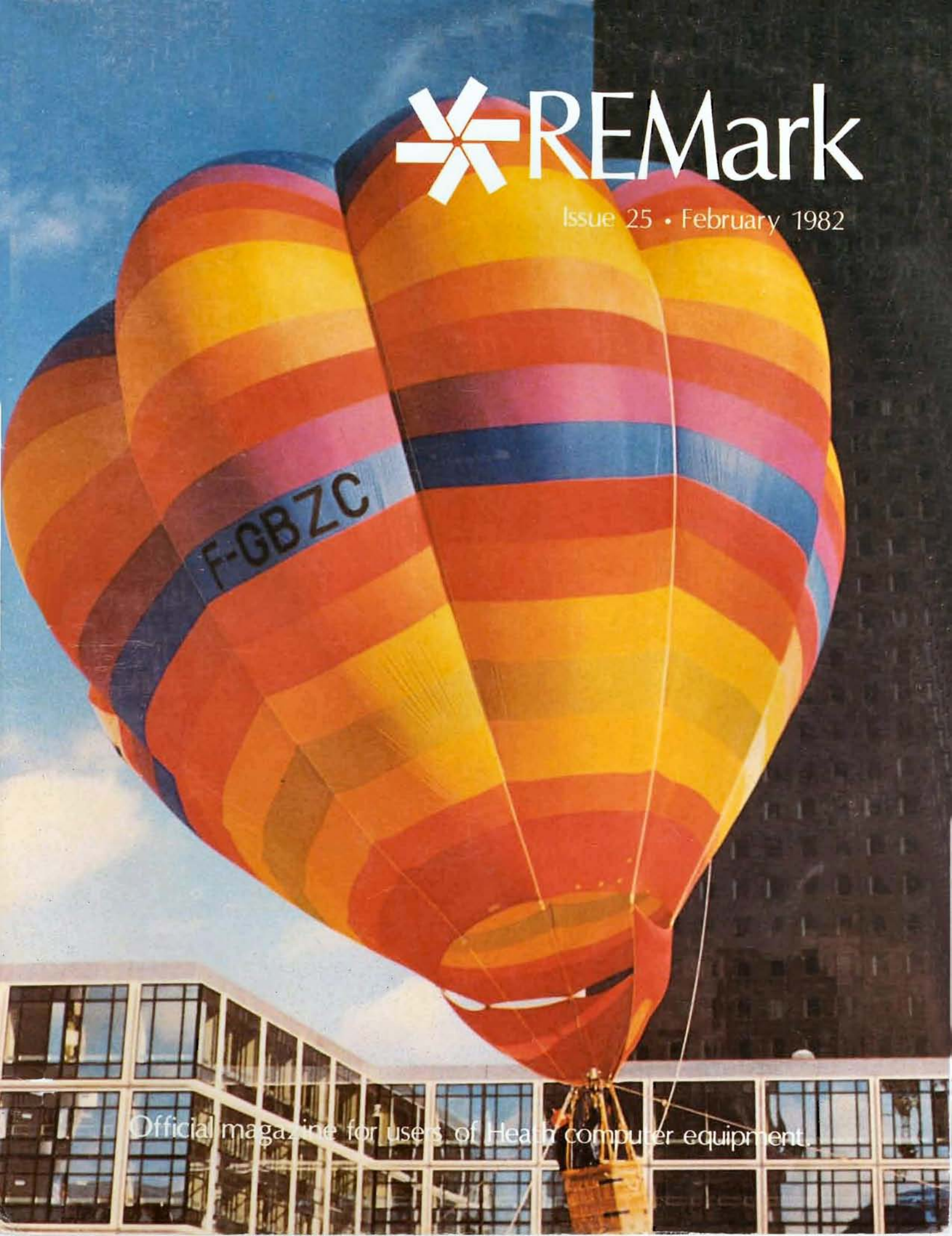

REMark

Issue 25 • February 1982

Official magazine for users of Heath computer equipment.

on the cover . . . .

The General Electric 'information' balloon at the SICOB Computer show in Paris, France.

Photo by John Beran

on the stack

>CAT

"REMark" is a HUG membership magazine published 12 times yearly. A subscription cannot be purchased separately without membership. The following rates apply.

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

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

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

Send payment to:

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

HUG Manager ..... Bob Ellerton
HUG Secretary ..... Margaret Bacon
Software Coordinator and Developer ..... Jim Blake
REMark Editor and Software Developer ..... Pat Swayne
Assistant Editor and Layout ..... Nancy Strunk
HUG Bulletin Board and Software Developer ..... Terry Jensen

REMark

PATCH HUG.GRP

It seems that something is always going on at HUG that causes some great effect on the user community. Well, no difference as we head along into 1982. We have already been delayed by weather around here a couple times and now we lose one of our best people to Zenith just as we are firing off the New Year. I'm speaking of Gerry Kabelman who most of you know through telephone contact with HUG. Gerry has been assigned the task of Marketing Coordinator for Zenith Data Systems. We wish Gerry the best in his new position.

As you can see, the above must be the bad news. How about some good news? Do you remember Jim Blake? Good ole' JB:? If you remember, he is the guy who really got HUG on its feet in the beginning. Well, Jim is rejoining the HUG staff to begin a new program that will be of great interest to those of you who write and use HUG software. Jim will describe this new offering in the following paragraphs and I'm sure that all of us who have known and worked with him will appreciate his return to our little group.

BE:

Jingles in your Jeans

This month marks the fifth anniversary of the Heath Users' Group. And, this magazine has documented the growing pains, successes and yes, the failures. The successes, for the most part, are because of you and your contributions. So, we start this new year with a plan to reward you for your efforts. A plan to put 'jingles in your jeans.' Money. Green money. The kind that DOESN'T jingle.

As you may know, there's a lot of great software out there that isn't in the HUG library because the author feels more may be gained by marketing it through other channels. And that's fine. We hope it continues. Now, however, an author may submit a program to HUG and, if selected for distribution through our various channels, we will pay the author a royalty on each unit sold! Typically 20% of the retail selling price.

Here is how it will work and some of the ground rules.

If you as the author of a program would like HUG to help you market your program, submit it along with complete documentation for our evaluation. If, in our judgement, we can distribute the program and make it a mutually rewarding experience, we will forward to you for your review and signature a simple agreement which outlines the terms and conditions. Then, at the end of each calendar quarter, you reach in your mail box and get your 'jingles'.

Some of the terms and conditions are:

* Since we receive many similiar programs, you must rely on your copyright, patent or other protection because we must be free of any confidentiality obligation.

* The program must be complete, ready to run and include the necessary documentation in a text file on disk as well as in printed form. The best programs are 'beginner ready', are user friendly, and when appropriate, use plenty of graphics or screen formatting.

* We have the right to modify, enhance in any form and will supply you with a copy of the final product.

* The agreement is NON-EXCLUSIVE which means you retain all rights, including the right to license it to someone else.

* The program may be distributed to all HUG members and through any of Heath's or its affiliate marketing channels.

If, for any reason, we decide not to distribute the program under this plan and you do not want the program in the regular HUG software exchange library, all materials will be returned to you. Let us emphasize that this plan is meant to supplement the current HUG software exchange library.

Direct all materials and correspondence to:

Jim Blake, Software Coordinator
Heath Users' Group
Hilltop Road
St. Joseph MI 49085

Phone: (616) 982-3463

And another thing... contributors of major articles that appear in REMark will receive a nice certificate suitable for framing and a one year extention of their membership.

These are just two of the changes and improvements we have planned for 1982. This is really going to be an exciting year! And I'm glad to be a part of it again.

:JB:

Turnkey Operations Using CP/M, MBASIC and SUBMIT

by Robert Conde
PCB Industries
11 Sugarbush Lane
Coram, NY 11727

REMark issue 21 presented two examples detailing how to "link" to a machine language program (ie. CP/M .COM type file). Both methods require that the area occupied by the CP/M CCP (Console Command Processor) not be overlaid by the users' program. Additionally, those methods tend to be CP/M version dependent as they reference absolute values (which may vary from version to version) to calculate the location of the CCP command buffer.

While developing "turnkey" business programs in MBASIC it soon became apparent that it would be very desirable to be able to:

1. Link to any .COM type file from an MBASIC program so that the MBASIC application program would be able to utilize commercially developed and highly efficient specialty programs such as Micropro's Supersort. Such programs are extremely fast and flexible. They are able to perform operations many times faster than would be possible using MBASIC implementations of the same function.

2. Be able to return to the MBASIC applications program from whatever .COM file(s) were run -- even when the source code for those programs is not available.

3. Allow the original BASIC program to dynamically determine which machine language programs would be run.

4. Be able to chain or link several .COM programs together as desired.

5. Perform all the above without operator interaction.

The method used to accomplish these goals uses the CP/M SUBMIT facility along with a little known fact concerning CP/M. When CP/M preforms a "warm boot" (which normally occurs when exiting .COM type programs), the operating system always searches Drive A for a file named $$$.SUB. If such a file exists CP/M will try to execute it. The only stipulations are:

1. The CP/M SUBMIT.COM program must also reside on Drive A.

2. The file $$$.SUB must contain acceptable commands as specified in the CP/M SUBMIT documentation. In addition, the first character in the file must be a count of the characters in the command(s), and the last two characters must be a null (zero) and the character "$".

The procedure explained here involves creating a file on Drive A (before exiting MBASIC) that will in turn link to a valid CP/M SUBMIT file that contains commands that will be sequentially executed. (Note that if you want to execute only one .COM file, you can link directly to it rather than to a submit file.) The following MBASIC routine details what is needed:

```
5000 REM  ROUTINE TO LINK TO .COM FILES
5010 OPEN "O",1,"$$$.SUB":REM  ESTABLISH
     WARM BOOT HOOK
5020REM  FN$ IS NORMALLY DETERMINED BY
     YOUR MAIN PROGRAM.  HERE WE ASSUME
     THAT FN$ WAS SET EQUAL TO "ACTION".
     HOWEVER, ANY VALID CP/M .SUB OR .COM
     FILE NAME IS ACCEPTABLE.
5030 FN$="SUBMIT "+FN$:REM  THE SPACE
     AFTER 'SUBMIT' IS NEEDED.  OMIT THIS
     LINE IF LINKING TO A .COM FILE.
5040 FN%=LEN(FN$):REM  DETERMINE COMMAND
     LENGTH
5050 REM  LOAD COMMAND INTO FILE
5060 PRINT #1,CHR$(FN%);FN$;CHR$(0);"$"
5070 REM  EXIT TO CP/M AND EXECUTE THE
     FILE.
5080 CLOSE #1:SYSTEM
```

When the routine reaches line 5080, the system will warm boot, and the file $$$.SUB will be run which in turn will execute the SUBMIT file "ACTION.SUB". It is , of course, necessary to have previously created the file ACTION.SUB using any convenient text editor. An example of how such a file might appear follows:

```
HSORT B:DATA.DAT B:NU.DAT
REN B:OLD.DAT=DATA.DAT
REN B:DATA.DAT=NU.DAT
MBASIC MENU /F:4
```

This sequence loads and executes the HUG CP/M sorting program HSORT, sorting the file DATA.DAT and placing the result in NU.DAT. After leaving HSORT, DATA.DAT is renamed to OLD.DAT and NU.DAT is renamed to DATA.DAT. Then MBASIC is reloaded and the program MENU.BAS is run.

Using this method of program transfer, a great deal of flexibility and power are available to the MBASIC programmer.

EOF

Patch Page

ACKACK PATCH

The ACKACK game from HUG disk no. 885-1112 contains an error that can cause it to crash randomly. Use PATCH.ABS (supplied with HDOS) to fix it with the patch that follows. We have listed the entire patch session, with user input in **bold print.**

```
>PATCH

PATCH Issue #50.06.00.

File Name? SY1:ACKACK.ABS

Address? 50034
050034 = 312/301
050035 = 010/312
050036 = 050/010
050037 = 062/050
050040 = 044/062
050041 = 050/044
050042 = 301/050
050043 = 311/^D
Address? ^D
Patch Issue #50.06.00.

File name? ^D
>
```

The symbols "^D" mean CTRL-D (hold down the CTRL key and type D). The above example assumes that you have ACKACK.ABS on a disk in SY1:. Do not make the patch if the old data (the numbers just before the slashes) in your copy of ACKACK do not match those shown above. Our stock is being updated, and you may have a corrected copy.

HUG'S SY.DVD AND THE H8 Z80 BOARD

HUG's SY: device driver (885-1095) does not work properly with the new Z80 board (HA-8-6) from Heath. The problem only affects disks that were initialized from a system disk containing the new SY.DVD and then sysgened. They will not boot on an H8 with the new Z80 board. To correct the problem, make the following patch to SY.DVD using PATCH.ABS:

Address	Old data	New data
006141	062	311

The patch can also be made using DUMP (from 885-1062) or a similar utility. Patch the 7th sector of SY.DVD (sector 006).

Address	Old data	New data
0661	32	C9

After the patch is made, disks you initialize will boot properly on an H8 with the Z80 board. You can also fix disks that were initialized before you made the patch by patching track 0 sector 0 as follows:

Address	Old data	New data
5B	32	C9

These disks will now boot properly if they are not otherwise damaged.

PATCHES TO CAT AND CCAT

The following patches are improvements to the alphabetizing directory programs CAT.COM from 885-1213 and CCAT.ABS from 885-1090. The patches should be made to the .ASM files, which should then be re-assembled.

