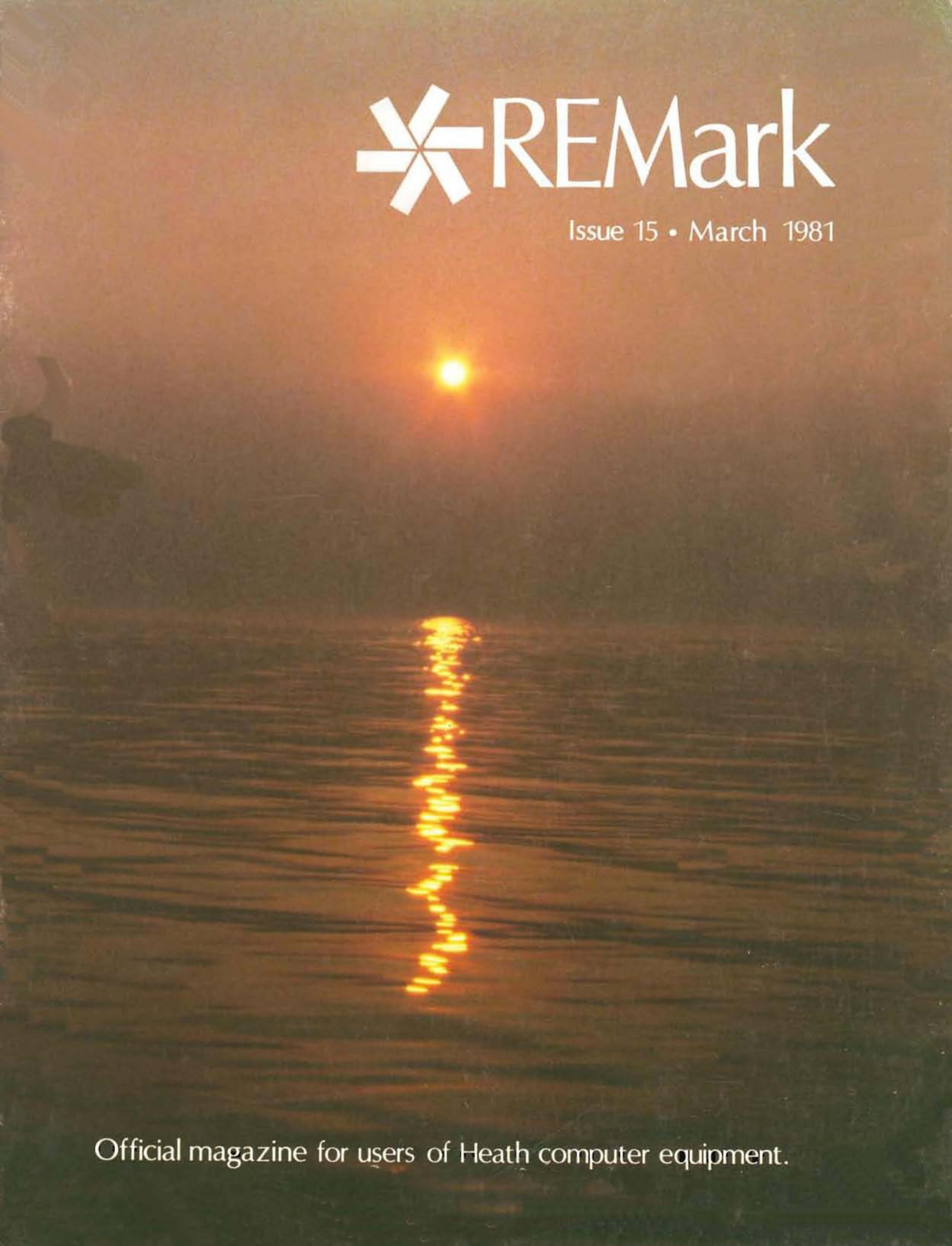
REMark

Issue 15 • March 1981



Official magazine for users of Heath computer equipment.

on the cover . . . .

Sunset on Lake Michigan

Photo by Gerry Kabelman.

on the stack

>CAT

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

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

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

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

Send payment to:

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

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

REMark

RENEWAL.HUG

ERROR -- FILE NOT FOUND

OUR OFFICE WIZARD reports a serious problem along with a big OOPS! It seems that the main computer used to co-ordinate new memberships and appropriate renewal information, has a major malfunction. Somebody forgot to tell the computer about you "oldtimers" that want to continue being HUG members! NOW WHAT? On examination of the situation (program), we found that the best, quickest, and most efficient method of correction would be an explanation to all readers so that you will be able to determine your expiration date and take us to "task" to ensure continued delivery of your "goodies". We have called on Nancy (THE WIZARD) to provide all of us with this information.

BE:

TAKE IT AWAY NANCY!

During the original program description for our "BIG" machine, we included a method by which "membership renewal" would be "looked" at by the computer approximately one month prior to a members expiration date. Well...as you can guess, somebody either didn't clean their ears or one of us was chewing bubble-gum. Because, the computer now generates renewal info about one month AFTER you have expired! We have received many complaints, and I am still trying to get rid of the gray hair this situation has caused.

AN EXPLANATION PLEASE....

Your HUG ID contains a twelve digit number. The first six digits are your personal numbers that identifies you as an individual. The next six digits are....AND I STRESS....your STARTING date as a member! (Some of you are probably saying to yourself....NO, that's not correct! Your ID does not agree with the date when you became a HUG member! You are correct. If you required renewal, your renewal was handled manually....for obvious reasons. The "BIG" machine didn't tell us about you. So, GET NEW!) Anyway, how can you decide when it's time to renew? Let's look at an example ....

Mr. X examines his ID to find the required information.

His ID is ----- xxxxxx800625.

Remember you use the last six digits or 800625. Mr. X joined HUG on JUNE 25, 1980. The first two digits (or 80) indicate the year. The second two digits (or 06) indicate the month. The last two digits (or 25) indicate the day. Given this data, Mr. X knows that his EXPIRATION will occur on JUNE 25, 1981, or one year from his "start" date.

To renew a membership to the Heath Users' Group, simply use the membership form found on the inside back cover of REMark (if you don't want to CUT ON DOTTED LINE, a copy will do just fine).

I hope that I have described this "OOPS" well enough. If you have any questions please feel free to give me a call at HUG. And, above all, please forgive me as, I think someone once said, "TO ERR IS HUMAN, TO REALLY SCREW UP REQUIRES A BIG MACHINE" or something like that!

P.S. We hear the BIG GUY is gonna get fixed! THREE CHEERS !!!!

NS:

A NOTE OF THANKS

Here's a quicky note of thanks to all of you that responded to our cry for information and articles to be printed in future issues of REMark. We have received some "HEAVY" stuff and we are looking forward to publishing this material.

If you have not sent a little "something" for REMark, please reconsider! It is your tips and suggestions, in particular, the small articles and programs that most of our readers find invaluable.

THANKS again - BE:

ATTENTION JB: You are now an OFFICIAL HUG member!

A KISS for Assembly Programming

Again, HUG has received many requests to review Assembly Language Programming techniques that will enable the beginner to "get started" with this fascinating area of computer software development. To make the following information most useful, we have selected a very small program which can be entered on both the H8 and H89 with little complication. And, to prevent total confusion, (mostly my own) the article will be presented in three parts for coming issues of REMark.

PROGRAM DESCRIPTION

The program selected will make use of the "horn" (speaker) of the H8 or the "bell" (speaker) of the H89. The program itself actually contains three major parts each of which is a smaller program capable of independent operation (more on this later). During this mini-course, you will be using the HDOS Assembler and Editor to construct the .ASM file (source), the .LST file (listing), and the .ABS file (runnable). As will be explained, you will be able to modify this program to fully understand the project on completion. The completed program (before modification) will do nothing more than "ring" the bell (or horn) five times before returning to the HDOS prompt (>) that you are already familiar with. However, several important factors will be covered including speed, similiarity to other languages, and familiarity with some of the "working parts" of the CPU itself.

STEP #1 ... DEFINE YOUR WORK

Every programmer knows the importance of a logical approach to solving a software problem. FIRST you must decide what it is you want the computer to do! We have already decided that we are going to use the "bell" in our computer. Further, we have decided for some wierd reason that the "bell" should "sound" five times. Well...if you were to examine the "listing" for the H8 and the H89 you would find two similiar items. Both have "monitors" (H8=PAM-8 and H89=MTR88) and both "monitors" have a routine known as "HORNO". "HORNO" makes the "bell" go one time! (NOTE: the monitor is the guy that "tells" the computer to do certain predetermined functions (ie turn on front panel-H8, show H: on screen of the H89.)) That's pretty handy! Someone thought to put a program just for us in a major part which "lives" in the computer! Ok, now we have a way to make the "bell ring" (one of the three major parts). Now we need some way for our computer to count to five (BASICites, how about a FOR-NEXT loop). (There's the second major part.) Lastly, because the computer is soooo fast and we are soooo slow, we will need a way to "WAIT" between "rings" to actually see (or should I say hear) if our computer counted to five. (There's the third major part.) Now we defined what we are going to need to get our work done!

1. RING THE BELL
2. COUNT TO FIVE
3. WAIT FOR US

STEP #2 .... SMOOTH FLOW

The second major step toward getting the job done is the logical FLOW of our program in the computer. To obtain the desired results, programmers have created the FLOW CHART (see fig. A). Let's read through this thing together and learn something about the chart and the innards of the CPU.

This chart is comprised of eight major parts or "blocks".

1. THE START is the beginning of the program which in BASIC terms relates to the first line. In Assembly language, the starting point or "line number" is usually an ADDRESS (refer to fig. B) We will pick the address of 100000 as the first "line"(the "A" after 100000 indicates that we have selected the OCTAL numbering system for reference with "A" denoting "split-octal" [nnnnnnA] later you will see a "Q" indicating "straight-octal" [nnnQ]). NOTE: OCTAL was selected since it is used as the front panel input for the H8 as well as the "command mode" input of the H89.

2. PUT FIVE IN B, (MVI B,FIVE or MoVe Immediate to the B register, #5) the next block of our chart, places the number five in a thing called the "B register" of the CPU. (There are seven registers in the 8080 CPU each of which has a letter assigned A,B,C,D,E,H,L.) As will be seen, we can manipulate these "registers" to provide useful functions.

3. CALL HORNO is our next step. The "CALL" is an instruction to the CPU which is EXACTLY like the GOSUB instruction used by BASIC. In fact, if you examine

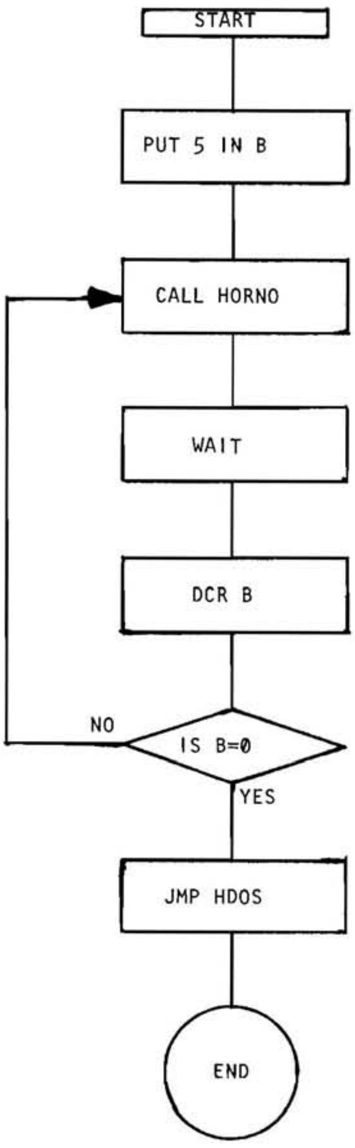


fig. A

the routine or listing for your monitor, under "HORNO", you will find somewhere a "RET" which is the same as a RETURN in BASIC thus completing the "GOSUB". (Easy isn't it!) Therefore, we can safely say that this "block" of our chart is a GOSUB routine that makes the "bell" ring one time!

4. WAIT is the next step of our program. WAIT is also an independent program within a program. This particular step causes a time delay and .... son-of-a-gun .... it's the same as a PAUSE in BASIC (more on this later).

5. DCR B is performed after WAIT. DCR B means that we subtract one (-1) from the original value of B (5 I think!). If we put five in B to begin with, we better have four after this step was performed. (Getting easier!)

6. Now it is DECISION time! "IS B=Ø" is the funny looking block that is used to determine if certain conditions have been encountered. In our program, we want to know if the B register is equal to zero YES or NO. If the answer is "NO" the program does a GOTO (BASIC again) or JNZ (Jump Not Zero) back to HORNO. If the answer is "YES" the program moves on to the next block. You will note that in each successive pass the B register is decremented by one (-1) until B=Ø. When B is zero, our program continues.

7. JMP HDOS is a GOTO (again!!!) or JuMP (unconditional) which passes from our "beeper" back to the HDOS prompt (>). Without going into great detail, this block was added to allow us to "get out" of our routine while NOT requiring a RESET to do so. (You experienced programmers .... QUIET PLEASE!!!!).

