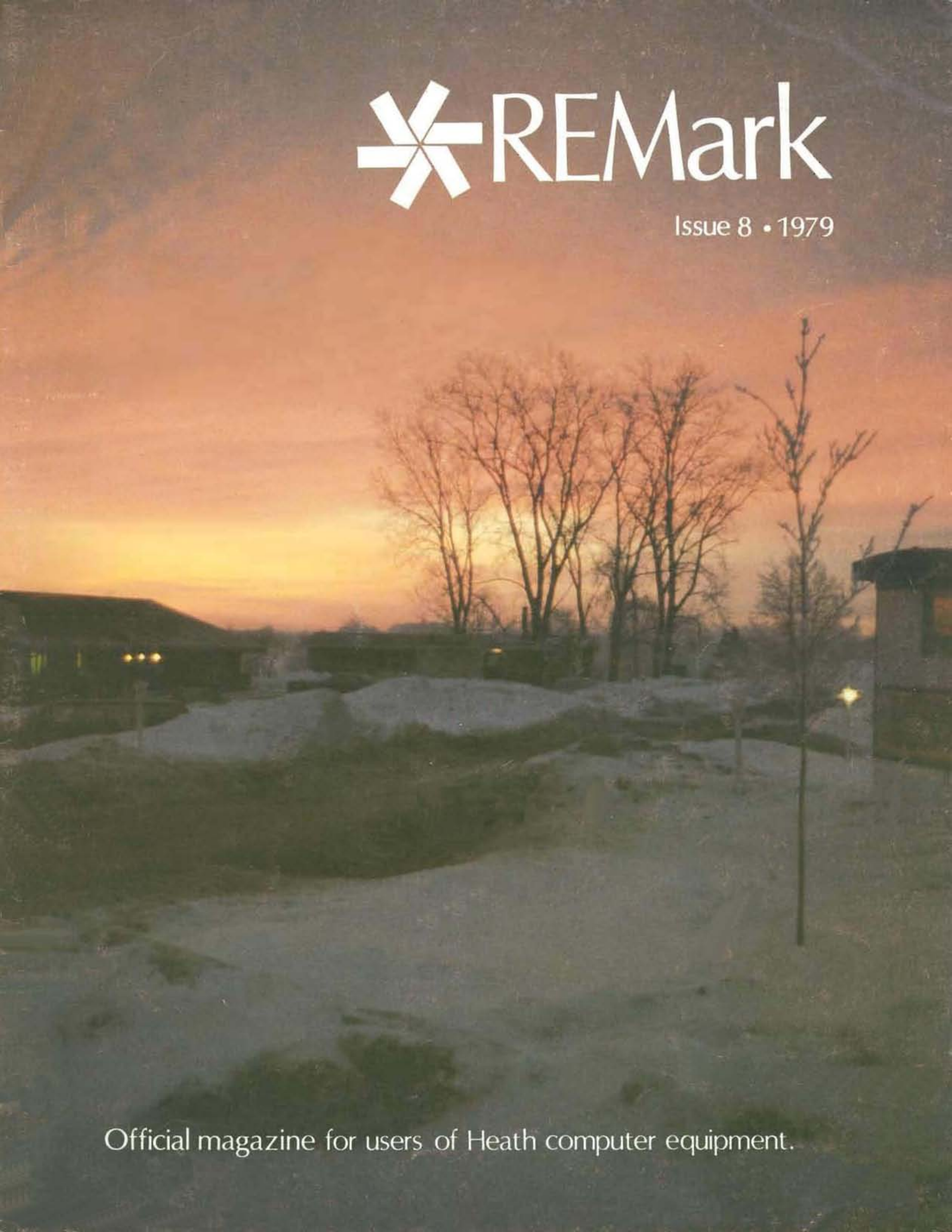
REMark

Issue 8 • 1979



Official magazine for users of Heath computer equipment.

COVER SHOT

It may be pretty, but it isn't sail boats, bright sunshine and Bikinis!

on the stack

>CAT

NOTICE CANADIAN MEMBERS

To avoid excessive shipping costs and to provide more expedient delivery to Canadian members, you may now place your orders for HUG software directly with Heath Canada at 1480 Dundas East Highway, Mississauga, Ontario Canada L4X 2R7.

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

	U.S. Domestic	Canada & Mexico	International
Initial	$14	$16	$24
Renewal	$11	$13	$18

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

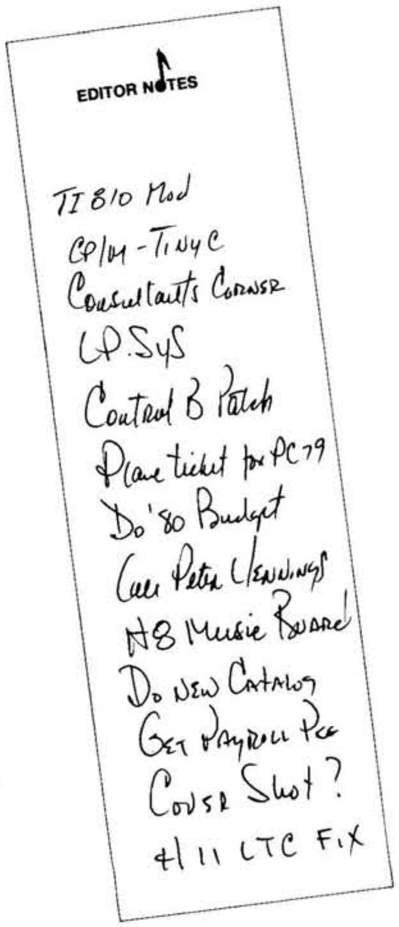
Back issues are available at $2.50 plus 10% handling and shipping. Request for magazines mailed to foreign countries should specify mailing method and add the appropriate cost.

Send payment to:

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

HUG Manager and Editor . . . . Jim Blake
Graphics . . . . .Ron Hungerford
HUG Secretary . . . . .Susan Gilfoyle
Software Developer . . . . .Gerry Kabelman

REMark



EDITOR NOTES

TI 810 Mod
CP/M - Tiny C
Consultant's Corner
CP.SYS
Control B Patch
Plane ticket for PC 79
Do '80 Budget
Call Peter Jennings
H8 Music Board
Do New Catalog
Get Payroll Pcc
Cover Shot?
H11 LTC Fix

" Ladies and Gentlemen... we have just learned that the POPE has been delayed in New York. Therefore, we expect to land in Philadelphia on schedule." That's how our trip to PC '79 began... and it got more interesting as the days went by . For instance, my hotel room faced the convention center where the POPE spoke to 12000 Priests and Nuns. And I couldn't make a move without the secret service and Philly's SWAT team peering through my window from atop the adjoining buildings. Well it was interesting... and so was PC '79. There were fewer exhibitors than past years and the crowd was lighter, but it made it easier to get around the hall and really see what was there. Microbyte was showing the new sixteen bit computers (8088 and 8086).. There was a couple hard disks shown and plenty of software.

Tiny c

HUG now has a copy of TINY C. Quoting from the owners manual, "TINY C is a versatile, expressive general purpose programing language which offers economy of expression, modern control flow and data structures, and a rich set of operators. ...TINY C is concise. It is a stripped down version of 'C'. TINY C is a structured programming language which has if-then-else, while loops, functions, global and local variables, and character and interger data types, pointer, and arrays." tiny-c (as it is actually written) is available for both HDOS and the HT-11 system from tiny c associates PO Box 269 Holmdel, New Jersey 07733.

HUG MEETING

Some questions that came up during the users' meeting were; When will we get more disk storage for the H8/H89-H11 and what will it be? ANSWER :" Heath recognizes that more storage is desirable and is actively seeking solutions to the problem commensurate with modern day technology." In other words DSDD and Hard disk are a good possiblity sometime next year." Since returning from Philadelphia, I've heard some talk of three drives for HDOS. If you want to vote on the subject, send me a note, quick!... and I'll let them know how you feel about it. QUESTION: Will there be a Z-80 CPU board (or 16 bit conversions) for the H 8? ANSWER: "NO." Next question: "I want an H 89, but need two drives." ANSWER: No problem. Buy an H 17. It will plug right into the H 89 controller card. QUESTION: "What is that white thing in your ear?" ANSWER: I got off the plane so fast that part of the ear phones stuck in my ear. I wondered why the stew gave me that funny look! QUESTION: "I have an old H 17 controller card that will not work with CP/M or Microsoft. How do I get a new one?" ANSWER: If you have one of the original controller cards, contact Heath Technical or your local store for details. The board will be exchanged. QUESTION: "When will you have CP/M programs in the library?" ANSWER: Any minute. We have physically down loaded the CP/M library to 5" diskettes and are in the process of making any necessary changes so they will work with the H 8 or H 89. Then as soon as the legal arrangements are made, the programs will be available. We will probably let you know by separate mailing.

NEW LOOK

Hope you like the new look of the magazine. This method may not be as fancy, but it goes together in about one-third of the time which allows us to be more current.

Look in your mail box. Should find a new software catalog . And the next issue of REMark will be there the first week in February. Happy holidays from Sue, Gerry and I.

:JB:

Organization! Confusion with a Structured Definition

Gene Bellinger
4929 Red Fox Dr.
Annandale, Va. 22003

It was intended that this article develop a modularized top down structured program to renumber BASIC programs. This was to be offered as a proof of the method proposed in Part I of this article. I say was rather than is, because the following offers some further thoughts on the previously proposed structures rather than the intentioned BASIC program.

This further examination appears essential as a result of some additional understanding which has evolved since completion of Part I of this article. At this time it seems the concepts are destined to undergo continuous refinement as understanding develops.

The basic premise was, and still is I suppose, to develop a means and method of some assistance in the development of logically correct BASIC programs. The utility of the assistance rests on the quality of the proposed concepts, and as such, the concepts must avail themselves to improvement. Thus what follows is a set of propositions intended to improve the utility of the initial concepts.

Proposition I

The REPEAT UNTIL structure is considered more fundamental than the DO WHILE structure.

The original statement of the allowable structural, i.e.

1. Sequence-of-statements
2. IF-THEN-ELSE
3. DO WHILE
4. REPEAT UNTIL

was intended, although not specifically stated, to be in an ascending order of complexity. The correct ordering of these structures appears to be:

1. sequence-of-statements
2. IF-THEN-ELSE
3. REPEAT UNTIL
4. DO WHILE

This determination resulted from the realization that the DO WHILE structure is a composition of the IF-THEN-ELSE and REPEAT UNTIL structures. That is:

```
DO WHILE(conditional-statement)
sequence-of-statements;
END;
```

is equivalent to:

```
IF conditional-statement THEN
   REPEAT
   sequence-of-statements;
   UNTIL(NOT conditional-statement);
```

The equivalence of these two descriptions shows that the DO WHILE is not a necessary structure, but its convenience makes it quite desirable.

Proposition II

The CASE structure is considered to provide a negative utility.

This may appear to be a rather rash statement in light of the fact that both PASCAL, and its newest offspring ADA, have implemented the CASE structure based on its determined utility. The justification of Proposition II is based on several points of developed understanding, as follows:

The previously proposed CASE structure defining the expression to evaluate as an integer was a rather restrictive definition of the structure. A definition of greater utility is:

```
CASE expression OF
expression-1 :
    sequence-of-statements-1;
expression-2 :
    sequence-of-statements-2;
expression-3 :
    sequence-of-statements-3;
ELSE sequence-of-statements-4;
END CASE;
```

This structural definition would implement with the expanded IF-THEN-ELSE structure as:

```
    IF expression = expression-1
       THEN sequence-of-statements-1;
    ELSEIF expression = expression-2
       THEN sequence-of-statements-2;
    ELSEIF expression = expression-3
       THEN sequence-of-statements-3;
    ELSE
       sequence-of-statements-4;
    END IF;
```

The CASE structure is generally preferred over its IF-THEN-ELSE equivalent due to the simplicity of statement.

In Part I of this article the traditional IF-THEN-ELSE structure was provided with an expanded definition based on the added utility it provided in the specification of a program PDL (i.e. Program Design Language specification). If the redefinition was justifiable based on the added utility, might the IF-THEN-ELSE structure lend itself to still another redefinition? The following is proposed:

The traditional IF-THEN-ELSE is rewritten as:

```
    IF conditional-statement
    THEN sequence-of-statements-1;
    ELSE sequence-of-statements-2;
    END IF;
```

Now the THEN is replaced with EQUALS true : to produce:

```
    IF conditional-statement EQUALS
    true : sequence-of-statements-1;
    ELSE sequence-of-statements-2;
    END IF;
```

Next, as the conditional--statement is a specific type expression and true is the evaluation result of a specific type expression, a transformation from specific to general results in:

```
    IF expression EQUALS
    expression-1 :
        sequence-of-statements-1;
    ELSE sequence-of-statements-2;
    END IF;
```

Now the replication of expressions is allowed, i.e.

```
    IF expression EQUALS
    expression-1 :
        sequence-of-statements-1;
    expression-2 :
        sequence-of-statements-2;
         .
         .
         .
    expression-n :
        sequence-of-statements-n;
    ELSE sequence-of-statements;
    END IF;
```

Thus we have an IF structure with a definition such that:

1. All the utility of the CASE structure is provided.
2. The traditional IF-THEN-ELSE structure is maintained as a special form of the IF structure.
3. The utility of the ELSEIF construct is maintained and emerges as a byproduct of the definition.

Statement 3 may not be obvious, but consider the following example:

```
    IF true EQUALS
    conditional-statement-1 :
        sequence-of-statements-1;
         .
         .
         .
    conditional-statement-n :
        sequence-of-statements-n;
    ELSE sequence-of-statements;
    END IF;
```

This is the expanded IF-THEN-ELSE structure previously defined with the ELSEIF construct.

The statement that the CASE structure provides a negative utility is now seen to have been based on the idea that the CASE structure need not exist as a separate structure, as its functional utility may be provided for elsewhere.

Note that nothing has been lost, as the previously defined forms of the IF structure, i.e. IF-THEN-ELSEIF-ELSE, may still be used as desired. The previously defined forms are simply specific forms of the new and more general IF structure definition.

Proposition III

The FOR-NEXT loop provides a utility which dictates its inclusion as an allowable structure.

The FOR-NEXT structure is defined as:

```
    FOR index := E1 TO E2 STEP E3
    sequence-of-statements;
    NEXT index;
```

where E1, E2, and E3 represent expressions.

The FOR-NEXT structure allows the repeated execution of a sequence of statements with an index automatically incremented from E1 to E2 by E3. The FOR-NEXT structure is defined in terms of simpler structures, but one must be careful as there exists two distinct major definitions. The structure may be defined in a manner which dictates the sequence of statements be executed at least once, or it may be defined in a manner which allows the sequence of statements to be executed zero (Ø) times depending on the values of E1, E2, and E3.