The first patch is for CAT, and results in slightly faster operation. The patch area is below the label NOTDBL, just before CALL JBIOS, and is shown here in bold print.

```
        LDA     NEWDRV  ;GET DRIVE
        MOV     C,A     ;IN C
        INR     E       ;NOT FIRST SELDSK
        CALL    JBIOS   ;CALL SELDSK IN BI
```

This patch puts an odd number in E, which causes the BIOS SELDSK function to only return the disk parameter address rather than performing a complete disk selection.

The next patch is for CCAT and fixes a problem that caused it to give an incorrect free sector report when reading a disk with no free sectors. The patch area is below the label PFILES, just before CALL GETSIZ.

```
        LXI     H,GRT   POINT TO GRT
        MOV     A,M     GET FIRST INDEX BY
        MVI     E,0     ASSUME NO SPACE
        ORA     A       ANY SPACE?
        JZ      NOSPACE
        CALL    GETSIZ  GET UNUSED SIZE
        MOV     E,A
NOSPACE MVI     D,0     DE = SIZE
```

This patch skips the call to GETSIZ if there is no free space. GETSIZ always assumes at least one free group on the disk.

Note: Our stock is being updated with the corrected programs, so your copy may have these changes already made.

PS:

Pascal Corner III

By: Henry E. Fale
QUIKDATA COMPUTER SERVICES, INC.
2918 S. 7th. St.
Sheboygan, WI 53081
(414) 452-4172

Welcome to part three of the Pascal Corner. I am pleased with the response I have received from readers of this column. As a brief review, in part one (Issue 22, November 1981) I covered some general points about Pascal. In part two (Issue 23, December 1981) I continued with some points on the Lucidata Pascal, illustrated a very simple Pascal program, and began building a slightly more involved Pascal program, TAXRATE. So far; Procedures were covered, some reserved words for console input and output, constants and variables, and the basic arithmetic functions.

I want to take a few lines here to give thanks and recognition to Fred Pospeschil. Fred has been proof reading my Pascal Corner before I submit it to REMark, and has been checking my Pascal Syntax. He has caught a lot of errors I would have let slip by. Although Fred bills himself as a 'beginner' Pascal programmer like myself, I find him too be much more of an advanced 'beginner'. Many thanks to Fred for his help.

Let's get started by continuing our program TAXRATE which we will finish in this section. We thus far defined the constants and variables, defined the procedures CLEARSCREEN and CALCULATE, and showed how to perform the necessary math functions by assigning a variable to a calculated value. We will now define the procedures for input and output.

In this program, like most others, there must be some operator input to the program. In this case we will want to input the price, from which the salestax and total price can be calculated. Keep in mind this is a very basic program to get you started. You can expand on these ideas to create a total point of sale package where many item prices may be inputted rather than just one. When this program is finished, you will have all the needed "software tools" to do just that. Since we are interested in input, let's name our procedure thus. You could give it any legal unique name you like. Remember from part two, that you must have a BEGIN and END in the procedure. Here is how I did mine.

```
PROCEDURE INPUT;
    BEGIN
        WRITE ('ENTER SELLING PRICE   ');
        READLN (PRICE);
        WRITELN
    END;
```

In analyzing this, note the procedure name is INPUT and the semicolon is required. Note also how the indenting, although not mandatory, makes the program easier to read and analyze. Get into the habit of doing this. The procedure has a BEGIN and an END to tell where the actual body of the procedure is. Also note the BEGIN has no semicolon, but the END does. This is proper format for procedures. The only time the END will have a period after it is the actual program end.

The first thing that executes in this procedure is the WRITE which causes the 'prompt string', ENTER SELLING PRICE, to be displayed on the console. Note the required semicolon after this line. All procedure lines like this must have a semicolon except the one just before the END statement. The next line, READLN (PRICE); will read the value input from the console and place it in the real variable (previously declared) labeled PRICE. This would be similar in a BASIC program to the line: INPUT"ENTER PRICE";PR . It is important to note here that when using the READLN statement, a Carriage Return is required since READLN means read the entire line. This is in contrast to the READ statement which will only take the first character typed and 'take off', similar to INPUT$(1) in MBASIC. The next statement, WRITELN simply outputs a blank line on the screen, and that's the end of that procedure--easy!

Now that we can input, there has to be a similar way to output to the console as well. We covered this briefly in our first example, but will now expand on the concept. Since I am outputting, I choose to name the procedure OUTPUT. Here's how it will look.

```
PROCEDURE OUTPUT;
    BEGIN
        WRITELN ('PRICE =',PRICE:7:2, '    SALES TAX = ',SALESTAX:5:2);
        WRITELN;
        WRITELN ('TOTAL DUE = ',TOTAL:7:2)
    END;
```

As proper format, the procedure OUTPUT has a BEGIN and END statement, again, note the semicolon placement. The first WRITELN line has introduced some new concepts. This line will cause the string PRICE = to be printed, and then the value which is stored in PRICE. What's this :7:2 all about? It simply tells Pascal to reserve 7 spaces (including decimal place) for the value PRICE, 2 of which are to the right of the decimal place, a necessity for working with dollars and cents. This can be thought of as similar to PRINT USING "####.##" in MBASIC. Since we are still in the same WRITELN statement, the string SALES TAX = will then be printed on the same line, followed by the value stored in the real variable SALESTAX which has also been declared as a VARiable or REAL type. This is allowed 5 places, 2 which are at the right of the decimal place (cents value). A blank line will be output to the console to keep things looking neat by the following WRITELN; statement. The last WRITELN statement is similar to the first, causing the string TOTAL DUE to be printed followed by the value stored in the REAL VARiable TOTAL. This again is alloted 7 spaces, 2 to the right of the decimal. That wasn't hard, was it? It's getting better all the time.

Since I feel it is important, let's dwell a moment on the print formatting. The actual form of this is WRITELN(Variable: field length : places after decimal). The field length indicates how many spaces you want to reserve on the screen for your named variable. The places after decimal is optional. It would not be used, for instance, for an integer number or a string. If the variable you are outputting is shorter than the allowed places for field length, the extra spaces will be filled with spaces to the left of the number. This results in automatic decimal point alignment when formatting dollars and cents. If the variable is larger, it will be displayed in scientific notation. The field length and decimal need not be an actual number, but can be a variable or constant which has been previously defined. For instance,

```
WRITELN (PRICE : CENTS + 5 : CENTS);
```

is perfectly valid. If 'CENTS' had been previously defined as 2, then this would evaluate to WRITELN (PRICE : 7 : 2). A feature of print formatting is that it automatically rounds off numbers. This can be used with strings in a similar way as shown in the following example.

```
STRING := 'THIS IS A TEST';
WRITELN (STRING : 10);
```

The above code will produce the output shown on the following line.

```
THIS IS A
```

There are all kinds of neat tricks and uses of this print formatting feature. Experiment with it and see what you can come up with. Now we return to our TAXRATE program again.

Now everything essential to this program has been covered except for boolean values and characters. A BOOLEAN value is true or false, a logic element. A CHARacter element is a single element used, for instance, to input a 'Y' or 'N' value from the console for YES or NO. In part two we assigned the variable A as a character, and the variable FINISHED as a boolean value. Here is a way to use these in making a decision. I chose this method to be able to go back to the beginning of the program and be able to input several different prices and get the appropriate sales tax and totals. It's much easier than running the program many times, since you stay in the program loop until you are actually done. I will create a new procedure called DONE, which will let me loop back to the beginning of the program when properly called from the program body. Pay close attention as this gets a little tricky.

```
PROCEDURE DONE;
    BEGIN
        WRITE ('DONE?    ');
        READ (A);
        IF (A='Y')
          THEN FINISHED := TRUE
          ELSE FINISHED := FALSE;
    END;
```

WOW! What happened here? Now can you see why I stressed using proper indention? It does not sound important when small simple programs and procedures are used, but in this one 3 levels of indenting help keep things straight. Let's analyze this procedure.

The Procedure DONE of course has its BEGIN and END. After that we cause the prompt string DONE? to be output to the console and READ the one character response (either Y for yes or N for no) in the CHARacter variable A. Please note here, that the READ is used rather than the READLN. This means that the character is placed in A without a Cariage Return needed as mentioned before. If READLN would have been substituted, a CR would have been required before the program would proceed. Now a new statement, the IF statement is introduced, and functions much like that encountered in a BASIC type program. What we are doing here is saying if Y was input, then the the BOOLEAN VALUE of TRUE is assigned to the BOOLEAN VARIABLE FINISHED. If anything other than 'Y' is input, a value of FALSE is assigned to this variable. We then come to the end of the procedure. So all this procedure is actually doing is assigning TRUE to FINISHED if Y was input, otherwise it assigns FALSE to FINISH. Why do we want to do this? The reason will be clear as we discuss the main program body.

You will encounter a new Pascal statement, the REPEAT-UNTIL statement. It tells the computer to REPEAT a number of statements (whatever is in the REPEAT-UNTIL loop) UNTIL a specified condition is true. It is in certain ways similar to the WHILE-DO statement which will be covered in a later Pascal Corner. In fact, a good amount of space will be devoted to program control with decision making. I just wanted to cover the IF-THEN and REPEAT-UNTIL so you have some software tools to work with.

Let's now put the whole program together, and I'll cover anything not previously covered. You can then enter this program and run it, or modify it to your needs.

```
(* PROGRAM LANGUAGE--LUCIDATA PASCAL

   PROGRAM TITLE--TAXRATE

   PROGRAM SUMMARY--CALCULATE TOTAL PRICE AND TAX
                   FROM PRICE ENTERED                    *)

PROGRAM TAXRATE;                  (* PROGRAM START     *)

CONST  TAXRATE = 0.04;            (* WISCONSIN SALES TAX   *)
       ESC     = 27;              (* TERMINAL ESCAPE CODE  *)

VAR    PRICE, SALEXTAX, TOTAL :    REAL;
       A                      :    CHAR;
       FINISHED               :    BOOLEAN;

PROCEDURE CLEARSCREEN;
     BEGIN
          WRITE(CHR(ESC),'E')
     END;

PROCEDURE CALCULATE;
     BEGIN
          SALEXTAX := TAXRATE * PRICE;
          TOTAL    := PRICE + SALESTAX
     END;

PROCEDURE INPUT;
     BEGIN
          WRITE ('ENTER SELLING PRICE    ');
          READLN (PRICE);
```

```
          WRITELN
     END;

PROCEDURE OUTPUT;
     BEGIN
          WRITELN ('PRICE =',PRICE:7:2,'   SALES TAX = ',SALESTAX:5:2);
          WRITELN;
          WRITELN ('TOTAL DUE = ',TOTAL:7:2)
     END;

PROCEDURE DONE;
     BEGIN
          WRITE ('DONE?   ');
          READ (A);
          IF (A='Y')
               THEN FINISHED := TRUE
               ELSE FINISHED := FALSE;
     END;


BEGIN                          (* MAIN PROGRAM BODY STARTS HERE *)
     FINISHED := FALSE;        (* SET BOOLEAN VAR 'FINISHED' TO FALSE *)
     REPEAT
          CLEARSCREEN;
          INPUT;
          CALCULATE;
          OUTPUT;
          DONE;
     UNTIL FINISHED
END.                           (* END PROGRAM *)
```

That's the whole program. The only thing we did not discuss is the main program body. It starts with the lone BEGIN statement and is followed by the REPEAT loop. Hold on a minute. Repeat? What's that? All the program between the REPEAT and UNTIL FINISHED will be done over and over again until the boolean value FINISHED turns up TRUE, which only happens in procedure DONE if a 'Y' is entered in response for DONE?. Pretty neat huh? To start with, finished is assigned boolean FALSE and will hold that value until changed in the procedure DONE. But until it takes on the TRUE value, the program loop will continue to repeat so you can enter values till the cows come home. Every time the loop is entered, you will go through the procedures as listed; CLEARSCREEN, INPUT, CALCULATE, OUTPUT, and DONE. When the FINISHED value is finally TRUE, then control exits the REPEAT loop to the next statement which is the END of the program (note the period)!

That was a very simple program used to demonstrate many basic Pascal keywords and applications. Next month I'll start by printing a small report on the comparisons of four different Pascal languages. Then we'll build a LOAN program to calculate loan payments and interests, and get into VARiables inside procedures. We will also drift into more powerful features like disk I/O and output to a printer device, space permitting. Try the above example, and try your hand at your own simple programs. If you come up with some neat stuff, send it in so I can look at it. If it's good, I'll return it with a gold star! There's much more coming in this Pascal Corner and this powerful language. A month may seem like a long time to wait if you're reading this article, but not if you're writing it, so hang in there! Pascal is here to stay. Until next month-------.