8. If you require an explanation of the last "block", GOTO the beginning of this article and type RESUME!!!

WAIT .... TAKE-A-PART

As mentioned earlier, our program contains three smaller programs. One is the counter, one is the "CALL" to HORNO which is a program built into the computer, and the last is WAIT, a program which we will examine fully. NOTE: WAIT will give you a good idea of just how fast a computer can perform a task once we get to the point of running our completed project. Further, note the similiarity of the WAIT routine to the NESTED FOR-NEXT LOOP used in common BASIC programs.

WAIT contains six major parts or "blocks"(see fig. C). All of these blocks would normally be compressed into the single block called WAIT in the main program FLOW CHART (see fig. A).

1. The first two blocks perform the same function. Therefore, we will discuss these together. Block one and block two PUT A NUMBER IN the D and E registers. The number selected was 377Q (remember straight-octal). This number is equal to 255 decimal. If you examine the Definition of Program Terms (see fig. B) you will see that TIME1 and TIME2 are "equated" to this number (377Q). If you were to select any other number (377O is the largest you could use here) the

```
******** START (BLOCK #1) ********

START   ORG     100000A STARTING ADDRESS (LINE NUMBER)

******** PUT #5 IN B (BLOCK #2) ********

        MVI     B,FIVE  PUT #5 IN "B" REGISTER

******** CALL HORNO (BLOCK #3) ********

LOOP    CALL    HORNO   GOTO MONITOR ROUTINE TO ACTIVATE "BELL"

******** WAIT (BLOCK #4) ********

        MVI     D,TIME1 PUT # IN "D" REGISTER FOR DELAY COUNT
LOOP2   MVI     E,TIME2 PUT # IN "E" REGISTER FOR DELAY COUNT
LOOP3   DCR     E       DECREMENT "E" REGISTER BY ONE
        JNZ     LOOP3   IF "E" IS NOT ZERO, DECREMENT AGAIN (LOOP3)
        DCR     D       WHEN "E" IS ZERO, DECREMENT D BY ONE
        JNZ     LOOP2   IF "D" IS NOT ZERO, RELOAD "E" WITH "TIME2" (WAIT)

******** DECREMENT B REGISTER (BLOCK #5) ********

        DCR     B       DECREMENT "B" BY ONE (OF FIVE)

******** DECISION - IS B=0 (BLOCK #6) ********

        JNZ     LOOP    IF "B" IS NOT 0 THEN GOTO HORNO (BLOCK #3)

******** GOTO HDOS (BLOCK #7) ********

        JMP     HDOSP   JUMP (GOTO) HDOS PROMPT TO FINISH PROGRAM

******** DEFINITIONS OF PROGRAM TERMS (EQUATE TABLE) ********

FIVE    EQU     005Q    FIVE = FIVE IN DECIMAL, HEX, OR OCTAL
HORNO   EQU     002136A THIS IS THE ADDRESS (LINE NUMBER) FOR THE "BELL"
TIME1   EQU     377Q    SOME #, THE AMOUNT INDICATING TIME DELAY
TIME2   EQU     377Q    SOME #, THE AMOUNT INDICATING TIME DELAY
HDOSP   EQU     040100A ADDRESS (LINE NUMBER) FOR RETURN TO HDOS PROMPT

******** END (BLOCK #8) ********

        END     START   THE "END" STATEMENT LOOPS TO THE "START" LABEL

*** NOTE: The "END" block appears after the DEFINITIONS which ARE NOT
          considered a portion of the flow chart "blocks".
```

fig. B

WAIT would be less between "beeps" in our main program.

2. As discussed earlier, the next block, DCR E, subtracts one (-1) from the original value of E (255 decimal).

3. Decision time! "IS E=0" YES or NO then creates a loop for 255 counts per the original value of E.

4. Now we DCR D or subtract one (-1) from D which had an original value of 255D.

5. Again a decision is requested (IS D=0, YES or NO). However, if the answer is NO or D is not equal to zero, then the E register gets "loaded" for another count of 255D. When D is zero, control is passed to our main program.

SUMMARY on WAIT

For every two-hundred fifty-five counts of E (decrements), D is decremented by one (-1). Therefore, the computer must subtract 255 x 255 times to get out of this NESTED loop or perform roughly 65,000

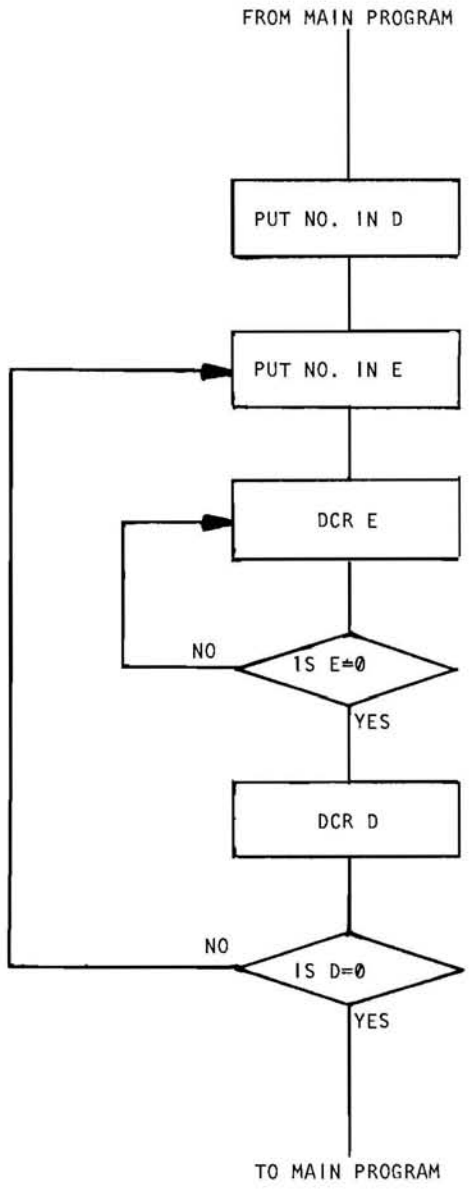


fig. C

steps to return to the original program. Amazing isn't it!!!

LET'S GET GOING ....

Well, we have DEFINED our work. We have looked at the FLOW needed for the computer to DO the work. Now it's time to "talk" to our machines or get them to "co-operate" with us! In the next issue of REMark, we will explore the use (or misuse!) of the HDOS Editor and Assembler in an effort to get this "mess" to run!

BE:

PAGED:

A Text Editor Anyone Can Use

By Scott Witt
79 Old Haverstraw Road
Congers, New York 10920

Word-processing is one of the most practical uses to which a microcomputer can be put, but most text editing programs are so cumbersome that they're a nuisance to use. A long list of commands must be memorized, and a simple slip of the finger can wipe out a text file reflecting many hours of work. Even the accomplished programmer soon tires of the time consuming and burdensome procedures required to prepare and edit text and assembly language listings.

That's certainly the way I felt after using some of the editor programs that were available for my H8 a couple of years ago. As a freelance writer with a contract to produce a book a year for a major American publisher, I needed a program that would enable me to enter and edit text rapidly. The hardware was there: the H19 terminal is extremely well suited for text-editing, and the H17 disk units provide rapid and reliable storage and retrieval of material. But the software was lacking. So, I wrote my own editor, designing it to be simple to operate and efficient to use. And then, thinking other people might benefit from an editor program that combines power with ease-of-use, I made it available to HUG members. (HUG P/N 885-1079 $25.00.)

It would be an understatement to say that PAGED (short for Page Editor) was well received. I've had mail from around the world and telephone calls from throughout the United States from enthusiastic users who say they've abandoned programs costing many times as much in favor of PAGED.

*An office manager from New Jersey said that seeing PAGED demonstrated in the local Heathkit Store convinced him to lay out several thousand dollars for an H19 and Diablo printer for his business.

*A Heathkit computer user from The Netherlands wrote: "As a writer of extensive assembly language programs as well as Fortran programs, I would like to thank you for the editor 'Paged' under HDOS. It's certainly a significant improvement over other editors I've used. More efficient and safer to use, it can also be used by non-programmers due to the extensive prompting."

(vectored to page 11)

H11 FORTRAN vs. BASIC

by Dr. Peter Vijlbrief, M. D.
Schoonoord 26
2155 ED Voorhout
The Netherlands

Editor's Note: The following two programs were sudmitted by Dr. Vijlbrief for our H-11 users. They are FORTRAN (listing 1) and BASIC (listing 2) versions of the popular "MASTERMIND" game. For those of you unfamiliar with MASTERMIND, it is a number guessing game. The player must find a hidden four digit number while the computer tells him how many digits are correct and in the correct places. These two programs give those familiar with BASIC but not with FORTRAN a chance to see how FORTRAN works. Our thanks to Dr. Vijlbrief for these programs.

Listing 1. MASTERMIND in FORTRAN (HT-11)

```
C       MASTERMIND IN FORTRAN
C       AUTHOR: DR. P. VIJLBRIEF - VOORHOUT
C
        TYPE 1
1       FORMAT (24X'MASTERMIND'/24X'**********'//
        1 1ØX'THE COMPUTER HAS A HIDDEN NUMBER OF FOUR DIFFERENT'/
        2 1ØX'DIGITS.  YOU HAVE TO FIND THAT NUMBER.'/
        3 1ØX'ENTER FOUR DIGITS AND HIT RETURN <CR>.  THE COMPUTER DISPLAYS'/
        4 1ØX'YOUR NUMBER AND FLAGS IT WITH TWO DIGITS SEPARATED'/
        5 1ØX'BY A SLASH (C/W).  THE FIRST DIGIT (C) COUNTS THE'/
        6 1ØX'RIGHT DIGITS IN THE CORRECT PLACE, THE SECOND'/
        7 1ØX'DIGIT (W) COUNTS THE RIGHT DIGITS IN THE WRONG PLACE.'/
        8 1ØX'IF YOU CAN NOT FIND THE HIDDEN NUMBER, TYPE ØØØØ <CR>,'/
        9 1ØX'THE COMPUTER GIVES YOU THE ANSWER.'//)
C
C       THIS NUMBER IS THE SEED FOR A RANDOM NUMBER GENERATOR
C
        TYPE 2
2       FORMAT (1ØX'TO START THE GAME, ENTER A TWO DIGIT NUMBER:'//)
        ACCEPT 3, K,M
3       FORMAT ( 2I1)
C
C       INITIALIZE
C
        AVER=Ø
        ITURN=Ø
        TURN=Ø
        TOT=Ø
C
C       BEGIN GAME
C
5       TRY=Ø
        ITRY=Ø
        ITURN=ITURN+1
        TURN=TURN+1
C
C       MAKE RANDOM NUMBER (FOUR DIFFERENT DIGITS, FIRST NOT=ZERO
C
1Ø      IONE=INT(RAN(K,M)*1Ø.)
        IF (IONE*1) 2Ø,1Ø,2Ø
2Ø      ITWO=INT(RAN(K,M)*1Ø.)
        IF (ITWO-IONE) 3Ø,2Ø,3Ø
3Ø      ITHREE=INT(RAN(K,M)*1Ø.)
        IF ((ITHREE-IONE)*(ITHREE-ITWO)) 4Ø,3Ø,4Ø
4Ø      IFOUR=INT(RAN(K,M)*1Ø.)
        IF ((IFOUR-ITHREE)*(IFOUR-ITWO)*(IFOUR-IONE)) 5Ø,4Ø,5Ø
C
C       INITIALIZE FO BEGIN OF PLAYER'S GUESS
C
5Ø      IGOOD=Ø
```

```
      IWRONG=0
      TRY=TRY+1.
      ITRY=ITRY+1
C
C     PLAYER'S GUESS
C
      ACCEPT 300, JONE,JTWO,JTHREE,JFOUR
C
C     DOES PLAYER QUIT?
C
      IF (JONE+JTWO+JTHREE+JFOUR) 55,500,55
C
C     SEARCH FOR CORRECT DIGITS IN CORRECT PLACES
C
55    IF (JONE-IONE) 70,60,70
60    IGOOD=IGOOD+1
70    IF (JTWO-ITWO) 90,80,90
80    IGOOD=IGOOD+1
90    IF (JTHREE-ITHREE) 110,100,110
100   IGOOD=IGOOD+1
110   IF (JFOUR-IFOUR) 130,120,130
120   IGOOD=1GOOD+1
C
C     SEARCH FOR CORRECT DIGITS IN WRONG PLACES
C
130   IF ((JONE-ITWO)*(JONE-ITHREE)*(JONE-IFOUR)) 150,140,150
140   IWRONG=IWRONG+1
150   IF ((JTWO-IONE)*(JTWO-ITHREE)*(JTWO--IFOUR)) 170,160,170
160   IWRONG=IWRONG+1
170   IF ((JTHREE-IONE)*(JTHREE-ITWO)*(JTHREE-IFOUR)) 190,180,190
180   IWRONG=IWRONG+1
190   IF ((JFOUR-IONE)*(JFOUR-ITWO)*(JFOUR-ITHREE)) 210,200,210
200   IWRONG=IWRONG+1
C
C     DISPLAY GUESS NUMBER AND FLAG IT
C
210   TYPE 350, JONE,JTWO,JTHREE,JFOUR,IGOOD,IWRONG
C
C     FIND OUT IF GUESS IS ALL CORRECT?
C
      IF (IGOOD-4) 50,220,50
C
C     FIGURE OUT AND DISPLAY SCORE
C
220   TOT=TOT+TRY
      AVER=TOT/TURN
      TYPE 400, ITRY,AVER
C
C     ASK FOR ANOTHER GAME
C
      ACCEPT 450,MM
      IF (MM-1) 600,5,600
300   FORMAT (4I1)
350   FORMAT ('+',30X,I2,I2,I2,I2,I3,'/',I1)
400   FORMAT (//'YOU FOUND IT IN ',I2,' TRIES.'
      1 /'YOUR AVERAGE IS: ',F4.1,' TRIES.'
      2 /'WANT ANOTHER GAME ? TYPE 1, ELSE 0.')
450   FORMAT ( I1)
500   TYPE 550, IONE,ITWO,ITHREE,IFOUR
550   FORMAT (//' THE RIGHT ANSWER IS: ',4I2)
      STOP
C
C     GIVE END SCORE
C
600   TYPE 650, ITURN,AVER
650   FORMAT (//' YOU HAVE FOUND ',I2,' NUMBERS IN AN AVERAGE'/
      1 ' OF ',F4.1,' TRIES.')
      END
```

