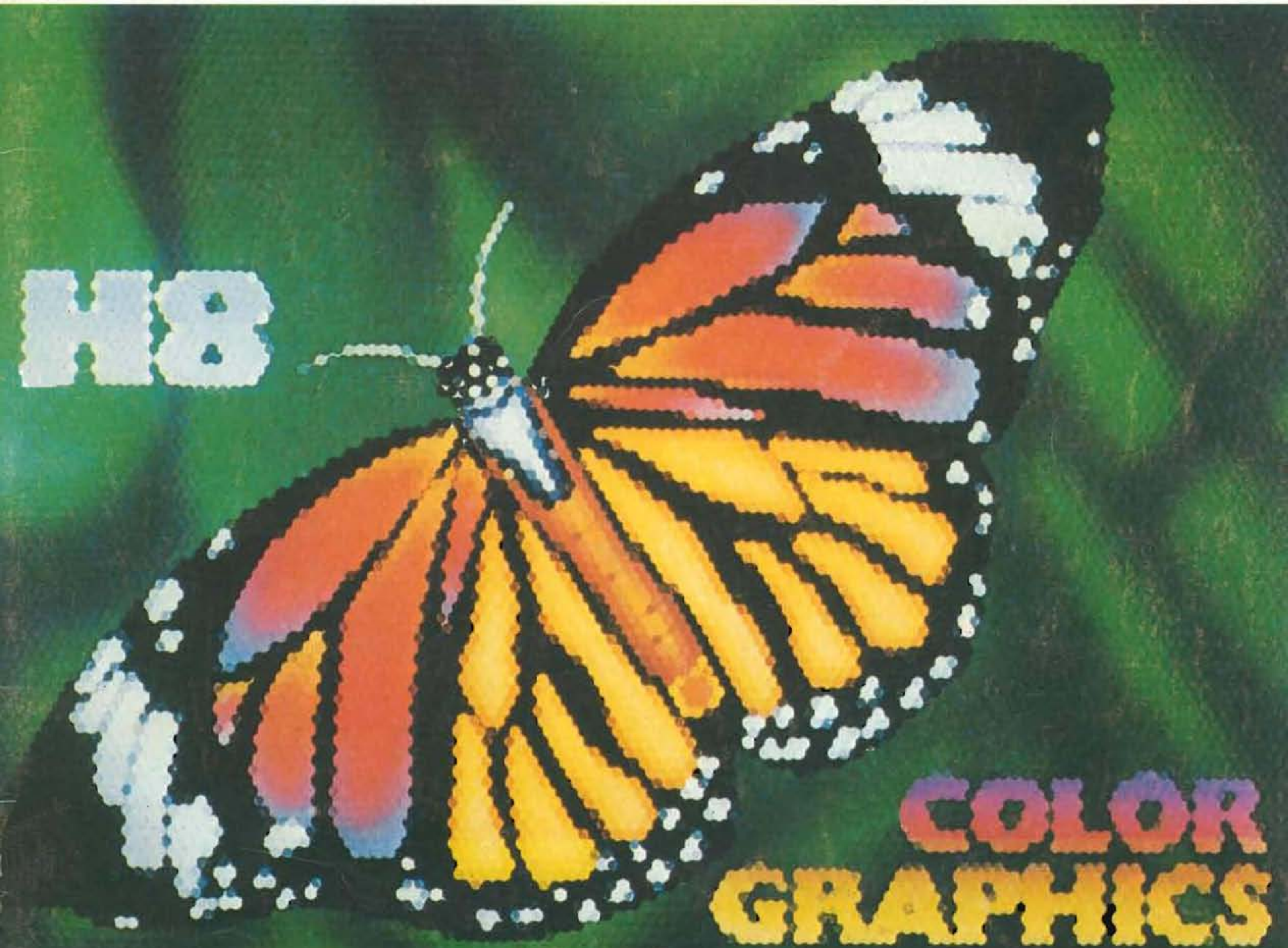
REMark

Issue 12 • November 1980



Official magazine for users of Heath computer equipment.

on the cover . . . .

Here's another of our great new wall posters.

on the stack

>CAT

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

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

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

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

Send payment to:

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

HUG Manager and Editor .... Jim Blake
Assistant Editor and
Software Developer .... Patrick Swayne
HUG Secretary .... Nancy Strunk
Software Developer .... Gerry Kabelman
Software Developer .... Jon Falkner

REMark

The New HDOS

By the time you read this, HDOS version 2.0 should be available. Most of you know by now that this is the HDOS that supports 8-inch drives (H47), but you may not know that it has a lot of other new features.

HDOS 2.0 comes with so many goodies that it takes up 3 5¼-inch disks. The first one contains the HDOS system that you boot up with. One of the others has all the device drivers in assembly source, including the disk drivers. The third disk contains the assembler (a neat new version -- more on that later), DBUG, and .ACM files for just about anything immaginable. The documentation includes the HDOS System Programmer's Guide, and .ACM files for everything in it are on the third disk.