The two alternate definitions are as follows:

```
    index := E1;
       REPEAT
       sequence-of-statements;
       UNTIL( SGN(E3)*index > E2 );
```

where SGN represents the SIGN function, and the sequence of statements must be executed at least once.

```
    index := E1;
       DO WHILE( SGN(E3)*index <= E2 );
       sequence-of-statements;
       index := index + E3;
       END;
```

or alternatively:

```
    index := E1;
       IF (SGN(E3)*index <= E2) EQUALS
       true:
          REPEAT
          sequence-of-statements;
          index := index + E3;
          UNTIL( SGN(E3)*index > E2 );
       END IF;
```

and the sequence of statements may be executed zero times.

An additional problem exists with the FOR-NEXT structure because E1, E2, and E3 represent expressions. The specific point in the definition of the structure where these expressions are evaluated becomes a very important consideration. The more standard definition or interpretation of the point of evaluation of these expressions is as follows:

```
    index := evaluation-of-E1;
    v2 := evaluation-of-E2;
    v3 := evaluation-of-E3;
       REPEAT
       sequence-of-statements;
       index := index + v3;
       UNTIL( SGN(v3)*index > v2 );
```

and as you can see there are many possible variations of this form that may be defined, or is more often the case, assumed. Even knowing this definition or interpretation, can lead to unforseen problems. Consider the following example:

```
    A := 0;
       FOR A := A+1 TO A+1
       PRINT A
       NEXT A;
```

which assumes the previous definition and as a result actually executes the PRINT A statements twice rather than once as might be assumed on first observation.

The next question, most likely, is concerning the expansion to the FOR-WHILE-NEXT structure! Thus:

```
    FOR index := E1 TO E2 WHILE
    (conditional-statement)
    sequence-of-statements;
    NEXT index;
```

which is quite easily implemented as:

```
    IF conditional-statement EQUALS
    true:
       index := E1;
       v2 := E2;
          REPEAT
          sequence-of-statements;
          index := index + v3;
          UNTIL( (index > v2) OR
             (NOT conditional-statement) );
    END IF;
```

about which no more shall be said least someone ask about a FOR-UNTIL-NEXT or even a FOR-WHILE-UNTIL-NEXT structure.

Summary

This article has attempted to provide some additional understanding with regard to the basic concept of structures. The intended result being the establishment of a new ordered list of allowable structures, i.e.

1. sequence-of-statements
2. IF
3. REPEAT UNTIL
4. DO WHILE
5. FOR NEXT

With some luck, the next article in this series may actually get to the development of a Modularized Top Down Structured Program!

EOF

REQUIRED PATCHES FOR EXTENDED BENTON HARBOR BASIC

UPGRADING	10.05.00
TO	10.05.01

NOTES: This patch corrects the problem of the CONTROL-B routine also setting the CONTROL-O flag.

```
-----------------------------------------------------------------
076.135   303   362   111
111.362   072   214   040   346   040   310   052   271
111.372   040   303   140   076
112.323   061
113.266   061
-----------------------------------------------------------------
```

STRING SORT

Bob Stillman
9709 Hillshire Dr.
Richmond Il. 60071

Presented here is a sort program that may be of some interest to other readers. This program was designed to sort on entry of string data which allows two things to 'HAPPEN' at the same time (almost) with the result of increased speed.

Other sort programs we have seen, wait until the complete 'FILE' has been entered, then go through a sort routine. This works, but does seem to take quite a bit of time. This program will keep the file in order as it is entered with the sort time being part of the data entry and almost unnoticeable.

The entry 'END' will exit you from the routine and send you to a print-out that can, of course, be changed to suit your needs. 'X' will contain the total number of entries. The program can be 'CALLED' at any time to sort a new entry. Just stick the new entry in 'N$' and call the routine. The value of 'X' will be increased and the new entry filed.

Sister strings, IE...address, city/town ect., can also be tied to this routine. Add them to the entry line '200', and the sort lines 250, 2010 and 2020.

```
10 REM       PROGRAM TO SORT ON DATA ENTRY
60 PRINT :PRINT :PRINT
70 PRINT "TO EXIT PROGRAM ENTER 'END'":PRINT :PRINT
75 :
85 REM --SET SIZE OF FILE TO SORT--
100 INPUT"MAXIMUM NUMBER OF FILES PLEASE : ";A:PRINT :PRINT
110 DIM N(A)
111 :
195 REM --INPUT ENTRY AND SET TERMINAL FLAG 'END' TO GOTO 3000--
200 PRINT "  ";X+1;:LINE INPUT "NAME TO FILE PLEASE: ";N$:IF N$=""THEN 200
210 IF N$="END" THEN PRINT :PRINT :GOTO 3000
213 :
215 REM --INCREASE FILE BY ONE 'X+1' AND START CHECK FOR ORDER--
220 X=X+1:FOR A=1TO X
221 :
225 REM --ENTRY FITS IN MIDDLE OF FILE SO GOSUB 2000 AND INSERT--
230 IF N$<N$(A) THEN GOSUB 2000
231 :
235 REM --IF 'R' FLAG SET ENTRY FILED BY GOSUB 2000, ALL DONE RETURN--
240 IF R=1 THEN R=0:PRINT :GOTO 200
241 :
245 REM --STICK ENTRY AT END OF FILE, NOTHING LARGER THERE THEN RETURN--
250 NEXT A:N$(X)=N$:PRINT :GOTO 200
251 :
1995 REM --SUB ROUTINE TO INSERT AN ENTRY IN THE MIDDLE OF THE FILE--
1997 :
1999 REM --MOVE FILE BEHIND ENTRY BACK ONE LOCATION TO MAKE ROOM--
2000 FOR B=X+1 TO A STEP -1
2010 N$(B)=N$(B-1)
2011 :
2015 REM --COUNT TO 'A' AND INSERT ENTRY.  SET 'R' FLAG AND RETURN--
2020 NEXT B:N$(A)=N$:R=1:RETURN
2021 :
2995 REM --PRINT OUT FILE AS IT WAS SORTED--
3000 FOR A=1TO X
3010 PRINT A;TAB(5);N$(A)
3020 NEXT A:STOP
4001 :
4007 REM --'X' WILL CONTAIN THE NUMBER OF ENTRIES AND CAN NOT OF COURSE
4009 REM   BE LARGER THAN THE ORIGINAL 'DIM' ENTERED AT THE START OF THE
4011 REM   PROGRAM.  THIS PROGRAM CAN BE USED AS A SUB-ROUTINE AND CALLED
4013 REM   WHEN EVER AN ENTRY IS MADE--
4015 :
4017 REM --ODD NUMBER LINES IN THE PROGRAM CONTAIN USER INFORMATION ONLY
4019 REM   AND CAN BE DELETED--
5001 REM
```

EOF

STRING MANIPULATION ON H8 USING THE CONSOLE DRIVER

BY ROGER GREENHALGH
4905 Old Mill Pl.
Raleigh N.C. 27612

Certain string operations in Benton Harbor BASIC are very slow. The statement MID$(A$,B ,1) can take a full minute for a line of text. Heath cannot, or at least will not, tell us how to locate in memory a particular string, hence, the difficulty in using a machine language program to process a string more rapidly.

This paper suggests the use of the Console Driver as a means of getting at a string. The BASIC PRINT statement sends the characters of a specified string into the Console Driver. There the stream can be intercepted and processed by a machine language program. The results can be put into the Console Driver type ahead buffer and taken from there by BASIC with a LINE INPUT statement.

The following rules facilitate this procedure.

1. Prepare a machine language program for the string manipulation desired. Characters will be presented to this program one at a time. They will be found in the accumulator. Place the program in memory above BASIC's stack.

2. In the BASIC program segment the string into sections, which when processed produce strings not longer than the length of the type ahead buffer. i.e. 28 characters. Too long a string will overwrite the control character table.

3. POKE into the machine language program the location of the first character of the type ahead buffer. This location is 040 156 or 8302 decimal.

   i.e. POKE A,110:POKE A+1,32.

4. POKE the starting character count into location 040 155 or 8301 decimal.

   i.e. POKE 8301,X
   (X usually will be zero).

5. POKE into the Console Driver the starting address of the program. A preferred branch point is at 040 365 or 8437 decimal.

   i.e. POKE 8437,A:POKE 8438,B.

6. The program should end with a RET or a branch back to $CDOUT, located at 040 111 or 8265 decimal. The latter will display the string before processing.

7. The BASIC program executes a PRINT command. The string flows through the Console Driver and the machine language program a character at a time.

8. The action in 5. is reset.

   i.e. POKE 8437,73:POKE 8438,32.

9. Before the processed string is read, it may be necessary to disable functions in the Console Driver. e.g. The value 32 decimal in location 040 307 or 8391 decimal changes character case. Location 041 or 8516 decimal holds the start of control character processing.

10. The BASIC program executes a LINE INPUT command to read in processed string. This string will be displayed.

11. The steps in 9, must be reset and the value zero put into 040 155 or 8301 decimal.

   i.e. POKE 8301,0

EOF

H8 EXPANSION

Larry Henderson
304 Wooster Drive
Bryans Road Maryland

Ever since I purchased and built the Heathkit H8 computer, I have been concerned with expansion of the original system to one that could utilize the full potential of the available printed circuit cards.

A happy circumstance was the recent purchase of a Mullen Electronics extender board (about $40), which I obtained to allow work on various circuits on the H8-2 parallel interface board. Quality of this board is outstanding and I believe would meet the high standards of Heath's product line. As things developed, there is an alternate use for this board which is not apparent to the casual user. This use is as the nearly perfect interface to a second Heath motherboard.

BOARD CONSTRUCTION

When I opened the plastic bag containing the kit for the extender board, I found a well sealed, bubble packaged, cushioned box which contained the board, a plastic bag containing all parts, and two sheets of documentation done in the clear and concise Heathkit style.

The board appears to be a very rigid fiberglass laminate, about one-eighth inch thick with heavy and wide conductor traces running on both sides of the board. Following the detailed instructions provided, I was able to complete installation of the four connectors in about ten minutes, including warming up the soldering pencil. Running a point-to-point test with an ohm meter assured that there were no solder bridges between traces. This took only a few minutes and is recommended not only for your peace of mind but to assure lack of smoke from the H8 startup. A nice feature of this board is that the +18 volt, -18 volt and +8 volt power line traces have removable links provided in them so that measurement of amperage in these lines is possible with appropriate instruments. These links are attached to pressed-in terminals on the board with screws. Two sets are located toward the top of the board and one set toward the bottom. I suggest that you install the bottom link on the BACK of the extender board. In addition to these necessary parts there are two other nice features worth mentioning. There is a rubber cushion which is to be stuck to the back of the board to rest on the power supply and two labels for the top of the board to identify each trace's function.

Here, a word of caution is in order prior to putting the unit to use. If your H8 is as full as mine, there is a board installed in every slot except the back connector (which Mullen recommends for the extender board). Clearance between the extender board and the board directly in front of it is very tight, and the chance of contact between traces on these boards is neatly assured. I suggest that an insulator be attached on the front of the lower half of the board. I used a sheet of clear vinyl plastic attached to the board with transparent tape, but any good insulator should do. With the link installed on the back of the board, clearance between these two boards is also increased.

BUSS EXTENSION

After completing construction, I noticed that the male pins, located at the top left of the extender board extend past the edge of the board in order to mate with the board to be placed in test. These pins fit into a Heath motherboard if one of the sets of male connectors are not attached, and allow addition of eight more slots for H8 expansion.

On my expansion motherboard, I had already mounted sixteen 25-pin connectors on the center locations. I then placed the male pins from the extender board into the BACK of the motherboard and, assuring the extender board was perpendicular to the motherboard, soldered the pins in place. Next, I glued the two boards together with one of the fast setting glues such as Eastman 910. I again ran a point-to-point test and eliminated one solder bridge and then installed the unit in the computer.

EXTENDED SYSTEM TEST

I placed the following boards in the H 8 and on the extended motherboard:

CPU		
Front Panel		
H8-2	1 each	Parallel
H8-16	1 each	RAM
H8-1 + H8-3 (8K)	3 each	RAM
H8-5	2 each	Serial
H8-17	1 each	Disk
Controller		

The following two tests were run:

The Memory Test of the 40K RAM (PC location 030 003 offset octal) was run for two hours with no failures.

The Test 17 (M) disk test to assure functioning of the system at high speed was run three times with no failures.

FUTURE PLANS

I intend to build a support bracket for the bottom of the new boards from light aluminum feedstock and to enclose these boards in an aluminum box attached to the top of the H8. If I run into a power deficiency with the H8 power supply, I intend to install a more powerful one to replace the 8 volt system now in the H8. Addition of several H8-7 Breadboard Cards with circuits to use the spare 4K space below the 8K starting address from more EPROM and RAM will occur during the next year.

With the capability of expanding the slot capacity of the H8, together with the quality of software and hardware from both Heathkit and others, the system becomes one of the most flexible and adaptable computers available to the hobbyist.

EOF

"H14 $150 SOONER"

Max Robinson
Rt 12 Box 437
Bowling Green KY 42101

Why would anyone want to operate their H14 printer from the same I/O port as their H9? Well, the most compelling reason I can think of is to provide hard copy output of their console terminal activity as required by HUG or to show a friend how a program works. While the most recent issue of Extended Benton Harbor BASIC does support the H8-4 and the assembler has a halt to let you configure a listing port from PAM-8 the editor cannot drive the H8-4. If you, like me, are buying an H8 system by saving your pennies, you may want to buy your H14 150 dollars sooner if possible. Well, it is!

THE THEORY

An RS 232 system is a voltage-operated system. This means the output resistance of the driver is much lower than that of the circuit being driven. This is somewhat, although not totally, analogus to the power company and your home appliances. Just as you are able to turn on more than one appliance at a time, you can connect more than one RS 232 load to one source. In theory then all you need to do is to connect the H14's data input in parallel with the H9's data input and all is well, not quite.

Unless you are content to run your system at 300 baud there is more to be done. When the character buffer in the H14 gets full it sends a control-S back to the computer to stop its output until the printer can catch up. You need to connect the data output from the H14 to the computer's data input but the H9 has that port tied up and you cannot put the H14's data output in parallel with the H9's data output. A logical "OR" of the two output signals is exactly what is needed and this can be provided by an oldfashioned diode "OR" gate. So much for the theory, does it work?

FACT

Almost! As pointed out by B.G. Chambers on pages 15 and 16 of issue number 6 of

REMark, the input resistance of the H8-5's RS 232 data input is too low. The diode or gate did not function properly until this modification was installed.