Listing 2. MASTERMIND in BASIC (HT-11)

```
10 REM AUTHOR: DR. P. VIJLBRIEF, VOORHOUT, THE NETHERLANDS
20 REM
30 PRINT TAB(25);"MASTERMIND"
40 PRINT TAB(25);"**********"
50 PRINT
60 PRINT TAB(10);"THE COMPUTER CONTAINS A HIDDEN NUMBER OF FOUR"
70 PRINT TAB(10);"DIFFERENT DIGITS."
80 PRINT
90 PRINT TAB(10);"TRY TO GUESS THE NUMBER."
100 PRINT
110 PRINT TAB(10);"ENTER A FOUR DIGIT NUMBER, SEPARATED BY COMMAS:"
120 PRINT TAB(10);"X,X,X,X <CR>"
130 PRINT
140 PRINT TAB(10);"THE COMPUTER DISPLAYS YOUR NUMBER AND FLAGS IT"
150 PRINT TAB(10);"WITH TWO DIGITS, SEPARATED BY A SLASH (C/W)."
160 PRINT TAB(10);"THE 'C' TOTALS THE AMOUNT OF RIGHT DIGITS IN THE"
170 PRINT TAB(10);"CORRECT PLACE AND THE 'W' TOTALS RIGHT DIGITS IN"
180 PRINT TAB(10);"WRONG PLACES."
190 PRINT
200 PRINT TAB(10);"IF YOU CAN'T FIND THE CORRECT NUMBER, ENTER:"
210 PRINT TAB(10);"0,0,0,0 <CR>.  THE COMPUTER GIVES YOU THE ANSWER"
220 PRINT TAB(10);"BUT THE GAME IS OVER !"
230 REM 'G' IS THE NUMBER OF GAMES PLAYED.
240 REM 'T' IS THE TOTAL OF TRIES TO GUESS THE NUMBER IN ALL GAMES."
250 REM 'C1' AND 'W' ARE THE CORRECT AND WRONG DIGIT COUNTS."
260 REM 'N' IS THE NUMBER OF GUESSES IN ONE GAME.
270 REM INITIALIZE
280 LET G=0
290 LET T=0
300 LET N=0
310 REM MAKE THE HIDDEN NUMBER AT RANDOM, FIRST DIGIT NOT ZERO
320 LET G=G+1
330 RANDOMIZE
340 A=INT(RAN(0)*10)
350 IF A=0 THEN 340
360 B=INT(RAN(0)*10)
370 C=INT(RAN(0)*10)
380 D=INT(RAN(0)*10)
390 IF A=B THEN 340\IF A=C THEN 340\IF A=D THEN 340
400 IF B=C THEN 340\IF B=D THEN 340
410 IF C=D THEN 340
420 REM A,B,C,D ARE THE FOUR DIFFERENT DIGITS OF THE HIDDEN NUMBER
430 LET C1=0
440 LET W=0
450 PRINT
460 INPUT P,Q,R,S
470 REM P,Q,R,S ARE THE PLAYER'S GUESS
480 REM DOES THE PLAYER QUIT?
490 IF P+Q+R+S=0 THEN 930
500 REM SEARCH FOR RIGHT DIGITS IN RIGHT OR WRONG PLACES
510 IF P<>A THEN 530
520 C1=C1+1
530 IF P=B THEN 540 \IF P=C THEN 540 \IF P<>D THEN 550
540 W=W+1
550 IF Q=A THEN 560 \IF Q=C THEN 560 \IF Q<>D THEN 570
560 W=W+1
570 IF R=A THEN 580 \IF R=B THEN 580 \IF R<>D THEN 590
580 W=W+1
590 IF S=A THEN 600 \IF S=B THEN 600 \IF S<>C THEN 610
600 W=W+1
610 IF Q<>B THEN 630
620 C1=C1+1
630 IF R<>C THEN 650
640 C1=C1+1
650 IF S<>D THEN 670
660 C1=C1+1
670 REM CANCEL LINEFEED AND REMOVE PLAYER'S INPUT (H19)
```

```
680 PRINT CHR$(27);CHR$(65);TAB(20);P;" ";Q;" ";R;" ";S;"    ";C1;"/";W
690 REM COUNT TRIES
700 N=N+1
710 REM LOOK FOR END OF GAME
720 IF C1<4 THEN 430
730 PRINT
740 PRINT TAB(10);"YOU FOUND THE NUMBER IN ";N;" GUESSES !"
750 REM COUNT THE TOTAL NUMBER OF GUESSES
760 T=T+N
770 REM DON'T GIVE AVERAGES AFTER ONE GAME
780 IF G=1 THEN 800
790 PRINT TAB(10);"YOUR AVERAGE IS ";T/G;" GUESSES."
800 PRINT
810 PRINT TAB(10);"ANOTHER GAME ? (TYPE YES OR NO)"
820 INPUT Y$
830 IF Y$="YES" THEN 870
840 IF Y$="NO" THEN 880
850 PRINT TAB(10);"TYPE 'YES' OR 'NO'!"
860 GOTO 820
870 PRINT \PRINT "TYPE YOUR GUESS: "\GOTO 300
880 PRINT
890 PRINT TAB(10);"AFTER ";G;" GAMES YOUR AVERAGE OF GUESS TO FIND"
900 PRINT TAB(10);"THE HIDDEN NUMBER WAS ";T/G
910 PRINT TAB(10);"THANKS FOR PLAYING !"
920 END
930 PRINT
940 PRINT TAB(10);"THE HIDDEN NUMBER WAS ";" ";A;" ";B;" ";C;" ";D
950 STOP
```

EOF

(vectored from page 7)

*A math professor from the University of Utah wrote: "Your editor rivals TV-Teco on the DEC. The members of the Heath Users' Group here in Salt Lake City feel that no other editor available is as easy to learn and use as PAGED."

The "extensive prompting" mentioned by the user from The Netherlands is something that was missing from editors I used prior to PAGED. When your mind is engrossed in writing a program (or, in my case, the chapter of a book) it's easy to lose your train of thought if you must recall a complex series of commands merely to change a word or paragraph. Computers being powerful instruments, there's no reason why any good program shouldn't be extremely easy to use.

Here are just a few of PAGED's built-in features:

ONE-LETTER COMMANDS. If you want to insert some lines, merely type "I" and the computer will display the entire word INSERT on the screen and then ask you where you want the insert to go. The same with E(dit) and all of the other commands. Rather than your having to remember to enter commands in a set pattern, the program asks you for the information it requires, and all of it is entered with a keystroke or two.

AUTOMATIC MARGIN. When you're using a computer, you might as well make full use of its computational capability. Thus PAGED has an option in which it will automatically start a new line once the current line has taken all the words it can. You don't have to watch the screen to keep track of the cursor, and you need never hit the carriage return key. If, upon later editing, the length of a line is changed, a simple command will re-format the entire paragraph -- or entire page, for that matter. A block of text that is, say, 72 characters wide can be narrowed to, say, 20 characters as fast as you can sneeze.

DOUBLE-SPACING. Need double-spaced material, but want to cram as much of it on the screen as you can? Then enter it in single-space format; when you're through, tell the program to double-space your material. A similar command does the reverse -- changing double-spaced material to single-spacing.

DISK RESET. Once PAGED is loaded into memory, you can reset, or even dismount, disks at will. You can even operate with no disks in the machine at all.

FILE STREAMING. Your file, be it text or a program listing, may be longer than the memory capacity of your computer allows. PAGED lets you stream it in to memory for editing, and then out to disk so that more material can be brought in.

FILE VERIFICATION. When streaming text into and out of a program, it's easy to forget which file is which. The "Verify"

(vectored to page 15)

Using RDT to Make a Hex-Octal Assembler

This article presents modifications to the new HDOS 2.0 assembler and its cross reference utility (XREF) that allow you to select output of addresses and data in either hex or octal. It also illustrates one of the uses of the RDT debugging tool (HUG part no. 885-1092) that was introduced in REMark issue #14. To make these modifications, you will need:

1. An HDOS 2.0 system disk.

2. A working disk containing ASM, XREF, and RDT (unless they are on your system disk) and ample work space. You will also need an editor on one of the disks.

Listing 1 is the source for the modification for the assembler, and listing 2 is the XREF modification. Type them in with your editor and assemble them. Then run RDT. Since this modification is obviously of interest to those who like hex notation, we will perform the work in hex. The first command we need to give to RDT, then, is

```
]BASE HEX
```

RDT is now set for input and output in hex. What we are going to do with it is combine the .ABS files you just made with the assembler and cross reference programs. The RDT instruction manual explains how to merge .ABS files like this. In this case the patch is not within the program but after it, so you might think that you can simply load the assembler, then load the patch, then save the whole thing back on disk. But it cannot be done that way because an .ABS file actually starts 8 bytes before its ORG address, and the patch would overwrite the end of the assembler. (Those 8 bytes contain the machine code I D, the program's origin, it's size, and it's entry point.) So we will use the method presented in the RDT manual. The first thing to do is load the patch for the assembler. If it is named ASMEXT.ABS, we would type

```
]LOAD SY1:ASMEXT   (you type this)

STARTS  4208       (RDT prints this)
ENDS    4292
```

RDT prints the starting and ending addresses of the patch in hex. The next step is to move the patch out of the way so it will not be overwritten by the assembler when we load it (because its end address is rounded up to the next 256 byte boundry by HDOS). We can move the patch by typing

```
]MOVE 4208,4292,5208
```

The patch is now exactly 1000H bytes farther up in memory. Now, we can load the assembler:

```
]LOAD SY1:ASM

STARTS  2280
ENDS    4207
```

Now all we have to do is move the patch back down and save the combined file on disk:

```
]MOVE 5208,5292,4208

]SAVE SY1:ASM,2280,4292,4208
```

The third address given in the SAVE command is the program entry point, which in this case is the entry point of the patch. When the assembler is run, it will start at that point. Note that the old assembler on SY1: is wiped out when we save the new one. We could have given the new one a different name and/or saved it on another disk to avoid erasing the old one.

We can now modify the XREF program using the above procedure. CAUTION: This patch is for the HDOS 2.0 XREF.ABS, not the HUG XREF.ABS. The RDT command sequence is as follows:

```
]LOAD SY1:XRFEXT

STARTS  2DE7
ENDS    2E1A

]MOVE 2DE7,2E1A,3DE7

]LOAD SY1:XREF

STARTS  2280
ENDS    2DE6

]MOVE 3DE7,3E1A,2DE7

]SAVE SY1:XREF,2280,2E1A,2DE7
```

As with the assembler, we could have given the cross reference program a different name when we saved it, but it would have to be renamed back to XREF.ABS before we ran it because the assembler links to a file with that name when you ask it to do a cross reference.

OPERATION

The patched assembler can be used in the immediate mode or command mode in the same way as the old assembler, except that it will ask you for the base you want. In the immediate mode, it looks like this:

```
>ASM SY1:ASMEXT=SY1:ASMEXT

Modified Heath Assembler

Want Hex or Octal? (H or O) <O> _
```

As soon as you type an H or O, the assembly starts. The default is octal, so any key hit other than H will produce octal output. You can hit RETURN twice after the command line if you want octal. Hex mode will be set if you type either an upper or lower case H. If you use the command mode, it looks like this:

```
>ASM

Modified Heath Assembler

Want Hex or Octal (H or O) <O> H

HDOS ASseMbler Issue #1Ø4.Ø6.ØØ

*
```

HOW THE PATCHES WORK

Both the assembler and cross reference programs have routines that cause binary numbers to be converted to ASCII in octal notation. The patches replace those routines with ASCII hex output routines when you specify hex. The assembler patch also removes the code that puts a period between the upper and lower bytes of an address if you specify hex. If you tell the assembler that you want a cross reference, it links to the XREF program when it finishes assembling. Since the end of the XREF program falls below the hex patch in the assembler, it is still there for the XREF extension to examine. If it finds that the assembler has been patched, it patches the cross reference program as well, so both will produce hex output. In addition to patching the output routine, the XREF patch also has to replace two ASCII zeros with spaces. If that is not done, hex addresses will have two trailing zeros in the cross reference listing.