When you boot up the new HDOS, it asks you to type spaces the first time, and then stores the baud rate information on the disk so that you do not have to type spaces from then on. It also stores the date, so you can shut off the power, re-boot, and the date is still there. You can even instruct HDOS 2.0 not to ask you for the date at boot-up, and it will use the one already stored. There is a new BOOT command that allows you to boot up on any drive while in the command mode. The drive you select becomes SY0:, and the others are re- assigned. HDOS 2.0 can support up to five drives: three 5¼-inch and two 8-inch. That's about 2.3 megabytes of on-line storage.

The INIT and SYSGEN programs in the new HDOS can use two drives, if you wish, so you do not have to do a lot of disk swapping. With SYSGEN, you can specify the switch /MIN, which will produce a system disk with only the files absolutely necessary for system operation. You can also say SYSGEN *.*, and make an exact copy of a disk. There is a TEST47 program for the 8-inch drives and a new version of TEST17 for the 5¼-inch ones. Both of these programs provide for hardcopy listings of your test results.

As I mentioned before, HDOS 2.0 comes with a new version of the assembler (ASM). It includes a cross reference utility that will give you a cross-referenced list of your program's symbols, if you wish. It allows .ACM files to be on any disk, and will search all disks automatically for them. You can specify which drive it is to start the search on. The documentation will include more HDOS interface information than previous versions.

All in all, the new HDOS is a very nice package, and well worth the switch, even if you are not contemplating getting 8-inch drives. It is better than any other "hobby" operating system that I know of, and rivals many "professional" systems. HDOS owners who have had their systems for less than a year will get 2.0 free. If you have had HDOS for more than a year, you will be able to get the new one for $55. The price for new owners is $150. HDOS 2.0 is in the new Christmas Heathkit catalog.

ROOM FOR MORE

As you go through the pages of this issue of REMark, you will notice that a number of the articles are signed PS:. This issue is being prepared before the last one is shipped (which is necessary to put out an issue per month), and there just were not enough articles to fill it, so I wrote some. With publication this often, there is room for more articles. If you have some ideas you think other Heath computer users could use, send them in. We prefer that you send your articles on disk or cassette, but I will take crayon on a grocery bag if it is a good idea.

STICK WITH HEATH

I'm sure you have noticed that some non-Heath manufacturers are beginning to offer CPU boards for the H8 with features that make them look fairly attractive. If you are considering one of these, you should be aware that they may not be compatable with some of the new products from Heath, such as the Extended Configuration Option in the new catalog. Stick with Heath, and you'll always have a system that works together.

PS:

"Turnkey" Operations with HDOS Version 1.6

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

ABSTRACT

Simple ways to accomplish automatic bootup with HDOS vs 1.6 are discussed. Some valuable, inexpensive utility programs are covered.

BACKGROUND

J. J. Thompson published an article on the HDOS "type-ahead buffer" in REM, issue 7, page 11 with a demo pgm in BASIC on page 23. Jack markets a CPM to HDOS (and vice versa) conversion program (write him at 281 Warren Avenue, Kenmore, NY 14217 for details).

Jim Blake had an assembly language listing of "PROLOGUE" on page 14 of REM, issue 9.

Jim Teixeira (62 Churchill Street, Sudbury, MA 01776) markets an HDOS modification program for $15.00 (slightly higher in selected Heathkit Electronic Centers).

Steve Robbins (4610 Spotted Oak Woods, San Antonio, TX 78249) markets a very nice line editor for the H8 and H89 and also wrote a letter to, and published in, BUSS, issue No. 25. Write Steve for details of his line editor.

HUG markets a disk containing a DUMP.ABS program and documentation, along with other nice utilities.

HUG also markets a disk with SUBMIT.ABS on it.

What do all of the above have to do with "turnkey" operation of HDOS? Read on and discover how easy it is to modify HDOS, without any knowledge of machine language!

In the process of "booting up", HDOS looks for any program named as PROLOGUE.SYS. If HDOS finds any executable (.ABS type) file named PROLOGUE.SYS, then it loads and runs it. As an example, you could rename BASIC.ABS to PROLOGUE.SYS; then, on bootup after entering the date routine, you would automatically wind up in BASIC, rather than at the monitor prompt (>). IF, on the other hand, HDOS does NOT find any executable program called PROLOGUE.SYS, THEN HDOS automatically loads and executes SYSCMD.SYS and this culminates with the monitor prompt (>). (This latter occurrence is what ordinarily happens when you "boot up" HDOS.)

However, disk users can use the "type-ahead" buffer to easily create a file named PROLOGUE.SYS by using SUBMIT (HUG) or DOCOM (a part of Jim Teixeira's SYSMOD package).