A word is in order about operation at 300 baud. At this speed, the printer is able to keep ahead of continuous data flow from the computer. The printer does not need to send anything to the computer and connection of the talk-back line is not necessary. But if you want to run at 4800 baud you need the circuit shown in the figure.

As you can see, the H14's data input is not exactly in parallel with the H9's data input. I reasoned that if the system were being operated with the H14's power switched off, there was a possibility of damage to the H14's input circuit. Thus I included a 3.3 K ohm resistor to protect the H14's input circuit. I have not proved this because I am in no mood to risk a level conversion chip on the gamble.

The switch is necessary to take the printer off-line. You might think that the off-line switch on the printer could be used for that purpose but the H14 is too smart! Even in the off-line position when its buffer gets full, it sends a control-S to stop the output. If you try to override with a control-Q from the keyboard, the H14 sends another control-S to stop the output. The data flow to the H14 must be interrupted to keep it quiet.

CONSTRUCTION

The two diodes, the 3.3 K ohm resistor and a 25 pin female connector are installed in a small aluminum mini-box. The 47 K ohm resistor is placed on the H8-5 board. The 25 pin connector may be mounted on one side of the box. But the existing cable which connects the H9 to the socket on the back of the computer. The ends of the cable should enter the ends of the box through rubber grommets. The diodes may be any small signal silicon type such as Heath part number 56-56. The switch may be any type. When installing the 47 K resistor on the H8-5 board use unused TTY jumper holes. -18 volts may be found on the jumper hole which is in common with pin 7 of the I/O connector. The other end of this resistor should be soldered in the left hand hole of the TTY jumper which would short out R 151. Sleeving should be placed over the resistor leads to prevent shorts.

OPERATION

The only trick to using the H14 in this mode is keeping the prompt from being printed. This can be done in BASIC as follows. Type in the following. Type in the following program with the print switch off.

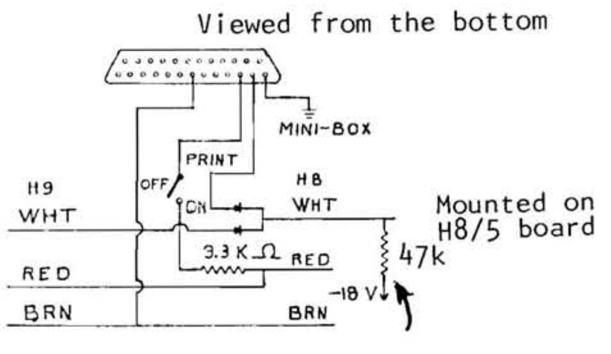
```
10 print
20 GOTO 20
```

With the print switch still off type RU. The computer will add the N. Turn the print switch on and hit return. Any characters which were in the buffer will be printed. Turn the print switch off and hit control-C. The buffer is now empty. To suppress "END AT LINE NNN' the last statement in your program should be

NNN GOTO NNN

After the program has finished executing you may turn the print switch to off and hit control-C. As the editors can see I have not found a way to suppress the printing of the TED-8 prompt. Remember that the contents of the buffer are not printed until a carriage return is received by the H14.

You may find that sometimes your computer seems dead and you may think a crash has occurred. You will most likely find that you have released the off-line switch on the H14. This causes the H14 to send a control-S which stops all computer output. Simply press in the off-line switch and your computer will come alive again. This is a common mistake because it is necessary to release the off-line switch to advance or reverse the paper. Happy programming!



EOF

H8 SHOEHORN

Automatic hardware boot-up

Howard Nurse
665 Maybell Ave.
Palo Alto Ca. 94306

Here is a circuit that you can build which will simplify your H8/H17 system start up procedure. The circuit is a sequencer which electronically pushes the H8 front Panel switches for you, in machine gun succession. The twelve-switch key pad on the front panel is a powerful tool when developing and debugging programs, but it can be a bit of a nuisance when bringing up the disk. To boot a disk requires that you first push up to 13 separate switches on the H8 front panel. The shoehorn sequencer allows you to start your disk boot using one switch. .sl The sequencer, shown in Figure 1, consists of six TTL chips which can be built on a small piece of perf board, or on the H8-7 breadboard card. Control wires from the sequencer are soldered to the back of the H8 front panel board where the key pad switch lugs are located. The start control line can be wired to a separate switch on the key pad, such as the "Cancel" switch, to a switch added to the H8 panel, or to your terminal.

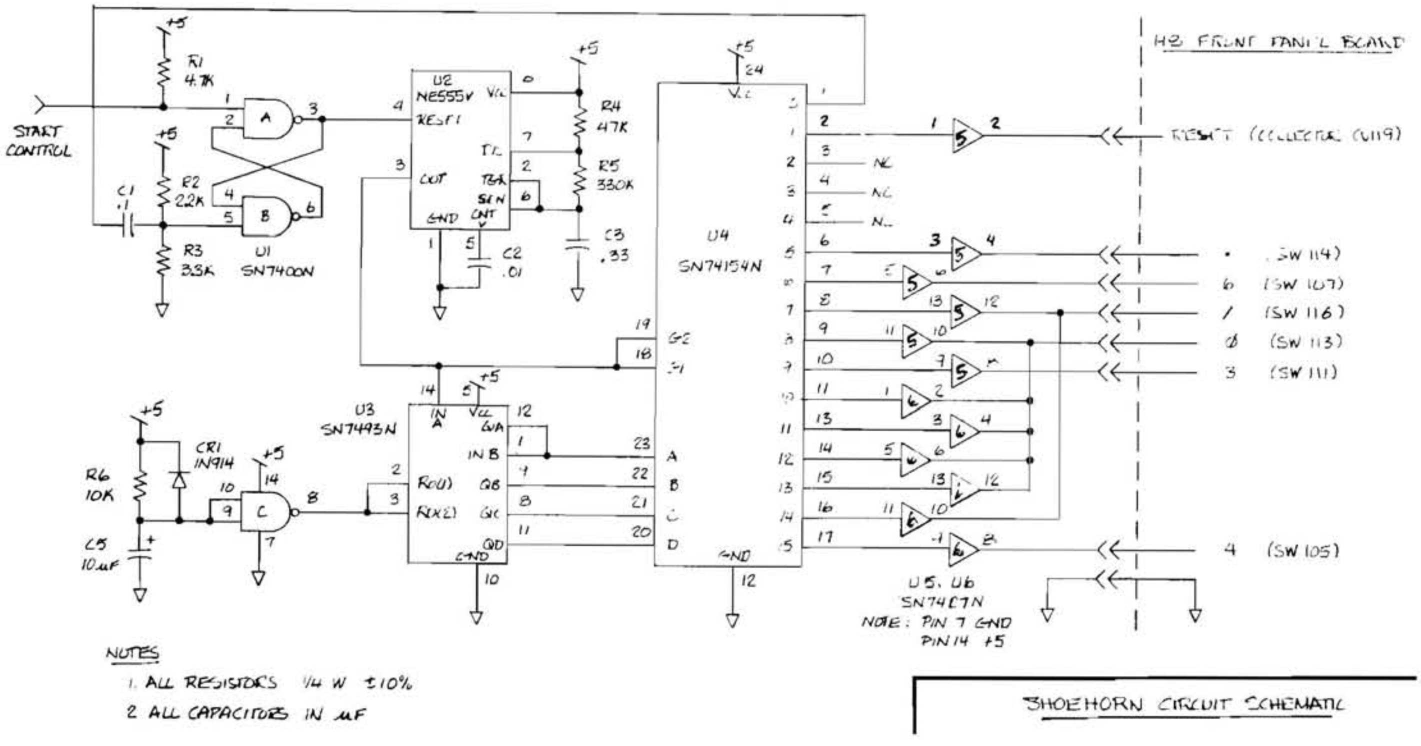
A 555 Timer (U2) generates the clock for the sequencer. The output from the timer feeds binary counter U3. The counter outputs serve as address inputs for 1-of-16 decoder U4. As the counter cycles from Ø to 15, a low logic level appears at successive outputs from the decoder. These outputs, when buffered by open collector drivers U5 and U6, are used to short the hot side of the desired key pad switches to ground in the correct order.

The switching sequence is initiated by momentarily grounding the start control input. The control latch (U1-A and U1-B) supplies a high logic level to timer U2 pin 4, which allows the timer to start oscillating. Decoder output "1" resets the 8080A microcomputer by grounding the collector of Q119 on the H8 front panel board. The next three outputs are unused to allow time for the H8 reset to complete. The sequencing ends when the control latch is reset by the leading edge of decoder output "Ø".

NAND gate U1-C and its associated circuitry resets the binary counter to a count of Ø on power-up.

Since there is uncertainty in the exact length of time required for the 8080a reset circuitry to respond, it may be possible for you to speed up your sequencer by lowering the value of R 5. If you wish to speed up the clock, some experimentation with the value of R 5 may be necessary.

EOF



BUGGIN' HUG

Dear Hug,

I just finished assembling my H27 Flexible Disk System and thought I would write and tell other users of the H11 system some of the things I have found out in the last six months.

First, I have had a problem with the Line Time CLock in the H11 since I first built it. The problem manifested itself in a clock program and caused the time to be five percent fast. At first I thought that it was a programming problem and suspended any further use of the LTC interrupts. Recently, with thoughts of getting the H27, I desired to have the 'TIME' function work properly. So I wrote a program using the console terminal interrupts as a known time source to actually check out the frequency of the LTC interrupts. I counted 1000 interrupts from a 110 baud terminal (the period was 100.827 mSEC) which should have gone for approximately 10 seconds, and simultaneously counted LTC interrupts. Periodically, I stopped the computer and checked the accumulated count (there should have been 6000 for exactly 10 seconds). The result was between five and seven percent high. In desperation, I connectd an oscilloscope to the LTC input on the H11 backplane (not a recommended operation!) and looked at the waveform. A quick observation showed a stable square-wave, but when I carefully triggered the 'scope on the negative edge (the edge the LSI-11 uses), I noticed an occasional brightening effect on the waveform. It appeared that the scope was double triggering. Aha! But, when I expanded the area on the screen, I noticed that there was a real second pulse occurring at the negative transition. Now the problem was to find out where it was coming from. Back-tracking through the circuitry, I couldn't see any junction that had the same problem on either edge of the waveform. On a hunch, I installed a capacitor across the 'JUMPER' connections that defeat the LTC. The extra pulse disappeared! The value of the capacitor was somewhat critical as it tended to slow down the fall time of the LTC signal. I settled on a ,01uF, 25V disc bypass capacitor and soldered it where the jumper was installed. Voila, the problem was fixed! The test programs I had written using the LTC now checked perfectly and I was able to determine the frequency of the A. C. line as 60.003 Hz, which isn't too bad for the local power company. I know of at least two other persons who have had this same problem and fixed it in the same manner. The computer service technician I talked to at Heath knew of no problem with the LTC nor of any fix for any problem!

When I finally completed the H27, I found an immediate need to initialize many floppy disks. The instruction manual with the H27 indicates a method, via uODT, to initialize disks by setting the various H27 registers to 'magic values'. I reasoned that this same sequence could be done in a program and run whenever needed. Now, when I buy a new box of disks, they are all initialized at one time and subsequently checked with PIP.

> Bob included a program to initialize diskettes but since such a program was published in the last issue of REMark, we chose not to include it here. :JB:

I noticed in the last issue of HUG that a reader was having a problem with FOCAL and its FCLK function. Apparently, the LTC interrupt routine was only counting time ticks in one 16 bit location, but two locations are reserved for counting ticks. The author of the article noticed that an INC instruction does not set the CARRY bit if the word overflows, as is needed for double-word counting of events. However, ther is a technique to get two words of counting, and in the same space available in FOCAL. The ADC instruction adds the CARRY bit into a word. and if there is a CARRY out (overflow), the CARRY bit is set at the conclusion of the instruction. Hence, a simple two instruction program will count using 32 bits:

```
ADC WORD1    ;ADD 1 TO WORD 1 IF CARRY SET

ADC WORD2    ;IF CARRY OUT, ADD 1 TO WORD 2
```

Now the problem is to get that initial add of 1 bit. The LSI-11 interrupt vector has available just the mechanism needed: Set the carry bit in the PSW when the LTC interrupts. This will cause a 1 to be added to WORD1 every time the LTC interrupts. If WORD! turns from 177777 to 0, it causes a CARRY out. This manifests itself as a 1 to be added to WORD2 in the second instruction! All that is needed to set the CARRY bit is change the word at location 102 (LTC interrupt vector PSW) by adding 1 to it: 340 becomes 341. This sets the CARRY bit everytime the LTC interrupts and causes correct operation of the 32 bit clock counters. This technique is used in RT-11 for the same function. It is also the first time I have ever seen a use for the condition code bits in the interrupt vector PSW.

Anyway, as an ex-paper-taper, I am extremely happy and pleased with the H27 system. My production of working programs has increased about 20 times, and I find that I make about ten edits of a program within 30 minutes, just to get it perfect. Try that with paper tape! It took 30 minutes just to punch out a source tape! And then it didn't work! One program that I found extremely helpful with paper tape was a co-resident editor/assembler. This was concocted by purchasing (from DECUS) a text editor (TED-11A), and making a routine to connect the editor to the PAL-11S assembler such that the assembler got its input data from the editor text buffer. This eliminated having to punch out a source tape until the program was void of assembler errors (although logic bugs might still be their). I calculated the speed of this combined routine at about 700 characters per second for PASS 1. The program even printed the decimal number of bytes in the source program. Typing CTRL P returned to the editor with no loss of text. This editor, however, is a slightly modified version of the one supplied by HEATH (ED-11), but I found the modifications closely resembled the RT-11 EDIT program. Now that I have the disk system, this program will probably just sit in the box of tapes. But it was a good program to use anyway.

So, keep up the good work, Jim, et al. I look forward to the next issue of REMark.

Regards,

Bob Meister
59 Glade St.
West Haven CT 06516

Dear Jim

The enclosed is submitted for your consideration a short, self contained, article for possible publication in REMark Magazine.

Regardless of what kind of home computer a person owns, it has been my observation that few owners use it for letter writing. Since I have made extensive use of my H8 for that purpose, starting with a basic letter writer with tape input and more recently using the technique herein described, I felt others would benefit from my experience.

I recognize that the material is rather simple compared with most of the articles appearing in REMark, but not all of the REMark readers are experienced computer hackers and thus might gain from some "easy stuff".

I continue to enjoy my H8/H17/H9/Expandor and look forward to every issue of REMark. Thank you again for the items and information which you have provided me in the past months.

Warm Regards,

Alden G. Finley
1429 Balmy Beach Drive
Apopka Fl. 32703

Here is the text of Gordon's article. :JB:

Dear Fellow H8 Owners,

You really ought to use your H8/H17 for writing both business and personal letters. It is convenient, saves time, provides a record, and makes good use of your computing equipment. Let me take you step-by-step through the easy way to do it.