PS: I have a program written in Heath's UCSD Pascal which is a simple disk to printer file copy. Nothing special, but it is easier to use and much faster than invoking the FILER and doing a T)ransfer . This was submitted to me for inclusion in my newsletter H-SCOOP by Terry Smedley/ Route 3 Box 1274C/ Hoquiam, WA 98550. Since I am covering Pascal here and not in my newsletter because of a lack of space, I am throwing in this program as an extra. Since I have already covered most of the Pascal used in this program, you should be able to follow what it is doing.

```
(* TRANSFERS TEXT FILE TO PRINTER *)
(*$L PRINTER:*)
(*$I-*)
PROGRAM PRINT;
VAR  LINE,FILEN          :STRING;
     PRTR,INP            :TEXT;
     LINECOUNT           :INTEGER;
```

```
     FOPOK,DONE          :BOOLEAN;

BEGIN
LINECOUNT := 0;
FORPOK := FALSE;
DONE := FALSE;

REPEAT
     WRITE ('FILENAME TO PRINT: ');
     READLN (FILEN);
     RESET (INP,FILEN);
     IF IORESULT <> 0 THEN
          IF LENGTH(FILEN) > 0 THEN
               BEGIN
               WRITELN('ERROR WHILE OPENING FILE: ',FILEN);
               WRITELN('PLEASE CHECK THE FILENAME.');
               WRITELN;
               END
          ELSE DONE := TRUE
     ELSE FORPOK := TRUE;

UNTIL ((DONE = TRUE) OR (FOPOK = TRUE));

(*$I+*)

IF (NOT DONE) THEN
BEGIN

REWRITE (PRTR,'PRINTER:');
READLN(INP,LINE);

WHILE (NOT EOF(INP)) DO
     BEGIN
     WRITELN(PRTR,LINE);
     LINECOUNT := LINECOUNT + 1;
     IF LINECOUNT = 60 THEN BEGIN
          PAGE(PRTR);
          LINECOUNT := 0;
          END;
     READLN(INP,LINE);
     END;

WRITELN(FILEN,' --> PRINTER:');
PAGE(PRTR);
CLOSE(INP);
CLOSE(PRTR);
END;
END.
```

You will notice no Procedures are used in this program, which is acceptable, although in large programs, may make understanding and de-bugging more difficult. Also notice that UCSD supports STRING variables, which Lucidata does not. In Lucidata, you have to build your own--something I'll cover later. Note the variable type defined as TEXT. It's similar to CHAR in Lucidata. Since this program is using disk and printer I/O, I'll leave it alone till a later time, but wanted to present it for those who have UCSD and want to do something useful with it.

EOF

ETUG -- ET/ETA-3400 USERS' GROUP

EDITOR'S NOTE: ETUG was formed in January 1981 to provide a central source of information on Heath Company's ET/ETA 3400 Microprocessor trainer. For further information write to:

ETUG
c/o Charles Van Dyke
11231 Oak St
El Monte, CA 91731

BUGGIN' HUG

Dear HUG,

In Issue 9 of REMark (February 1980), you published an article illustrating a simple way to use all of the functions of the H-19 terminal (graphics, reverse video, cursor addressing, etc.) in basic programs. This was accomplished by creating basic variables equal to the escape codes of the different H-19 functions and sending them to the H-19 in PRINT statements.

That program greatly simplified the task of setting up screen layouts for all of the MBASIC programs I have written since. Since that time many other REMark articles have also contributed to better programming on my part. With that in mind I would like to return the favor by sharing with you a complementary program to the H-19 program mentioned above.

This program is for use with the new H-25 printer. Like the H-19 program, it sets MBASIC variables equal to the various H-25 escape codes. Since the H-25 escape codes are more complicated than the H-19, this is even a bigger help in simplifying formatting of hard copy output to the H-25 using all of its features.

Using the H-19 and the H-25 programs together at the beginning of my programs allows me easy access to all of the wonderful features built into both machines. I have been pleased with the performance of the H-19 for two years and the H-25 is already proving to be a real winner. I'm glad I waited for it.

```
100 E$ =CHR$(27)              '  E$  = Escape Code
110 C$ =CHR$(13)              '  C$  = Carriage return code
120 ER$=E$+"c"                '  ER$ = Power up reset
130 ED$=CHR$(14)+C$           '  ED$ = Double width characters
140 EI$=CHR$(20)              '  EI$ = Reverse index
150 EG$=E$+"[10m"+C$          '  EG$ = Graphics mode
160 EP$=E$+"[11m"             '  EP$ = Print regular characters (exit graphics)
170 E1$=E$+"[1w"+C$           '  E1$ = 10     cpi (horizontal)
180 E2$=E$+"[2w"+C$           '  E2$ = 12     cpi      "
190 E3$=E$+"[3w"+C$           '  E3$ = 13.2   cpi      "
200 E4$=E$+"[4w"+C$           '  E4$ = 16.5   cpi      "
210 E5$=E$+"[1x"+C$           '  E5$ = 6 lpi  (vertical)
220 E6$=E$+"[2x"+C$           '  E6$ = 8 lpi
230 E7$=E$+"[3y"+C$           '  E7$ = 8.5 inch form length
240 E8$=E$+"[0y"+C$           '  E8$ = 11  inch form length
250 E9$=E$+"[5y"+C$           '  E9$ = 7   inch form length (check with top stub)
260 EE$=E$+"[?7h"             '  EE$ = Discard text beyond right margin (no wrap)
270 EW$=E$+"[?7]"             '  EW$ = Wrap leftover text to next line
280 EM$=E$+"[;":EN$="s"+C$    '  EM$;n;EN$ = Set right margin (n=margin ie;132)
```

This program stores the escape codes of various H-25 functions, such as set vertical pitch, in MBASIC variables. This makes it very easy to control the H-25 from within a program so that the H-25 can be used to its full advantage.

Several things should be noted about the escape codes. Some require a terminating character, such as a carriage return, after their use. Where this is required I have put the carriage return code into the MBASIC variable so you do not need to. Because of this, I recommend that you set the switch for auto line on the back of the H-25 to off. This will prevent a line feed during the transmission of the escape code.

The escape code for enter double width mode has several rules. The code must be the first character sent in a line, and it must be sent at the start of every line to be printed double width. For MBASIC, this means that the previous print statement must not have ended with a ";".

Examples: To send the power up reset code, (set the H-25 to the switch settings on the back) type:

```
Print #1,ER$;
```

The ";" is optional. If you don't include it the paper will advance one line.

To print a double width line type:

```
Print #1,ED$;"This is a double width line."
```

To change cpi to 16.5, lpi to 8 and enter the graphics mode type:

```
Print #1,E4$;E6$;EG$;
```

To set the right margin at 80 type:

```
Print #1,EM$;80;EN$;
```

The rest of the escape codes work in the same manner as the examples above. I have not included some of the functions, such as vertical or horizontal tabs as these may need some customizing for your application. I have also only set up set form feed variables for 3 standard form lengths. Simple modifications of E7$,E8$ or E9$ to the proper escape codes for your forms can be done easily.

Sincerely,

Bob Meyers
807 S Racine
Chicago, IL 60607

Dear Bob,

I would like to announce to the members of HUG that the USUS (UCSD System Users Group) software library is now available in H-8/89 format. There are 10 volumes of software available to USUS members. Each volume consists of one H-11 or three H-8/89 disks. The disks can be ordered from me for $17.00 per volume for the H-8/89 disks or $10.00 for the H-11 disks. The H-89 disks also include some extra goodies in the extra 60 or so blocks of space. The software includes utilities, games (Wumpus, Startrek, a really mean Othello, Blackjack, and the original Adventure), a LISP interpreter, a text formatter, a Pascal text formatter, modem handlers, printer spoolers, database programs and primitives, sort routines and a bunch of other really good stuff. Most of the programs are non-trivial and represent a lot of effort by some very skilled Pascal programmers. They are indispensible as examples of practically every programming construct in the language.

You must be a member of USUS in order to receive the disks. Membership is $20.00 per year, payable to USUS c/o:

Chip Chapin, Secretary
P.O. Box 1148
La Jolla, CA 92038

The P-System is alive and growing. Pascal is far faster than most any BASIC and it runs well on Heath computers. If you don't have the P-System, then get it. It is more user friendly than either HDOS or CP/M and it runs on almost any computer.

Sincerely,

George W. Schreyer
412 N Maria Avenue
Redondo Beach, CA 90277

The Ugly American???

Bob Ellerton just walked past me in the hall and said that if I didn't get my next article into his office within 24 hours he was going to write an obituary column under the heading of 'Heath on foreign shores'. Well, I would not want anyone to think that I'm dead yet, not by a long shot; so here we go.

One of the first things an average American notices when visiting a European country is that most of the 'natives' don't understand most of what he is saying (ESPECIALLY England) unless he is using sign language. Sign language alone can be embarrassing especially when put in rather delicate situations.

All of this was made quite clear to me when I stepped off the plane in Frankfurt West Germany and realized I did not know where I was going to live; but I was going to be there two or more years.

This all leads up to the basic contents of this article. How does one make a computer product, designed for English speaking users, operate in an environment where the users do not speak English?

There are three basic approaches to this problem.

1.) Don't sell the product to people who don't speak English.

2.) Give away a free English language course with each computer.

3.) Convert the computer to operate using all the alphabetic characters as used by the operator.

Well, approach number one will not work. For one thing what would happen to all those people who live in California ???? Approach number two will not work very well, the learning curve would be far too long. It would take two years before the poor guy would learn how to 'boot'. That only leaves us with approach number three as a real solution.

Converting the existing 'English' language keyboard to another languages character set was one of my first projects in Europe.

It started with the easy ones, (German, Swedish, etc) and then built up to the more difficult ones, (French, Italian, etc).

Before starting, one must remember that if you want to follow some of the standards set by the big guys (IBM, DEC, etc.) then you have 127 different ASCII characters. Many of these are control characters so when you set yourself down to work out the problem you really have only about 95 ASCII characters that can be altered to another configuration.

Well, lucky for me most of this has been done already and the ASCII character sets for Swedish, French, German, Norwegian, Finish, Danish, Italian, Hebrew, Russian, and Spanish have been more or less standardized. However, Arabic, Greek, Indian and most of the other middle east and far eastern languages have not been standardized. Setting up one of these character sets, Arabic for example, can be very frustrating and requires creative imagination to say the least.

Because we only have about 95 'changeable' codes to work with naturally when we change the English keyboard to German, for example, it will look different.

The German alphabet has 30 alphabetic characters as opposed to English which has 26. German, in addition to the 26 'normal' English characters has four extra, 'Ä', 'Ö', 'Ü', and 'ß'.

This will mean that the English type keyboard will require not only five 'bit map' changes to produce the German characters (don't forget the upper/lower case requirements), but it will also require that the location of the given keys be placed in 'funny' locations on the board. For example the 'Y' and the 'Z' are swapped one for the other. Also when the 'Ä', 'Ö', 'Ü', and 'ß' are put into place the following characters are lost, '[','\]','\','¦','@' (see the example).

Please note that when I'm speaking of character changes I speak of changing the way the character 'looks', not its ASCII code. A few of you real 'sharpies' just began to realize another problem (I heard your brain do a head load).

When we go to change the character set, the appearance of the characters change but not the ASCII code. This of course means that some of the software we use with English will go 'wacko' when we start to use German character keyboards.

For example, Autoscribe uses the '¦' to indicate a new page but under the German character set the '¦' is now a lower case 'ö'. Which means that every time a German writes a 'ö' he will drive an unmodified Autoscribe into the funny farm.

To solve this problem involves modifying Autoscribe to use a new 'new page' control character. This very thing was done with all the languages used in Europe.

Other software, such as MBASIC really do not need changes to be made, due to the fact that MBASIC will print anything within the string operation. For this reason computer languages such as Fortran, MBASIC, Cobal, and anything in assembler can print out their text in all languages.

Special application programs, such as Autoscribe, Magic Wand, Word Star, etc. require that they have alterations made within the structure of the code to account for the control operations.

So how DO we change things within the firmware of the Z/H89 and Z/H19 to give us a new character set? Depending on what the age is of your computer or terminal, you have a set of four or three PROMs that contain all the code for the generation of both printable and non printable characters.

The early models of the 89/19 combination contained four TMS2716 EPROMS. One character generator (IC473), one keyboard encoder (IC430) and two that contained the code to control the Z19 (IC423 and IC422). Later models contained masked PROMs, and about a year ago we changed the two control code PROMs from a 'split code pair' (IE 2x 2716) to an 'all in one' PROM (IE 1x 2732).

The changes in the control code were rather minimal except when French and Italian were implemented and I will not get into that. The changes in the character generator and keyboard encoder are where most of the action took place.

The keyboard encoder and the PROM that goes with it converts the switch matrix (keyboard switches) into a series of absolute addresses for the keyboard encoder PROM. When, for example you press the letter 'A' on your keyboard, the keyboard encoder converts the switch (on the keyboard) into an address. This address is an eight bit memory location within the keyboard encoders PROM. This address contains the ASCII code for the letter 'A'. When this address is accessed it will put this ASCII code (A=41h) on the internal buss of the terminal system where it then goes flying out your serial port to the computer. By going to each address you can change the ASCII code that is located there. In this way you can change, and swap around keys from one location to another. The keyboard encoder does other things also, like check if you have a shift operation to transmit an upper or lower case 'A'. A lower case 'A' would of course be a different address within the KB encoder PROM and therefore a different ASCII code, in this case a lower case 'A' would be 61h.

Now that you have a basic idea of how we get the proper ASCII code out the serial port, let's take a quick look as to how we get those little swiggles on the screen of the C.R.T.