THE TYPE-AHEAD BUFFER

HDOS always reserves a space in memory for this buffer. All input from your terminal keyboard (from the monitor prompt >) goes into this buffer and the input is not acted upon by HDOS until you hit the Return key. For example, when you type CAT from the monitor prompt (>), first the "C" goes into the buffer, then "A", then "T". Nothing happens otherwise until you hit the Return key, at which time the directory of the disk is displayed on your terminal. Try typing CAT, hit Return and immediately type CAT and hit Return again (note that the directory will be in the process of being displayed while you type in the 2nd CAT and return). You will see a total of 2 listings of the directory, one right after the other. Now, since the type-ahead buffer holds 100 characters, if you had a way of stuffing this buffer with several commands, and if you could easily call up ALL of these commands with one keystroke, then ALL of these commands would be executed one after the other! Well, you CAN easily do this with either SUBMIT or DOCOM. I prefer DOCOM since it is part of Jim's SYSMOD package and this package also lets you use 1 or 2 letter abbreviations to do many functions of HDOS. For example MOUNT SY1: becomes M1 with SYSMOD.

Here is a practical example of all that is required using DOCOM (it is assumed that you have purchased Jim's package and modified HDOS according to Jim's very clear documentary - it takes only a few minutes.)

From the monitor prompt (>) type:

```
DOCOM X=M1;LOAD LP:;MBASIC MYPGM          (now, hit the Return key).
```

The above creates on your disk an executable program called X.ABS. Now from the monitor prompt (>) simply type X and hit Return., The disk in SY1: will be mounted, LP: will be loaded, MBASIC will be loaded and your MBASIC program called MYPGM.BAS will be run and all this is done automatically for you.

Note the ";"s which appear in the DOCOM line above. These are converted to Returns (more accurately "new line characters" in HDOS) by the DOCOM function of SYSMOD. And, when you type X and hit the return key, X.ABS stuffs the type-ahead buffer with the commands entered. Then the commands are executed for you one after the other. All automatically.

NOW, consider that if we had named the program (instead of X) PROLOGUE.SYS (or if we renamed X.ABS to PROLOGUE.SYS), then on boot up we would not even have to type the X and hit Return! We are approaching "turnkey operation".

Next we come to Steve Robbin's role in this saga. He showed how to easily modify the bootup process so that with vs 1.6 of HDOS, one does NOT have to "TYPE SPACES" to determine the BAUD rate (or have to hit Return key thereafter). Steve's instructions, slightly modified, follow. (From BUSS, issue 25 - write BUSS, 325 Pennsylvania Ave., SE, Washington, DC 20003 for subscription information.)

All that remains is to be able to bypass the "date routine" of HDOS. Well, Jim's little package also allows you to modify HDOS to do just that! You simply answer the question about the "date routine" "N" during the SYSMOD process and your resulting system disk won't stop and ask you for the date.

By combining all of the above, we now have true "turnkey" operation. You simply turn on your computer and disk drive(s) and place disk(s) in drive(s). Now, with the H89 you simply type B and hit Return. That is all! You are automatically dropped off in a running MBASIC program named MYPGM in the illustration we have used. With the H8 (we assume you have Heath's "PAMGO" ROM) you just hit the GO key to accomplish everything.

Suppose you do want the date in memory but have followed the above routine. Simply use the following MBASIC program (use it as the target of a GOSUB in your program) and this will poke the correct date into the appropriate places in memory.

One further note. Each version of HDOS seems to have a different location in memory for the "type-ahead" buffer. We are using HDOS version 1.6.

DIRECTIONS FOR PREPARING AUTOBOOT DISKS FOR HDOS VS 1.6

You will need:

HUG's disk VIII - contains DUMP.ABS & DUMP.DOC. Order from HUG.

SYSMOD.ABS & Instructions - obtainable from Jim Teixeira.

HDOS, vs 1.6. You probably already have this.

What you will end up with:

2 SYSTEM disks, one which will be a completely autoboot disk, which - with PROLOGUE.SYS which you can easily create with DOCOM (Part of Jim's SYSMOD) - will drop you off in a running MBASIC program if you so desire; the other disk will stop and allow the date to be entered as usual. Note that with the first disk you can always poke the date into correct memory locations with a suitable MBASIC program; or date can be entered from monitor prompt > - see Jim's instructions with SYSMOD. Note that 2 new System disks will also have, in addition to the automatic boot routines, Jim's very desirable HDOS system modifications - try them, you will NEVER use unmodified HDOS again!

To boot an "AUTOBOOT" disk from an H89, simply type B and hit return. You do NOT have to type "spaces"! (The type spaces message will appear since this message is contained in ROM - simply disregard this message.) Similarly, you do NOT have to type spaces when using an H8!

DETAILED INSTRUCTIONS -

Prepare a copy of HDOS distribution disk vs 1.6 and delete everything with command DELETE *.*. Mount HUG's DUMP.ABS disk in SY1:. Prepare a hard copy listing of SY1:DUMP.DOC and study it thoroughly.

Run DUMP.ABS by typing SY1:DUMP and hit return.

You will only be working with Drive 0, and Track 0.

Answer "drive" question with 0, "track" question with 0, and "sector" question with 0. Answer "modify this sector" question with Y and hit return. Answer "enter address (0-FF in Hex)" question with 16 and hit return. (Note that this is 0010 at left, 6th column at top). Display will now say 0016: 41 A . At this point type Z and hit return. Now, carefully type S, then T, then A, then N, then D, then B, then Y, then 7 dots (periods). After the 7th period, hit the return. Display asks if you want to write it back to disk - answer Y and hit return.