First, load your letter writing disk into SY0:. (It should have the following files on it: EDIT.ABS or ED.ABS, your address, and the addresses of all of your correspondents. The latter in individual files labeled XXXX.A.) Second, call up EDIT or ED, or even use HDOS, and compose the body of your letter. After any editing, put it to disk using a label XXXX .LTR, where XXXX is a short name for your correspondent and is the letter sequence number. Third, ready your printer and call up PIP. Fourth, type <AT:=AGF.A>. (or LP:=AGF.A. Of course you will use your own address file label.) Fifth, adjust paper to suit and type <AT:=TT:> and then the date, <return>. The salutation (if you didn't include it in the body of the letter), and then type <CTRL D>. Sixth, type <AT:=XXXX .LTR> <RETURN> and the body of the letter will be typed. Seventh, type <AT:=TT:> and enter your close, and signature line if desired. Then type <CTRL D>. Eighth, adjust paper to about 8 lines up from the bottom and type <AT:=XXXX.A> <RETURN> and the mailing address will be printed. Ninth, remove paper, sign it and place it in a window envelope, stamp, and mail it. Tenth, copy your letter over to a nonbootable letter file in SY1: and erase it from the SY0: disk.

There it is. Just like this letter appears here. You have a record copy on file, can make a hard copy if you desire, don't have to address the envelope, don't have to use your address book, don't have to hunt up the gummed labels that you made and misplaced. You have a neat looking letter, and have demonstrated another job that your computer can do for you.

Future improvements might include a text writer like HUG's H8 Text Formatter and a short ABS program that would include the sender's address and get the date from the HDOS file.

H14 Mailing Label Info

In attempting to run a mailing label program, I found that the paper backing of the labels was too translucent and the paper-out detector shut down the printing. I removed a strip of sprocket-holes from one side of fan-fold paper (11" long) and fed it thru the paper guides on the printer at the left end (where the paper-out detector is) up past the platen and on to the sprocket. The other end was brought forward and over-lapped with the first end on the sprocket and cemented to the first end, forming a loose loop. (Don't cement the loop to the sprocket!!) The label material (Dennison Tabulating labels, Item 42-451-0) was then fed into the printer. I ran off over 300 labels and the paper loop was none the worse for wear. Of course this cheats the paper-out detector, but is better than throwing away 5,000 labels. The labels come in continuous fan-fold so the probability of running out is minimal.

Albert G. Shafer
683 S. W. 7th Street
Boca Raton FL 33432

Dear Jim,

Enclosed is my article, "Permutations for the H8", which should be of interest to REMark's readers, especially those who want to investigate Assembly Language. I'm working on a follow-up article, "Combinations for the H8."

I just received your letter about user-design hardware. I am planning to enter a useful circuit+software, one which should be very easy for users to implement.

You may remember a letter I wrote earlier this year on my plans for building a better tape operating system. Here is the status of the project: Eddie Baby is working, as you know. (John's Editor is HUG P/N 885-1034) My assembler is now working. The listings enclosed with my article were produced by my assembler. I don't want to distribute it yet because it is not fully debugged. It is somewhat more powerful than HASL, as it can produce relocatable and linkable object modules (like H11 software). So far I have only used it for absolute programs, since I haven't written the linking librarian needed to take advantage of separately assembled modules. This I hope to do in the next month or so. The object modules produced cannot be loaded using PAM-8, since they are in an incompatible format. I do have a small loader for loading my object modules.

Once I got my assembler working, I stopped using HASL (I haven't used it since!) My assembler is co-resident with Eddie Baby, so that assembly is much faster and more convenient. There is even a facility for transmitting listings over my modem to another computer (which is how the listings enclosed were produced.)

My plans for the future (near and far): (1) get a linking librarian written and working, (2) rewrite Eddie Baby and the assembler so that they are more compact, (3) get those pieces of software into a form that can be distributed, and distribute it through HUG. Such a system would allow the user to produce extensive assembly language programs using a tape system.

Later, I may re-write my Q (the language of the Future -- today!) compiler to work well with the tape system.

Dear Jim,

I have noticed with considerable interest that Heath is at least considering the support of additional user designed circuits, and even better, putting the INTEL 8088 on the H8 Buss. Such actions should be strongly encouraged as they will keep the H8 system "ALIVE" for a longer period of time by providing growth potential for both the hardware and the users software. The state of art in computer electronics is advancing rapidly and without upgrade potential, existing systems will be avoided by potential purchasers for fear that they will be left behind.

Specifically, the H8 with its 8080A MPU is rapidly falling out of favour with the more serious or professional programmers as they would like to use the language/instruction set capabilities provided by the newer 16 bit micros such as the 8086/8088, Z8000, etc. I will be the first to admit that the 8080A is capable for handling any application I may have and that my H8/H9 system is normally input/output bound rather than CPU bound.

Even a two to five fold increase in internal processing speed would be useful and appreciated, however, in many instances this is not the major reason for wanting to move to working with an 8086 class MPU. For many of us the ability to write our programs with the facilities provided by an 8088 type MPU is the most important factor. The INTEL MCS-86 product description, dated February 1979, states that "depending on program type, execution speeds 7 to 12 times faster than 8080A speed can be expected. As the same time, the program is typically 10% to 25% shorter". Although some recoding may be necessary to achieve the savings, these factors would increase the value of existing memory boards and other system components. The reason I purchased a computer was to learn, and be able to program in, various low and high level languages. That I will use the system for word processing, CAI, and other applications is of secondary importance.

Since personal computer owners usually have only limited financial resources to invest in their system, they are highly concerned with the costs involved with upgrading their system. Clearly, the MPU and its supporting circuits represent only a small fraction of the total cost of a complete system. Requiring a user to junk or trade off all of a system so as to be able to use an upgraded MPU shows a lack of concern for the equipment owners and a poor business strategy on the part of the vendor. Vendors such as IBM and INTEL would not be the companies they are if they had not been sensitive to these important factors. We already see the 9900 and 8086 MPUS being offered on the S100 Buss in such a way that little, if any, changes to the rest of the system are required. That INTEL produced the 8088 indicates that they recognize both the users desire for and profitability of, this type of approach to system upgrading.

I plan on having my computer system for a long time and will want to move to a 16 bit MPU sometime in the future. My willingness to make additional investments in my present system would be greatly increased if I knew that a reasonably priced 8088 CPU card would be made available ,even if it took several years. For myself, one of the main selling points of the H9/H19 terminals and H14 printer is that they are buss independant and can be moved to an upgraded CPU if Heath doesn't provide one. For this reason, I purchased the H14 printer rather than disk storage. Except for embedded applications, the general purpose microcomputer community has moved from the 4 bit to 8 bit processors and the move to 16 bit processors is on the way.

For my purposes, the H8 is an excellent machine and its disadvantages are greatly overshadowed by its advantages. Providing a hardware compatible upgrade CPU card would greatly increase the value of the existing H8 mainframes and input/output devices as well as encourage the purchase of additional ones.

INTEL has developed an 8086 assembler/converter which is capable of assembling/converting 8080 code so as to protect this considerable investment. Microsoft has already announced an 8086 Basic and Industrial Programming Inc. has announced a real time operating system for the 8086. Although the 8086/8088 processors are still new and quite expensive, major software offerings and tools are already being made and a lot more should be available by the time the 8086/8088 processor prices come down to a reasonable level. As an example, an 8086 version of CP/M is scheduled for release in the near future.

To keep the H8 a viable long range system, I would recommend the following be at least considered.

1. An 8088 CPU card with supporting BASIC, PASCAL, and ASSEMBLER languages.
2. An expansion unit to the Heath and/or S100 BUSS.
3. Bit mapped graphics.
4. H9 upgrade kits (upper/lower case, 24 lines, etc.). Many of us might have the money to upgrade our H9 but not enough to replace it.
5. Higher disk storage capacity per disk plus additional drives.
6. Lower priced 16K/32K synamic RAM cards.

These of course, are but one individual's desires and view of the market. In the interest of the present and potential Heath company itself, I strongly recommend a comprehensive direct mail user survey to determine the true market/user demand for future hardware and software products. Until such a survey is initiated readers of this letter are encouraged to forward their thought to me at the above address and I will consolidate and forward them to HUG and BUSS.

With best wishes for our mutual success in developing and using state-of-the-art microcomputer technology.

Fred W. Pospeschil
3108 Jackson St.
Bellevue NE 63005

Reference REMark 5, Program Directory Program Pg. 6. When the program is used with an H19 terminal, two lines must be changed;

```
60 IF Z = 27 GOTO 135
65 IF Z<>13 THEN PRINT CHR$(7);: GOTO 135
```

From what I can see the H9 outputs the most significant bit as a 1, the H19 as a 0. Also, if the H8-4 card is used, line 55 will have to be changed to reflect the proper port (decimal) in the PIN portion of the line.

Hope this info might be of some help to someone else who has had the same problems.

Brian L. Hansen
315 Roast Meat Hill Rd.
RFD 2
Killingworth CT. 06417

EOF

ETA-3400 MEMORY MOD

James F. Gregory
407 Wonderly Ave.
Dayton Oh. 45419

I am using the new ETA 3400 accessory and trainer as my computer system. After putting this system into operation, I found that the 1/2K of memory on the Trainer was not available for use. The following is a modification I made to the Trainer that might be of interest to other users of this system. The Trainer RAMs are addressed by IC2, IC3, and IC20. Only IC2 is to be modified. In checking the decoding of IC 2 the following addresses are available for the added 1/2K of memory.

```
IC2 output 2 (pin 3) gives addresses 4000, and 4100
    output 3 (pin 4) gives addresses 6000, and 6100
    output 4 (pin 5) gives addresses 8000, and 8100
    output 5 (pin 6) gives addresses A000, and A100
```

Since the Trainer needs the RAM's addressed as built to operate without the accessory, I mounted a SPDT slide switch to the edge of the circuit board, on angle brackets, near IC 1 and cut a hole in the side of the case to hide the switch so it could not be accidentally thrown. The foil from IC2 pin 1 to IC3 pin 13 was cut. Pin 1 of IC 2 was wired to one side of the switch and the center of the switch goes to IC3 pin 13. The other side of the switch is wired to your choice of pin 2,3,4 or 5 of IC2. By operating the switch, the Trainer can be used without the accessory or with the accessory and the extra 1/2K of memory. This extra memory can be used for subroutines and can be called by the USR function of BASIC. I have been using this system for 2 1/2 months and have been happy with it.

EOF

TABLE OF MEMORY LOCATIONS FOR HDOS TYPE AHEAD BUFFER FOR VERSION 1.5 of HDOS

LOCATION	OCTAL VALUE	DECIMAL VALUE
HDOS Starting Address (low byte)	Ø4Ø 32Ø	84ØØ
HDOS Starting Address (high byte)	Ø4Ø 321	84Ø1
Offset to Line Counter Byte	Ø11 275	2493
Offset to Queue Tail Pointer (low byte)	Ø11 3Ø1	2497
Offset to Queue Tail Pointer (high byte)	Ø11 3Ø2	2498
Offset to Queue Head Pointer (low byte)	Ø11 3Ø3	2499
Offset to Queue Head Pointer (high byte)	Ø11 3Ø4	25ØØ
Offset to Pointer to Buffer Start (low byte)	Ø11 3Ø5	25Ø1
Offset to Pointer to Buffer Start (high byte)	Ø11 3Ø6	25Ø2
Offset to Pointer to Buffer END+1 (low byte)	Ø11 3Ø7	25Ø3
Offset to Pointer to Buffer END+1 (high byte)	Ø11 31Ø	25Ø4
Offset to Buffer Start	Ø11 314	25Ø8
Offset to Buffer End	Ø12 Ø6Ø	26Ø8

(See Jack Thompson's article in issue 7:JB:)

SOFTWARE FOR H8/H88/H89 SYSTEM . . .

HUG P/N 885-1036	TAPE IV
Cassette Tape	$ 9.00
HUG P/N 885-1037	VOLUME IV
Documentation--Hardcopy	$12.00

EDUCATIONAL PROGRAMS

EDUCATION GAME BASIC/10.02/H8/24k

An entertaining means of being drilled in math.

MATH DRILL BASIC/10.02/H8/16k

A speed and accuracy math drill.

UNIVERSAL QUIZ BASIC/10.02/H8/24k

Learn the state capitols, the presidents, or any other quiz you build.

CHEMISTRY SYMBOL QUIZ BASIC/10.02/H8/24k

Teaches the 50 most common elements.

ELECTRONIC CALCULATION PROGRAMS

AC CIRCUITS BASIC/10.02/H8/24k

Calculates circuits such as series RL, inductive-capacitive reactance, resonant frequency and others.

RC CHARGE & DISCHARGE BASIC/10.02/H8/24k

Calculate most anything about the function T=RC.

MCROWAVE PATH BASIC/10.02/H8/16k

Calculate the median received signal level for a microwave path.

OHM'S LAW BASIC/10.02/H8/16k

Employs the twelve basic Ohm's law formulas.

F.M. POWER BASIC/10.02/H8/16k

Calculate the legal parameters of operation of a commerical FM transmitter.

POWER SUPPLY DESIGN BASIC/10.02/H8/16k

Design a P.S. with zener regulation.

FINANCIAL PROGRAMS

REAL ESTATE DEPRECIATION
BASIC/10.02/H8/16k

Find the depreciation schedule for properties.

BUSINESS ANALYSIS BY GEOGRAPHICAL AREA
BASIC/10.02/H8/16k

Categorize customers by types of service, location and price of service.

GENERAL PURPOSE BONDS BASIC/10.02/H8/16k

Determine the interest and update the cash value of bonds.

ANNUITIES BASIC/10.02/H8/16k

Calculate the legal withdrawals per month.

FINANCE BASIC/10.02/H8/24k

Finance is a personal financial program.

CAR DEAL BASIC/10.02/H8/24k

Calculate the dealer's price and profit for a car.

CAR GAS BASIC/10.02/H8/24k

Keep track of MPG, gallons, money spent, gas tax and the miles driven.

GAME PROGRAMS

COMPUTER POETRY BASIC/10.02/H8/24k

Applies rules of logical syntax to lists of user provided words.

TWILIGHT GOLF LEAGUE BASIC/10.02/H8/32k

Calculate and store all necessar, records for 12-team and 24-player golf league.

ENCRYPTO BASIC/10.02/H8/24k

Players try to decipher an encrypted message.

BATTING AVERAGE BASIC/10.02/H8/24k

Maintain baseball batting averages.

WHEEL OF FORTUNE BASIC/10.02/H8/16k

Bet on the numbers and win or lose your bank roll.

UTILITY PROGRAMS

BREWMEISTER BASIC/10.02/H8/32k

A complete aid to making beer.