At the same time that the ASCII code for the letter 'A' was flying out your terminals serial port many other things were also going on. The terminal logic card (TLC) of your Z89 and Z19 has its own computer built into it along with a sophisticated IC known as a CRTC (cathode ray tube controller). This CRTC, among other things, is forever monitoring the internal buss of the TLC just waiting for an ASCII code to come zooming along. When one arrives (in this case the 'A'), it will then decode this 41h into another logical address. This address is contained within the character generator PROM (IC473). This address contains some specific information that the CRTC can use to put the character up on the screen of the CRT. It is here within the character generator PROM that all the information is kept that will produce (in this case) the letter 'A'.

Starting at internal address 0410h, there are 10 memory locations that contain the correct binary bits in the correct sequence to write the letter 'A' over a period of 10 scan lines as controlled by the CRTC. The CRTC "knows" from the ASCII code where to start, in our case it adds a zero to 41h to get the address 410h. From there it also "knows" how many addresses above 410h it must read to get all the binary information for the given letter. In this binary information, the binary '0' is black and the binary '1' is white (or green).

To see an even better example of this, just look very closely at the characters on the screen of your terminal and you will notice that each letter is constructed with tiny dots. These dots are the binary '1's found in the ASCII code locations of the character generator.

This is about as far as I really care to go with regard to an explanation of how the characters are generated, but I think you will now understand to some degree why and how our wonderful little home wreckers have managed to communicate

in other languages.

In closing I would like to point out that all code examples were given in HEX for a reason. Whenever work is done in the area of PROMS and EPROM etc., it is always referenced in HEX. This is more or less a manufacturing standard.

OK, OK, enough is enough. If the phone rings just one more time I'm going to scream....

YEEEEEEEEEEEEEEEEAAAAAAAAAAAAAA.

So many of you have called wanting to know about some of the products I spoke of in REMark Issue 21 that my office is beginning to look like a PIER #1 import export shop. In an attempt to answer your questions, I will list the following items of interest and the address to write to. All the following locations may be addressed in English except England which must be written in "Oxford" English, (do not thy know of this old chap?). Address will follow this article.

England

a. WH89-PAL/C, three port parallel Centronics compatible I/O.

b. 12 bit/ 8 channel A/D I/O.

c. H89-20ma, 20ma active/passive converter.

d. PROM1, EPROM burner for the '89' series systems.

e. MX89, memory map pixel graphics for the '89' system.

Germany

a. IEEE-488 I/O

b. German character set for the '19' and '89' systems.

c. Autoscribe / German.

d. Device drivers for all current Olympia printers, (I.E. HDOS)

France

a. Segate Technology 5meg storage system for the '89'.

b. Honeywell Bull, 10x10meg storage system for the '89' and the '11' systems.

c. Autoscribe / French.

d. Bar code reader, (reads 3 internationally known bar codes automatically).

e. French character set for the '19' and '89' systems.

Sweden

a. Swedish character set for the '19' and '89' systems.

Should you have any questions please feel free to write me and I will answer as soon as I can, but barring the fact that I am on the road at times it will take a week or three.

John R. Beran

Contact Addresses:

England

Heath Electronics - U.K., Ltd.
Bristol Road
Gloucester, Gl2 6EE
England
Telex/cable 43179 BWGLOW G.

Att: HEATHKIT

FRANCE

Heath/Zenith S.A.
47 Rue de la Colonie
F-75013 Paris, France
Telex 200205F

Germany

Heath GMBH
D-6072 Dreieich - 1
Robert Bosch Strasse 32-38
Postfach 10 20 60
West Germany Telex 0417986

Sweden

Heath Scandanavia, A.B.
Norr Malarstrand 76, Box 12081
102 23 Stockholm 12, Sweden
Telex 14238 HeathSW

John Beran

New HUG Software

885-1115 NAVPROGseven $20.00

NAVPROGseven Aircraft Navigation
and
Flight Planning

Author: Alan Bose
2514 Essex Court
St. Joseph, MI 49085

Requirements: NAVPROGseven is designed to run on either the H8/H19/H17 or the H89 using HDOS versions 1.5 to 2.0. It requires a dual-drive system, a minimum of 48K, Microsoft Basic and a line printer.

Introduction: NAVPROGseven is a database management system designed for pilots flying cross-country. The system is built around a latitude-longitude referenced navigation program designed to prepare a flight log that is ready for use in the cockpit. The system stores performance data about each aircraft you fly, navigation data about each checkpoint, airport or navaid you fly over, and saves this information for easy access on subsequent flights.

The title of NAVPROGseven comes from the features and functions designed into the system, many of which are not found in similar programs:

1. Easy input & revision of the airport/navaid data base.
2. Two RNAV (area navigation) functions that return the latitude & longitude of a location based on cross-bearings from known points.
3. Aircraft performance data stored for each plane you fly.
4. Easy access & display of airport & checkpoint information (using standard FAA identifiers) as you plan your route of flight. Automatic flight planning selects naviads closest to your great circle route and prepares several alternate routings. Often flown routes can be saved for later use.
5. Great circle navigation between checkpoints using aircraft performance data, and printout of ready-to-use flight log. Ideal for use with the Heathkit OC-1401 Navigation Computer.
6. Climb/descent profiles calculated based on aircraft performance data.
7. Multiple sort criteria to organize airport/navaid data into easy-to-read printouts. VECTORED TO 25

885-4002 REMark Volume II $20.00

REMark Volume II is now hot off the press. Volume II contains Issues 14 to 23 of REMark bound in one handsome book which, when used with Volume I and Issue 24 of REMark, will totally complete the library of printed material from the Heath Users' Group. REMark Issue 24 serves as the cross-reference to the articles contained in all previous publications of REMark. In the future, HUG will publish, on a yearly basis, another Volume which will contain issues from the previous year. Volume I has been selling quickly, so be sure to get your name on the order list for Volume II.

885-1031 Music 8 and 89 $20.00

Music 8 and 89 are new adaptations of James Gorgan's program, Music. Modifications were performed to Jim's program to allow use with the H-89 computer as well as the H-8 computer. Some hardware modifications will be required to allow proper operation with this particular software for both the H-8 and the H-89. These modifications are explained in the README.DOC along with the operating instructions. Fortunately, the hardware modifications are easy to perform in both cases.

PLAY8.ASM and PLAY89.ASM source codes are provided so that the software can be used on either the H-8 or the H-89. Further, the two ASM files can be compared for a learning experience.

MUSIC.BAS and MUSICM.BAS, the compiling routine for the various "scores", are provided for the user that has either Extented Benton Harbor BASIC (MUSIC.BAS) or MicroSoft BASIC (MUSICM.BAS).

The program requires at least one disk drive, HDOS, an H-8 or H-89 with at least 48K of memory and either the H-19 or H-9 when used with the H-8. As stated previously, you will be required to make easy hardware modifications to either the H-8 or the H-89 to obtain proper results. Several compositions are complete and ready to play by typing the the name of the tune for either the H-8 or H-89. Scores for other compositions are provided so that you can practice with the compiling routine which uses the HDOS EDIT program, the HDOS ASM (Assembler) and your choice of E.B.H. BASIC or MBASIC.

HUG Product List

Part Number	Description		Selling Price
CASSETTE SOFTWARE (H8 and H88)			
885-1008	Volume I Documentation and Program Listings (some for H11)		$ 9.00
885-1009	Tape I Cassette		$ 7.00
885-1012	Tape II BASIC Cassette		$ 9.00
885-1013	Volume II Documentation and Program Listings		$ 12.00
885-1014	Tape II ASM Cassette H8 Only		$ 9.00
885-1015	Volume III Documentation and Program Listings		$ 12.00
885-1026	Tape III Cassette		$ 9.00
885-1036	Tape IV Cassette		$ 9.00
885-1037	Volume IV Documentation and Program Listings		$ 12.00
885-1039	WISE on Cassette H8 Only		$ 9.00
885-1057	Tape V Cassette		$ 9.00
885-1058	Volume V Documentation and Program Listings		$ 12.00
HDOS SOFTWARE (H8/H17 or H89 -- 5-inch only)			
MISCELLANEOUS COLLECTIONS			
885-1024	Disk I H8/H89		$ 18.00
885-1032	Disk V H8/H89		$ 18.00
885-1044	Disk VI H8/H89		$ 18.00
885-1064	Disk IX H8/H89		$ 18.00
885-1066	Disk X H8/H89		$ 18.00
885-1069	Disk XIII Misc H8/H89		$ 18.00
GAMES			
885-1010	Adventure Disk H8/H89		$ 10.00
885-1029	Disk II Games 1 H8/H89		$ 18.00
885-1030	Disk III Games 2 H8/H89		$ 18.00
885-1031	Music 8 & 89 H8/H19 and H89		$ 23.00
885-1067	Disk XI Graphic Games .ABS and B H BASIC (H19/H89)		$ 18.00
885-1068	Graphic Games (H19/H89)	*	$ 18.00
885-1088	Graphic Games (H19/H89)	*	$ 20.00
885-1093	Dungeons and Dragons Game Requires H89 or H8/H19	*	$ 20.00
885-1096	Action Games (H19/H89)	*	$ 20.00
885-1103	Sea Battle Game (H19/H89)		$ 20.00
885-1111	HDOS MBASIC Graphic Games	*	$ 20.00
885-1112	HDOS Graphic Games		$ 20.00
885-1113	HDOS Fast Action Games		$ 20.00
885-1114	Color Raiders and Goop (HA-8-3)		$ 20.00
UTILITIES			
885-1019	Device Drivers (HDOS 1.6)		$ 10.00
885-1022	HUG Editor (ED) Disk H8/H89		$ 15.00
885-1025	Runoff Disk H8/H89		$ 35.00
885-1043	MODEM Heath to Heath H8/H89		$ 21.00
885-1050	M.C.S. Modem for H8/H89		$ 18.00
885-1060	Disk VII H8/H89 SUBMIT, CLIST, FDUMP, ABSDUMP, etc.		$ 18.00
885-1061	TMI Cassette to Disk H8 only		$ 18.00
885-1062	Disk VIII H8/H89 (2 disks) MEMTEST, DUP, DUMP, DSM		$ 25.00
885-1063	Floating Point Disk H8/H89		$ 18.00
885-1065	Fixed Point Package H8/H89		$ 18.00
885-1075	HDOS Support Package H8/H89		$ 60.00
885-1077	TXTCON/BASCON H8/H89		$ 18.00
885-1079	HDOS Page Editor		$ 25.00
885-1080	EDITX H8/H19/H89		$ 20.00
885-1082	Programs for Printers H8/H89		$ 20.00
885-1083	Disk XVI RECOVER, etc.		$ 20.00
885-1089	MACRO, CTOH, and misc Utilities		$ 20.00
885-1090	Misc. HDOS Utilities CCAT, HPLINK, AH, MBSORT, etc.		$ 20.00
885-1092	RDT Debugging Tool H8/H89		$ 30.00
885-1095	HUG SY: Device Driver HDOS 2.0		$ 30.00
885-1098	H8/HA-8-3 Color .ABS/.ASM		$ 20.00
885-1099	H8/HA-8-3 Color in Tiny Pascal		$ 20.00
PROGRAMMING LANGUAGES			
885-1038	WISE on Disk H8/H89		$ 18.00
885-1042	PILOT H8/H89		$ 19.00
885-1059	FOCAL-8 H8/H89		$ 25.00
885-1078	HDOS Z80 Assembler		$ 25.00
885-1085	PILOT Documentation		$ 9.00
885-1086	Tiny Pascal H8/H89		$ 20.00
885-1094	HUG Fig-Forth H8/H89 2 Disks		$ 40.00
BUSINESS, FINANCE AND EDUCATION			
885-1047	Stocks H8/H89		$ 18.00
885-1048	Personal Account H8/H89		$ 18.00
885-1049	Income Tax Records H8/H89		$ 18.00
885-1051	Payroll H8/H89		$ 50.00
885-1055	Inventory H8/H89	*	$ 30.00
885-1056	Mail List H8/H89	*	$ 30.00
885-1070	Disk XIV Home Finance H8/H89		$ 18.00
885-1071	SmBusPkg III 3 Disks H8/H19 or H89	*	$ 75.00
885-1091	Grade and Score Keeping	*	$ 30.00
885-1097	Educational Quiz Disk H89 or H8/H19	*	$ 20.00
DATA BASE MANAGEMENT SYSTEMS (DBMS)			
885-1107	Amateur Radio Logbook and TMS		$ 30.00
885-1108	Telephone/Mail Info. System	*	$ 30.00
885-1109	Retriever (2 disks)		$ 40.00
885-1110	Autofile		$ 30.00
885-1115	Aircraft Navigation DBMS H8/H89		$ 20.00
AMATEUR RADIO			
885-1023	RTTY Disk H8 Only		$ 22.00
883-1106	Morse-89 H8/H19 or H89		$ 20.00
* Means MBASIC is required			
H11 SOFTWARE			
885-1008	Volume I Documentation and Program Listings (some for H11)		$ 9.00
885-1033	HT-11 Disk I		$ 19.00
CP/M SOFTWARE (5-inch only)			
885-1201	CP/M (TM) Volumes H1 and H2	%	$ 21.00
885-1202	CP/M Volumes 4 and 21-C	%%	$ 21.00
885-1203	CP/M Volumes 21-A and B	%%	$ 21.00

VECTORED TO 32

DND and HDOS 2.0

The HDOS version of HUG's **Dungeons and Dragons** game is supplied on a bootable disk using an altered version of HDOS 1.6. It was done that way because the game is simply too large to run under HDOS 2.0, which uses more memory than 1.6. This article will present a method for running DND under HDOS 2.0 without altering the game itself, and the steps to take to make a bootable disk with the game on it.