PS:

Listing 1. Modification for ASM

```
        TITLE   'EXTENSION TO THE HEATH ASSEMBLER'
        STL     'BY PATRICK SWAYNE  24-DEC-8Ø'
* ASMEXT.ASM
* THIS IS AN EXTENSION TO HEATH'S ASSEMBLER VERSION 1Ø4.Ø6.ØØ
* IT ALLOWS THE USER TO SELECT OUTPUT IN OCTAL OR HEX

        ORG     42Ø8H           FIRST ADDRESS AFTER ASM

* EXTERNAL REFERENCES AND CONSTANTS

$MOVE   EQU     18AAH           H17 ROM MOVE ROUTINE
$TYPTX  EQU     195EH           STRING PRINTER
CRLF    EQU     ØAH             CR - LF
OLDSTRT EQU     3B6CH           OLD ASM ENTRY POINT
NUMO    EQU     34AFH           NUMBER OUTPUT ROUTINE
PERIOD  EQU     326EH           PERIOD OUTPUT ROUTINE
$CONASC EQU     34B8H           CONASC ADDRESS IN ASM
.SCIN   EQU     1               HDOS INPUT ROUTINE
.SCOUT  EQU     2               HDOS OUTPUT ROUTINE
.CONSL  EQU     6               CONSOLE SETUP ROUTINE
.CLRCO  EQU     7               CLEAR CONSOLE

START   CALL    $TYPTX
        DB      CRLF,'Modified Heath Assembler',CRLF
        DB      CRLF,'Want hex or octal? (H or O) <O>',24ØQ
        XRA     A
        LXI     B,8181H
        SCALL   .CONSL          SET UP CONSOLE FOR SINGLE CHAR
INCH    SCALL   .SCIN
        JC      INCH            GET RESPONSE
        SCALL   .SCOUT          ECHO CHARACTER
        ANI     5FH             MAKE CAPITAL
        CPI     'H'             HEX?
        JNZ     EXIT            WANT OCTAL, START THE ASSEMBLER
        LXI     H,NUMO
        LXI     D,HEXOUT        SET POINTERS
        LXI     B,HEXEND-HEXOUT NO. OF BYTES TO MOVE
        CALL    $MOVE           INSERT HEX OUTPUT ROUTINE
        XRA     A               ZERO IN A
        LXI     H,PERIOD        PERIOD OUTPUT ROUTINE
        MOV     M,A             ZERO IT OUT
        INX     H
        MOV     M,A
        INX     H
        MOV     M,A
EXIT    MVI     A,CRLF
        SCALL   .SCOUT          PRINT CR, LF
        SCALL   .CLRCO          MAKE SURE CONSOLE IS CLEAR
        XRA     A
        LXI     B,ØFFH
        SCALL   .CONSL          RESTORE CONSOLE TO LINE MODE
        JMP     OLDSTRT         GO TO ASSEMBLER

* HEX OUTPUT ROUTINE
* REPLACES THE OCTAL OUTPUT ROUTINE
* IN THE ASSEMBLER IF NEEDED

HEXOUT  PUSH    PSW             SAVE BYTE
        RRC                     GET HIGH NIBBLE
        RRC
        RRC
        RRC
        CALL    $CONASC         CONVERT TO ASCII
        POP     PSW             RESTORE BYTE
CONASC  ANI     ØFH             GET LOW NIBBLE
        ADI     9ØH             CONVERT TO ASCII
        DAA
        ACI     4ØH
        DAA
        MOV     M,A             STORE NUMBER
        INX     H               MOVE POINTER
        RET
HEXEND  EQU     *
        END     START
```

Listing 2. Modification for XREF

```
        TITLE   'EXTENSION TO THE HEATH XREF UTILITY'
        STL     'BY PATRICK SWAYNE  24-DEC-8Ø'
* XRFEXT.ASM
* THIS IS AN EXTENSION TO THE ASSEMBLER'S XREF UTILITY
* THAT DETERMINES WHETHER NUMBERS SHOULD BE OUTPUT IN
* OCTAL OR HEX, AND MODIFIES XREF IF REQUIRED.


* EXTERNAL REFERENCES, ETC.

$MOVE   EQU     18AAH           H17 ROM MOVE ROUTINE
OLDSTRT EQU     228ØH           OLD XREF STARTING POINT
ZEROS   EQU     27BDH           ASCII ZEROS ARE HERE
NUMO    EQU     2CB6H           NUMBER OUTPUT ROUTINE
PATCH   EQU     34AFH           ADDRESS OF PATCH IN ASM
$CONASC EQU     2CBFH           CONASC ADDRESS IN XREF
MI.PPSW EQU     ØF5H            PUSH PSW INSTRUCTION
SPACE   EQU     2ØH             ASCII SPACE

        ORG     2DE7H           FIRST ADDRESS AFTER XREF

START   LDA     PATCH
        CPI     MI.PPSW         IS ASM MODIFIED?
        JNZ     OLDSTRT         IF NOT, START XREF
        LXI     H,ZEROS         POINT TO ASCII ZEROS
        MVI     M,SPACE         REPLACE WITH SPACE
        INX     H               MOVE TO NEXT ZERO
        MVI     M,SPACE         REPLACE WITH SPACE
```

```
         LXI     H,NUMO
         LXI     D,HEXOUT            SET POINTERS
         LXI     B,HEXEND-HEXOUT NO. OF BYTES TO MOVE
         CALL    $MOVE               INSERT THE HEX ROUTINE
         JMP     OLDSTRT             IF SO, START XREF

* HEX OUTPUT ROUTINE
* REPLACES OCTAL ROUTINE IN XREF
* IF NECESSARY

HEXOUT   PUSH    PSW                 SAVE BYTE
         RRC                         GET HIGH NIBBLE
         RRC
         RRC
         RRC
         CALL    $CONASC             CONVERT TO ASCII
         POP     PSW                 RESTORE BYTE
CONASC   ANI     ØFH                 GET LOW NIBBLE
         ADI     9ØH                 CONVERT TO ASCII
         DAA
         ACI     4ØH
         DAA
         MOV     M,A                 STORE NUMBER
         INX     H                   MOVE POINTER
         RET
HEXEND   EQU     *
         END     START
```

EOF

(vectored from page 11)

command tells you at a glance which files are open for read and write.

PAGE FORMAT. The program deals in "pages" which are really 24-line screenfuls of text (the 25th line on the screen is the command-line). Thus the first 24 lines of your file constitute Page 1, with succeeding pages made up of ensuing blocks of 24 lines each. Editing and the other commands deal with the page that is displayed on the screen at the time. You can jump to any page at will. Each line on the page is numbered so you can refer to a particular line for editing merely by entering the line number. In addition, there's a multi-edit capability that allows you to edit a particular string wherever it appears.

LINE CENTERING. Sometimes the simplest of features can come in the most handy. Centering a line of type, as in the title of an article, is easily accomplished with PAGED.

There are, of course, many other useful features contained in PAGED. What you've seen above are just some of the goodies that help make it easier to use. A man from the West Coast called to say he'd finally found a program that his wife could use. She knows nothing about computers and doesn't want to learn. But she does need to do word-processing, and she finds PAGED a snap.

What makes it equally gratifying to me is the fact that professional programmers also find PAGED a great help. One of them wrote: "It has been four months since the purchase of your editor. During this time it has served as my development editor for a number of programs. I use an H89 connected via a modem to a DEC-20 or PDP-11/60."

And, finally, this comment from a HUG member: "I'm pleased to see that HUG is offering this, and many other useful programs, at a fraction of what they might cost in the commercial marketplace. They have made my computer a tool instead of a toy."

EOF

H8COMM/H89COMM Modification

885-1Ø43 Mod for HDOS 1.6

A simple modification to the TRANSUB.ACM file will allow the H8COMM/H89COMM to operate under version 1.6 of HDOS.

The modification is to add an asterisk (*) in front of the TWO lines that have "STA .ZFLAG" near the beginning of the TRANSUB.ACM source code file. A reassembly is required before operation.

EOF

New HUG Software

885-1207 TERM and H8COPY $20.00

TERM was developed for HUG by Jim Buszkeiwicz of the Heath Technical Consultation Group as a CP/M answer to CPS. It is capable of providing the following features:

1. Full or half duplex operation
2. It can send or recieve disk files.
3. It works on H8's and H89's.
4. It can automatically log on to TIMENET or MicroNET

TERM comes with full documentation on disk. It is designed to work with CP/M version 2.0 or higher.

H8COPY allows you to copy files from HDOS to CP/M. The program comes in assembly source and can be assembled for either ORG 4200H CP/M or ORG 0 CP/M by changing an EQUate. It has two modes of operation, direct and ASCII. In the direct mode, a file is copied as is. In the ASCII mode, HDOS newline characters are replaced with the carriage return-line feed sequence used in CP/M as the file is copied. The ASCII mode also can convert the @ sign used to extend logical lines in HDOS MBASIC to the reverse carriage return-line feed sequence used in CP/M MBASIC if you wish.

Movin' On

by Terry Jensen

It's inevitable...every place of business gets hit sooner or later by "growing pains" by its customers and "advancing pains" by its personnel...well, once again both of these pains have hit HUG.

HUG's continued growtn as any other business demands more from its services and our new "Chief" is attacking this growth head-on (more later). Sometimes, however, little is noticed of the employee that moves on to higher "rungs" of employment.... well, here in HUG, we do indeed miss the loss of Jon Falkner as part of the "gang".

Jon has moved back into Heath Company where his technical knowledge will be utilized helping the Heath customer. I fortunately had the opportunity to work with Jon for about three months... soaking his brain any chance I could for facts, short-cuts and generally any knowledge I could use to help me learn. Most of the time I was getting more than I could handle. Well, now Jon is gone and yours truly must "fill his shoes". No easy task!!!

"Boss" Bob has now had time to evaluate HUG and its purpose and he has some excellent ideas and intentions. "Captain" Bob has issued his "orders" and we are determined to carry out those "orders".

As for myself, I am now in the exasperating position of sitting in Jon's old office with "Big Bad" Bob lerking over my shoulder. Am I looking for sympathy?... No... on the contrary I am fascinated by the challenge and opportunity I have found myself. (I wonder how many users' would give their right arm to be in my situation.)

As I glance around my office I see, next to the door, a desk with a calender for keeping notes. On the desk are two phones one for conversation and the second for MicroNET. Between the phones sits a DIABLO 1640 for replying, by use of AUTOSCRIBE, all those wonderful letters that come in each day. Along the back wall sits a table with an H-8, H-17 and H-19. Next to the table the DECWRITER sits waiting for my next hardcopy to be printed.

Opposite the door sits a second desk which contains my files, disk files and miscellaneous things. On top of the desk is an H-11A, H-27 system interfaced to the H-8, H-19. Many of my personal and "closest" Heath manuals are filed here also. Along the last wall I find a bookshelf where most of the manuals I will ever need (and some I have never seen before) are nicely and professionally filed patiently waiting for the opportunity to be opened and read. A chalk board will be placed above the bookshelf to act as a status board on "what's happenin'" including future projects.

Scattered on the walls around the room are H-8, H-89 posters with every issue of REMark (excepting issue #1). Taped under the REMark magazines are lists, charts, and tables easily accessable for reference.

Finally, above and behind the H-19 terminal are good, spiritual notes and Bible verses. I have been told that we will reflect what we read and what we place in the environment that surrounds us.

With Jon having gone on, and Bob pressing for good things for HUG, it is my intention to accomplish and fulfill my responsibilities to Jon, Bob, and ultimately you the user!