DAY OF WEEK BASIC/10.02/H8/16k

Compute the day of week for a given date.

. . . AND SOME MORE

PHILATELIC PHYLE BASIC/10.02/H8/16k

A stamp inventory program which will store, sort and display data on stamps.

BASIC LETTERS BASIC/10.02/H8/24k

Prints big letters on terminal or printer.

TICKER TAPE BASIC/10.02/H8/16k

Displays an alphanumeric message across the H8 front panel.

KEY PAD ENTRY BASIC/10.02/H8/16k

Allows data entry into a BASIC program through the H8 front panel.

HASL8 SYMBOL GENERATOR ASM/H8/16k

Generate a cross reference list after an assembly.

BASIC LINE EDITOR ASM/H8/16k

Enables the revision of text on a line of an Extended BASIC program.

HUG P/N 885-1029 DISK II GAMES 1
H8 Disk Software $18.00

SPACE WARS ABS/H8/H17/32K

A machine code version of the popular space wars game.

GAME OF LIFE ABS/H8/H17/16K

A machine code version of a game in which cells are reproduced and changed.

DEATHSTAR BASIC/H8/H17/24K

Deathstar is an interactive game, based upon the popular movie "STAR WARS"(C).

CAMEL BASIC/H8/H17/24K

Camel is a desert survival game.

STOCK MARKET GAME BASIC/H8/H17/32K

A stock market simulation game which provides excitement out of making (or loosing) large sums of money.

FOOTBALL BASIC/H8/H17/24K

This football game uses the H-8 keypad to select offensive and defensive plays.

QUBIC BASIC/H8/H17/24K

Qubic is a three dimensional Tic-Tac-Toe game based on a 4 by 4 by 4 playing cube.

HUG P/N 885-1030 DISK III GAMES 2
H8 Software $18.00

ATOM 2000 BASIC/H8/H17/40K

The time is year 2000. An atomic war has taken place, your assignment is to survive.

SUBMARINE BASIC/H8/H17/24K

CRYPTO BASIC/H8/H17/24K

Crypto is a decoding game.

ANIMAL BASIC/H8/H17/20K

You think of an animal and the computer tries to guess the name of the animal.

MASTERMIND BASIC/H8/H17/24K

BATTLESHIP BASIC/H8/H17/24K

The game of Battleship is to locate and sink the ships hidden on a grid.

BASEBALL BASIC/H8/H17/24K

HUG P/N 885-1032 DISK IV MISCELLANEOUS
H8 Disk Software $18.00

MORSE ABS/ASM/H8/H17/24k

Morse translates characters typed on the terminal into Morse Code using the H8 speaker.

FORMAT ABS/ASM/H8/H17/16k

Text formatter.

CALENDAR BASIC/H8/H17/24k

MAP OF BASIC SYMBOLS ABS/H8/H17/24k

A debugging aid for BASIC programs which creates a table of variable names.

KCAT & ZCAT ABS/H89//H8/H17/16k

KCAT will print a listing of all non-system files and ZCAT will print all files on the the disk.

MUSIC LIBRARY SYSTEM BASIC/H8/H17/24k

PRINT-A-FILE BASIC/H8/H17/32k

--EDIT

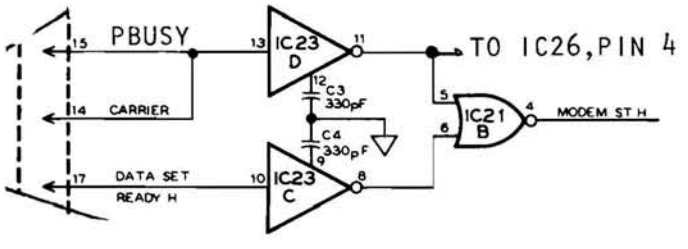
H-11-5/TI 810 Modification

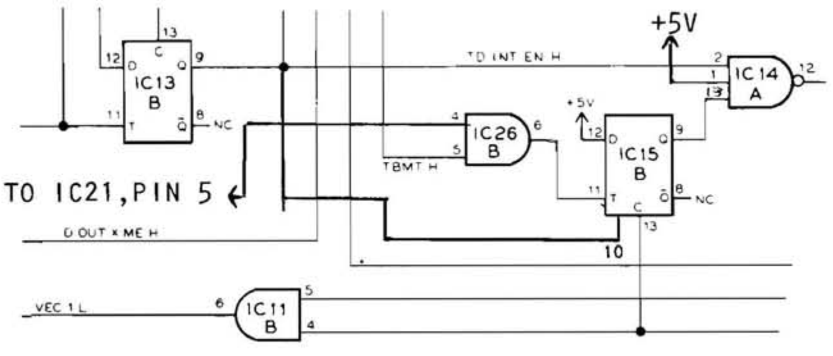
By: Randy Borchardt
Heath Systems Engineer

There has recently been a great deal of interest in serially communicating to the TEXAS INSTRUMENTS model 810 printer from the H-11-A computer. The serial card (H-11-5) currently offered by HEATH was not designed for interfacing printers which make use of hardware handshaking via a "printer busy" signal. However, a few "cuts and jumpers" will allow the H-11-5 to operate in this manner.

Since HT-11 is set up to use interrupt controlled I/O, the purpose of these cuts and jumpers is to allow the printer's busy signal (PBUSY) to control whether or not the UART's transmitter buffer empty signal causes an interrupt to be passed onto the computer's bus.

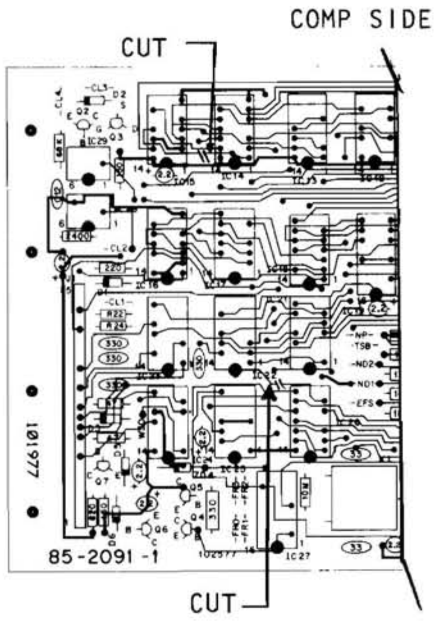
The portions of the schematic which are affected are shown below and may be cut out and taped over the original.

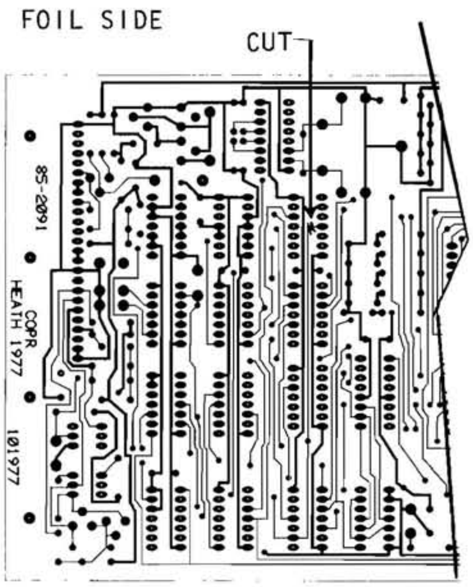




There are two methods of modifying the circuit card. However, one method requires removal of an IC socket so the second method will be described. Three cuts and four jumper wires are required. The artwork shown below is for the latest revision of the circuit board, so if your board is of an older vintage, be sure that you cut the proper foils (fortunately, this area of the boards hasn't changed too much, but it has changed).

First, make the two cuts as indicated on the component side of the circuit board. Next, make the cut indicated on the foil side of the board.





Now, install the four jumpers as follows:

FROM	TO
IC 21, pin 5	IC 26, pin 4
IC 15, pin 11	IC 26, pin 6
IC 13, pin 9	IC 15, pin 10
IC 14, pin 1	IC 14, pin 14

The modifications shown above are such that the modified card will still communicate with a HEATH H-14 line printer. The only difference between the hardware handshake of the H-14 and the standard TI-810 is that the busy signals of the two units are opposite in polarity.

This presents the need for a choice. The alternatives are whether to move a jumper plug on the TI-810's CPU board (this must be done if you wish to remain compatable with the H-14) or move one end of one of the jumper wires just installed on the H-11-5.

If you do not wish to remain H 14 compatible or do not wish to open your TI-810, return to the modified H-11-5 and move the jumper from IC 21, pin 5 to pin 4 of the same IC.

Otherwise, proceed to install the IRC (inverted reverse channel) option in your TI-810. To do this, remove the top cover from the printer (five screws) and then remove the metal wrap which covers the card cage (three screws). Next, remove the CPU card which is the one with the green card pull. Now, while holding the board as though viewing it from the front of the printer, locate jumpers E7, E8, and E9. Move the jumper plug from E8 and E9 to E8 and E7. Now replace the card and covers. If you wish, there is a checklist inside the ribbon cover of the printer where you can check that the IRC option has been installed. Now that the computer and the line printer are ready to go, a cable is needed to connect them. The following schematic shows the type of cable to use. Two male "D" connectors, some five conductor cable, and two short jumper wires are required.

```
TO PRINTER:                                                     TO COMPUTER:
     ----                                                            ----
   :  1 :------------------------------------------------------: 1 :
   :  3 :------------------------------------------------------: 3 :
   :  7 :------------------------------------------------------: 7 :
   : 11 :------------------------------------------------------: 4 :
   : 20 :------------------------------------------------------: 20 :
   :  6 :---                                                   :    :
   :  8 :---:                                                  :    :
   :  9 :---                                                   :    :
     ----                                                            ----
```

(mark this end "PRINTER")
Happy Printing!

PROGRAMMABLE BAUD RATE FOR THE H11-5

George Roth
11030 Brookfield
Livonia MI

I recently completed the H19 CRT terminal kit and I am very impressed with the engineering, quality and features of this unit.

The capability of "programmable baud rate" brought to mind the possible application of using the H19 baud rate generator to drive the H11-5 serial interface board as stated under "Introduction" on page 2 of the H11-5 assembly manual. Further checking revealed that the serial interface was capable of having its baud rate generator jumpered for Multiplexed Input (Im). The serial board rate generation is accomplished by IC27 a CMOS multi-function chip. It was determined that the crystal oscillator and its output to the -8 volt supply circuitry still functioned with the multiplexed input jumpers in place. The clock output on pin 11 then became a function of an external clock input to pin 15.

Modifications of the serial board were the installation of jumpers FRO, FR1, FR2 and FR3, connecting a wire between pin 15 of IC27 and J1 terminal 5 and connecting J1 terminal 4 to ground.

Cont'd. On Next Page

A twisted pair was connected to terminals 15 (signal) and 14 (signal gnd) on the H11 rear panel "AG" serial connector.

I determined that the required 16x baud rate signal was available at pin 15 of the programmable baud rate generator U451 on the "Terminal Logic Circuit Board" of the H19. An unused line driver was available on U452. Its pin 2 was isolated from its connection to pin 31 of U451 and a jumper installed between pin 2 of U452 and pin 15 of U451. Also, pin 3 of U452 was jumpered to connector 404-2 on the TLC board. The twisted pair from the H11-5 serial interface board was connected directly to J404.

The modifications work beautifully with complete control of the system baud rate from the H19 terminal allowing you to slow the system to a readable rate when desired.

One word of caution. If you desire to change the baud rate when the computer is outputing data, first stop the output with Control S, take the H19 off-line, modify the baud rate by typing Escape r then one of the capital letters A (110 baud) through L (9600 baud) (M for 19.2k :JB:)and returning to the computer by releasing the off-line key and typing Control Q.

I hope others will find this simple modification as rewarding as I have.

EOF

INTEGRAL DATA . . . DATA

Walter A. Rison
PO Box 134
Pisgah MD 20640

This circuit just might be of interest to someone with an Integral Data IP-125 or IP-225 printer, trying to run parallel with the H8-2. The problem with running the IP-125 parallel is that after a few lines of print, the printer will hang (carriage goes all the way over to the left limit and stays there). After talking to Integral Data about this problem, they told me that the printer was getting two strobes without sending an acknowledge for the first one. This will screw up the buffer. I found this to be true, so I set out to correct the problem.

The circuit in Fig. 1 will correct this problem. This circuit will first delay the acknowledge by about 1.2 milliseconds, then send a 1.2 microsecond pulse to the H8-2. Also I found the cable connecting the printer to the H8-2 to be critical. I used a flat ribbon cable with every other wire grounded. The following is how to build and connect the interface.

1. Remove jumper Z2-Z4 in the printer .
2. Install jumper Z1-Z3 in the printer .
3. Set the dip switches in the printer for parallel operations as shown in the user's manual.
4. Build the circuit as shown in Fig. 2 .
5. Connect the cable to the circuit as follows: Pin 19 from printer to pin 1 of IC1 pin 5 of IC2 to H8-2 channel pin 10 All other connections are standard see table 1.
6. Mount the PC board on the H8-2 as shown in Figure 2 with a nylon screw and nut. Also put some kind of insulator under the board.

7. The +5 VDC and ground are taken from the H8-2.

This circuit has been working on my printer for about 6 months with no problems. And is now working on an IP-225 with no problems.

When I first started to connect the IP-125 to the H8-2 I also had a problem with the printer dropping characters. This was found to be a noisy clock going to the USARTs. I fixed this by replacing IC141 with an SN 74LS14.

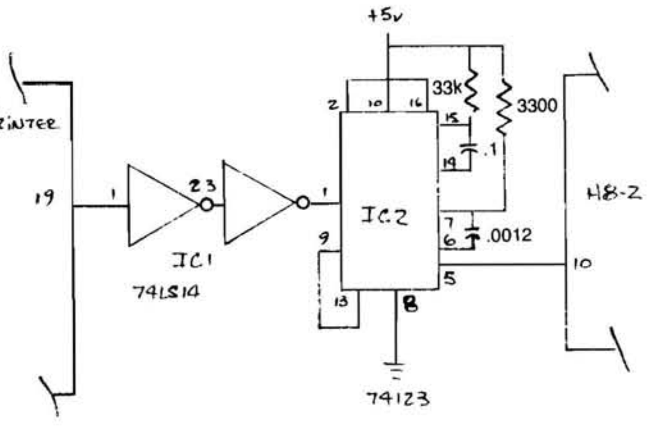


Fig. 1

TABLE I

Interface Cable Connections
This cable should be a ribbon cable with every other wire ground.