Getting More Memory

One of the reasons why MBASIC is memory hungry is that it loads both of the HDOS overlays into memory when it starts. The first overlay is used for the file commands (OPEN, CLOSE, FILES, etc.), and the second is used for the RESET command. If you alter MBASIC so that it does not load in the second overlay, you gain about 2k more free memory space and lose only the RESET command. The following patch will cause MBASIC to load only the first overlay (this patch is for version 4.82 only).

Address	Old data	New data
160241	076	000
160242	001	000
160243	377	000
160244	010	000
160245	322	000
160246	217	000
160247	152	000

Make the patch using PATCH.ABS (supplied with HDOS 2.0). Make sure that the old data matches that shown here before you enter the new data. The program DNDRUN.ABS on the DND disk also loads in both overlays, so it will also have to be patched as follows.

Address	Old data	New data
042204	076	000
042205	001	000
042206	377	000
042207	010	000

The program DNDRUN is not necessary for operation of the game, but it makes a nice picture on the screen and loads in MBASIC and the menu program for you.

Making a DND System Disk

The DND package of programs is designed to be run from a system disk, with the game starting automatically when the system is booted. I will present two methods for making system disks. The first method is the "easy" method that does not require any patches to the system. The second method will produce a disk that works like the original HDOS 1.6 system that DND comes on.

To make the "easy" system, first make a minimum system disk using SYSGEN/MIN. Then copy SET.ABS to it (or run SET from another disk) and enter SET HDOS NODATE. Then delete SET.ABS and copy the following files to the system disk from the HUG DND disk: DNDRUN.ABS, DND.BAS, DND.DAT, START.BAS, and MENU.BAS. Also put the patched version of MBASIC on the system disk. Rename the file DNDRUN.ABS to PROLOGUE.SYS. Now enter BYE and reboot on your new system disk. If this is the first time you are booting the disk, you will have to type spaces to start the boot process, but every time after that it will go right to the game without any intervention.

To make a system disk that works like the HDOS 1.6 DND disk, first initialize a new disk. Perform the following patch using DUP (from HUG 885-1062) or a similar utility if you are using a 40 track single sided disk and the hard sector (H17 type) controller. This patch reduces the size of DIRECT.SYS to make more space on the disk.

Track 13 Sector 7

Address	Old data	New data
DD	48	44
FE	82	00

Track 14 Sector 8

40	4B	41
44	41	00
48	00	4B

If you have the soft sector controller (H37), initialize the disk at double density to get more space. If you have an 80 track drive, you will have all the space you need. Sysgen the disk you have just made using SYSGEN/MIN and set the NODATE option as described before. Then make the following patches using DUP or a similar utility.

Track 13 Sector 2

Address	Old data	New data
53	E0	00
6A	E0	00

These patches remove all flags from the files SYSCMD.SYS and PIP.ABS. After making the patches, delete those two files from the disk. Then copy all of the files from the DND disk to the disk you have just made using PIP *.* (or ONECOPY *.*), except for DNDRUN.ABS. Copy your patched version instead.

VECTORED TO 32

The BASIC Memory Grabber

Don't Fall Into This FOR/NEXT Trap!

By: Roy S. Reichert

Scenario: You have just written a masterpiece program in Microsoft BASIC. After working diligently to debug it and make the best use of available memory, you finally put it to work running a big job. Everything looks great! It is running well and.....oops! What's this? All of a sudden you see a screen diagnostic: "Out of memory"!

Must have missed something here. So you go in and change the amount of string space given in the CLEAR statement. Perhaps you even reduce the constants used in the DIMension statement. There.....that should do it. Now to try again.

Things look good. That must have solved the problem. Looks like everything is O.K. now.....oops! What, again? "Out of memory"!!!

What is happening here? Well, you may have just fallen into one of the most insidious traps in MBASIC. I call it the "BASIC Memory Grabber". This trap is insidious because it is not easily detected, there may be no errors in your code, and the program may work correctly in all other respects. Indeed, the trap may not even become apparent until after you have been using the finished program for some time.

To explain the probelm, let me illustrate with the following code:

```
1000 CLEAR
 .
 .
1200 FOR T=A TO B
 .
 .
1500 IF(exp1)GOTO 1610
 .
 .
1600 NEXT T
1610 FOR T=X TO Y
 .
 .
2000 IF(exp2)GOTO 2210
 .
 .
2200 NEXT T
2210 REM
 .
 .
2500 IF(exp3)GOTO 1200
```

This code represents a simple logical flow of a typical program. The main body of code is represented by the lines having dotted "line #'s". Note that there is nothing wrong with the logic itself; it is totally consistent with the rules of the language, and it works. (The term "exp" is an expression whose value is determined by the program acting on some form of data).

If the data being processed results in a situation where all of the "GOTO"'s are executed repeatedly, the program will actually use up all available memory and halt! If you want to try it, enter this simplified version of the logic into your system and run it:

```
1000 CLEAR
1010 FOR T=1 TO 10
1020 GOTO 1040
1030 NEXT T
1040 FOR T=1 TO 10
1050 GOTO 1070
1060 NEXT T
1070 PRINT FRE(0)
1080 GOTO 1010
```

The statement at line 1070 prints out the amount of free memory space available in your system. Using HDOS, as this program runs you will see this number get smaller and smaller until it eventually reaches near zero and the program halts with a diagnostic. If you are using CP/M, the printed number will not change, but the program will still halt with the diagnostic.

What is going on? Well, it is safe to assume that we are experiencing a "stack" problem. HDOS allows the use of all available memory for building the push-down stack which holds the pointers for GOSUB's, FOR/NEXT loops, etc. CP/M dedicates a reserved stack-space for this purpose and free memory is not affected. What we see happening here is simply the growth of this stack to fill all allowable space. We have managed to confuse the interpreter in such a way that the stack is not properly "popped" clear as we jump out of the FOR/NEXT loops.

The effect only occurs because we have used the same variable ("T") for the index of both loops. This does not violate any rules of MBASIC since the FOR

statements initialize the value of T each time, but the interpreter cannot handle this properly.

If you change one of the index variables (change "T" to "K" in lines 1040 and 1060) and re-run the program,the problem does not occur. The memory number printed will remain constant and the program will run indefinitely.

You may have noted that in the first test, under HDOS, memory gets "grabbed up" in 34 byte increments. If we change the test to include a third loop (see below), memory is taken in 51 byte increments. Thus, there appears to be one 17 byte stack entry left behind by each loop.

```
1000 CLEAR
1010 FOR T=1 TO 10
1020 GOTO 1040
1030 NEXT T
1040 FOR T=1 TO 10
1050 GOTO 1070
1060 NEXT T
1070 FOR T=1 TO 10
1080 GOTO 1100
1090 NEXT T
1100 PRINT FRE(0)
1110 GOTO 1010
```

If we change the index of the middle FOR/NEXT (change "T" to "K" in lines 1040 and 1060), the problem goes away! This is particularly interesting because the first and last FOR/NEXT loops are still logically adjacent, yet the interpreter handles the stack properly. You will find that the problem only exists when two or more loops, using the same index variable, logically jump to the start OF EACH OTHER in an unbroken chain! In this last test, the last GOTO jumps to the start of another loop, but the first GOTO does NOT jump to the start of another loop with the same index variable ("T"). This second loop used "K". The chain was broken and the problem did not occur. I am somewhat at a loss to explain exactly what happens here since I have no access to the MBASIC source code.

Failure to execute a NEXT statement prior to executing another FOR statement, probably makes the interpreter "think" that these are "nested" loops, thus, the use of the same index variable would be confusing. If we were to re-write our test routine so that each FOR/NEXT loop went the full range of the index, with a normal "fall-out" from the loop, then the problem would not occur. This would allow the interpreter to flush the stack properly and no trap would exist. However, such code is frequently inefficient, clumsy and slow. Jumping out of FOR/NEXT loops is a reasonable technique to use as you understand what you are doing.

In summary then, it is clear that jumping out of FOR/NEXT loops has its accompanying hazards. Although many programmers argue that this is always dangerous, the reason usually given is that the value of the index variable is not always preserved. The problem shown here is not concerned with the VALUE of the index variable, but rather seems to be related to the ADDRESS of the index. (Remember that variable names are nothing more than relocatable address codes).

It is reasonable to expect such a logic structure to crop-up in any program during development and debugging. Although the program will work as intended, it may be a "memory grabber" without any warning. Use care in building such logic structures. Use different variable names for loop indices and test the program thoroughly. If you are not careful, the "memory grabber" will get you!

Local HUG News

As promised, we have listed the Local Heath Users' Groups that have formed throughout the country and the world (*). This list was comprised of all groups that responded to our request letter for current information. If your group does not appear on the list, or, if some of the information is incorrect, please let us know so that we may make necessary changes to our records for the next printing of this data.

We hope this list will help those of you who are in the same general area as some of these groups to become familiar with what they have to offer. Further, communication with others who have the same interest in Heath/Zenith computers will aid you in growing with your computer in the future.

If your club or group publishes a newsletter, you may wish to contact other groups for an exchange of information. Many of the groups are more than willing to exchange materials to enhance the total knowledge of the various computer products.

Current Local HUG Clubs As Of 13-Jan-81

AK, Eagle River
Alaska HUG
P.O. Box 951
Eagle River, AK 99577
907-694-9908 Group Size 20
Contact Person: Ben Sevier

CA, Anaheim
ANAHUG (Anaheim HUG)
330 E. Ball Road
Anaheim, CA 92805
714-776-9420 Group Size 120
Contact Person: John Belsher, President
3rd Thursday 7:00 PM at HEC

CA, El Cerrito
ECHUG (El Cerrito HUG)
6000 Potrero Avenue
El Cerrito, CA 94530
415-236-8870
Contact Person: Alan Biocca
4th Wednesday at HEC

CA, Fresno
FresHUG (Fresno HUG)
4833 East Santa Ana
Fresno, CA 93726
209-291-6258 Group Size 4
Contact Person: Harlen Collins

CA, Glendora
Southern CA H11 Users Group
430 W. Highland Avenue
Redlands, CA 92373
714-886-4766 Group Size 40
Contact Person: Dr. M.J. Di Girolamo
Meets at 625 E. Palm, Glendora, CA

CA, Los Angeles
Los Angeles HUG
24025 Fernlake Drive
Harbor City, CA 90710
213-539-4276 Group Size 20
Contact Person: Dean Gibson c/o Ultimeth Co.
1st Thursday 7:00 PM at HEC

CA, Redwood City
BAHUG Bay Area HUG
2001 Middlefield Road
Redwood City, CA 94063
415-365-4915 Group Size 219
Contact Person: Bob Bance, Sec.
2nd Tuesday 7:00 PM at HEC

CA, Riverside
Tri-HUG
5705 Via Sotelo
Riverside, CA 92506
714-683-2929 Group Size 20
Contact Person: Kenny Adcock

CA, San Diego
San Diego HUG
12202 Kingford Court
El Cajon, CA 92021
714-561-2540 Group Size 170
Contact Person: Richard Cobb
1st Wednesday 7:00 PM at Parkway Jr HS La Mesa

CA, Woodland Hills
LUVAHUG
22504 Ventura Blvd.
Woodland Hills, CA 91364
213-883-0531 Group Size 40
Contact Person: Paul S. Townsend
2nd Thursday 7:00 PM at HEC

CO, Denver
DENHUG (Denver HUG)
P.O. Box 20422
Denver, CO 80220
303-394-2082 Group Size 96
Contact Person: Alfred K. Carr, Treasurer
BB 303-422-3409
2nd Monday 7:00 PM at HEC

CT, Avon
CONHUG (Connecticut HUG)
8 Huckleberry Lane
W. Simsbury, CT 06092
203-658-2944 Group Size 35
Contact Person: Tom Carborne
1st Wednesday at HEC
H11 Special Interest Group

FL, Fort Myers
P.O. Box 05-37
Tice, FL 33905
Contact Person: Robert Sloat
Just getting started

FL, Miami
Miami Amateur Computer Club
4705 W. 16th Avenue
Hialeah, FL 33012
305-823-2280 Group Size 35
Contact Person: Ralph Boyd
At HEC

FL, Orlando
HUG of Central FL Computer Sc.
135 Stonyridge Drive
Longwood, FL 32750
305-339-8853 Group Size 34
Contact Person: Jim Donlon, President
4th Wednesday at various locations

HI, Honolulu
HUGH (HUG Hawaii)
1255 Nuuanu Avenue # 1405
Hononlulu, HI 96817
808-531-8843 Group Size 45
Contact Person: Jim Branchaud, President
3rd Saturday at Mililani, 1st Wednesday at Kalihi

IL, Champaign
CCCC (Champaign Cty Comp Club)
412 Dorchester
Mahomet, IL 61835
312-586-5100 Group Size 12
Contact Person: Roger Fraumann

IL, Downers Grove
I-HUG (Illinois HUG)
6116 Lane
Downers Grove, IL 60516
312-971-1660 Group Size 25
Contact Person: Len Bateman
3rd Wednesday at various locations

IL, Peoria
CIHUG (Central Illinois HUG)
408 Bess Street
Washington, IL 61571
309-745-8313 Group Size 17
Contact Person: Ronald Morgan, President
3rd Sunday at 3 PM (Jan, Mar, May, Jul, etc.)