<TLJ>

HUG Product List

Part Number	Description	Selling Price
CASSETTE SOFTWARE		
MISCELLANEOUS COLLECTIONS		
885-1008	Volume I Documentation	$ 9.00
885-1009	Tape I Cassette	$ 7.00
885-1012	Tape II BASIC Cassette	$ 9.00
885-1013	Volume II Documentation	$ 12.00
885-1014	Tape II ASM Cassette H8 Only	$ 9.00
885-1015	Volume III Documentation	$ 12.00
885-1026	Tape III Cassette	$ 9.00
885-1036	Tape IV Cassette	$ 9.00
885-1037	Volume IV Documentation	$ 12.00
885-1057	Tape V Cassette	$ 9.00
885-1058	Volume V Documentation	$ 12.00
UTILITIES		
885-1034	Character Ed Cassette H8 Only	$ 11.00
885-1035	ED/ASM/DEBUG Cassette H8 Only	$ 11.00
PROGRAMMING LANGUAGES		
885-1039	WISE on Cassette H8 Only	$ 9.00
885-1040	PILOT on Cassette H8 Only	$ 11.00
885-1045	FOCAL Cassette H8 Only	$ 11.00
885-1085	PILOT Documentation	$ 9.00
AMATEUR RADIO		
885-1027	Morse8 Cassette H8 Only	$ 14.00
885-1028	RTTY Cassette H8 Only	$ 11.00
HDOS SOFTWARE		
MISCELLANEOUS COLLECTIONS		
885-1024	Disk I H8/H89	$ 18.00
885-1032	Disk V H8/H89	$ 18.00
885-1044	Disk VI H8/H89	$ 18.00
885-1060	Disk VII H8/H89	$ 18.00
885-1062	Disk VIII H8/H89 (2 Disks)	$ 25.00
885-1064	Disk IX H8/H89	$ 18.00
885-1066	Disk X H8/H89	$ 18.00
885-1069	Disk XIII Misc H8/H89	$ 18.00
885-1083	Disk XVI Misc H8/H89	$ 20.00
GAMES		
885-1010	Adventure Disk H8/H89	$ 10.00
885-1029	Disk II Games 1 H8/H89	$ 18.00
885-1030	Disk III Games 2 H8/H89	$ 18.00
885-1031	Disk IV Music H8 Only	$ 23.00
885-1067	Disk XI H8/H19/H89 Games	$ 18.00
885-1068	Disk XII MBASIC Graphic Games	$ 18.00
885-1088	MBASIC Games Disk	$ 20.00
UTILITIES		
885-1019	Device Drivers (HDOS 1.6)	$ 10.00
885-1022	HUG Editor (ED) Disk H8/H89	$ 15.00
885-1025	Runoff Disk H8/H89	$ 35.00
885-1043	MODEM Heath to Heath H8/H89	$ 21.00
885-1050	M.C.S. Modem for H8/H89	$ 18.00
885-1061	TMI Load H8 Only	$ 18.00
885-1063	Floating Point Disk H8/H89	$ 18.00
885-1065	Fix Point Package H8/H89 Disk	$ 18.00
885-1075	HDOS Support Package H8/H89	$ 60.00
885-1077	TXTCON/BASCON H8/H89 Disk	$ 18.00
885-1079	HDOS Page Editor	$ 25.00
885-1080	EDITX H8/H19/H89	$ 20.00
885-1082	Programs for Printers H8/H89	$ 20.00
885-1092	RDT Debugging Tool H8/H89 Disk	$ 30.00
PROGRAMMING LANGUAGES		
885-1038	WISE on Disk H8/H89	$ 18.00
885-1042	PILOT on Disk H8/H89	$ 19.00
885-1059	FOCAL-8 on Disk H8/H89	$ 25.00
885-1078	HDOS Z80 Assembler	$ 25.00
885-1085	PILOT Documentation	$ 9.00
885-1086	Tiny Pascal Disk	$ 20.00
BUSINESS AND FINANCE		
885-1047	Stocks H8/H89 Disk	$ 18.00
885-1048	Personal Account H8/H89 Disk	$ 18.00
885-1049	Income Tax Records H8/H89 Disk	$ 18.00
885-1051	Payroll H8/H89 Disk	$ 50.00
885-1054	SmBusPkg II 3 Disks H8/H19/H89	$ 60.00
885-1055	MBASIC Inventory Disk H8/H89	$ 30.00
885-1056	MBASIC Mail List H8/H89 Disk	$ 30.00
885-1070	Disk XIV Home Finance H8/H89	$ 18.00
885-1091	Grade and Score Keeping	$ 30.00
AMATEUR RADIO		
885-1023	RTTY Disk H8 Only	$ 22.00
885-1052	Morse8 Disk H8 Only	$ 18.00
H11 SOFTWARE		
885-1008	Volume I Documentation	$ 9.00
885-1033	HT-11 Disk I	$ 19.00
CP/M SOFTWARE (version 1.43 -- ORG 4200H)		
885-1201	CP/M (TM) Volumes H1 and H2	$ 21.00
885-1202	CP/M Volumes 4 and 21-C	$ 21.00
885-1203	CP/M Volumes 21-A and B	$ 21.00
885-1204	CP/M Volumes 26/27-A and B	$ 21.00
885-1205	CP/M Volumes 26/27-C and D	$ 21.00
885-1206	CP/M Games Disk	$ 21.00
CP/M SOFTWARE (version 2.2 -- ORG 0)		
885-1027	TERM and H8COPY	$ 20.00
MISCELLANEOUS		
885-0017	H8 Poster	$ 2.95
885-0018	H89 Poster	$ 2.95
885-0019	Color Graphics Poster	$ 2.95
885-4	HUG Binder	$ 5.75

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

$10,000 FIRST PRIZE!

John Hopkins Launches National Search --

Personal Computing to Aid the Handicapped

The first national search for ideas and inventions through which the full spectrum of personal computing technology can be harnessed to assist the handicapped has been announced by the John Hopkins University.

To be conducted by the Applied Physics Laboratory of the University, and with the National Science Foundation and many computer companys as co-sponsers, the effort will be highlighted by a national competition for ideas, devices, methods, and computer programs to help handicapped people overcome difficulties in learning, working, successfully adapting to home and community settings. Categories that may be addressed include computer-based aids for the blind, deaf and mentally retarded; for individuals with learning disabilities, neurological or neuromuscular conditions; and the orthopedically handicapped.

One hundred awards will be made, including a $10,000 grand prize, personal computer equipment, other cash prizes, computer training and certificates of merit. Entires will be sought from computer specialists, full-time high school and college students, and from interested people generally, including those with handicaps. Regional and national awards will be made in all categories. National awards will be presented at a banquet in the fall of 1981 in the Washington, D. C. area.

Paul L. Hazan, director of the Personal Computing to Aid the Handicapped project, said the competition is a challenge to the American people to use their conceptual skills in bringing forth practical aids based on computer technology that will help an individial or group of people with a handicap. "Just as important will be the opportunity provided the inventors and developer to make contact and form partnerships with the handicapped in a way that can lead to wide acceptance and use of the new computing technology," Hazan stressed.

Orientation meetings are being scheduled at major rehabilitation centers throughout the United States to bring together potential "inventors," handicapped people and professionals in habilitation rehabilition fields. Special presentations also will be made nationwide at chapter meetings of the Association for Computing Machinery (ACM), Institue of Electrical and Electronics Engineers (IEEE), and personal computer clubs.

Contestants will have until June 30, 1981 to prepare and submit their entires.

To obtain additional information including a descriptive flyer and contest application, write to:

Personal Computing to Aid the Handicapped
John Hopkins University
Post Office Box 670
Laurel, MD 20810

EOF

Some Thoughts on Writing Game Programs

by Roy S. Reichert
29 Blazier Road
Warren, New Jersey 07060

We all like to play games -- especially those that run on a microcomputer. When you look at the software offerings in the catalog of any User Group, HUG included, you cannot help but notice the high percentage of game programs offered. I suspect, however, that just the desire to play them is not the prime reason for the existence of so many game programs. The fact is that we, as owners and users of such sophisticated systems as the modern microcomputer, are inclined to explore the full potential of these systems and we want to have fun while we do it. So, we not only play games -- we create them as well.

It has long been recognized in the programming field, that the development of game programs is a valuable and highly effective way for the programmer to develop new techniques, algorithms and insights into the complexities of hardware and software. The development of games gives the programmer the opportunity to be creative without artificial constraints. While not always successful in terms of a highly popular game, the effort is always profitable in the experience and knowledge gained.

I too, am a fan of the computer game. Many a pleasant hour has been spent answering the challenge of a good board game, searching for treasures in a dungeon, solving an intricate puzzle, etc. Also, I have spent many hours writing computer games of various types, primarily for the entertainment of friends and family. The experience gained in these activities has prompted me to formulate some ideas as to the proper approach to game design. This is certainly not intended to be a treatise on game theory, but simply an outline of some common-sense ideas to be considered in game development.

Programming is a profession. If we are going to write a program, even for the simplest of games, we should strive to be professional in our approach. There is no reason why we cannot do this and still enjoy what we are doing. Not only does it develop good habits, but it creates a sense of pride in the results.

Before you develop a game program, think about the people who will play the game. It is most likely that they will not all be programmers such as yourself. You must not fall into the trap of making assumptions about the players' understanding of the computer system. Also, do not succumb to the temptation of taking the easy way out of a programming problem. Except where hardware or software limitations dictate otherwise, take the rough road, if necessary, to make things as clear and sensible to the player as possible. Failure to do so could spell doom to the popular acceptance of your game. Remember that you want to write it once, so others will play it many times.

Let's briefly examine some of the things that contribute to the enjoyment of playing a well designed computer game. Assuming that you have an idea for a fantastic computer game which will make you famous (if not rich), what can be done to optimize the chances for success?

1. PROVIDE A "HELP" FUNCTION

For games that have a large repertoire of input commands and/or values, the new player has trouble remembering the options available for play. Such a game should provide a HELP command which can be used to produce a summary display of all options, commands, prompts, etc., which may be needed. It should be made clear to the player from the very beginning, that the HELP command is available. Also, this command should be available at any time during play, if possible.

2. PROVIDE "REFRESH" CAPABILITY

This applies primarily to board games, but would be desirable in any game where a video display was generated at the outset, and kept displayed throughout the game. No matter how well regulated the computer power-supply may be, or how clean the power coming in may be, there is always the possibility of a "glitch" causing a random distortion or "smear" of the display. Without some way of invoking a REFRESH (or RESET) command, the player is forced to tolerate the damage caused to the display, or else abort the game and start over. A proper implementation of such a command will erase the screen, re-display the board (or other display) and complete the display with all playing pieces in place as they were before invoking the command. A good example of such a command is the RESET BOARD command used in the OTHELLO game by Richard Musgrave on HUG disk #885-1068. This feature can be used effectively in conjunction with the HELP command above, in cases where the display uses the entire screen and the screen must be erased in order to display a HELP summary.

3. PROVIDE A "HEARTBEAT"

Many games exhibit long delays during play, while the computer performs some internal task such as executing a logic-tree search to determine a move. This often happens with games written in BASIC. During these delays, the anxiety that builds up in a player can be distressing.

One way to relieve this anxiety is to give the system a "heartbeat" to assure the player that something is indeed happening. A common way of doing this is to cause the computer to print a steadily growing string of characters on the screen. Dots (....) are frequently used. My own preference is to use asterisks (****) as they are more easily discernable at a glance.

This type of heartbeat has the advantage of showing the relative period of time elapsed by its length. Implementation of such a feature is via a PRINT statement placed within the loop where the computer is performing its lengthy task. An example is as follows:

```
00010 REM GAME ANALYSIS LOOP
00200 FOR N=A TO B
00800 PRINT "*";
00900 NEXT N
01000 END
```

Details, of course, depend on the game but the concept is simple and effective.

One of my favorite "heartbeats" is what I call the "spinner". It makes use of the H19 or H89 graphics to create a single-character display which appears to spin on the screen. The advantage is that the "spinner" requires only one character location on the screen and therefore, by using direct cursor addressing, can be placed anywhere without interfering with the rest of the game display.

The "spinner" is best implemented via a subroutine, which is called from a GOSUB placed in the program loop. It may, in fact, be desirable to place several GOSUB's throughout the loop to give the "spinner" the proper rotational speed. Again, this depends on the particular application. An example of the implementation of the "spinner" follows. Type it into your system and run it. This routine places the "spinner" near the lower right corner of the screen.

```
00010 DIM S$(3)
00020 S$(0)="a":S$(1)="y"
00030 S$(2)=",":S$(3)="x"
00040 E$=CHR$(27):F$=E$+"F":G$=E$+"G"
00050 PRINT E$;"E"
.
.
00200 REM GAME PROCESSING LOOP
00210 C=0  00220 FOR N=1 TO 20
.
.
00400 FOR K=1 TO 15:REM DUMMY TASK
00410 NEXT K
00420 GOSUB 1000
.
.
00500 FOR K=1 TO 15:REM DUMMY TASK
00510 NEXT K
00520 GOSUB 1000
.
.
00600 FOR K=1 TO 15:REM DUMMY TASK
00610 NEXT K
00620 GOSUB 1000
.
.
00700 NEXT N
00710 PRINT E$;"Y6h "
.
.
00900 END
01000 REM SPINNER ROUTINE
01010 PRINT E$;"j";E$;"x5";F$;E$;"Y6h";
01020 PRINT S$(C)
01030 PRINT G$;E$;"k";E$;"y5";
01040 C=C+1:IF C=4 THEN C=0
01050 RETURN
```

4. DO INPUT PRE-PROCESSING

This simply means that the game program should be written to prevent problems caused by input errors by the player. The program should process each input command, move, etc., to determine if it is as expected. If there is an error, simply loop back and request a repeat of the input, rather than allow the program to abort or get into trouble trying to work with incorrect information.