IP-125/225 EIA Connector		Interface Circuit	H8-2
PIN	NAME	PIN	PIN
3	STROBE	-	11
7	SIGNAL GROUND	-	9
19	ACKNOWLEDGE	1	10
14	DATA BIT 0	-	1
13	DATA BIT 1	-	2
12	DATA BIT 2	-	3
11	DATA BIT 3	-	4
10	DATA BIT 4	-	5
9	DATA BIT 5	-	6
15	DATA BIT 6	-	7

PARTS LIST

(1)	74LS14	INTEGRATED CIRCUIT
(1)	74123	INTEGRATED CIRCUIT
(1)	3300 ohm 1/4W	RESISTOR
(1)	33K 1/4 W RESISTOR	
(1)	.0012UF DISC	CAPACITOR
(1)	.1UF DISC CAPACITOR	

EOF

CONSULTANT'S CORNER

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

Q: Certain keys on my H-9 terminal print double or triple, causing errors in my program. How can I stop this?

A: Capacitor C405 (keyboard circuit board) should be changed from .001 uF to .005 uf ceramic. This increases the length of time required for a keystroke to be considered valid by the encoder. In extreme cases, a .01 uF capacitor can be used, but this may limit the maximum typing speed that can be used on the keyboard. If individual keys continue to "bounce" after this mod, they should be replaced.

Q: I want to use the H-14 printer with my H8-4 serial card. I understand that the port should be 340, but what interrupt should I use?

A: If you are using cassette software, you should enable interrupt 3 for that channel. If you are using the disk system, the interrupt will be ignored by the system; but it should be disabled if you do not also use cassette software.

Q: When using EDIT, I sometimes get "No Free Space on Media" when I finish the editing and BYE, even though I haven't made the file any larger. Why does this happen, and how can I get around it?

A: HDOS will normally not delete your old file until the new one has been completely written (for safety reasons). This requires that there be enough space on the disk for 2 copies of the program. You can defeat this safety by giving a second NEWOUT command, using the same filename. The original file will be immediately deleted, and the space will be reclaimed. NOTE: you must have least 2 free sectors for this to work (because of the second NEWOUT), and if the BYE operation fails anyway (because your file became larger while EDITing) you will have no copies on disk.

Q: I have an H-8 disk system and an H8-2 parallel I/O card. I would like to use a parallel printer, but the only device drivers supplied are for H8-4 or H8-5 cards. How can I get the system to use my H8-2?

A: The device driver for the H8-5 (ATH85.DVD) works equally well with the H8-2 parallel card. Be sure to assign the correct port number (using SET) and disable interrupts on the H8-2. Full hardware handshaking will be required of your printer.

Q: I use HASL-8 with my H-8 cassette system. Sometimes, a particular program will produce "SEQUENCE ERROR" after repeated tries through the assembler. Other programs assemble correctly. What could cause this?

A: You probably forgot to include the END statement. HASL-8 will continue to read the source tape until it encounters the END statement. If none is present, the tape will be read past the end-of-file, causing the error when it encounters the next file on the tape.

EOF

BP.DVD WITH THE H8-5 CARD

Philip Little
639 Maryanna Lane
Monrovia CA 91Ø16

Now that you have a floppy disk and an H8-4 multiport serial I/O card, perhaps you feel that you don't need yur H8-5 serial and cassette interface card anymore. Before you pull it from your system forever, consider this. First of all, many of your favorite cassette programs will not run under HDOS. It would be worth keeping the card in the computer for this reason alone. Secondly, the now unused serial interface port can be put to good use. It can be used to interface your H8 to an inexpensive baudot printer. This means you do not have to take up another card slot in your computer with a parallel interface card in order to run a baudot machine. Even if you already have an H14 printer, it is still nice to have something to make temporary copies of programs and text that does not use expensive printer paper.

The H8-4 can be used to interface with a baudot machine, but BP-DVD must be altered considerably. For those of us who are more hardware oriented, it might be easier to use BP.DVD with as few changes as possible and modifying the H8-5. This can be done by disconnecting the baud rate generator on the card and replacing it with the 555 timer portion of the circuit in Howard Nurse's article "An ASCII/BAUDOT DRIVER for the H8 System" in REMark issue 2. The circuit is so compact that it can be put on a small circuit board and mounted on the face of the H8-5 interface card. I used the connections themselves to hold the board in place. You may want to use better construction practices. Just make sure that the 5Ø K pot shaft faces up so that it can be adjusted. Power for the circuit is available on the H8-5 card.

Pin three of the 555 timer goes to the "Serial I/O TX Speed" hole on the H8-5 card. The other hardware change that needs to be made is to disconnect the interrupt jumper that was used when the card was used with your console. I also changed the port address for 372 to 374 because I was not sure if HDOS would be confused if I used a port that was assigned as an alternate terminal for something else.

I am using a model 11 teletype with an old tube type terminal unit. It was easy to interface with RS-232. All I had to do was connect the RS-232 output across the grid of the keying tube. If you do not have a terminal unit, a keying transistor and a power supply will probably work as well.

The conversion from ASCCI to BAUDOT is performed by BP.DVD from the HUG special H8/H17 software releases. The only changes necessary to the software are:

1. Change the data port number from Ø7Ø to 374. (Ø7Ø cannot be used because the first number must be a three if you wish to keep your cassette interface.)

2. Make sure the I/O port is initialized for a baudot word. (BPDVD comes configured for an 8 bit word.)

That's all there is to it. Teletype paper is cheap and you can print all you want without running up a big paper bill.

EOF

REQUIRED PATCHES FOR HASL-8

```
UPGRADING  Ø4.Ø5.ØØ
TO         Ø4.Ø5.ØØ.1

NOTES:  This patch corrects the page numbering problem when HASL-8
        is used with the H14.
-----------------------------------------------------------------

Ø53364  315  Ø27  Ø71
Ø71Ø27  Ø41  264  Ø7Ø  247  311
Ø75ØØ1  ØØ1

-----------------------------------------------------------------
```

PERMUTATIONS FOR THE H8

John Beetem
856 Allardice Way
Stanford CA 94305

This article describes an efficient method for generating all the permutations of N things. As examples, the method is applied to the Eight Queens problem and scrambled word games. 8080 assembly language programs are given for solving permutation problems on the H8.

A permutation is "an ordered arrangement of a set of objects." (Webster) The number of different permutations of a set of N objects is N!. ("N factorial".) N! is equal to the product of all the positive integers up to and including N. (6! = 6x5x4x3x2x1 = 720) 0! is defined to be 1: there is only one permutation of zero objects.

There are many problems which can be solved by generating all the permutations of a set of objects and doing something with each permutation. Two such problems are the EIght Queens problems and scrambled word games. Since N! increases extremely quickly as N increases, humans are not well suited to solving these problems by the above method. However, computers are well suited, provided that N does not get too large.

THE EIGHT QUEENS PROBLEM

(see BYTE Oct. 1978, Jan. Feb. Aug. 1979)

The Eight Queens problem consists of placing 8 queens on a chess board (8x8) so that no queen can take any other queen. Since a queen can take another piece horizontally, vertically, or diagonally, the solution is not trivial. However, there are in fact 92 possible solutions!

One method of solution: No queen could be in the same row or column as any other queen. Since there are eight rows, eight columns, and eight queens, there must be exactly one queen in each row and one queen in each column. The chessboard can be represented as an eight element array Q such that the queen in column "i" is in row Q(i). For example, if the Q array contains (1, 3, 2, 4, 5, 8, 7, 6), the chessboard looks like:

```
1    Q +   +   +   +
2    +   Q   +   +
3      Q   +   +   +
4    +   + Q +   +
5      +   + Q +   +
6    +   +   +   + Q
7      +   +   + Q +
8    +   +   + Q +
```

By representing the board using the Q array, we have made sure that no queen can take another horizontally or vertically.

The next step is to take all permutations of the Q array, and for each permutation see if any queen can take another diagonally. If no queen can take another, then the permutation is printed out.

SCRAMBLED WORD GAMES

There are many games which require the players to make words by taking permutations of letters. Some of these games are: Anagrams, Scrabble, and Jumble.

One way to find a valid permutation (i.e. an English word) given a set of letters, is to output all permutations and have a human choose a valid word. Unfortunately, most of the permutations are invalid so this method is very time consuming. One way to speed things up is not to output words that start with pairs of consonants which do not appear in English (e.g. "ds", "jl".)

PERMUTATION METHOD

Program QUEENS is used to solve the Eight Queens problem. QUEENS calls a subroutine PERM, which generates all the permutations of an array of up to 255 bytes. PERM processes each permutation by calling a subroutine VISIT, which is changed for each problem. In the Eight Queens problem, VISIT makes sure that no queen can take another diagonally, and prints the permutation if this is so.

PERM is a recursive subroutine: it calls itself. PERM does the following when called:

(1) If the array to be permuted contains one byte, then go to VISIT, since there is only one permutation of one thing.

(2) If the array contains N 1 bytes, then permutations are computed as follows:

  (a) Leave the first byte in the array alone, and find all the permutations of the remaining N-1 bytes by calling PERM.

  (b) Switch the first byte with the second byte, then find all permutations of the last N-1 bytes by calling PERM, and then switch the two bytes back.

  (c) Switch the first and third bytes, call PERM, and switch them back.

  .
  .
  .

  (n) Switch the first and Nth bytes, call PERM, and switch them back.

That is the whole algorithm. It may take some thought and experimenting with small arrays to convince yourself that this actually works.

INITIALIZATION

The main program for QUEENS is ridiculously simple. All it does is point HL to the beginning of the Q array and point BC to one past the end of the Q array. The Q array tells where the queens are: the queen in column "i" is in row Q(i). The row number is indicated by an ASCII character, '1' - '8'. This makes it easy to print the permutations, and makes no difference if two row numbers are subtracted.

VISIT

VISIT is called each time a new permutation of Q is ready. VISIT uses a double loop to see if any queens are on the same diagonal. (We already know that two queens cannot be in the same row or column.) Two queens are on the same diagonal if (Q(i)-Q(j)) = (i-j) and i<>j.

If no two queens are on the same diagonal, then the Q array is printed on the output device with one space between each digit. (See BYTE Jan. 1979, p.162 for similar results.)

PERFORMANCE

On my H8, it takes 20 seconds to solve the Eight Queens problem with no output. It takes 39 seconds if results are displayed.

JUMBLE

Program JUMBLE is used to solve scrambled word problems. It works as follows: the H8 waits for the user to enter a word through the console terminal. (The word maybe scrambled.) The RETURN key is used to terminate the string of characters. Subroutine PERM is then used to generate every possible permutation of the letters. The VISIT routines checks the first two letters of the permutation to see if they are an invalid pair of consonants. TABLE is used to store the valid combination. If the first two letters are valid, then the permutation is printed on the console.

COMMENTS ABOUT THE PROGRAMS

Both programs use the standard Heath console driver (that is, the old standard before the H8-4!) The console driver must be loaded with JUMBLE or QUEENS. By the way, the listings are in lower case because an assembler other than HASL-8 was used to assemble them.

Jumble and Scrabble are registered trademarks.

```
        Title   'QUEENS'
*
        org     41200A
*
$wchar  equ     40147A
$prscl  equ     40152A
*
Queens  call    $prscl
        mvi     a,012Q
        call    $wchar
        lxi     h,Q
        lxi     b,Q+8
        call    perm
        hlt
        jmp     Queens
*
Q       db      '12345678',15Q
*
* subroutine PERM
* generates all possible permutations of an array of bytes
*
* HL=address of first byte; unaffected by subroutine.
* BC=address of last byte+1; B is destroyed, C is unchanged. 1<=BC-HL<256
* AF,DE destroyed by subroutne.
*
* Calling sequence:
*
*
* [load HL]
* [load BC]
* Call Perm
*
perm    mov     a,l                 If there is only one byte to permute (trivial case)
        inr     a
        cmp     c
        jz      visit               Then process permutation
        mov     d,h                 DE points to beginning of array
        mov     e,l
loop    mov     b,m                 Switch byte at beginning of array
        ldax    d                   with byte pointed to by DE
        mov     m,a
        mov     a,b
        stax    d
        push    h                   Save HL,DE
        push    d
        inx     h                   Permute array with one fewer element
        call    perm
        pop     d                   Restore DE,HL
        pop     h
        mov     b,m                 Switch bytes back
        ldax    d
        mov     m,a
        mov     a,b
        stax    d
        inx     d                   Move DE to next element of array, so that if will be
        mov     a,e                 first on the next cycle
        cmp     c                   If we are not at the end of the array
        jnz     loop                Then loop back.
        ret                         Else return to previous level
*
```

```
* VISIT
* If no queens are on the same diagonal, thn output the Q array
Visit   lxi     h,Q              For i=1 to 7   [BASIC equivalent]
Vis1    mov     d,h                For j=i+1 to 8
        mov     e,l
        inx     d
* Same diagonal?
Vis2    ldax    d                    If (Q(j)-Q(i)
        sub     m
        mov     b,a
        mov     a,l                        = i-j
        sub     e
        cmp     b
        rz
        cma                              or Q(j)-Q(i) = -(i-j) )
        inr     a
        cmp     b
        rz                              Then RETURN
        inx     d                      Next j
        mov     a,e
        cmp     c                      [C points to one past end of Q]
        jnz     vis2
        inx     h                Next i
        mov     a,l
        inr     a
        cmp     c
        jnz     vis1
* Print permutation
        lxi     h,Q              HL poins to Q array
Vis3    mvi     a,' '            print space
        call    $wchar
        mov     a,m              print Q(i)
        inx     h                i=i+1
        call    $wchar
        mov     a,m              End of array?
        cpi     15Q
        jnz     vis3
        call    $wchar           Print <cr>
        mvi     a,12Q
        call    $wchar           Print <lf>
        ret
*
        end     Queens
```

```
        Title   'JUMBLE'
*
        org     41200A
*
$rchar  equ     40144A
$wchar  equ     40147A
$prscl  equ     40152A
*
Jumble  call    $prscl
        mvi     a,012Q
        call    $wchar
        lxi     h,text
        mov     b,h
        mov     c,l
loop1   call    $rchar
        call    $wchar
        stax    b
        inx     b
        cpi     15Q
        jnz     loop1
        mvi     a,12Q
        stax    b
        dcx     b
        mvi     d,14q
loop2   call    $wchar
        dcr     d
        jnz     loop2
        call    perm
        call    $rchar
        jmp     jumble
* subroutine perm
* generates all possible permutations of an array of bytes
*
* HL=address of first byte; unaffected by subroutine.
* BC=address of last byte+1; B is destroyed, C is unchanged. 1<=BC-HL<256
* AF,DE destroyed by subroutne.
*
```

```
* Calling sequence:
*
*
* [load HL]
* [load BC]
* Call Perm
*
perm    mov     a,l             If there is only one byte to permute (trivial case)
        inr     a
        cmp     c
        jz      visit           Then process permutation
        mov     d,h             DE points to beginning of array
        mov     e,l
loop    mov     b,m             Switch byte at beginning of array
        ldax    d               with byte pointed to by DE
        mov     m,a
        mov     a,b
        stax    d
        push    h               Save HL,DE
        push    d
        inx     h               Permute array with one fewer element
        call    perm
        pop     d               Restore DE,HL
        pop     h
        mov     b,m             Switch bytes back
        ldax    d
        mov     m,a
        mov     a,b
        stax    d
        inx     d               Move DE to next element of array, so that if will be
        mov     a,e             first on the next cycle
        cmp     c               If we are not at the end of the array
        jnz     loop            Then loop back.
        ret                     Else return to previous level
*
Visit   lda     text            If first char is vowel
        call    vowel
        jz      display         Then display line
        cpi     's'             Display all words that start with s
        ani     37Q             Convert consonant to table index
        add     a
        add     a
        add     a
        lxi     h,table-15
        mvi     d,0
        mov     e,a
        dad     d
        lda     text+1          Get second char
        call    vowel           If vowel
        jz      display         Then display line
        cpi     'y'             If y then display
        jz      display
        mvi     d,6             See if in table
loop3   cmp     m
        jz      display
        inx     h
        dcr     d
        jnz     loop3
        ret
* Display Line
Display lxi     h,text
Dis1    mov     a,m
        inx     h
        call    $wchar
        cpi     12Q
        jnz     dis1
        ret
*procedure vowels
* "Z" set on vowel
vowel   ori     40Q             convert to lower case
        cpi     'a'
        rz
        cpi     'e'
        rz
        cpi     'i'
        rz
        cpi     'o'
        rz
        cpi     'u'
        ret
```

```
*
table   db      'blr-----'
        db      'chlrz---'
        db      'drw-----'
        db      'e-------'
        db      'fjlr----'
        db      'ghlnr---'
        db      'h-------'
        db      'i-------'
        db      'j-------'
        db      'khlnr---'
        db      'll------'
        db      'mn------'
        db      'n-------'
        db      'o-------'
        db      'phlnrst-'
        db      'q-------'
        db      'rh------'
        db      's-------'
        db      'thrwz---'
        db      'u-------'
        db      'v-------'
        db      'whr-----'
        db      'x-------'
        db      'yt------'
        db      'zl------'
*
text    ds      10
        end     Jumble
```