Now, type CTRL B, and reboot your disk. You will see a message that says "STANDBY.......".

What you have just done above is substitute the standby message for the "ACTION? BOOT" one.

NEXT, using exactly the same procedure as outlined in detail above, change the 4 bytes beginning at A5 address (00A0 at left, col 5 above) from BOOT to spspspsp. Simply hit the space bar 4 times after you type the Z and return; after the 4th space just hit return. Write it back to disk and then CTRL B. Boot up again. We simply removed the BOOT message, replacing it with 4 spaces.

NEXT, using same procedure, change the 4 bytes beginning at 54 on Track 0, Sector 4. (Note that this is sector 4)! However, from now on do NOT type the Z as you will NOT be entering ASCII values. Enter each value and hit return. After the last value is entered and return is hit, then type a space and hit return.

Change these 4 bytes to 21, 0C, 00, C9. Then space & hit return, write back to disk and CTRL B. Now reboot. You no longer will have to TYPE SPACES TO DETERMINE BAUD RATE. This "fix" by Steve will set the BAUD automatically on bootup to 9600 on either H8 (with 4 port serial board) or H89.

NEXT, using the same procedure, change 3 bytes beginning at 25 on Track 0, Sector 0 to C3, 22, and 23. Then space and return, write it back, and hit CTRL B. (Note that this is at a different address than that originally used by Steve in his letter to BUSS.) This "fix" removes the CR you used to have to hit after typing spaces to set the baud rate.

Now, dismount SY1: and mount in SY1: the disk containing Jim's SYSMOD package. You will want to make a hardcopy printout of Jim's documentation. Then, copy SYSMOD.ABS to the system disk we are preparing. Now, dismount and remove Jim's disk.

Next, use either DUP or FTCOPY and make a duplicate of disk currently in SY0:. We now have 2 new system disks ready for modification by SYSMOD.

Follow Jim's instructions with one disk (after bootup). Simply type SYSMOD and hit return. Answer question about date Y and hit return. In a few seconds, you have a modified system disk. Now, delete SYSMOD.ABS, BYE and repeat procedure with other disk, but answer date question N (it defaults to N with a Return). When the sysmod pgm is finished, delete it. (Type HELP and hit return from the monitor prompt > to see all of Jim's "goodies" now present on your new system disks.)

You now have 2 system autoboot disks. One bypasses the date routine, and the other does not. Otherwise, the disks are identical.

You can use PROLOGUE to put you in a running MBASIC program. And, if you have booted up the disk without the date routine, here is an MBASIC program to poke the date into memory from any MBASIC program. Use MERGE command to merge it with desired program, and then GOSUB to it where desired. Don't forget to save pgm in ASCII before merge.

MBASIC date routine, compliments of Fox Hill Farms:

```
  10 'POKEDATE.BAS    delete this line and insert GOSUB 5000 at @
                      appropriate place in desired program.
5000 PRINT "Enter DATE, then hit Return."
5010 PRINT
5020 PRINT "Use this format  DD-MMM-YY"
5030 PRINT
5040 LINE INPUT "(Example: 10-JUN-80)... ";DA$
5050 IF LEN(DA$)=8 THEN DA$="0"+DA$
5060 IF MID$(DA$,3,1)<>"-" THEN 5460
5070 IF MID$(DA$,7,1)<>"-" THEN 5460
5080 IF ASC(MID$(DA$,1,1))>51 THEN 5460
5090 IF ASC(MID$(DA$,2,1))>57 THEN 5460
5100 FOR W=4 TO 6
5110 IF ASC(MID$(DA$,W,1))>91 THEN GOSUB 5480
5120 NEXT W
5130 IF MID$(DA$,2,1)<"0" AND LEN(DA$)>8 THEN 5460
5140 IF LEN(DA$)<8 OR LEN(DA$)>9 THEN 5460
5150 IF RIGHT$(DA$,2) <"70" THEN 5460
5160 EE$=MID$(DA$,4,3):FOR I=1TO3:IF ASC(MID$(EE$,I,1))>90 THEN 5460ELSE NEXT I
5170 GOSUB 5250
5180 DA$=LEFT$(DA$,3)+DD$+RIGHT$(DA$,3)
5190 REM   %%% next 3 lines put in ASCII memory %%%
5200 FOR I=8391 TO 8383 STEP -1
5210 POKE I,ASC(RIGHT$(DA$,1))
5220 DA$=LEFT$(DA$,LEN(DA$)-1)
5230 NEXT I
5240 RETURN
5250 DD$=MID$(DA$,4,3):RR$=LEFT$(DA$,2):RR=VAL(RR$)
5260 YY$=RIGHT$(DA$,2):ZZ=(VAL(YY$)-70)*2
5270 IF INSTR(1,"AUGSEPOCTNOVDEC",DD$)<>0 THEN YY=ZZ+1 ELSE YY=ZZ
5280 REM   %%% next line puts 'year value' in 'coded date' memory %%%
5290 POKE 8393,YY
5300 JJ=32
5310 IF DD$="JAN" THEN PP=JJ+RR:DD$="Jan":GOTO 5440
5320 IF DD$="FEB" THEN PP=JJ*2+RR:DD$="Feb":GOTO 5440
5330 IF DD$="MAR" THEN PP=JJ*3+RR:DD$="Mar":GOTO 5440
5340 IF DD$="APR" THEN PP=JJ*4+RR:DD$="Apr":GOTO 5440
5350 IF DD$="MAY" THEN PP=JJ*5+RR:DD$="May":GOTO 5440
5360 IF DD$="JUN" THEN PP=JJ*6+RR:DD$="Jun":GOTO 5440
5370 IF DD$="JUL" THEN PP=JJ*7+RR:DD$="Jul":GOTO 5440
5380 IF DD$="AUG" THEN PP=RR:DD$="Aug":GOTO 5440
5390 IF DD$="SEP" THEN PP=JJ+RR:DD$="Sep":GOTO 5440
5400 IF DD$="OCT" THEN PP=JJ*2+RR:DD$="Oct":GOTO 5440
5410 IF DD$="NOV" THEN PP=JJ*3+RR:DD$="Nov":GOTO 5440
5420 IF DD$="DEC" THEN DD$="Dec":PP=JJ*4+RR
5430 REM   %%% next line puts 'day-month value' in 'coded date' memory %%%
5440 POKE 8392,PP
5450 RETURN
5460 PRINT:PRINT CHR$(27)CHR$(112)"Error - enter again!"CHR$(27)CHR$(113)
5470 PRINT:GOTO 5000
5480 MID$(DA$,W,1)=CHR$(ASC(MID$(DA$,W,1))-32):RETURN
```