IL, Rockford
Blackhawk Bit Burners
325 Beacon Drive
Belvidere, IL 61008
815-544-5206 Group Size 35
Contact Person: Frank D. Dougherty

IN, Indianapolis
Indianapolis HUG (IHUG)
3390 Peppermill Drive #2C
West Lafayette, IN 47906
317-257-4321 Group Size 60
Contact Person: Robert Wild, President
2nd Wednesday 7:15 PM at HEC

KS, Wichita
Wichita HUG
1909 Siefkin
Wichita, KS 67208
316-681-3456 Group Size 18
Contact Person: David Horwitz
2nd Sunday of ODD months 2:00 PM at E. Pike Bldg.
Corner of Webb and Kellog in Wichita

LA, Kenner
NOHUG
1900 Veterans Blvd.
Kenner, LA 70062
504-467-6321 Group Size 60
Contact Person: Nathan Gifford
1st Wednesday at 7:30 PM at HEC

MA, Northampton
Hampshire Computer Club
Box 685
Northampton, MA 01061
617-584-7159 Group Size 100
Contact Person: Ed Judge, Secretary
2nd Tuesday 7 PM at McConnel Hall Smith College
Beginners Group 1st Tuesday

MA, Peabody
HUG North Shore
6 Susan Drive
Saugus, MA 01906
617-233-2941 Group Size 60
Contact Person: Hal Messinger, President
BB 617-531-9332 24 hours
2nd Wednesday Hilltech Bldg Danvers

MA, Pittsfield
BERCHUG (Berkshire County HUG)
73 Waverly Street
Pittsfield, MA 01201
Contact Person: Paul E. Ouellette, President

MD, Baltimore
Baltimore HUG
6106 Marlora Road
Baltimore, MD 21239
301-323-6093 Group Size 70
Contact Person: William Frey
3rd Thursday 7:00 PM at HEC

MI, Detroit
Metro Detroit Area HUG
7716 Winona
Allen Park, MI 48101
313-928-7423 Group Size 50
Contact Person: Chuck Dattolo

MI, Kalamazoo
SMHUG (Southwest Michigan HUG)
623 Wildwood Place
Kalamazoo, MI 49008
616-349-3535 Group Size 50
Contact Person: Al Jacobs, Secretary/Treasurer
4th Saturday 1 PM at Western Michigan University
Moore Hall, Rm 1034, News Letter

MI, Saint Joseph
SJHUG (Saint Joseph HUG)
Saint Joseph, MI 49085
Group Size 33
Contact Person: Vance Fisher, Chair Person
1st Tuesday 7:00 PM at various locations
Check HEC for place of meeting.

MN, St. Paul-Minneapolis
SMUGH
8895 72nd Street
Cottage Grove, MN 55016
612-459-4382 Group Size 100+
Contact Person: Steve Howard, President
Last Monday at 7:00 PM (Alt. St Paul & Mpls)

MO, St. Louis
SLHUG (St. Louis HUG)
3794 McKelvey Road
Bridgeton, MO 63044
314-291-1850 Group Size 120
Contact Person: Mike Davis, President
BB 314-291-1854 after hours ONLY
2nd Wednesday 7:30 PM at HEC

NC, Charlotte
HUG Charlotte
2721 Picardy Place
Charlotte, NC 28209
704-374-6997
Contact Person: Jim Simpson

NC, Fayetteville
Cape Fear Computer & HUG
2454 Vandemere Avenue
Fayetteville, NC 28304
919-485-4586 Group Size 11
Contact Person: Jerry Mills, President
Bi-Weekly 2:00 PM on Sundays at homes.

NE, Omaha
OMAHUG (Omaha HUG)
9207 Maple Street
Omaha, NE 68134
402-391-2071 Group Size 200
Contact Person: Chuck Juvenal, Chairman
3rd Sunday 6:30 PM at HEC

NJ, Fairlawn
HUGNJ (HUG of New Jersey)
3507 Broadway
Fairlawn, NJ 07410
201-791-6938 Group Size 85
Contact Person: Mel Beiman
BB 201-791-3015 24 hours
3rd Monday 8 PM at HEC

NY, Buffalo
BUG (Buffalo Users Group)
3585 South Benzing Road
Orchard PArk, NY 14127

716-662-7122 Group Size 50+
Contact Person: Jon Hodge
3rd Sunday 1 PM at HEC

NY, Long Island
Jeri-HUG (Jericho HUG)
15 Jericho Turnpike
Long Island, NY 11753
513-334-8181 Group Size 80
Contact Person: Bob Lippman
2nd Thursday 7:30 PM at HEC

NY, Rochester
RHUG (Rochester HUG)
937 Jefferson Road
Rochester, NY 14623
716-773-0193
Contact Person: Joanne Lang, Chairperson
Last Tuesday at 7:00 PM at HEC

OH, Cincinnati
Cincinnati HUG
10133 Springfield Pike
Woodlawn, OH 45215
513-771-8850 Group Size 50
Contact Person: Roger Svoboda
2nd Tuesday 6:30 PM at HEC, $10.00 Dues/year
Newsletter I/O Port

OH, Cleveland
NOHUG (Northeastern Ohio HUG)
4705 Tanglewood Place
Lorain, OH 44053
Group Size 40
Contact Person: Art Petkosek
2nd & 4th Thursday 7:00 PM at Maple Hts. Library

OH, Columbus
Columbus HUG
2500 Morse Road
Columbus, OH 43229
614-475-7200 Group Size 50
At HEC

OH, Dayton
Wright-Patterson HUG
4110 Spruce Pine Court
Dayton, OH 45424
513-236-4915 Group Size 36
Contact Person: Jim Moore, President
1st Thursday 4 PM at Wright-Patterson AFB

OH, Toledo
THUG (Toledo HUG)
4804 Mt. Airy Road
Sylvania, OH 43560
419-882-3626 Group Size 30
Contact Person: John F. Priebe, President
Last Sunday 8 PM

OK, Oklahoma City
OKC TUGS
2727 NW Expressway
Oklahoma City, OK 73112
405-848-7593 Group Size 40
Contact Person: Bob Perry
2nd Sunday at 1:00 PM at HEC
BBS 405-848-9329 24 hours

PA, Frazer
FUG (Frazer Users Group)
1641 Princess Anne Drive
Lancaster, PA 17601
717-397-3146 Group Size 60
Contact Person: Dave Hendrie, President
1st Sunday 4 PM at HEC

PA, Pittsburgh
PittsburgHUG
3482 William Penn Highway
Pittsburgh, PA 15235
412-824-3564 Group Size 35
Contact Person: John C. Schultz, President
3rd Thursday 7:00 PM at HEC

RI, Warwick
HUG-,RI' (HUG of Rhode Island)
558 Greenwich Avenue
Warwick, RI 02886
401-738-5152 Group Size 150
Contact Person: Walt Phaneat
2nd Wednesday 8 PM at HEC

TN, Memphis
Memphis HUG
6874 Kirby Brooks Drive
Memphis, TN 38115
901-362-8860 Group Size 4
Contact Person: Morris Proctor
Meets at National Cotton Council

TX, Dallas
DFW HUG (Dallas-Fort Worth)
2715 Ross Avenue
Dallas, TX 75201
214-826-4053 Group Size 70
Contact Person: Henry Gardiner, President
1st Thursday and 15 days later (Wed.) at 7:30 PM
At HEC BB 214-742-1380

TX, Houston
HUG-H
7798 Braniff
Houston, TX 77061
713-644-5689 Group Size 75
Contact Person: Tom McCormick, President

TX, San Antonio
San Antonio (SAHUG)
7111 Blanco Road
San Antonio, TX 78216
512-341-8876 Group Size 65
Contact Person: Tom Schneider
First Tuesday at HEC, 7:30 PM

TX, Witchita Falls
Nortex HUG (North Texas HUG)
4510 Allendale Road
Wichita Falls, TX 76310
817-692-1241
Contact Person: Alan D. Martin

UT, Midvale
UHUG (Utah HUG)
58 E. 7200 South
Midvale, UT 84047
801-566-4628 Group Size 75
Contact Person: Don Greene, President
2nd Wednesday 7:00 PM at HEC

VA, Fairfax
CHUG (Capital HUG)
P.O. Box 2653
Fairfax, VA 22031
301-283-6260 Group Size 400
Contact Person: Larry Henderson, President
3rd Monday 7:30 PM at Fairfax High School
Special Interest Group for H11s

VA, Richmond
Richmond HUG
1724 Blakemore Road
Richmond, VA 23225
804-232-2925 Group Size 8
Contact Person: Jim Scott
2nd Monday at various locations

VA, Virginia Beach
THUG (Tidewater HUG)
1055 Independence Blvd.
Virginia Beach, VA 23455
804-460-0997 Group Size 90
Contact Person: John E. Smith, President
1st & 3rd Tuesday at 7:00 PM at HEC

WA, Spokane
SPOHUG (Spokane HUG)
RFD 1 Box 676
Spokane, WA 99204
509-448-9727 Group Size 18
Contact Person: Charles Ballinger
Newsletter

WA, Vancouver
Portland-Vancouver HUG
516 SE Chkalov Drive
Vancouver, WA 98663
206-254-4441 Group Size 25
Contact Person: Richard Crawford
1st Thursday at 7:30 PM at HEC
Portland OR and Vancouver Area

CANADA, Alberta
HUC (Heath Users of Canada)
101 5809 Macleod Trail South
Calgary, Alberta T2H 0J9 CANADA
403-252-2688
Contact Person: Gary Selman

CANADA, Ottawa Ontario
HUG ,O' (HUG Ottawa)
866 Merivale Road
Ottawa, ONTARIO K1Z 5Z6 CANADA
613-728-3731 Group Size 30
Contact Person: Brain Fultz, President
2nd Wednesday 8:00 PM at HEC

CANADA, Toronto
THUG (Toronto HUG)
1480 Dundas Street E.
Mississauga, ONT. CANADA L4X 2R7
416-273-3797 Group Size 25
Contact Person: Bill Smith

CANADA, Vancouver BC
VHUG (Vancouver HUG)
3058 Kingsway
Vancouver BC V5R 5I7 CANADA
604-437-7626 Group Size 27
Contact Person: Eric Worthy
Last Monday 7:30 PM at HEC

GERMANY, Sprendlingen
HUG-Deutschland
Robert-Bosch-Strasse 32-38
D-6072 Dreieich W. GERMANY
06103/3808 Group Size 200
Contact Person: Egon Becker/Lydia Luguet

GERMANY, Frankfurt
Frankfurt HUG
American Consulate General FRDCO
APO NY, NY 09757
566187 Group Size 3
Contact Person: Carl Lovett

HOLLAND, Apeldorn
Dutch HUG
Hofstraat 30
7311 KW Apeldorn HOLLAND
Group Size 70
Contact Person: Evert Jan Stokking

PANAMA CANAL
Canal HUG
P.O. Box 1112
APO Miami, FL 34001
84-4094 Group Size 6
Contact Person: Michael Gulick, President
1st Tuesday 7:30 PM at Howard Air Force Base

PUERTO RICO, Rosario
PRHUG (Puerto Rico HUG)
P.O. Box 765
Rosario, PR 00746
809-892-4677 Group Size 5
Contact Person: Norberto Collado Rivera

****

The Heath Users' Group will hold a meeting at the West Coast Computer Faire in San Francisco. The exact meeting date is yet to be established. However, when we have more details, we will pass them on via the HUG BB on Micronet and on the SOURCE POST HUG catagory.

****

(*) NOTE: List of areas which previously had a HUG group but have not returned our most recent request for data.

AZ, Phoenix
CA, Bakersfield
CA, Pomona
CA, San Jose
FL, Fort Lauderdale
FL, Tampa
GA, Alanta
IL, Northbrook
KS, Mission
KY, Louisville
SC, Columbia
TX, Abilene
TX, Corpus Christi
WI, Milwaukee
CANADA, Montreal
GUAM

VECTORED FROM 16

The comprehensive flight log tells you field elevation at your departure & destination airports, navaid frequencies enroute, distances for each leg & total remaining, true & magnetic course, magnetic heading corrected for wind and magnetic variation, ground speed corrected for wind & climb and/or cruise leg segments, ETE & ETA for each leg, fuel usage based on climb and/or cruise with startup/taxi/takeoff fuel accounted for, fuel remaining, and a warning if reserves will be less than VFR or IFR minimums.

In addition, a synopsis of the flight tells you fuel used, reserves in gallons and time, fuel/time/distance used to climb, how far out from your destination you should begin your descent, and how fast your descent should be to maintain a gentle 2 degree descent profile.

The printed flight log also has distance & true course, along with RNAV cross-bearings conveniently arranged for easy entry into the Heathkit OC-1401 Aircraft Navigation Computer for in-flight use.

The system comes complete with sample aircraft & route data, and the airport/navaid database has over 100 airports, navaids, intersections and checkpoints already on file.

Program Content: There are eight programs called by seven user-selectable items on the master menu. The master menu includes the following commands:

DIRECTIONS & GUIDANCE
INPUT/REVISE AIRPORT & NAVAID DATA
INPUT AIRCRAFT PERFORMANCE DATA
AUTOMATIC ROUTE PREPARATION
AIR NAVIGATION & FLIGHT PLANNING
NAVIGATE PRE-PLANNED ROUTE
SORT & LIST DATA ON FILE

DIRECTIONS & GUIDANCE: This prints the documentation file which explains the operation of the NAVPROGseven system.