EOF

LP.SYS THAT 'KNOWS' ABOUT TABS, PAD CHARACTERS

```
.W.TITLE LP  V01/HT-11
; HT-11 LINE PRINTER (LP) HANDLER
;
;
;THIS VERSION WRITTEN BY:     CHUCK SADOIAN
;                             PO BOX 112
;                             DINUBA, CALIF   93618
;JULY 16, 1979
;
;SUPPORTS THE FOLLOWING OPTIONS:
;                 WIDTH
;                 CR
;                 FORMO
;                 HANG
;                 LC
;                 CTRL
;                 TAB (NEW)
;                 PAD  (NEW)
;THE SET LP TAB CAUSES THE HANDLER TO PASS TAB CHARACTERS DIRECTLY
;TO THE LINE PRINTER, WHILE THE SET LP NOTAB CAUSES THE HANDLER TO
;SIMULATE TABS WITH THE PROPER NUMBER OF SPACES.
;
;THE SET LP PAD=N SETS THE NUMBER (N) OF PAD CHARACTERS (NULLS) THAT ARE
;SENT AFTER A CARRIAGE RETURN.  NOTE THAT YOU CAN CHANGE THE CHARACTER
;REQUIRING THE PAD CHARACTERS FROM A CR TO ANY OTHER CHARACTER BY SETTING
;'PADCHAR' TO EQUAL THE OCTAL CODE OF THAT CHARACTER.

LP$CSR=177514
HDERR   = 1
CR      = 15
LF      = 12
FF      = 14
HT      = 11
CNTRLQ  =        21                    ;XON CHARACTER
CNTRLS  =        23                    ;XOFF CHARACTER
COLSIZ  =        132
PADCHAR =        15
;DEFINE REGISTERS
R0=%0
R1=%1
R2=%2
R3=%3
R4=%4
R5=%5
SP=%6
PC=%7

 .ASECT
 .= 400
        .WORD   30.
        .RAD50  /WIDTH /
        .WORD   <O.WIDTH-400>/2+40000
        NOP
        .RAD50  /CR    /
        .WORD   <O.CR-400>/2+100000
        NOP
        .RAD50  /FORMO /
        .WORD   <O.FORMO-400>/2+100000
        BMI     LPERR-ERROPT+.
        .RAD50  /HANG  /
        .WORD   <O.HANG-400>/2+100000
        .WORD   40
        .RAD50  /LC    /
        .WORD   <O.LC-400>/2+100000
        BNE     IGNORE-CTROPT+.
        .RAD50  /CTRL  /
        .WORD   <O.CTRL-400>/2+100000
        BEQ     TABSET-TABOPT+.
        .RAD50  /TAB   /
        .WORD   <O.TAB-400>/2+100000
        .WORD   10.
        .RAD50  /PAD   /
        .WORD   <O.PAD-400>/2+40000
        0
O.WIDTH:MOV     R0,COLCNT
        MOV     R0,RSTC+2
        CMP     R0,R3
        RTS     PC
O.CR:   MOV     (PC)+,R3
         BEQ    RSTC-CROPT+.
        MOV     R3,CROPT
        RTS     PC
O.FORMO:MOV     (PC)+,R3
         BEQ    BLKO-FFOPT+.
        MOV     R3,FFOPT
        RTS     PC
O.HANG: MOV     (PC)+,R3
         BMI    RET-ERROPT+.
        MOV     R3,ERROPT
        RTS     PC
O.LC:   CLR     R3
        NOP
        MOV     R3,LCOPT
        RTS     PC
O.CTRL: MOV     (PC)+,R3
         BNE    PC1-CTROPT+.
        MOV     R3,CTROPT
        RTS     PC
O.TAB:  MOV     (PC)+,R3
         BEQ    HDWTAB-TABOPT+.
        MOV     R3,TABOPT
        RTS     PC
O.PAD:  MOV     R0,R3
        NOP
        MOV     R3,PAD
        RTS     PC
.ENABL LSB
BEGIN:  MOV     @#204,@#200
        MOV     @#206,@#202
        SUB     #LPINT-LRINT,@#200
        MOV     LPCQE,R4
        ASL     6(R4)
        BCC     LPERR
RET:    BIS     #100,@LPS
        BIS     #100,@#LP$CSR-4         ;ENABLE INPUT SIDE
        RTS     PC
;
;INPUT SIDE OF SERVICE ROUTINE

LRINT:  MOVB    @#LP$CSR-2,R4
        BIC     #177600,R4
        CMP     #CNTRLQ,R4
        BNE     2$
        BIS     #100,@LPS               ;ENABLE PRINTER AFTER CNTRL Q
        BR      1$
2$:     CMP     #CNTRLS,R4
        BNE     1$
        CLR     @#LP$CSR
1$:     BIS     #100,@#LP$CSR-4
        RTI
        BR      LPDONE
LPINT:  JSR     R5,@$INPTR
        .WORD   ^C<4>&340
        MOV     LPCQE,R4
        TST     @(PC)+
LPS:    .WORD   LP$CSR
ERROPT: BMI     RET
        TST     (R4)+
FFOPT:  BEQ     BLKO
        TST     (R4)+
LPNEXT: TSTB    @LPS
        BPL     RET
        TST     PADCNT
        BEQ     3$
        DEC     PADCNT
        CLR     R5
        BR      PC1
3$:     ASLB    (PC)+
TABFLG: .WORD   0
        BNE     TAB
IGNORE: MOVB    @(R4)+,R5
        BIC     #177600,R5
        TST     (R4)
        BEQ     LPDONE
        INC     (R4)
        INC     -(R4)
        TST     R5
        BEQ     IGNORE
        CMPB    #40,R5
        BHI     CHRTST
        CMPB    #140,R5
        BHIS    PCHAR
        SUB     (PC)+,R5
.ASECT
.=1000

LRSTRT: .WORD   204
        .WORD   LPINT-.
        .WORD   340
LPSYS:
LPLQE:  .WORD   0
LPCQE:  .WORD   0
LCOPT:  40
PCHAR:  DEC     (PC)+
COLCNT: .WORD   COLSIZ
        BLT     IGNORE
        ASLB    (PC)+
TABCNT: .WORD   1
        BEQ     RSTTAB
PC1:    CMPB    R5,#PADCHAR
        BNE     PRINT
        MOV     PAD,PADCNT
PRINT:  MOVB    R5,@(PC)+
LPB:    .WORD   LP$CSR+2
        BR      LPNEXT
CHRTST: CMPB    #HT,R5
TABOPT: BEQ     TABSET
        CMPB    #LF,R5
        BEQ     RSTC
        CMPB    #CR,R5
CROPT:  NOP
        CMPB    R5,#FF
CTROPT: BNE     IGNORE
RSTC:   MOV     #COLSIZ,COLCNT
RSTTAB: MOV     #1,TABCNT
        BR      PC1
TABSET: MOV     TABCNT,TABFLG
TAB:    MOV     #40,R5
        BR      PCHAR
HDWTAB: ASLB    TABCNT
        BEQ     RSTTAB
        DEC     COLCNT
        BR      HDWTAB
BLKO:   INC     -(R4)                   ; BLKO ONLY ONCE
        CMP     (R4)+,(R4)+             ; POINT TO NEXT CHARACTER
        MOV     FF,R5
        BR      RSTC
LPERR:  BIS     #HDERR,@-(R4)
.DSABL  LSB
LPDONE: CLR     @#LP$CSR-4
        CLR     @LPS
        MOV     PC,R4
        ADD     #LPCQE-.,R4
        MOV     @#54,R5
        JMP     @270(R5)
PADCNT: .WORD   0
PAD:    .WORD   0
$INPTR: .WORD   0
LPHSIZ=.-LRSTRT
.END
```

```
INSTALLATION INSTRUCTIONS

After you've typed in the source, so:
.R ASEMBL
*LP.OBJ=LP.MAC
ERRORS DETECTED: 0
FREE MEMORY: NNNNN. WORDS
*^C
.R LINK
*SY:LP.SYS=LP.OBJ
 *^C
.R PATCH
PATCH H01-2
FILE NAME--
*SY:MONITR.SYS/M
*17000;0R
*0,13632/     400    466
*E
.
DONE!x
```

EOF

MEETINGS and CLUB NOTICES

TUCSON ARIZONA

The Pima Community College Computer Club (PC4) has been formed at the East Side Campus at 7830 East Broadway and meets the Second Friday of each Month at 7:30 PM. Several system demonstrations have been held and more are planned. Contact Mike Blicharz (749-9157) or Saul Levy (793-0670).

HAWAII — HUGH

HUGH is not very hugh... but strong and you can contact the Hawaiian group at the Aloha Computer Club, Inc. at P.O. Box 4470 Honolulu 96813 or phone them at 808-254-2319 after 5 PM Hawaiian time.

DANVERS MA

Growing steadily since the fall of 78, HUG Northshore has over 200 programs in the library and meets the second Wednesday of each month at (7:00 PM) at the HILL TECH Bldg. 88 Holten Street (3rd Floor). For a free copy of their newsletter write them at P.O. Box 112 Danvers Ma. 01923.

WASHINGTON D.C.

William Johnson wants to form a Users' Group near the D.C. area.. Contact him at P.O. Box 65 Woodbridge Va. 22194.

CANADA

The H8 Users of Ontario have their monthly meeting at the Heath Electronic Centre in Mississauga. For more information concerning their fall schedule, contact Roger Harkness at (416) 429-7847 .

MOVCHR

```
        .TITLE MOVCHR
        .GLOBL MOVCHR
;       CAN BE INVOKED EITHER AS A SUBROUTINE OR A FUNCTION
;
;       CALL MOVCHR (A,B,L)      BY: GARY SIFTAR
;       OR                           9006 S. 199 E. AVE.
;       J = MOVCHR (A,B,L)           BROKEN ARROW OK. 74012
;
;       MOVES FROM BYTE ARRAY 'B' TO BYTE ARRAY 'A' FOR LENGTH 'L', OR
;       UNTIL A NULL BYTE IS ENCOUNTERED IN ARRAY 'B'. IF 'L' IS GREATER
;       THAN ZERO, AND IF A NULL IS ENCOUNTERED IN 'B' BEFORE 'L' IS
;       REACHED, THEN 'A' WILL BE PADDED WITH BLANKS OUT TO A LENGTH
;       OF 'L'.  IF 'L' IS PASSED AS ZERO, THEN .TITLE MOVCHR
        .GLOBL MOVCHR
;       CAN BE INVOKED EITHER AS A SUBROUTINE OR A FUNCTION
;
;
;       MOVES FROM BYTE ARRAY 'B' TO BYTE ARRAY 'A' FOR LENGTH 'L', OR
;       UNTIL A NULL BYTE IS ENCOUNTERED IN ARRAY 'B'. IF 'L' IS GREATER
;       THAN ZERO, AND IF A NULL IS ENCOUNTERED IN 'B' BEFORE 'L' IS
;       REACHED, THEN 'A' WILL BE PADDED WITH BLANKS OUT TO A LENGTH
;       OF 'L'.  IF 'L' IS PASSED AS ZERO, THEN BYTES ARE MOVED UNTIL
;       A NULL IS ENCOUNTERED IN 'B', AND NO PADDING IS DONE.  IF
;       'MOVCHR' IS INVOKED AS A FUNCTION, THEN THE NUMBER OF BYTES
;       MOVED IS RETURNED IN 'J'.  THE VALUE RETURNED IS ONLY THE
;       NUMBER OF BYTES OF DATA MOVED FROM 'B' TO 'A',  AND DOES
;       NOT INCLUDE PADDING (IF ANY).
;
;       NOTE:  ALL THREE ARGUMENTS MUST BE PASSED.  TO DEFAULT THE
;       LENGTH, PASS 'L' AS ZERO.
;
        .MCALL  .REGDEF
        .REGDEF
;
MOVCHR:MOV      2(R5),R1          ;ADDR OF ARRAY A
        MOV     4(R5),R2          ;ADDR OF ARRAY B
        CLR     R3                ;BYTES-MOVED COUNTER
        MOV     @6(R5),R4         ;VALUE OF L
        BEQ     30$               ;BRANCH IF LENGTH PASSED WAS ZERO
;
;       A NON-ZERO LENGTH WAS PASSED
;
10$:    TSTB    (R2)              ;NULL BYTE IN ARRAY B?
        BEQ     20$               ;IF SO, GO TO BLANK-PADDING LOOP
        MOVB    (R2)+,(R1)+       ;MOVE A BYTE FROM B TO A
        INC     R3                ;INCREMENT COUNTER
        CMP     R3,R4             ;HAVE L BYTES BEEN MOVED?
        BLT     10$               ;IF NOT, LOOP AROUND AGAIN
        MOV     R3,R0             ;IF SO, RETURN NUMBER OF BYTES MOVED
        RTS     PC                ;RETURN TO CALLER
;
20$:    MOV     R3,R0             ;RETURN ACTUAL BYTES OF DATA MOVED
25$:    MOVB    #40,(R1)+         ;MOVE A BLANK TO ARRAY 'A'
        INC     R3                ;INCREMENT COUNTER
        CMP     R3,R4             ;HAS ARRAY 'A' BEEN PADDED UP TO L BYTES?
        BLT     25$               ;IF NOT, LOOP AROUND AGAIN
        RTS     PC                ;IF SO, RETURN TO CALLER
;
;       LENGTH WAS PASSED AS ZERO
;
30$:    TSTB    (R2)              ;NULL BYTE IN ARRAY 'B'?
        BEQ     40$               ;IF SO, EXIT LOOP
        MOVB    (R2)+,(R1)+       ;MOVE A BYTE FROM 'B' TO 'A'
        INC     R3                ;INCREMENT COUNTER
        BR      30$               ;LOOP UNTIL FIND NUL BYTE
40$:    MOV     R3,R0             ;RETURN NUMBER BYTES MOVED
        RTS     PC        ;RETURN TO CALLER
        .END
```