For example, if the program expects a numerical value and the player erroneously enters an alphabetical character, this should be detected. The same holds true for the reverse situation. To illustrate, study the following examples...

This input expects a positive integer number and will reject anything else, with a repeat request for input:

```
00100 S=0:LINE INPUT "RANGE ?";A$
00110 GOSUB 500:IF S=1 GOTO 100
00120 REM CODE TO PROCESS VALUE (A$)
00130 A=VAL(A$)
.
.
00500 REM NUMERIC INPUT PRE-PROCESSOR
00510 L=LEN(A$)
00520 FOR N=1 TO L
00530 K=ASC(MID$(A$,N,1))
```

```
00540 IF K<48 OR K>57 THEN S=1:GOTO 560
00550 NEXT N
00560 RETURN
```

This input expects an alphabetic command word and will reject anything else, with a repeat request for input:

```
00100 S=0:LINE INPUT "COMMAND ?";A$
00110 GOSUB 500:IF S=1 GOTO 100
00120 REM CODE TO PROCESS COMMAND (A$)
.
.
00500 REM ALPHA INPUT PRE-PROCESSOR
00510 L=LEN(A$)
00520 FOR N=1 TO L
00530 K=ASC(MID$(A$,N,1))
00540 IF K<65 OR K>122 THEN S=1:GOTO 570
00550 IF K>90 AND K<97 THEN S=1:GOTO 570
00560 NEXT N
00570 RETURN
```

Several things should be noted about the above examples. In both cases input is done using string variables. This is always safer in that the code will accept any of the keyboard characters which might be entered, correct or otherwise. Entering an alphabetic character when the input expects a numeric entry, spells disaster!

Both examples use subroutines to do input pre-processing. The same routines are then available to handle any input requirement in the game. Variations of the routines are easily implemented to handle special case input strings.

In the second example, notice that the pre-processor will allow both upper and lower case input. The command word processor would have to do likewise, but this is a recommended practice. It is entirely reasonable to expect a player to input both upper and lower case commands. Similarly, it is unreasonable to prohibit them, from the players point of view.

5. ECHO ALL INPUT

Program languages such as Microsoft BASIC, Assembler, etc., permit input entry without the use of the RETURN key to terminate the input line. This is often done when the expected input is a known number of characters. For such entries as moves on a Chess Board, Othello Board, etc., there is no need to require the player to hit RETURN after each entry. Such features should be used carefully or their convenience can become more like an annoyance.

Provision should always be made to "echo" the input back to the terminal so the player can see what has been entered. In this way, incorrect entries can be seen and the necessary correction more easily understood. Where the program is written to avoid the need for RETURN after each entry, the use of input pre-processing as discussed above, is especially important since the player does not generally have the use of the DELETE key to correct entry errors before they are "digested" by the program.

In many applications, it is possible to establish "defaults". This simply means that unless the player has a specific input in mind, the program can assume an input entry which is appropriate at that instant. To invoke the default, the player simply types one key, and the program proceeds according to a built-in procedure. Two examples follow....

Using Benton Harbor Basic:

```
09000 LINE INPUT "Do you want to move
      first? (Y/N) <Y> ";A$
09010 IF A$="N" OR A$="n" GOTO 9500
09020 A$="Y"  09030 REM PLAYER MOVES FIRST
.
.
09500 REM COMPUTER MOVES FIRST
```

Using Microsoft Basic:

```
09000 PRINT "Do you want to move
      first? (Y/N) <Y> ";:A$=INPUT$(1)
09010 IF A$="N" OR A$="n" THEN
      PRINT A$:GOTO 9500
09020 A$="Y":PRINT A$
09030 REM PLAYER MOVES FIRST
.
.
09500 REM COMPUTER MOVES FIRST
```

In the first example, the input must be terminated by a RETURN. If no character precedes this, A$ will have a null value. The program will detect this and assume the request to be "Y".

The same is true in the second example except that here the program requires only one character for input. This character can be any key, not necessarily a RETURN. All keys other than "N" and "n" will be interpreted as "Y".

Note that where the default route is taken, the variable A$ is specifically defined as "Y". This ensures the proper value for A$ in cases where the variable is used further in the program. Also, in the second example, the value is "echoed" to the terminal, since the INPUT$(1) statement does not do so. The use of the prompt "(Y/N) <Y>" tells the player that a Y or N input is expected and that anything else will be interpreted as Y.

6. WATCH THOSE SPELLING MISTAKES!

As mere human beings, we all make spelling errors. One reason why text editors are so popular is the ability to correct these mistakes easily. Still, it seems that too many spelling errors get out in public programs.

"What?", you say, "Spelling errors in purchased software??!!" Yes indeed! I have corrected many spelling errors in programs which I have purchased, HUG software included. There is really no excuse for this either. Reasonable care should be taken to check for such errors, even to the extent of having someone else check your work.

There is an expression which goes, "You only get one chance to make a first impression." I think that most of us will agree that first impressions are the ones which last longest. If we have produced a game with spelling errors, the first-time player can readily get the idea that there are other unforseen errors in store. This takes the edge off of the enjoyment of the game for many. To make matters worse, if the player is not a programmer, prospects for going into the code and fixing the mistake are grim. Take the extra time to check for spelling mistakes and fix them before they come back to haunt you.

7. DON'T OVERDO THE WISECRACKS

When heard the first time, a good joke is worthy of a good laugh. But when repeated over and over, the humor soon wears thin, often to the point of aggravation. Many games attempt to display a certain artificial intelligence from the computer by generating output statements simulating conversation with the player. Frequently, these statements take on a sarcastic tone in an attempt at teasing the player for a poor move, input error, etc.

An example seen frequently is, "You can't move there, you Dummy!", or something similar. When placed in the program so that they appear frequently, such statements lose their humor quickly and become a drag to the player. Such statements are best placed where they appear relatively infrequently during the game.

It is certainly appropriate for the program to produce "fireworks" on the screen when the computer wins, and to bow out gracefully with moderate grumblings on the screen when the human wins. I find this amusing and since it only happens once per game, it is not likely to get tiring.

By all means, avoid the use of scatology (Yes, I have seen it!) or ethnic terms in these comments. There are plenty of opportunities to be humorous without risking insult to the player.

8. WHEN THE GAME ENDS, "CLEAN UP"

Although I have listed this last, I find this to be one of my biggest gripes with games I have purchased. Often, the game will use such features as graphics, reverse video, the 25th line on my H89 terminal, etc. When ending such a game, I find I am left with a terminal that has these options set and I must generate the escape codes to restore things back to normal.

It is so simple for the program to "clean-up" this mess that I can't understand why it isn't done more often. For example, look at the following code:

```
09000 E$=CHR$(27):G$=E$+"G":L$=E$+"y1"
09010 C$=E$+"y5":V$=E$+"q"
09020 PRINT G$;L$;C$;V$
09030 END
```

This code exits from graphics, disables the 25th line, turns on the cursor and exits reverse video. It might even be desirable to do a screen erase, depending on the application. Again, the non-programmer is at your mercy here. Try to end your game by returning the terminal to its normal status.

CONCLUSION

I am sure that those of you who enjoy writing computer games, as well as playing them, will have other ideas about what makes a successful game. What I have discussed here is not really crucial to the success of a game; that is a subject which goes far deeper than we have room for here. But, these suggestions will go a long way toward putting on that final polish to make a good first impression on the player, and to help maintain a long enjoyable life for the program.

I have modified almost every game program I have ever purchased, to implement the above ideas. They have proven their worth in actual use. Moreover, many of these principles are appropriate for programs other than games. Take a professional approach to your programming. You will enjoy it more and learn it better.

EOF

HUG BUG....Digital Research's documentation manual indicates that the PRN: function of PIP is the same as LST: with TAB expansion and paging. However, PRN: is more similiar to the LPT: function. Using PRN: as LST: can cause your computer to "lock".

Sorting and Merging Sequential File Data

(WHEN TO SORT AND WHEN TO MERGE)
(HOW TO KEEP LARGE FILES
SORTED WITH A MINIMUM OF MEMORY)

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

ABSTRACT:

A short discussion of fundamentals of sorting and merging is presented. The programs referred to are MBASIC programs.

Definitions:

SORT - to put in order, alphabetize, etc.
MERGE- to combine one sorted list with another sorted list to produce a third list which is sorted and contains all the items in the first 2 lists.

I have previously listed 2 short MBASIC programs, one for using the Shell-Metzner sort, and one for merging. These are program 4 and program 17 in my "random file" article in REM issue 10. Both are for use on sequential files, such as a mailing list or an inventory list. At the time I wrote the merge program I wrote it as an intellectual exercise and until quite recently really did not know when, if ever, I would ever use it. One of the business lists I maintain is a sequential data list with patient's last name at the beginning of each record. Each record consists of numerous data fields separated from each other by a delimiter (I used "\" as the delimiter), and each record is separated from the next record by a "return" (new line character to HDOS). I started this file with 20 records. Since I desired to keep the records in ascending alphabetical order by last name, I simply created (and then appended to) the file using programs similar to other programs listed in the above article in REM issue 10. However, since I knew that this list would grow and grow (probably to about 5000 records before the list is completed over the next year), I sorted (alphabetized) the list using HUG's SORTER.ABS program, since I wished to utilize all available memory (the MBASIC interpreter takes up a LOT of memory and Bill Moss's SORTER program is very small, is in machine language, and uses the Shell Metzner algorithm which provides for an extremely fast sort -- for example, it will sort 500 records, each 80 characters long, in about 7 seconds!)

Each time 10 or so records were appended to the bottom of the list, I simply sorted the entire list (remember that this is a VERY fast SORT). This continued UNTIL I ran out of memory for the sort! At this time there were only about 300 records in the list BUT each of these records were about 128 characters long.

WHAT TO DO?

My first thought was to do what I had done with another business list, this one about 1000 records long, with each record averaging about 75 characters. Again, the first field was last name followed by first name. Experimentation with this list had shown that SORTER could sort about 500 of these records without running out of memory. So, I had written a simple MBASIC program which simply line inputted, line by line, from the list to be sorted, then outputted to 2 temporary files. One file (A) received all the records whose first character (the last name was the first field in all these records) was less than "M", and the other file (B) got the remaining records. Then, I went to the monitor (>) and ran SORTER on file A, then on file B, then simply concatenated the 2 files using PIP. The MBASIC line which output to temporary file A or temporary file B was:

```
200 IF LEFT$(X$,1)<"M" THEN PRINT #2,X$@
    ELSE PRINT #3,X$
```

I was about ready to implement this procedure with my present file which ran out of memory, when it occurred to me that sooner or later I would need 3 temporary files, then more, and more, etc., all to be concatenated after individual sorting, as my list grew larger and larger. At this point a "light flashed and a bell rang"! Since my original list was always sorted, and since I was only adding 10 or 20 records at a time, why not just sort the records to be added, then merge the 2 sorted lists? WHY NOT INDEED?

This, of course, proved to be the solution I needed. It turns out that this procedure is much faster than sorting a large list of records that is already in order (already sorted) with just a few unsorted records appended to the bottom of the list!

(vectored to page 25)