INPUT/REVISE AIRPORT & NAVAID DATA: All facilities are identified and accessed by their FAA identifiers, or if one is not assigned you can make one up. You don't have to worry about duplicating identifiers since the program accepts this as normal. For example, DPA is the identifier for DuPage County Airport near Chicago, but DPA is also the identifier of the DuPage VORTAC (a VHF omni-directional navigation beacon) located 4.3 nautical miles to the west. Since the two are not co-located, separate entries should be made for each. Later, when you request "DPA" the computer will find both and ask which one you want.

During new data entry, the computer will prompt you for the following:

FACILITY CODE: What's there - an airport, a non-directional beacon, a VORTAC, etc.
FREQUENCY of navigational aid
NAME: city & state
LATITUDE
LONGITUDE
MAGNETIC VARIATION
ELEVATION

The RNAV functions can be used to calculate latitude & longitude based on bearings from two points that the computer already has on file, or the distance & bearing from one known point. Data for each checkpoint is stored on a random file, five to a sector. A random index holds only the identifiers and tells the computer the relative location of each data block. A separate file holds the RNAV cross-bearings. You can revise or delete data as often as you like, or you can let your library of checkpoints continue to grow.

INPUT/REVISE AIRCRAFT PERFORMANCE DATA: Data for each aircraft is filed according to the plane's N-number. The program will prompt you for the following data from the plane's Operations Manual: fuel used for startup/taxi/takeoff; normal cruising altitude; fuel/time/distance to climb to cruise altitude; true airspeed; fuel consumption; fuel on board; and the cost per hour to operate the plane.

AUTOMATIC ROUTE PREPARATION: After entering your departure & destination points, this program scans the database and automatically prepares several routings using checkpoints along your great circle route. With each pass the checkpoints are closer & closer together. Four separate routes can be prepared for a 500 nm flight in about 8 minutes.

NAVIGATION AND FLIGHT PLANNING: Using this program simply enter the number of checkpoints you'll be using, and the identifiers for each checkpoint. The data for each checkpoint is retrieved and displayed as each identifier is entered. If the route is one you expect to fly again it can be saved for future use.

Next enter the name of the Flight Service Station you'll close your flight plan with, and the winds forecast for each checkpoint. The computer then asks for your cruising altitude, true airspeed, fuel consumption and fuel on board. However, the computer also retrieves & displays the "normal" situation entered with the aircraft data, and by simply

responding with carriage returns this data is used, or you can enter new data to be used for the flight.

After entering your cruising altitude, the computer again refers to the aircraft data and uses the departure elevation to calculate your climb profile, and the destination elevation to figure your descent profile.

With NAVPROGseven you can cycle through the flight repeatedly, plugging in new variables and printing out the best flight profiles at the end of each cycle -- comparing fuel burn vs. time vs. cost in order to get the best use out of your plane.

After the flight log is printed out, another program is called automatically to compare the checkpoints used in the flight with the RNAV cross-bearings on file, which are also printed.

NAVIGATE PRE-PLANNED ROUTE: This program retrieves the checkpoints for a flight that was saved at an earlier date. If checkpoint data has been revised since the last time the route was flown, the new data will be used. A printout can also be made of all routes on file.

SORT & LIST DATA ON FILE: The sort program uses a Schell-Metzner sort routine to printout the airport/navaid database in easy-to-use listings by Identifier; by State & Identifier; by State & City; by State, Facility & Identifier; or unsorted straight from the file.

Comments: NAVPROGseven was written for aircraft navigation but is not limited to pilots alone. The great circle navigation and radio beacon cross-bearings can be helpful to sailors who also have a need to navigate efficiently, though at a slower pace. To assist in modifying the program the documentation includes references to books and articles on navigational mathematics.

The system is menu driven and includes a prologue for "turn-key" operation. The programs are self-prompting with one-key responses and many safety checks that allow the user to go back to the menu and start over.

As mentioned above, the program comes with sample data files that allow the pilot to "test fly" the system before creating his own database, and there are over 100 airports and navaids already on file that will get the pilot off to a running start.

Heath Related Products

Howard Nurse of COMMSOFT recently called and suggested that he has software and hardware available for the "Ham" types out there that is capable of the same operations as described in Issue 22 of REMark. He offers a variety of other software also. Since Howard is a Ham and has written some of the articles and software that HUG offered back in the old days, it seems obvious that he has much experience in this field. We suggest that you obtain further information by writing:

COMMSOFT
665 Maybell Avenue
Palo Alto, CA 94306

Phone: (415) 493-2184

Editors Note: We received the following information that would supplement John Beran's article on changing the ASCII information to create new characters.

Dear HUG,

My company produces custom engraved keytops for Heath/Zenith computers, plus a special set for the Magic Wand word processing program. This particular kit provides the user with fast recall of commands. I've also supplied custom keytops to users who just want to change the color of some keys on the keyboard.

I believe that our products will be of interest to your readers. Response from Heath/Zenith users has been good.

Sincerely,

Ken Kaplan
Arkay Engravers, Inc.
Box 916
2073 Newbridge Road
Bellmore, NY 11710

Ray Livingston of Livingston Logic Labs recently sent us a letter describing many neat products that compliment both the H-8 and H-89. Ray has the PC-17 Double Density 5" Controller for the H-8 and H-89 as well as the necessary supporting software. For further details on the offerings of Livingston Logic Labs, we suggest that you write Ray at the following address:

Livingston Logic Labs
P.O. Box 5334
Pasadena, CA 91107

CP/M Part III

In last October and November issues of REMark, Issues 21 and 22, we started a series of articles on CP/M for the beginner. This issue will continue with the series and is intended to get you up and running with CP/M.

In the last issue, we discussed the four subsections of CP/M. The Console Command Processor (CCP), which is the interface between the computer and the user. The Basic Input/Output System (BIOS), which handles the input and output operations between the computer and any peripherals. The file management controller of CP/M is called the Basic Disk Operating System (BDOS). And lastly, the Transient Program Area (TPA), which is the area in memory that is reserved for user selected programs.

We briefly touched on the "cold" and "warm" boot procedures. Also, we defined ambiguous and unambiguous file names and file extensions. This brings us to the point where we can

Bootup CP/M.

Let's begin!

First, it must be assumed that you have made the necessary hardware changes, i.e. the Extended Configuration or "Org 0" Modification, as explained in issue 21. You must have purchased either the 5 1/4" or 8" CP/M operating system, as detailed in the same issue. For ease of teaching, we will assume the 5 1/4" system.

Having done these steps, we are ready to turn on the computer and "cold boot" the system.

Turn on your computer and any/all disk drives that may be part of your system. Insert your CP/M Distribution Disk I in your primary drive, which is called drive A:.

For the H-89 users, do a "SYSTEM RESET" by pressing the SHIFT and RESET keys on your keyboard. Type a "B" for "Bootstrap" at the "H:" prompt that appears on the screen. Your system will respond with the word "BOOT". Type a Carriage Return, <CR>, and you will be off and running.

The system will display a message on the screen that will say:

"32 K HEATH/ZENITH CP/M 2.2.XX"

where the "XX" is the current version of the CP/M operating system.

The "cold boot" procedure was pointed out in the diagram that was pictured in issue 22. The "Bootstrap" routine from Track 0, Sector 1 is being read into low RAM and will execute causing the CP/M Monitor to be loaded into High Memory. That will cause the screen to display the first series of messages of the CP/M "First Time" bootup routine. See "Display I". (We'll continue with that shortly.)

For the H-8 users, do a "System Reset" by simultaneously depressing the "0" and "RST/0" keys on the front panel of the H-8, and then press the the number "1" or "4" key, on the front panel, and you will be . . well, almost off and running. This will need some explanation.

Your screen will also display the "32K HEATH/ZENITH CP/M 2.2.XX" message as explained above. At that point, your system will appear to stop. You will need to type the space bar a few times and then the CP/M Monitor will be loaded into High Memory. The screen will display the first series of messages of the CP/M "First Time" bootup routine.

Why is typing the space bar necessary with the H-8? With the H-89, CP/M assumes a default BAUD rate value of 9600 BAUD. (BAUD is the rate at which the CRT (or your screen) can communicate with the microprocessor.) Therefore, the "Bootstrap" routine automatically loads the CP/M Monitor into High Memory. However, with the H-8, CP/M does not assume any value for the BAUD rate of the CRT. This is because there are a couple of possible CRT's available that can run with the H-8. (The H-89 has only one, the built in H-19 terminal.) Thus the need for typing the space bar to determine the BAUD rate of the CRT.

(Just a small note; CP/M, with the H-8, does not know whether the system has the H8-5 or the H8-4 SERIAL I/O card. By typing spaces, the system determines the baud rate and also which SERIAL I/O card is used.)

Let's look more closely at this

"First Time Bootup Routine".

CP/M does some hardware checking, of your entire system, each time it is called upon to do so. This "First Time Bootup Routine" does this checking automatically and displays the results on your screen. This gives you the opportunity to verify that you and CP/M agree on what your hardware system consists of.

The "First Time Bootup Routine" is actually a transient program of CP/M, called CONFIGUR. On your "Distribution Disk I", the first thing that appears on your screen is the results of entering this program, CONFIGUR. So, it is actually CONFIGUR that does the hardware checking. (We will talk about CONFIGUR later in the article.)

Looking at "Display I", we see that the "Bootup Routine" or CONFIGUR starts by indicating the basic but important verification data. It displays the version number and your serial number of the CP/M system that you have purchased. It then lets you know that its purpose is to configure "the CP/M operating system to a particular hardware environment".

CONFIGUR then displays your hardware environment as it determines it to be. If the information given is not what you feel your system consists of, check any connectors for proper connection. Once you agree with CP/M, you are ready to continue.

Your Distribution Disk I, disk "A:", is write protected, which will not allow you to make any changes to it. So, at the "STANDARD SYSTEM (Y OR N)?" question, a "Y" or <CR> is sufficient for a response. This will cause the "A>" prompt to appear on the screen, meaning, you are now at the CP/M command level.

Now that you have reached the command level of CP/M, you are ready to begin the first steps to creating your own SYSTEM disk. Let's look at a brief outline of the next steps that you will need to take.

I. FORMAT a disk
II. MOVCPM5 (create a system image)
III. SYSGEN (write the system image)
IV. PIP *.* (copy the transient files)
V. DUP (option of two disk system)
VI. REBOOT SYSTEM disk
VII. CONFIGUR the SYSTEM disk

After this point, you will be ready to go on and explore CP/M.

First, let's take a more detailed look at how to go about creating a SYSTEM disk.

FORMAT

When you receive a diskette (disk) from a manufacturer of said product, the disk will not be in the proper format for use with the CP/M operating system. That means the disk will not be recognizable by the system and will be rejected as a bad disk.

In order for CP/M to recognize any disk, it needs to be initialized or formatted. The transient program which accomplishes this procedure is called FORMAT. (For HDOS users, this is equivalent to INIT.) FORMAT is also used to reinitialize used CP/M disks.

In order to run this initialization, at the "A>" command prompt, enter "FORMAT" and answer the questions appropriately. The instructions of FORMAT are contained in your CP/M Manual and will not be detailed here. The important thing to remember for single drive users is that you will always be using drive A:. For the others, choose an appropriate drive.

FORMAT "blanks" the disk by placing the byte E5H in each byte location of a sector on a disk. The disk may then be used as a data disk under CP/M or it may be made into a system disk by running MOVCPM5 and SYSGEN.

MOVCPM5

Most HDOS users are probably asking "what is MOVCPM5?" This is the first transient command that does not relate to any commands of HDOS.

Do you recall the first line that CP/M places on the screen when the Distribution Disk I is booted up? "32K HEATH/ZENITH CP/M 2.2.XX". The 32K is not a value that is pulled from the air, nor is it the amount of memory in my computer. My H89 has 48K of memory. (Refer to "Display I".)

CP/M has the ability to be configured for any sized memory computer . . AND it can be reconfigured for any memory size on any particular machine, up to the maximum amount of memory of the system. CP/M is shipped with the system configured at 32K, because most systems have at least 32K of memory.

This means that even though my computer has 48K of memory, CP/M only recognizes 32K, until it is told that the system has more memory. MOVCPM5 (or MOVCPM8 for 8" disks) is the program that reconfigures the CP/M system to a particular memory size.

After you have FORMATted a disk for your new SYSTEM disk, place the Distribution Disk I into A: and type "MOVCPM5" (or MOVCPM8 for an 8" system). MOVCPM5 will determine the amount of memory and construct the new CP/M system "image" for the new memory size and return to the command level.

With MOVCPM5 you have created a new system image IN MEMORY, it has not been stored on the disk. You now need to store this to your SYSTEM disk. To copy the system image from memory to disk (or from disk to disk), CP/M has the transient program

SYSGEN.

SYSGEN copies the system image from a source, be it memory or another disk, to a destination disk.

To do this, enter "SYSGEN" at the A> prompt. The next question will ask for "SOURCE DRIVE NAME (OR RETURN TO SKIP):", enter a RETURN (at this time). For "DESTINATION DRIVE", choose whatever is appropriate for your system and place the FORMATted disk in same. Then enter a RETURN and the new system image will be copied to your formatted disk.