THE ICING ON THE CAKE (IF your terminal is an H19, or you have an H89)

You will have noticed that, although we have eliminated all the non-essentials on bootup, the screen displays all the various messages such as version of HDOS, how many K of RAM are in operation, and MBASIC"s sign on message, etc. In case you wonder how to get rid of all these messages we include this routine. Assume that you have prepared an AUTOBOOT disk without the date routine and are using PROLOGUE.SYS to put you into a running MBASIC program of your choice.

We found a "bug" (a "feature"?) in some of the commands that work with the 25th line. This "bug" apparently sends the cursor off the screen. We reasoned that if we could incorporate this at the start of our autoboot, then, not only would the cursor move off screen, but also the messages would be printed off screen! Tried it and it works! Here is how.

First, add the following line at the beginning of your target MBASIC program (otherwise you won't see anything!):

```
6 PRINT CHR$(27)+"H"+CHR$(27)+"E":REM   home the cursor and clear the screen
```

Now, follow the routine we detailed earlier using DUMP where we put in the "STANDBY" message. Drive 0, Track 0, Sector 0. Answer "modify this sector" prompt with a Y and hit Return. Answer "enter address" question with 14 (instead of 16) and hit Return. Type Z and hit Return.

Now, very carefully, type in order the following. Note that you do NOT type the commas, note that there are 2 consecutive ESCs at one point, note that ESC means you hit the ESC key 1 time, be sure and enter lower or upper case exactly as indicated; and disregard the fact that a "tab" will be performed whenever you hit the ESC key. Here we go:

```
ESC,E,W,A,I,T,ESC,x,1,ESC,Y,8,ESC,ESC,y,1
```

Now, hit Return and write to disk. Exit DUMP with a CTRL B. DONE!

Now, reboot.

What we substituted for the "STANDBY message" was an ESC E to clear the screen, then the word WAIT, then we enabled the 25th line, then sent the cursor to the beginning of 25th line, and then used the "bug" that we previously found to get off the screen (don't know why it works but it does!)

FINAL NOTE

Some folks have asked us why we went to the trouble of preparing autoboot disks, when CPM can do the same thing. The reason is simply that we like the many features of HDOS. We like to have a date printed in the directory when we manipulate files, etc. We have NEVER lost any data using HDOS. These features are, and have been, very important to us.

SPECIAL NOTE FOR USERS OF HEATH EXTENDED BH DISK BASIC

You can use Benton Harbor disk BASIC and the above autoboot procedures to drop you off in a running HBASIC program. Here is how to do it:

Create a PROLOGUE.SYS file as outlined above which calls an .ABS file named as you desire. For example, using DOCOM:

DOCOM PROLOGUE.SYS=x Now hit the return key.

The above line will create a PROLOGUE which calls x.ABS. Now, create the x.ABS file as follows:

DOCOM X=BASIC;OLD "MYPGM.BAS";RUN   (now hit the return key).

The above line has created a file called X.ABS, which when called by PROLOGUE.SYS will load BASIC into memory, then will load a BASIC program called MYPGM.BAS, and then will run it. Be sure to include a line in your BASIC program at its beginning to home the cursor as outlined previously. (Otherwise the screen will remain blank, if you used the "ICING ON THE CAKE" way of booting up!)