Tiny BASIC Tricks

Because of the effort to get REMark back on schedule, I did not have time to write a Tiny BASIC Tricks column in the last issue, and have not had much time for it this time. So for this column I will present for your enjoyment a Tic-Tac-Toe game for ET3400 BASIC. This game was sent to us by George I. Brown, 2428 Eck Dr., Raleigh, NC 27604.

```
10 REM TIC-TAC-TOE PROGRAM BY GEORGE BROWN. 8/80
20 PR"THIS IS A TAC-TAC-TOE PROGRAM.  THE COMPUTER WILL PRINT"
30 PR"A TIC-TAC-TOE DIAGRAM WITH ALL ITS POSITIONS NUMBERED FROM"
40 PR"1 THROUGH 9.  IT WILL THEN PRINT ITS FIRST MOVE USING X.  ON"
50 PR"THE LINE BELOW, IT WILL PRINT 0=? AND WAIT FOR YOUR MOVE."
60 PR"AFTER YOU DECIDE YOUR MOVE, YOU TYPE ITS NUMBER AND THEN"
70 PR"PRESS THE RETURN KEY.  THE COMPUTER WILL RESPOND BY PRINTING"
80 PR"ITS NEXT MOVE.  THIS PROCEDURE WILL CONTINUE UNTIL THERE IS"
90 PR"A WINNER OR A DRAW.  OK, LET'S START."
100 PR"     I     I"
110 PR"  1  I  2  I  3"
120 PR"     I     I"
130 PR"-----I-----I-----"
140 PR"     I     I"
150 PR"  4  I  5  I  6"
160 PR"     I     I"
170 PR"-----I-----I-----"
180 PR"     I     I"
190 PR"  7  I  8  I  9"
200 PR"     I     I"
210 PR
220 PR"X=5"
230 GOSUB 1420
240 IF O=1 GOTO 310
250 IF O=2 GOTO 450
260 IF O=3 GOTO 550
270 IF O=4 GOTO 720
280 IF O=7 GOTO 920
290 IF O=8 GOTO 1090
300 IF O=9 GOTO 1190
310 PR"X=3"
320 GOSUB 1420
330 IF O=7 GOTO 360
340 LET X=7
350 GOTO 1360
360 PR"X=4"
370 GOSUB 1420
380 IF O=6 GOTO 410
390 LET X=6
400 GOTO 1360
410 PR"X=8"
420 GOSUB 1420
430 IF O=2 THEN LET X=9
440 GOTO 1450
450 PR "X=7"
460 GOSUB 1420
470 IF O=3 GOTO 500
480 LET X=3
490 GOSUB 1420
500 PR"X=1"
510 GOSUB 1420
520 IF O=4 THEN LET X=9
530 IF O=9 THEN LET X=4
540 GOTO 1360
550 PR"X=9"
560 GOSUB 1420
570 IF O=1 GOTO 600
580 LET X=1
590 GOTO 1360
600 PR"X=2"
610 GOSUB 1420
620 IF O=8 GOTO 650
630 LET X=8
640 GOTO 1360
650 PR"X=4"
660 GOSUB 1420
670 IF O=6 GOTO 700
680 LET X=6
690 GOTO 1360
700 LET X=7
710 GOTO 1450
720 PR"X=3"
730 GOSUB 1420
740 IF O=7 GOTO 770
750 LET X=7
760 GOTO 1360
770 PR "X=1"
780 GOSUB 1420
790 IF O=2 THEN LET X=9
800 IF O=9 THEN LET X=2
810 GOTO 1360
820 PR"X=1"
830 GOSUB 1420
840 IF O=9 GOTO 870
850 LET X=9
860 GOTO 1360
870 PR"X=3"
880 GOSUB 1420
890 IF O=2 THEN LET X=7
900 IF O=7 THEN LET X=2
910 GOTO 1360
920 PR"X=1"
930 GOSUB 1420
940 IF O=5 GOTO 970
950 LET X=9
960 GOTO 1360
970 PR"X=8"
980 GOSUB 1420
990 IF O=2 GOTO 1020
1000 LET X=2
1010 GOTO 1360
1020 PR"X=4"
1030 GOSUB 1420
1040 IF O=6 GOTO 1070
1050 LET X=6
1060 GOTO 1360
1070 LET X=3
1080 GOTO 1450
```

```
:RUN
THIS IS A TAC-TAC-TOE PROGRAM.  THE COMPUTER WILL PRINT
A TIC-TAC-TOE DIAGRAM WITH ALL ITS POSITIONS NUMBERED FROM
1 THROUGH 9.  IT WILL THEN PRINT ITS FIRST MOVE USING X.  ON
THE LINE BELOW, IT WILL PRINT 0=? AND WAIT FOR YOUR MOVE.
AFTER YOU DECIDE YOUR MOVE, YOU TYPE ITS NUMBER AND THEN
PRESS THE RETURN KEY.  THE COMPUTER WILL RESPOND BY PRINTING
ITS NEXT MOVE.  THIS PROCEDURE WILL CONTINUE UNTIL THERE IS
A WINNER OR A DRAW.  OK, LET'S START.
     I     I
  1  I  2  I  3
     I     I
-----I-----I-----
     I     I
  4  I  5  I  6
     I     I
-----I-----I-----
     I     I
  7  I  8  I  9
     I     I

X=5
O=? 1
X=3
O=? 7
X=4
O=? 2
X=6 AND THE COMPUTER WINS!
TYPE 1 TO CONTINUE OR 0 TO STOP.
THEN PRESS THE RETURN KEY.
? 1

X=5
O=? 3
X=9
O=? 1
X=2
O=? 8
X=4
O=? 6
X=7 AND THE GAME IS A DRAW.
TYPE 1 TO CONTINUE OR 0 TO STOP.
THEN PRESS THE RETURN KEY.
? 0
```

Continuation of Listing

```
1090 PR "X=7"
1100 GOSUB 1420
1110 IF O=3 GOTO 1140
1120 LET X=3
1130 GOTO 1360
1140 PR"X=1"
1150 GOSUB 1420
1160 IF O=4 THEN LET X=9
1170 IF O=9 THEN LET X=4
1180 GOTO 1360
1190 PR"X=7"
1200 GOSUB 1420
1210 IF O=3 GOTO 1240
1220 LET X=3
1230 GOTO 1360
1240 PR"X=6"
1250 GOSUB 1420
1260 IF O=4 GOTO 1290
1270 LET X=4
1280 GOTO 1360
1290 PR"X=2"
1300 GOSUB 1420
1310 IF O=8 GOTO 1340
1320 LET X=8
1330 GOTO 1360
1340 LET X=1
1350 GOTO 1450
1360 PR"X=";X;" AND THE COMPUTER WINS!"
1370 PR"TYPE 1 TO CONTINUE OR 0 TO STOP."
1380 PR"THEN PRESS THE RETURN KEY."
1390 INPUT A
1400 IF A=0 THEN END
1410 GOTO 210
1420 PR"O=";
1430 INPUT O
1440 RETURN
1450 PR"X=";X;" AND THE GAME IS A DRAW."
1460 GOTO 1370
```

EOF

(vectored from page 23)

Since I always need "hard copy" of any new file just after sorting the "records to be appended", it was simple enough to add to the merge program (program 17) so that everytime we "PRINT #3," we also "PRINT #4," (a 4th file was opened with output to "LP:" - hence, "PRINT #4," - remember to tell MBASIC you will need 4 files when you load MBASIC as your manual instructs). This modification of the merge program provided me with "hard copy" of the new list, at the SAME time the new merged list was being created and output to disk (PRINT #3,).

WORKS GREAT and I only wish I had thought of this procedure before.

So, carrying the above a step further, it became obvious that you could create enormous lists (up to your disks' capacities) with very little memory overhead, and you could do everything in MBASIC. You simply could create the original list with only 10 to 20 records, and sort it. Then, instead of appending to the list and sorting the whole thing, you would create a temporary file of the NEW unsorted records, sort them and then merge them with the original previously sorted list, etc., etc..

The concept is so simple and logical, but SORTER.ABS is SO fast that the obvious (merging) had become "unobvious" to me!

EOF

BUGGIN' HUG

MAN'S BEST FRIEND (BASICally)

Dear HUG,

I seem to have a minor problem that you might be able to shed some light on. It all started ten months ago when I got my H-88 computer. Since that time my wife complains that I'm REPEATedly LOCKed in my workshop with my new found friend. She also complains that all I ever do is eat, sleep, and talk computer. IF this is true THEN I could END up in the doghouse. I am hoping a word of encouragement from you to my wife might RESTORE her feelings for me and my toy and convince her that I don't always have my mind PROGRAMmed for that computer.

I don't know how I accomplished it, but, I was able to convince her to get me a DISK drive for Christmas so I could make my unit into an H-89. This was an OUTstanding addition, and gave me a whole new toy to play with. Since that time I have spent every minute I could get away with CHAINed to the chair in front of it. When I RETURN HOME from work I PEEK around the corner to make sure the coast is CLEAR, and then RUN to my workshop to ESCAPE from the pressures of work. When my wife POKEs her head into my room, she always has some derogatory REMark to make in an attempt to SHIFT my attention away from my H-89. Now I'm SCRATCHing my head wondering if I should RESET my way of thinking in order to SAVE our marriage. She has threatened to BUILD a barrier in front of my new companion or REPLACE it with an old BOOT. I feel it is time to take drastic STEPs and LET her have her way. She has LISTed her grievances and I'll attempt to CLEAR up as many as possible by FREEing some of my time to be with her. A couple of minutes set aside for her should be sufficient (I hope).

Possibly all is not lost, because I caught her ENTERing the workshop to play "Adventure". She had CLOSEd the door so I wouldn't see, but I stopped to watch anyway.

So, what I need is a word from the wise in order to DELETE her way of thinking so I can RETURN once again to a very relaxing hobby. I still can't understand why she insists that all I ever think about is my computer!

```
*BYE
SURE?Y
```

Jim Doherty
18357 Citrus Edge
Azusa, CA 91702

Jim....

It's obvious that you should PUT her in some "TWISTY LITTLE MAZE" and tell her you will be back to GET her later!

Uncle HUG

Dear HUG,

I recently modified my H-14 and find the mod to be such a convenience to me that I thought I might pass it on to you.

The modification was prompted by a chronic problem of mine (and I suspect a lot of other folks), namely, that of sitting in front of the CRT and wondering why nothing is happening. To my chagrin, the problem is usually the result of having left the H-14's ON LINE switch in the "out" position; having earlier set it this way to program the printer.

Quite simply, the modification consists of installing an SPST, normally closed push button switch (Radio Shack #275-1548) in the line from the ON LINE latching switch to pins 8 and 15 of the input buffers U109 and U110 respectively. Thus, when the new switch is in place, just leave the ON LINE switch "IN". To use the FEED FWD or FEED REV, just press your choice then press (bringing the above mentioned pins to +5 Volts, enabling the buffers) and release the mod button. Once enabled, the FEED buttons will function until you release them. Likewise for the FORM FEED (of course the FORM FEED waits until that switch is released). The function of the ON LINE switch remains unaltered. The ON LINE switch must be "IN" for the new mod button to function.

(vectored to page 28)

Dear HUG,

Occasionally it is necessary to know the binary equivalent of a decimal number. However, unless you have a conversion chart handy, then it can be a tedious procedure to deduce the proper sequence of "1s and Øs". Since I could not find such a chart and did not wish to do a conversion manually on each number that I desired, I wrote two simple programs to handle the chore for me.

The larger of the two listings runs the slowest. About six times slower actually. The main reason being the use of the FOR/NEXT loops. Also, it uses exponential notation as a part of the algorithm.

The shorter program uses a division routine on each number and determines the correct bit sequence depending on the remainder after dividing by two. The end result is shuffled around in string format before the final conversion is printed.

Both programs , of course, give the same results, one simply runs faster than the other. Both are written to output to a printer so that a listing can be generated for future reference. The largest number that will be converted is 255 (decimal).

Ray Massa
125 Aspen
Birmingham, MI 48ØØ9

```
1Ø REM DECIMAL TO BINARY CONVERSION
2Ø REM Written by Ray Massa
3Ø OPEN"O",#1,"LP:"
4Ø PRINT#1,"128 ";"64  ";"32  ";"16  ";"8   ";"4   ";"2   ";"1"
5Ø PRINT#1,"================================="
6Ø FOR M=Ø TO 255
7Ø Z=M
8Ø FOR X=8 TO Ø STEP -1
9Ø IF 2^X<=Z THEN 12Ø
1ØØ NEXT
11Ø REM
12Ø IF X=Ø THEN A=1
13Ø IF X=1 THEN B=1:Z=Z-2:GOTO 8Ø
14Ø IF X=2 THEN C=1:Z=Z-4:GOTO 8Ø
15Ø IF X=3 THEN D=1:Z=Z-8:GOTO 8Ø
16Ø IF X=4 THEN E=1:Z=Z-16:GOTO 8Ø
17Ø IF X=5 THEN F=1:Z=Z-32:GOTO 8Ø
18Ø IF X=6 THEN G=1:Z=Z-64:GOTO 8Ø
19Ø IF X=7 THEN H=1:Z=Z-128:GOTO 8Ø
2ØØ REM
21Ø PRINT#1,
22Ø PRINT#1,H;G;F;E;D;C;B;A;"  ****  ";M
23Ø A=Ø:B=Ø:C=Ø:D=Ø:E=Ø:F=Ø:G=Ø:H=Ø:
24Ø NEXT M
25Ø CLOSE#1
26Ø END
```

```
1Ø REM DECIMAL TO BINARY CONVERSION
2Ø REM Written by Ray Massa
3Ø OPEN "O",#1,"LP:"
4Ø FOR M=Ø TO 255
5Ø Z=M
55 IF Z=Ø THEN BI$="ØØØØØØØØ"
6Ø T=Z/2
7Ø IF T<1 THEN 11Ø
8Ø IF T=INT(T) THEN BINARY$="Ø"+BI$ ELSE BI$="1"+BI$
9Ø Z=INT(T)
1ØØ GOTO 6Ø
11Ø BI$="ØØØØØØØ1"+BI$
12Ø BI$=RIGHT$(BI$,8)
13Ø PRINT#1,BI$;"   ***   ";M
14Ø BI$=""
15Ø NEXT M
16Ø CLOSE#1
17Ø PRINT:PRINT
18Ø END
```

(vectored from page 26)

There is room for the new switch immediately above the ON LINE latching switch and in the same horizontal plane as the POWER and HIGH TEMP LEDs. The switch is interposed into the red lead of the eight lead ribbon under the insulating shield.

Sincerely,

William R. Rousseau, M.D.
1388 Highland Ave.
Salem, OH 44460

Editor's Note: If you perform modifications to your Heath equipment, these modifications should be removed if your unit requires service by the factory or Heath centers. The technicians working on your product will have no way of knowing what the results hardware changes will cause. And, in many cases, they are unfamiliar with the specifics of the modification and what it is installed for! (They really appreciate your co-operation in this regard.)

