABS16K	$18.00

CRUNCH converts a program into the Huffman coding system to save space. The Huffman coding system can reduce the size of a file by 30%. The routine is intended for backing up files.

EXPAND is used to convert the above files back to the orginal form.

COMPARE is used to compare to HDOS files to each other and report the findings.

MEMORY is a routine for testing the memory in a H8.
The disk includes 56 sectors of doc.

HUG P/N 885-1068	MicroSoft BASIC Games
H8/H19//H89/48K Disk Software	$18.00

A game package for use with the H8 and H19 or the H89. All games use the graphic features built into terminal. Some of the games are Hangman, Othello, Tic-Tac-Toe, Robot Nim, Dice Game, Slot Machine, Flying an Airplane and others.

```
10 OPEN 'DIRECT' FOR INPUT AS FILE #1
20 REM - VERIFY DATE AS TODAY
30 INPUT #1:Q$\IF Q$=DAT$ THEN 50
40 PRINT 'DIRECTORY NOT CURRENT--RESET <IN PIP> AND RETURN'\STOP
50 DIM F$(13),FO$(13),F(13)
60 READ N
70 FOR I=0 TO N-1
80 READ F$(I)\FO$(I)=F$(I)\F(I)=0
90 REM - ADD SPACES FOR NAMES UNDER SIX CHARACTERS BEFORE EXTENSION
100 P=POS(F$(I),'.',1)
110 IF P=7 THEN 130
120 F$(I)=SEG$(F$(I),1,P-1)$' '$SEG$(F$(I),P,LEN(F$(I)))\GO TO 100
130 NEXT I
140 REM - FIND LISTING OF EACH FILE
150 INPUT #1:Q$
160 IF LEN(Q$)<25 THEN 230
170 FOR I=0 TO N-1
180 P=POS(Q$,F$(I),1)
190 IF P=0 THEN 210
200 F(I)=VAL(SEG$(Q$,P+10,P+14))
210 NEXT I
220 GO TO 150
230 CLOSE #1
240 REM - DATA # OF FILES TO BE FOUND FOR SIZING
250 DATA n
260 REM - NAMES OF FILES FOLLOW; FILES FOR SEARCH OF CURRENT DIRECTORY
270 DATA 'filenm.ext','filenm.ext','filenm.ext'...for n items
.                                          Note that the extension
.                                          is required
.
.
.
other program lines
.
.
.
930 REM - OPEN Q$ FOR OUTPUT, ALLOCATE 10% MORE BLOCKS, MINIMUM +5
940 FOR I=0 TO N-1
950 IF FO$(I)=Q$ THEN 970
960 NEXT I\PRINT 'FILE NOT FOUND...ERROR'\STOP
970 Q1=INT(1.1*F(I))
980 IF Q1-F(I)>=5 THEN 990 \Q1=F(I)+5
990 GOSUB 1000 \RETURN
1000 REM - OPEN Q$ FOR OUTPUT(Q1) AS FILE #Q2
1010 OPEN 'OVERLY' FOR OUTPUT(1) AS FILE #7
1020 PRINT #7:'1050 OPEN Q$ FOR OUTPUT('$STR$(Q1)$') AS FILE #'$STR$(Q2)
1030 CLOSE #7
1040 OVERLAY 'OVERLY.DAT'
1050 REM - OVERLAY SITS HERE
1060 RETURN
```

Using these techinques indirect sizing of output files is possible and pseudo-indirect reading of file size is possible as well.

Enjoy.

Yours,

Doug McNeill

Douglas H. McNeill, M.D.

```
00010 REM MENU.BAS       ***SEE TEXT ON PAGE 25***
00020 PRINT CHR$(12);TAB(30);"***MOS***"
00030 PRINT TAB(24);"MENU OPERATING SYSTEM"
00040 PRINT "     WRITTEN BY R. PEARCE FOR VICKI ON H-8/17"
00050 CLEAR:DIM P$(100):OPEN "SY1:FILES.DAT" FOR READ AS FILE #1
00060 FOR I=1 TO 100:INPUT #1,;P$(I):IF P$(I)<>"EOF" THE THEN NEXT I
00070 CLOSE #1:P$(I)="":P=I-1:FOR I=1 TO P STEP 3
00080 IF P$(I)<>"" THEN PRINT TAB(10);I;P$(I);
00090 IF P$(I+1)<>"" THEN PRINT TAB(30);I+1;P$(I+1);
00100 IF P$(I+2)<>"" THEN PRINT TAB(50);I+2;P$(I+2)
00110 NEXT I:PRINT
00120 LINE INPUT "ENTER PROGRAM NUMBER (0 TO END) ";X$
00130 IF ASC(X$)<48 OR ASC(X$)>57 GOTO 120
00140 IF VAL(X$)>P THEN PRINT "NUMBER TO HIGH":GOTO 120
00150 IF VAL(X$)=0 THEN UNFREEZE "SYSCMD.SYS"
00160 IF P$(VAL(X$))="BASIC" THEN END
00170 P$="MENU":P1$=P$(VAL(X$)):CLEAR P$(:CHAIN "SY1:"+P1$+".BAS"
```

ATTN: MicroNET USERS

MicroNET subscribers may get weekly news from HUG by using the MicroNET BULLETin board. After signing on to BULLET, type SCAN NOTICE HUG. We anticipate some pretty exciting news to show up under this category in the next few weeks. Our Account number is 70003,207. Also, H-11 users can gain access to a special H-11 Bulletin Board by (at the command level) typing IRUN MNET11.CMD[70110,403]. This service is provided by Chuck Sadoian.