EOF

EDITOR'S NOTE: If you are going to use SUBMIT to create your PROLOGUE.SYS, and you have the version that works with HDOS 1.5, you can modify it to work with HDOS 1.6 as follows. In the file STUFF.ACM (that came with SUBMIT on HUG no. 885-1060) change the line SUI 7 to SUI 9, and change MVI L,215Q to MVI L,041Q. Then re-assemble the SUBMIT.ASM source. If you modify HDOS as per the article "Changes for HDOS Boot-Up" in REMark #11, you will not need the SYSMOD program mentioned in the above article.

PS:

A Proposed Graphics Command Set for High Level Languages

Now that Heath has the hardware for graphics (the H19/H89), I think it is time for graphics software. In this article, I will present a proposed graphics command set for use in high level interpreters. The idea is to make it easier and faster to produce graphics than with the current methods available, which involve the CHR$ function and a lot of string variable assignments (see REMark issues #9 and #11). There will instead be a set of commands to perform the most common graphics functions. (This discussion involves the BASIC language, but these commands should be adaptable to other languages as well.)

The first and most important command is PLOT. Its format is:

```
PLOT (x,y)=a,b,c,...
```

where x,y is the cursor position and a,b,c, are the graphic characters to be plotted. The 33 graphic characters produced by the H19/H89 will be numbered 1 to 33 with the number 34 used to specify a space (blank cell). You can specify any character in reverse video by putting a minus sign before its number. Minus 34 produces a blank in reverse video, which is a cell filled in with white. The cursor position is specified by column and line. Plotting starts at the specified column and continues to the right for as many characters as are listed following the equal sign. The command PLOT (12,40)=-34 would put a small white rectangle in the center of the screen. The following program will draw a small sail boat in the center of the screen:

```
10 PLOT (10,40)=-2
20 PLOT (11,40)=-34,-2
30 PLOT (12,40)=-34,-34,-2
40 PLOT (13,40)=31
50 PLOT (14,38)=2,-34,-34,-34,-34,21
```

In standard BASIC, the character numbered 2 is CHR$(95), and -2 is that character in reverse video. Number 21 is CHR$(114), number 31 is CHR$(124), and -34 is a space in reverse video. The PLOT command can also be used to position the cursor by leaving off the equal sign and following arguments. PLOT (5,25) places the cursor on line 5, column 25.

The PLOT command will be able to take any expression for an argument, such as a variable or complex expression. If you wanted to draw a line down the left side of the screen, you could say:

```
10 FOR I=1 TO 24
20 PLOT (I,1)=31
30 NEXT I
```

To support the PLOT command with other screen formatting capabilities, the following commands will be added.

Command	Use
CLS	Clears the screen.
ELI	Erases the line the cursor is on.
ETP	Erases from the top of the page to the cursor position.
EBP	Erases from the cursor position to the bottom of the page.
COF	Turns cusor off.
CON	Turns cursor on.
SCP	Saves cursor position.
RCP	Resores cursor to saved position.

These commands would be used in BASIC like other commands, and would take no arguments. To clear the screen before drawing the sail boat in the previous example, change line 10 to read:

```
10 CLS: PLOT (10,40)=-2
```

To make it easier to control screen formatting for normal PRINT statements, three single character functions will be added. They are the reverse bracket (]) to set reverse video, the normal bracket ([) to cancel it, and the exclaimation point (!), to enable printing on the 25th line. The use of these is best illustrated by examples:

```
10 PRINT ];"THIS LINE IS IN REVERSE VIDEO."
20 PRINT [;"THIS LINE IS NOT."
30 PRINT !;];"REVERSE VIDEO ON LINE 25."
```

Each use of the 25th line function (!) will erase what was there before, so PRINT !;"" will erase the 25th line. After the function is finished, the cursor will return to the starting point, that is, to where ever it was before you printed something on the 25th line. If you want to leave the cursor on the 25th line, you can use PLOT, which can also be used to put graphics there.

My first use of the graphics command set will be in a disk version of my Super Tiny-8 BASIC (see REMARK issue 9). Since ST-8 is a very fast BASIC, it may be possible to do simple animation with it. Later, I may try to add these commands to Benton Harbor BASIC or something. If you have suggestions or improvements, I hope you will let me know.

PS:

Maintain a Library in Microsoft BASIC

by D. J. Powers

Those of us who are familiar with compiler languages like FORTRAN are familiar with the techniques used in creating and maintaining libraries of subroutines. Typically, these subroutines are general in nature and are useful to many of the main programs that are written. As an example, a routine to calculate the Julian date would be useful in programs which do elapsed time calculations. In FORTRAN, such a subroutine would exist as an object module after being compiled by the FORTRAN compiler. The subroutine object module could then be linked to the FORTRAN mainline program via a LINKER or TASK builder utility. Sometimes it is convenient to store many object modules in what is called a LIBRARY. For example, a Library may consist of all the TIME/DATE subroutines that you have written. Libraries are created and maintained by a utility program called a LIBRARIAN.