You have just created a SYSGENed disk under CP/M, which is configured for the memory size of your system. SYSGEN does not place or copy any of the transient programs from the Distribution Disk to the SYSTEM disk.

One important note: You will need to reboot the system after the SYSGEN procedure, if you have created a new memory size image. (You will really have no choice, as the system stops.)

There is only one more step to complete in order to finish your SYSTEM disk. Your first SYSTEM disk should contain all the transient programs or "files" of the Distribution Disk I. To copy the files from the Distribution Disk I to your SYSTEM disk, you will need to use the transient utility program called

PIP.

Peripheral Interchange Program (PIP) is the CP/M utility which will copy or move one/any/all files from one disk to another. If your system contains

a printer, PIP will be your media for making a hardcopy of your disk files. This will be one of your most widely used utilities.

PIP is not limited to unambiguous filenames. The wild cards "?" or "*" may be used to transfer any combination of ambiguous filenames. (To any HDOS users, PIP can be considered an "old friend".)

You are now ready to copy the Distribution Disk files to your SYSTEM disk. The following is the command line for invoking PIP for any number of drives, including a single drive:

```
A>PIP B:=*.*[R]
```

(The [R] is a "flag" that will allow PIP to copy Read Only (R/O) files.)

For a two or three drive system, you will insert the Distribution Disk I in A: and your SYSTEM disk in B:. Very straight forward. For a single drive system this will need an explanation.

The BIOS of CP/M creates three "logical" disk drives, even when you have only one "physical" drive. The "logical" drives are "mapped" or rerouted to your "physical" drive, drive A:. For example, if you call for "TYPE B:SAMPLE.DOC" at the command prompt A>, CP/M will prompt you to "Put disk B in 5.25 inch drive 0 and press RETURN". Then it will begin displaying the file SAMPLE.DOC on the screen.

When using PIP, this is exactly what happens with a single drive system. CP/M will prompt you to replace your SOURCE and DESTINATION disks at the appropriate times.

Two NOTES at this time:

1) For a single drive system this process is extremely slow and tedious. For each file on the disk, it will take at least two disk swaps (one for the file name and one or more for the file).

2) The DUP utility (which has not been mentioned up to this point) can only copy to "physically" separate disks. It does not "map" to drive A:.

At the conclusion of PIP, you will have completed your SYSTEM disk, which will now be indentical to your Distribution Disk I, with the exception that your SYSTEM disk will be configured for the amount of memory of your computer. The important point is that the SYSTEM disk will contain all the transient programs of the Distribution Disk.

NOTE: The order of execution of FORMAT, MOVCPM5, SYSGEN, and PIP may vary slightly. FORMAT must be run first, however PIP may precede MOVCPM5 and SYSGEN.

This completes the process of creating a SYSTEM disk. No matter how many drives your system has, you will need to know the above steps. However, for computer systems that have more than one drive, FORMAT and PIP can be "bypassed" by using the CP/M transient program

DUP.

DUP is the utility program, which creates a duplicate copy of one disk onto another. The only stipulation is that the disks must be the same size and same density.

DUP, as noted above, will only copy from "physical" drive to "physical" drive. It does not "map" to a "logical" drive. It is for this reason, that single drive systems cannot use DUP, and bypass the aforementioned steps.

To make a copy with DUP, reboot with the Distribution Disk I in A: and a blank disk in B:. (Please note: You do not need to FORMAT a disk, before using DUP.) Enter "DUP" at the command prompt A> and follow through the menu. (For detailed instructions, refer to your CP/M Users Manual.)

At the conclusion of DUP, your SYSTEM disk will be a duplicate of the Distribution Disk I, including the 32K memory image size. Enter "MOVCPM5" to construct a new memory image. Run "SYSGEN" to copy the system image from memory to your SYSTEM disk on B:.

You will now have a SYSTEM disk indentical to the SYSTEM disk described above after using FORMAT, MOVCPM5, SYSGEN, and PIP.

Be sure to make a copy of your CP/M Distribution Disk II. The Distribution Disk II does not need to be a SYSGENed disk, therefore, you need only FORMAT a data disk and PIP the files to the data disk. You can then put your CP/M Distribution Disks away in a safe, cool, dry storage area.

Now that you have completed your SYSTEM disk, let's take a look how to set up the disk for normal operations of CP/M by looking closer at the transient program

CONFIGUR.

Your CP/M Users Manual has the details on the complete list of available menus for CONFIGUR, so we will not go into great detail here. We will point out to you a few of the basic settings, that will help you get started.

As explained earlier, CONFIGUR does the hardware checking when you first bootup on your Distribution Disk I. CONFIGUR is much more than a "hardware checking" utility.

CONFIGUR is the utility that allows the user to customize his system in any number of ways and, after execution, the CP/M BIOS will recognize any hardware environment that is specified. (For HDOS users, this is similar to the device drivers and their subsequent SET commands.) CONFIGUR does the configuring of the BIOS for any/all peripherals.

CONFIGUR may be run from the command level at any time, and can be considered one of the most important transient programs of CP/M. Your system, even a very simply one, has many different settings that can be user customized by reCONFIGURing the CP/M BIOS.

```
*****************************************
*                                       *
*  32K HEATH/ZENITH CP/M 2.2.XX         *
*                                       *
*  HEATH/ZENITH CONFIGURATION PROGRAM    *
*  VERSION 2.2.XX                       *
*  SERIAL NUMBER: YYY-YYYYY             *
*                                       *
*  THIS PROGRAM CONFIGURES THE CP/M OPERATING SYSTEM TO A
*  PARTICULAR HARDWARE ENVIRONMENT.     *
*                                       *
*  PLEASE WAIT DURING HARDWARE VERIFICATION...
*                                       *
*  H/Z89 WITH 48K OF RANDOM ACCESS MEMORY (RAM)
*  01 MINIFLOPPY DRIVE(S)               *
*  CRT BAUD RATE IS  9600               *
*  03 ADDITIONAL SERIAL PORTS FOUND     *
*                                       *
*                                       *
*  DRIVE A DISK IS WRITE PROTECTED.     *
*  MODIFICATIONS WILL NOT BE MADE TO THE DISK FOR THIS CONFIGUR RUN.
*                                       *
*  STANDARD SYSTEM (Y OR N)? <Y>:_      *
*                                       *
*****************************************
```

DISPLAY I

```
*****************************************
*                                       *
*  CP/M CONFIGURATION                   *
*                                       *
*  A    SET TERMINAL AND PRINTER CHARACTERISTICS
*  B    SET DISK PARAMETERS             *
*  C    CHANGE THE DEFAULT I/O CONFIGURATION
*  D    AUTOMATIC PROGRAM CONTROL       *
*                                       *
*  X    CONFIGURE, MAKING CHANGES TO MEMORY ONLY
*  Y    CONFIGURE, MAKING CHANGES TO BOTH MEMORY AND DISK
*  Z    QUIT, MAKING NO CHANGES         *
*                                       *
*  SELECTION:                           *
*                                       *
*                                       *
*                                       *
*                                       *
*                                       *
*                                       *
*                                       *
*                                       *
*****************************************
```

DISPLAY II

For example, in "Display I", the output is all upper case characters. CP/M is configured to "force" all lower case to upper case on output. This is just one simple, but important modification that we will change through CONFIGUR.

BAUD rates, PORT values, Nulls after <CR>, Disk Step rate, Error messages, Peripheral Defaults, and Automatic Execution of the Command Line are all options that can be modified through CONFIGUR. Making modifications will give you the opportunity to become more familiar with the menu levels of the CP/M CONFIGUR.

Notice that "Display II" is the main menu of CONFIGUR. From this menu, you can choose an option by selecting the appropriate letter.

NOTE: It is possible to leave CONFIGUR, without making any changes to the BIOS, by entering a "Z". "X" makes the changes in memory, but, upon reboot, no changes will have been made to the BIOS on the disk.

To get started, bootup on your SYSTEM disk, and this time, enter an "N" at the "STANDARD SYSTEM (Y OR N)?" question. That will bring you to the "CP/M CONFIGURATION" or main menu of CONFIGUR as shown in "Display II".

We have three changes that we want to make to the BIOS at this time. 1) We do not want to have lower case letters forced to upper case. 2) When pressing the DELETE key, we want it to "backspace" rather than echo the character. 3) We no longer want to "default" into executing CONFIGUR at bootup.

From the main menu, enter an "A", which will bring up the menu for setting the terminal parameters. At this point, by entering an "F", it will "FORCE OUTPUT TO UPPER CASE ON CRT: FALSE". Then enter an "L" and CONFIGUR will "ECHO ON DELETE: FALSE".

Enter a "Y" as your choice for "FINISHED, MAKE CHANGES AND RETURN TO MAIN MENU". At the main menu, enter a "D" to go to the menu that controls the automatic bootup execution. Enter an "A" and you will set "RUN AUTOMATIC COMMAND LINE ON COLD BOOT : FALSE".

NOTE: When entering a selection, a single keystroke is all that is required. In our examples, the selected letters, "F", "L", and "D" from their respective menus, will change the value of "TRUE" to "FALSE". (The RETURN key will NOT be pressed to complete execution of a selection.)

Now, enter a "Y" to return to the main menu. Enter a "Y" and the changes will be stored in memory and on the disk. Reboot your SYSTEM disk and you are ready to start using your CP/M operating system.

We have come to a point where you will have to begin to use and play with CP/M on your own. This article has covered many points to get you started.

It should be noted again, that these series of articles are not intended to replace the CP/M Users' Manual and Users' Guide. The approach is one in which the steps of operations are explained in the order of proper execution. You will need to refer to your CP/M Users' Manual for details to any of the information that is supplied in this series.

It is not necessary for the next few series of articles to have a set pattern. Each issue will bring more helpful information for aiding you in your use of CP/M.

< TLJ >

HUG/Heath Sale

As some of you may have noticed, Heath has reduced some of prices of the products that were offered in the HUG/Heath Sale described in REMark Issue 22. Heath has agreed to offer another extention for the following items and an increased price break for those items that were reduced in the Heathkit Catalog. As described previously, you must include an exact copy of page 30 of REMark Issue 22 to take advantage of the sale through January 31, 1982. The sale of the following items will be terminated on March 31, 1982, and will only include the following extended items for members of the Heath Users' Group. You will be required to send a copy of page 30 of REMark 22 and a copy of this page for this latest sale.

ITEM		PRICE	
H-17-3	Three-Drive Modification	$ 59.00	(Extended to March 31, 1982.)
HA-8-3	H8 Color Board	$325.00	(New price)
HA-8-8	Extended Config. Option	$ 39.00	(New Price)
HOS-1-SL	HDOS 2.0 Source listings	$ 65.00	(Extended to March 31, 1982.)

NOTE: All prices shown will expire on March 31, 1982.

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

CUT ALONG THIS LINE

HUG MEMBERSHIP RENEWAL FORM

When was the last time you renewed?

Check your ID card for your expiration date.

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

Name _____

Address _____

City-State _____

Zip _____

REMEMBER — ENCLOSE CHECK OR MONEY ORDER

CHECK THE APPROPRIATE BOX AND RETURN TO HUG

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

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

VECTORED FROM 18

Also, copy the patched MBASIC to the disk. Now when you boot this disk, it will come up with a menu that will allow you to play the game, read the documentation, or duplicate the disk. The duplicate function will only work if your disk is 40 track single sided hard sector. Otherwise, the disk can be copied using SYSGEN *.*.

Note: We are going to re-do our HDOS DND disk and release it as a data disk rather than a bootable disk in the future, with instructions for setting up your own system disk. A program will be included that will make the patch to MBASIC for you.

PS:

CATALOG CORRECTION

The latest Heathkit catalogs incorrectly state that CP/M version 2.2.03 is for the H89 only. It will work on an H8 as long as it has either the Extended Configuration Option (HA-8-8) or the Heath Z80 board (HA-8-6). HDOS 3.0 should also work on the H8.

VECTORED FROM 17

885-1204	CP/M Volumes 26/27-A and B	%% $ 21.00
885-1205	CP/M Volumes 26/27-C and D	%% $ 21.00
885-1206	CP/M Games Disk	%% $ 21.00
The above CP/M products are 2 disks each.		
885-1207	TERM and H8COPY	$ 20.00
885-1208	HUG Fig-Forth H8/H89 2 Disks	$ 40.00
885-1209	Dungeons and Dragons Game MBASIC and H89 or H8/H19	$ 20.00
885-1210	HUG Editor	$ 20.00
885-1211	Sea Battle Game for CP/M	$ 20.00
885-1212	CP/M Utilities I	$ 20.00
885-1213	CP/M Disk Utilities	$ 20.00
885-1214	Amateur Radio Logbook	$ 30.00

% Means CP/M 1.43 only (ORG-4200)
%% Means CP/M 1.43 or 2.2 (Heath)
Other CP/M disks are for 2.2

MISCELLANEOUS

885-0017	H8 Poster	$ 2.95
885-0018	H89 Poster	$ 2.95
885-0019	Color Graphics Poster	$ 2.95
885-4	HUG Binder	$ 5.75
885-4001	REMark VOLUME I	$ 20.00
885-4002	REMark VOLUME II	$ 20.00

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

Heath Users' Group
Hilltop Road
St. Joseph MI 49085