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 _____

REMEBER — ENCLOSE CHECK OR MONEY ORDER

CHECK THE APPROPIATE BOX AND RETURN TO HUG

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

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

HUG P/N 885-1069 DISK XIII MISCELLANEOUS H8/H89/ASM/ABS/16K $18.00

INFO is a routine that will report the status of the TT: device driver and the size of RAM in the system.

REM is a renumber program that will allow the user to set the starting number and the increment number. For BH BASIC.

XREF is a cross-reference generator routine for the HDOS assembler. The XREF uses a LST file as input and creates a new file with the cross-ref. at the end.

READ is a small routine to read a file from the disk to the TT: in octal.

HUG P/N 885-1077 TXTCON/BASCON H8/H89/ASM/ABS/16K Disk Software $18.00

Due to the lack of space on the Heath distribution disk these programs can no longer be offered on that disk. This disk is offered to the new computer owner who may need these programs to convert their cassette programs to disk. TXTCON converts text (editor) files and BASCON converts BASIC files to the disk operating system. Included is the complete source code including an assembly language programmers delight of ACMs.

HUG Product List as of 01-Jun-80

P/N	Description	Price
885-1008	Volume I Documentation	$ 9.00
885-1009	Tape I Cassette H8 Only	$ 7.00
885-1010	Adventure Disk H8/H89	$ 10.00
885-1012	Tape II BASIC Cassette	$ 9.00
885-1013	Volume II Documentation	$ 12.00
885-1014	Tape II ASM Cassette H8 Only	$ 9.00
885-1015	Volume III Documentation	$ 12.00
885-1019	Device Driver Disk H8/H89	$ 10.00
885-1022	HUG Editor (ED) Disk H8/H89	$ 15.00
885-1023	RTTY Disk H8 Only	$ 22.00
885-1024	Disk I H8/H89	$ 18.00
885-1025	Runoff Disk H8/H89	$ 35.00
885-1026	Tape III Cassette H8/H88	$ 9.00
885-1027	MORSE8 Casstte H8 Only	$ 14.00
885-1028	RTTY Cassette H8 Only	$ 11.00
885-1029	Disk II Games 1 H8/H89	$ 18.00
885-1030	Disk III Games 2 H8/H89	$ 18.00
885-1031	Disk IV MUSIC H8 Only	$ 23.00
885-1032	Disk V H8/H89	$ 18.00
885-1033	HT-11 Disk I	$ 19.00
885-1034	Character Ed Cassette H8 Only	$ 11.00
885-1035	ED/ASM/DEBUG Cassette H8 Only	$ 11.00
885-1036	Tape IV Cassette H8/H88	$ 9.00
885-1037	Volume IV Documentation	$ 12.00
885-1038	Wise on Disk H8/H89	$ 18.00
885-1039	WISE on Cassette H8 Only	$ 9.00
885-1040	PILOT On Cassette H8 Only	$ 11.00
885-1041	Sm Bus Pkg I (2 Disks) H8/H89	$ 50.00
885-1042	PILOT on Disk H8/H89	$ 19.00
885-1043	MODEM Heath to Heath H8/H89	$ 21.00
885-1044	Disk VI H8/H89	$ 18.00
885-1045	FOCAL Cassette H8 Only	$ 11.00
885-1047	Stocks H8/H89 Disk	$ 18.00
885-1048	Personal Account H8/H89 Disk	$ 18.00
885-1049	Income Tax Records H8/H89 Disk	$ 18.00
885-1050	M.C.S. Modem for H8/H89	$ 18.00
885-1051	Payroll H8/H89 Disk	$ 50.00
885-1052	Morse8 Disk H8 Only	$ 18.00
885-1054	SmBusPkg II 3 Disk H8/H19/H89	$ 60.00
885-1055	MBASIC Inventory Disk H8/H89	$ 30.00
885-1056	MBASIC Mail List H8/H89 Disk	$ 30.00
885-1060	Disk VII H8/H89	$ 18.00
885-1061	TMI Load H8 ONLY Disk	$ 18.00
885-1062	Disk VIII H8/H89 (2 Disks)	$ 25.00
885-1063	Floating Point Disk H8/H89	$ 18.00
885-1064	Disk IX H8/H89 Disk	$ 18.00
885-1065	Fix Point Package H8/H89 Disk	$ 18.00
885-1066	Disk X H8/H89	$ 18.00
885-1068	Disk XII MBASIC Games	$ 18.00
885-1069	Disk XIII Misc H8/H89	$ 18.00
885-1077	TXTCON/BASCON H8/H89 Disk	$ 18.00
885-1201	CP/M Volumes H1 & H2	$ 21.00
885-1202	CP/M Volumes 4 & 21 C	$ 21.00
885-1203	CP/M Volumes 21 A & B	$ 21.00
885-1204	CP/M Volumes 26/27 A & B	$ 21.00
885-1205	CP/M Volumes 26/27 C & D	$ 21.00

Heath Users' Group
Hilltop Road
St. Joseph MI 49085

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

POSTMASTER: If undeliverable, plese do not return.