So much for compiler languages. How do we accomplish the same thing with an interpreter like Microsoft BASIC? Remember that the software tools such as the Compiler, the Linker, and the Librarian are not available! As a first step, how do we create a subroutine module? We do it with a Text Editor or directly in Microsoft BASIC. The BASIC subroutine looks like any other BASIC program except that is terminated by a RETURN statement. For example:

```
10 REM
20 REM  Subroutine -- Sort Strings
30 REM
    .
    .
40 RETURN
```

Now how can we save this routine so that it can be easily retrieved when we need it? We can use the SAVE command with ASCII output specified. For example:

```
SAVE "SORT",A
```

We must specify the A for ASCII so that we can use the MERGE command later. In FORTRAN, modules are referenced by module name, but in Microsoft BASIC modules are referenced by line number. For example:

```
120 GOSUB 1000
```

will execute the subroutine starting at line number 1000. What we need then is a numbering scheme in Microsoft BASIC to allow main programs to call BASIC subroutines, and at the same time avoid any line numbering conflicts which could exist between main programs and subroutines. A typical line numbering scheme may look like this:

<u>Line Numbering Range</u>	<u>Routine Function</u>
1-9999	Main programs
10000-19999	Sort subroutines
20000-29999	I/O subroutines
30000-39999	Time/date subroutines
40000-49999	Miscellaneous routines
50000-59999	Etc.

Let us assume that we have a main program which needs to use some subroutines which exist and which have predefined starting line numbers. Our main program may look like this:

```
10 DIM W(50),R(50)
15 REM -- Get data records from file --
20 GOSUB 21000
25 REM -- Sort records using bubble sort --
30 GOSUB 10000
35 REM -- Calculate elapsed time for each sorted record --
40 GOSUB 32000
```

```
        45 REM -- Print results with standard report generator --
        50 GOSUB 44000
        55 END
```

How do we merge the main program with the subroutines we need? Remember that we do not have a LINK utility available. The following example will show how this is done. In Microsoft BASIC we issue these commands:

```
        LOAD "MAIN"
        MERGE "GETREC"
        MERGE "BUBSRT"
        MERGE "ETIME"
        MERGE "REPORT"
```

Now the main program and subroutines are linked together to form a runnable program. Note that the subroutines must be SAVEd in ASCII format in order for the MERGE to work.

Now, let us summarize how subroutine libraries can be maintained in Microsoft BASIC. First, create subroutine modules and save them with the SAVE command specifying ASCII format. Be sure each subroutine has a unique range of line numbers and a predefined starting line number so that each main program can access it with the same number. Next, set up a library of your subroutines. Finally, link the subroutines into a program as needed with the MERGE command. The advantages of this approach are: 1) modularity of program design, 2) re-usable common code, and 3) in some cases it saves disk space.

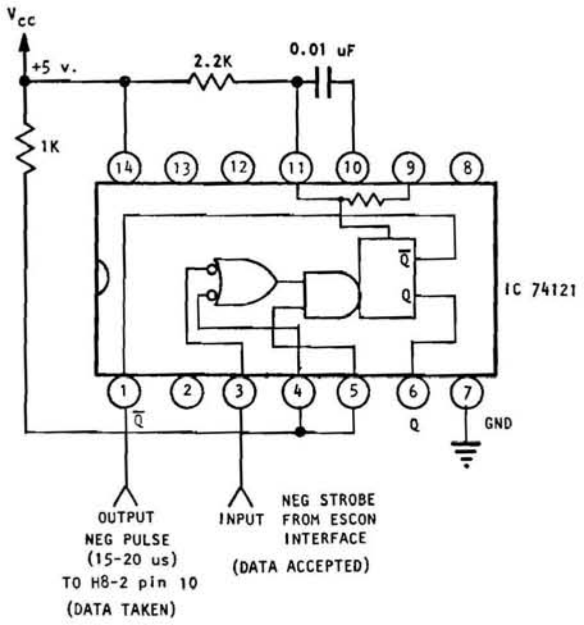
EOF

Editors Note: One more thing should be mentioned about this approach. You will need, in addition to pre-defined line numbers, a pre-defined set of variables for passing parameters between subroutines. It would be a good idea to keep a .DOC file of the variables handy. Also, once you have MERGEd a program, you should SAVE the whole thing if you intend to use it again.

PS:

H8-2 Interface Correction

The diagram below should have appeared with the article "H8-2 Interface" in REMark issue #11.