Dear HUG,

The back cover of REMark Issue 12, Nov. 1980, contains a Sunrise - Sunset program. Lines 50 and 60 are listed as:

```
00050 L1=(L1-INT(L1))/60+L1
00060 L2=(L2-INT(L2))/60+L2
```

But they should read:

```
00050 L1=(L1-INT(L1))/0.6+INT(L1)
00060 L2=(L2-INT(L2))/0.6+INT(L2)
```

These changes will correctly convert degrees and minutes to decimal degrees.

The answer for MONTH 3, DAY 16, LATITUDE 37.57 and LONGITUDE 91.46 will then be:

```
SUNRISE GMT 12 hours 21 minutes
SUNSET GMT 24 hours 10 minutes
GIVING 11 hours and 48 minutes of
daylight.
```

Sincerely,

M.G. Keeney
MICHIGAN STATE UNIVERSITY
Computer Science Department
Computer Center
East Lansing, MI 48824

Non-Heath Products

Henry Fale, of PORTZABEE COMPUTER SERVICES, publishes a monthly newsletter known as H8SCOOP. Henry's H8SCOOP is very current on Heath Equipment and associated gear that, for the most part, is fully compatible with your Heath computer. If you would like more details on subscription rates for H8SCOOP, Henry can be contacted at PORTZABEE COMPUTER SERVICES; 2981 S. 7th St.; Sheboygan, Wisconsin 53081.

Minit-Forms Paperware

VIDEO LAYOUT SHEET-used to layout graphics on the H19, H89, H88, Z19 & Z89 terminals and computers.
Size 11" x 14-1/4". Model # MF-001

GENERAL PURPOSE PROGRAMMING SHEETS-used for BASIC and assembly language programming, back sides carries info on the Heath and Zenith terminal escape sequences plus area for user defined variables.
Size 8-1/2" x 11". Model # MF-002.

FLOWCHARTING WORKSHEET-this sheet carries a 5 X 9 array of programming positions. Each position has 5 general programming symbols plus connecting lines both on center and between each position, both vertical and horizontal allow for easy flowcharting without the use of templates.
Size 11" x 14-1/4". Model # MF-003.

Two NEW forms now available are a PRINTER LAYOUT SHEET-132 columns wide by 60 lines high (MF-004) and a OVERLAY made of plastic for use in plotting your forms for printing (MF-005). A grease pencil can be used on the new OVERLAY.

All forms are printed in fade-out blue on #60 white offset stock, all are padded in 50's with a 60 point chipboard backing and standard three hole drilled for ring binders.

The forms are currently offered at $2.50 per pad, (overlay MF-005 $2.50 each when ordered with other forms else $4.00 each) mix or match, with minimum of 5 pads, which includes shipping (within the continental U.S.).

Order from: Minit-Man Printing
211 E. Allegan Street
Otsego, MI 49078
Phone orders accepted: 616-694-9141
VISA, MC accepted.

Minit-Forms are also available from most Heathkit Electronic Centers.

HUGBB Via MicroNET

PART I: MicroNET for Beginners

1) What is MicroNET?
MicroNET is a computer service provided by the Personal Computing Division of CompuServe Incorporated in Columbus, Ohio. By connecting your personal computer or terminal to the MicroNET system you are able to "talk" to a large computer. This service opens up some "neat" features not available from a small, single computer.

The MicroNET (MNET) system allows you to write and edit programs and data files and store them on the system. You are able to run your own programs and able to access data files and programs in the MNET library. Another feature is the ability to communicate with other users or the MicroNET staff. (For more detail on the options and features of the MicroNET system see the MicroNET User's Guide.)

2) Who can use MicroNET?
Valid MicroNET members have been issued a MicroNET number called a "User ID;" which enables them to access the system. This ID number allows the member to "log on" and "log off" the MNET system at "legal" hours determined by a priority level from their User ID.

3) How do I become a member of the MicroNET system?
If you are interested in becoming a member of the MicroNET computer system, you will need to fill out a "Information Service Request and Agreement" and mail to CompuServe. These forms are available at most HEATH stores, here at Heath Users' Group or write to:

> Information Service Division
> CompuServe Incorporated
> 5000 Arlington Centre Blvd.
> Columbus, Ohio 43220

The cost to become a member is $9.00. The charge to operate on the MNET system is approximately $5.00 an hour. (This charge will vary because some of the features have no charge to the member.)

Do NOT send any money with your "application" as MNET will bill you direct or bill you by credit card which you will provide on this form. In return you will receive a valid User ID and MicroNET User Guide which will enable you to access the MicroNET system.

4) If I become a member what do I need to get on or "talk" to MicroNET?
Your computer will "talk" to the MNET system by telephone . . . so be sure your computer is set up close to a standard telephone receiver. You will need a MODEM, which is the "interface" between your computer and MNET. The modem is simply the "holder" of the telephone receiver which controls the receiving and sending of messages. (See your computers owners manual for instructions on where and how to connect the modem to your computer.)

At this point the hardware interface is complete. . . now it would be nice to have a software interface that would allow us to receive and send complete files from our computer and in particular from our disk storage.

There are software interfaces on the market, HUG has the MODEM COMMUNICATIONS SYSTEM (MCS) while Softstuff (tm) has the COMPUTERIZED PHONE SYSTEM (CPS). I currently use CPS to do the monitoring so any mention of software interfacing from this point on will pertain to CPS. (In future issues I will explain MCS if I feel there is a need . . . both modem systems are similiar in their functions and have fairly accurate documentation.)

Now you are ready to "talk" to the MNET system. . . except for a telephone number in your area, which MicroNET provides or call 1-800-848-8990.

5) What is the HUG Bulletin Board?
The HUG Bulletin Board (HUGBB) is a service of Heath Users' Group for the members of HUG who are also members of the MNET system. The HUGBB has its own "operating system" for receiving and sending messages between HUG members.

This allows HUG members to "meet" with fellow HUGGIES in conversation that is centered around our general interests.

The members of the HUGBB are able to access the HUGBB Data Base. This file explains the programs (games or utilities) that are available from our DIRectory. Any new information or frequently ask questions will be posted on the Data Base. These are the general uses of the HUGBB.

6) What does the HUG Bulletin Board have to do with MicroNET?
The HUGBB is only a very small part of the MicroNET system and to the HUGBB member a very important part. However, we must remember that MicroNET is MicroNET and HUG is not able to advise or support the MNET system.

The HUGBB requires the facilities of MNET to function and for that we are grateful (no comment "blown" users). We have to use their COPY and transfer functions to exist as a service to our members.

7) What is "TYMNET" and how does it relate to MicroNET?
TYMNET is a telephone service to more populated areas that MicroNET users may use so as to call MNET on a local number saving on long distance charges. There is an addition fee on your MNET account of $2.ØØ per hour for the use of TYMNET. (MicroNET also has a list of these phone numbers.)

At this point you should have a basic picture as to what MicroNET and the HUG Bulletin Board are and what they are intented to do. Now for those of you who have a computer or terminal, a modem, a valid User ID, and a standard telephone receiver you are now ready to access MNET and the HUGBB.

PART II: TALKING TO MNET AND HUGBB

Before you can actually access the MicroNET system you must set your modem to properly receive the "signal" from MNET. The modem must be set to "full duplex" and "originate mode". (See the modem instructions to set same.)

You should now be ready to access the MNET system . . .so

a) Dial the MicroNET (or TYMNET) telephone number in your area. (We will assume for our purposes that you are calling a TYMNET phone number.)

b) When you hear a continuous, high-pitched tone, properly position the telephone handset into the acoustic coupler on the modem. (If you do not hear the high-pitched tone, redial until you do.)

At this time the software interface (CPS) and TYMNET take command. It will ask you to identify your terminal . . enter "A" and no carriage return. You will then be asked to LOGIN so . . type "CPS" and a carriage return, then it will ask for a password enter "WELCOME".

You are now at the point of actually logging on MNET. The system will pause slightly after entering the password "WELCOME" after which a " C" will be displayed on the screen and then you will be asked to enter your User ID and then your Password. At this time the system will identify itself and then display the MNET prompt which is "OK". You are now on MicroNET and if you prefer you may proceed to "play" with the MNET.

To get on the HUGBB from the MicroNET prompt, enter "IRUN HUG[7ØØØØ,21] and then you will enter the HUG Bulletin Board. Now refer to REMark issue #13 on page 19 for a sample session on the HUGBB.

Even though you are running on the HUGBB, you are not a member of the HUGBB and you therefore are not able to access the <X>Data Base or the <V>Int Log. To be added to the HUGBB, <L>eave a message to "SYSOP" asking to be added . . . include your HUG membership number so as we can verify your status. Then the next time maintenance is done you will be recognized by the HUGBB system and be able to access the areas for HUG members only.

When you have concluded your monitoring of MNET and the HUGBB be sure you <T>erminate from the HUGBB and then enter "BYE" at the MNET prompt when you are ready to get off the system. CPS also has a logoff procedure which is accomplished by pressing the white (or gray) key at the top of your keyboard.

We have taken you from an introduction of the MicroNET system through to the HUG Bulletin Board in this little article. If you have problems "getting on" the HUGBB, try following through the procedure again. If you still have problems give me a call and I will try to see what is the problem. Once you are on the HUGBB and you have questions about it, first "play" with it for a while and I am sure you will find the answer yourself (the bulletin board is not too difficult to follow through).

This article should give you a fairly good idea what MicroNET, TYMNET, and the HUGBB are and how you can use them as a member of the Heath Users' Group.

PART III: NOTE to all HUGBB Users

The HUGBB is definitely having some problems . . . The HUG Bulletin Board may be changing within the near future possibly even by the time of the release of this issue of REMark. We are hoping for an easy transistion period so as not to cause any "discomforts" to you as the user. The change-over is to be fairly simple with the user not being too aware of the change . . . However, many of the functions may require some experimentation on your part as the user.

SYSOP <TLJ>

Local HUG News

The Pacific Northwest Heath Users' Group (PNHUG) has published their first newsletter. They hope to provide their newsletter from four to six times a year depending on the information provided by their membership. The first attempt looks good. The newsletter indicated alternate meeting locations with odd months scheduled for the Southcenter Heath Electronics Center and even months scheduled for the Seattle Heath Electronics Center. These HEC stores can be contacted for further information.

NEW CLUB FORMING

Kenny Adcock is forming a new Users' Group in Riverside California. Anyone interested can reach Kenny by writing him at 5705 Via Sotelo; Riverside, California 92506 or calling (714)-683-1143 (leave message).

OVER THE BIG WATERS....

FHUG (the Frankfurt Heath Users' Group) of Frankfurt, Germany has been formed. Anyone interested in becoming a member is requested to contact Carl Lovett by writing American Consulate General FRDCO; APO New York, New York 09757; or calling 566187.

HEC STARTING NEW GROUP

The Heath Electronics Center located at 5285 Roswell Road in Atlanta, Georgia 30342 is starting to form a local group to support Heath systems in their area. Interested individuals or groups may contact the center for additional information. Their phone number is (404)-252-4341.

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

CUT ALONG THIS LINE

HUG MEMBERSHIP RENEWAL FORM

When was the last time you renewed?

Check your ID card for your expiration date.

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

Name ____________________

Address ____________________

City-State ____________________

Zip ____________________

REMEMBER — ENCLOSE CHECK OR MONEY ORDER

CHECK THE APPROPRIATE BOX AND RETURN TO HUG

NEW MEMBERSHIP FEE IS:

RENEWAL RATES			
US DOMESTIC	$15 ☐		$18 ☐
CANADA	$17 ☐	US FUNDS	$20 ☐
INTERNAT'L*	$22 ☐	US FUNDS	$28 ☐

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

HUG Posters

Dress up your computer room or the kids room with these fine color posters developed for HUG by Ray Massa. Each poster measures a large 24 inches by 18 inches and is protected by an individual mailing tube to prevent damage while in transit to your selected wall!

H8 POWER is the first poster and was featured as the front cover for Issue #11 of REMark. The HUG part number for this item is 885-17.

H89 ROCK was also shown as the rear cover on Issue #11 of REMark and is available as your second choice by ordering HUG part number 885-18.

H8 COLOR GRAPHICS is featured as the third poster you can choose from. This fantastic piece of art was chosen as the front cover for Issue #12 of REMark. This particular poster is available as HUG part number 885-19.

Each of the above posters may be ordered directly from the Heath Users' Group or you may purchase them from your local Heath Electronics Center. If your local store is out, ask them to order for you. These posters are available for only $2.95 each.

Making Waves in Tiny BASIC

This TINY BASIC program is used to demonstrate the geometric mathmatics required to construct a parabola (lines 10 to 90) followed by a reverse computation (lines 100 to 200) to form a continuous sine-wave pattern on your terminal. (Anyboby for Football??!!!)

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

Heath
Users'
Group
Hilltop Road
St. Joseph MI 49085

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

POSTMASTER: If undeliverable, please do not return.

885-2015