EOF

BASIC IDEAS

Douglas H McNeill MD
Poynette Family Practice Center
Poynette,Wi. 53955

```
10 REM: PROGRAM FOR CONVERSION OF A FILE FROM ALL UPPER CASE TO
20 REM: UPPER AND LOWER CASE (E.G.NAME AND ADDRESS FILES)
30 REM: BY D. MC NEILL 30-AUG-79
40 PRINT "ENTER FULL NAME OF FILE FOR CONVERSION TO UPPER/LOWER ASCII"
50 PRINT\PRINT TAB(10);"FILE ";\INPUT F$
60 PRINT\PRINT "ENTER FULL NAME OF FILE FOR OUTPUT"\PRINT
70 PRINT TAB(10);"FILE ";\INPUT F1$
80 OPEN F$ FOR INPUT AS FILE #1
90 OPEN F1$ FOR OUTPUT AS FILE #2
100 IF END #1 THEN 130
110 INPUT #1:Q$\GOSUB 180 \PRINT #2:Q$
120 GO TO 100
130 CLOSE #1\CLOSE #2
140 PRINT\PRINT "HAVE YOU MORE FILES TO CONVERT (Y OR N) ";
150 INPUT Q$\IF Q$<"Y" THEN 170
160 GO TO 40
170 END
180 REM: SBR FOR LOWER CASE ASCII CONVERSION
190 REM: ENTRY WITH Q$\EXIT WITH RECONSTITUTED LOWER CASE STRING AS Q$
200 W9$=""
210 IF Q$<>"" THEN 220 \GO TO 370
220 Q1$=SEG$(Q$,1,1)\W9$=W9$&Q1$
230 FOR X9=2 TO LEN(Q$)
240 X9$=SEG$(Q$,X9,X9)
250 IF ASC(X9$)<65 THEN 350 \IF ASC(X9$)>90 THEN 350
260 IF Q1$=" " THEN 350 \IF Q1$="." THEN 350
270 IF X9>4 THEN 310 \ON X9-1 GO TO 340 ,280,290
280 IF SEG$(Q$,1,2)<>"MC" THEN 340 \GO TO 350
290 IF SEG$(Q$,1,3)<>"MAC" THEN 300 \GO TO 350
300 Y8$=" MC"\GO TO 330
310 Y8$=" MC"\Y9$=" MAC"
320 IF SEG$(Q$,X9-4,X9-1)=Y9$ THEN 350
330 IF SEG$(Q$,X9-3,X9-1)=Y8$ THEN 350
340 X9$=CHR$(ASC(X9$)+32)
350 Q1$=X9$\W9$=W9$&X9$
360 NEXT X9
370 Q$=W9$\RETURN
```

MBASIC IDEAS

How to FIELD sub records. Supposing you want to create a random file with a record length of 64 bytes:

```
FOR I=0 TO 3
 FIELD  1, (I*64) AS DUMMY$, 18 AS NAME$, 26 AS ADDRESS$
    14 AS CITY$, 6 AS ZIP$
NEXT I
  .
  .
  .
FOR I = 0 TO 3
 LSET N$(I)=NAME$(I)
  .
  .
NEXT I
```

MORE IDEAS

INPUT$ | Here's how to use INPUT$. A$=INPUT$(1). The VAL(A$) will return the value of the character typed on the keyboard. The character will not be echoed and no CR is required. A$=INPUT$(4) allows 4 characters to be input without echo (password?).

STRING$ | If you want a string of '-'s across the page, use;
PRINT STRING$(80,"-")

MUSIC BOARD FOR THE H8

David Troendle
7230 Chadbourned Rd
New Orleans LA 70126

A system for music synthesis is now available for the H8. The system consists of a board which plugs into the H8 bus, and support (HDOS) software, including a music compiler, which can synthesize up to four parts of harmony in stereo.

This plug-in module, called the DAC-2, is a dual channel eight-bit D/A (digital to analog) converter circuit with two 6-pole low pass active filters. The active filter eliminates the inherent noise associated with the switching elements. While the DAC-2 has been designed for music synthesis, it can be used in virtually any D/A application including voice synthesis and X-Y plotting. The board has outputs for each channel both before and after the low pass filter. This would allow smooth or stepped voltage output.

The music system comes with the DAC-2 board, a forty-page user manual, and software. The principle components of the software consist of the music compiler (called "COMPOSE") and the player software (called "PLAY"). A utility program (called "WAVE") is also supplied which aids in the analysis of the harmonic components of a wave form. This software makes the creation of music for the DAC-2 board almost as easy as reading sheet music.

The music language is designed so that the constructs of the language are similar to that in standard musical notation. The music language has constructs for a key signature, repeats with multiple endings, da capo (D.C.), dal segno (D.S.), segno ( ), coda symbols ( ), and FINE. These constructs function exactly as their equivalent standard musical notation.

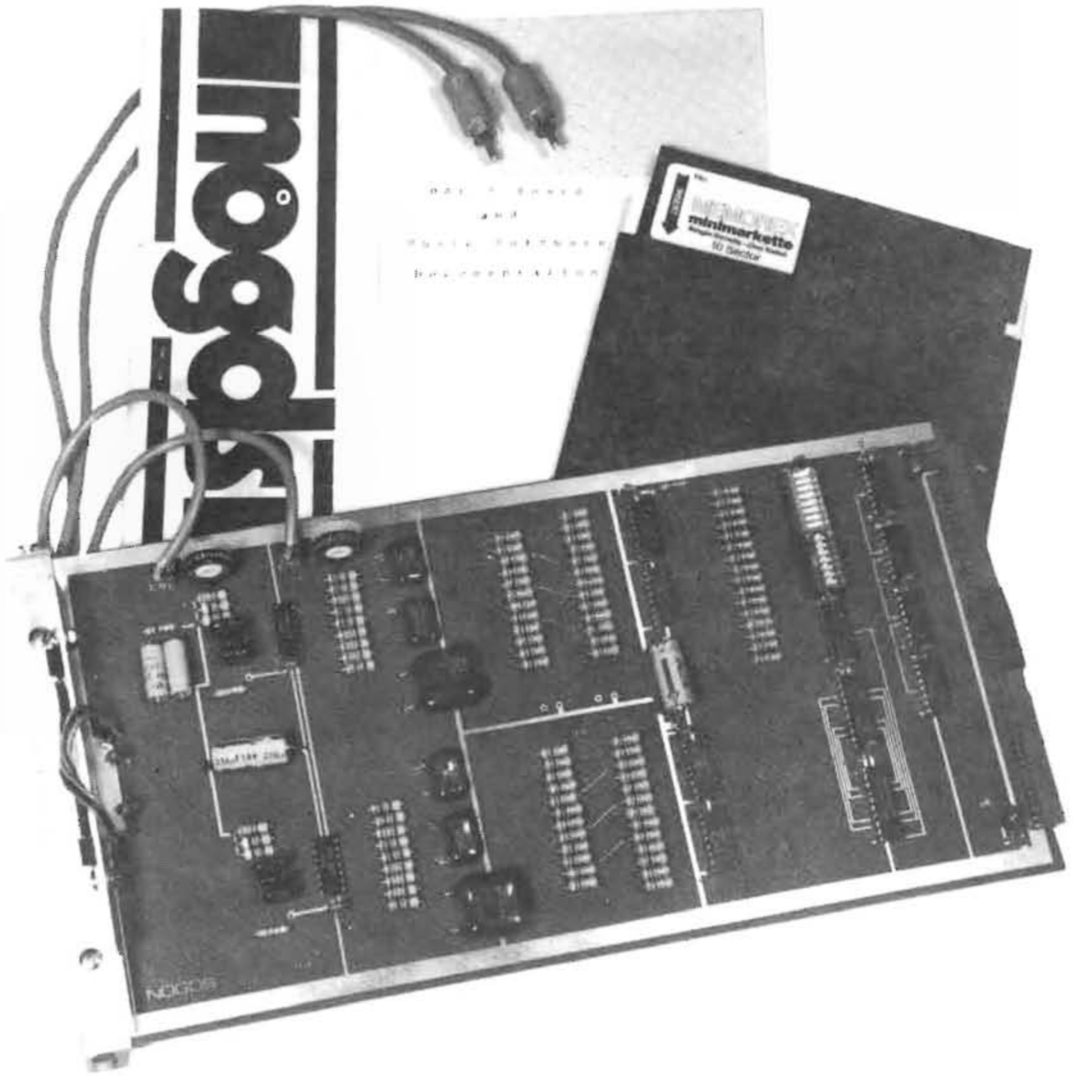
The creation of musical notes is simplified by defaulting its octave and time to the previous note. For instance, if the previous note was a quarter note, then the duration for the current note is assumed to be a quarter note of the same octave unless explicitly specified. By making use of this context dependence, the amount of time required to enter a song as well as the number of errors made is greatly reduced.

It will be necessary to be able to read sheet music in order to write in the music language. With a little experience, one can expect to be able to enter and debug a moderatley complicated song in approximately three hours.

Songs are initially entered in the music language with the aid of the text editor. Next the "COMPOSE" program is run. The speed the song is played, harmonics for each voice, and filename of the song are specified. A play file is produced in approximately one minute. By varying the harmonics and hence the waveform, you can control the timbre (musical quality).

The board, software, and documentation is available from several sources, incuding the Heathkit Electronic Center at 1900 Veterans Memorial Blvd. Kenner La. 70062 for $139.95. All warranty claims, if any, should be directed to the manufacturer.

EOF



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

You can determine your expiration date by examining the last six digits of your ID number — example: 780202 indicates your membership began 02/02/78 and expires one year from then.

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 APPROPIATE BOX AND RETURN TO HUG

			NEW MEMBERSHIP? FEE IS:
RENEWAL RATES			
US DOMESTIC	$11 ☐		$14 ☐
CANADA	$13 ☐	US FUNDS	$16 ☐
INTERNAT'L*	$18 ☐	US FUNDS	$24 ☐

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

THE BACK PAGE —

ET-3400 CONTEST

As promised, here is the software contest for you ET/ETA-3400 users. The prize is a gift certificate redeemable by mail order only for $250 in merchandise. The rules are simple:

1. The program should be one of maximum use and interest to the other members and and be well documented.

2. Easy to implement and use.

Submit your program, along with full operating instructions before January 21 of 1980. Use cassette tape if applicable. If you submit a hand copied listing, please comment each line of code. If you are using any hardware in addition to the ET-3400, please explain and provide drawings.

10.02 or 10.05 BASIC

Many of our newer members have acquired Extended B. H. Basic with their new system. All of HUG cassette software was developed before 10.05 was released and therefore many of the programs will not run under Version 10.05. Furthermore, 10.02 is no longer available from Heath. Well, to solve this problem, we have 10.02 available for five bucks. (P/N 885-1046).

>BYE

HUG PRODUCT PRICE LIST

PART NUMBER	DESCRIPTION	PRICE
	Keep your volumes neat with a:	
266-945	Cassette Holder (holds 2 cassettes)	$ 2.45
885-4	and a 'HUG' binder	$ 5.75
885-1008	Volume I Documentation	$ 9.00
885-1009	Tape I — H8 Cassette Tape	$ 7.00
885-1010	Adventure — H8 Disk	$10.00
885-1012	Tape II — H8 (BASIC) Cassette	$ 9.00
885-1013	Volume II Documentation	$12.00
885-1014	Tape II — H8 (Assembly) Cassette	$ 9.00
885-1015	Volume III Documentation	$12.00
885-1018	HDOS Programming Guide	$ 5.00
885-1019	HDOS Device Driver — H8 Disk	$10.00
885-1022	HDOS Editor — H8 Disk	$15.00
885-1023	RTTY Communication Processor — H8 Disk & Documentation	$22.00
885-1024	Disk I — H8 Software	$18.00
885-1025	Runoff Word Processor — H8 Disk	$35.00
885-1026	Tape III — H8 Cassette Tape Financial & Amateur Packages	$ 9.00
885-1027	Morse 8 — H8 Cassette & Documentation	$14.00
885-1028	RTTY Communications Processor — H8 Cassette & Documentation	$11.00
885-1029	Disk II — H8 Software 'Games 1'	$18.00
885-1030	Disk III — H8 Software 'Games 2'	$18.00
885-1031	Disk IV — H8 'Music' 2 Disks	$23.00
885-1032	Disk V — H8 Misc. Software	$18.00
885-1033	Disk I — HT11 Misc. Software	$19.00
885-1034	Character Editor — H8 Cassette Tape & Documentation	$11.00
885-1035	Co-resident Editor/Assembler — H8 Cassette Tape & Documentation	$11.00
885-1036	Tape IV — H8 Misc. Software*	$ 9.00
885-1037	Volume IV Documentation*	$12.00
885-1038	WISE — H8 Disk Software	$18.00
885-1039	WISE — H8 Cassette Tape	$ 9.00
885-1040	PILOT — H8 Cassette Tape & Documentation	$11.00
885-1041	Small Business Package (2 disks & DOC.)	$50.00
885-1042	PILOT — — H8 DISK	$19.00
885-1043	MODEM — — H8 DISK	$21.00
885-1044	DISK VI — — H8 SOFTWARE	$18.00
885-1045	FOCAL — — H8 CASSETTE	$11.00
885-1046	EXT. BH BASIC 10.02.01	$ 5.00

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.

885-2008