PULSE STRETCHER

Biorhythm Correction

Heath Biorhythm #40.00.00 will produce an incorrect day of the week in January or February of a leap year. To correct this, add the following lines to the program.

```
00375 X=Y: X1=M: GOSUB 1000: T=T-X

00385 X=Y4:X1=M4:GOSUB 1000:T1=T1-X

00395 X=Y1:X1=M3:GOSUB 1000:T2=T2-X

01000 IF X/400=INT(X/400) THEN 1030
01010 IF X/100=INT(X/100) THEN 1050
01020 IF X/4 <>INT(X/4)   THEN 1050
01030 IF X1>2 THEN 1050
01035 X=1
01040 RETURN
01050 X=0
01060 RETURN
```

Using AT: in MBASIC

by William W. Moss, M.D.
1057 Riverview Lane
Bradenton, Florida 33529

Here is a program written in MBASIC which allows you to directly access an alternate terminal for both input and output via either an H8-4 or an H89 serial I/O board. The routines given provide for either single character or line I/O.

The data to be output or recieved is transferred via the variable X$. The variable ATP% is the port address for the alternate terminal, which is set for 320 (octal). It can be revalued to any user selected value by changing line 1040. The variables X%, X1$, and X2$ are used for internal variables in the routines and thus should not be used by the calling program.

The UART initialization routine (lines 1040-1120) sets the baud rate to 300. If another baud rate is desired, refer to the operation manual for the serial I/O board used to determine the proper values for the UART divisor latch.

The program as submitted demonstrates "line input" via the alternate terminal.

```
100 'DEMONSTRATION OF ALTERNATE TERMINAL I/O IN MBASIC
110 ' by William W. Moss - 7/17/80
120 '
130 CLEAR 1000'          Declare string space
140 GOSUB 1040'          Initialize the Uart
150 GOSUB 6040'          Print CR/LF
160 X$="INPUT A LINE"
170 GOSUB 5040'          Print prompt on AT:
180 GOSUB 4040'          Input line from AT:
190 A$=X$'               Save the string input in A$
200 GOSUB 6040'          Print CR/LF
210 X$=A$: GOSUB 5040' Print the string on AT:
1000 '
1010 'Initialization routine for uart
1030 '
1040 ATP%=&O320'         Base port address for alternate terminal
1050 OUT ATP%+4,&O20'  Set loopback
1060 OUT ATP%+3,&O200' Divisor lock access
1070 OUT ATP%,&O200'   Set divisor LSB (300 Baud)
1080 OUT ATP%+1,1'      Set divisor MSB (See manual for other Baud rates)
1090 OUT ATP%+3,3'      8-bit words
1100 X%=INP(ATP%)'      Clear garbage from reciever
1110 OUT ATP%+4,0'      Turn off loopback
1120 RETURN
2000 '
2010 '
2020 'Single character input routine (with echo)
2030 '
2040 IF (INP(ATP%+5) AND 1) = 0 THEN 2040'       Wait for data entry
2050 X%=INP(ATP%) AND &O177'                     Get byte and clear parity bit
2060 OUT ATP%,X%'                                Echo it back at AT:
2070 X$=CHR$(X%)'                                Convert to character
2080 RETURN
3000 '
3010 '
3020 'Routine to output a character to AT:
3030 '
3040 IF (INP(ATP%+5) AND 32) = 0 THEN 3040'      Wait for clear transmitter
3050 IF LEN(X$)<>0 THEN OUT ATP%,ASC(X$)'        Transmit character
3060 RETURN
4000 '
4010 '
```

```
4020 'Input a line from AT:
4030 '
4040 X1$=""
4050 X$="? "
4060 GOSUB 5040
4070 GOSUB 2040
4080 IF X%<>13 THEN X1$=X1$+X$: GOTO 4070'      Get characters until CR
4090 X$=X1$
4100 RETURN
5000 '
5010 '
5020 'Print a line on AT:
5030 '
5040 X2$=X$
5050 FOR X%=1 TO LEN(X2$)
5060 X$=MID$(X2$,X%,1)
5070 GOSUB 3040
5080 NEXT X%
5090 RETURN
6000 '
6010 '
6020 'Output a CR-LF to AT:
6030 '
6040 X$=CHR$(13)+CHR$(10)
6050 GOSUB 5040
6060 RETURN
```

EOF

Set HDOS Stand-Alone

Are you running out of space on your disk? Do you wish you could get rid of the system files and just run the stuff you need? Well, you're in luck. There is an undocumented (and probably unsupported) feature of HDOS that allows you to do just that. To take advantage of it, initialize a disk and copy the following files to it: SYSCMD.SYS, PIP.ABS, and ERRORMSG.SYS (optional).

Then on any bootable disk that includes the SET program, give the command:

```
SET HDOS STAND-ALONE
```

HDOS will modify itself and notify you that the feature is possibly dangerous. If you're nervous you can undo the damage by saying:

```
SET HDOS NOSTAND-ALONE
```

If you want to proceed, give the command:

```
RESET SY0:
```

and you will be asked to replace the disk in SY0:. Put in the disk you prepared with SYSCMD.SYS on it, and you're running without the system on your disk, giving you much more space to play with.

Stand-alone mode loads the HDOS overlays into your upper memory, so you will need to switch back to a bootable disk after finishing a very large program that wipes out the HDOS overlay area.

Another benefit of running stand-alone is that you can plug along for hours, happily exchanging disks in SY0:, without ever needing to "type SPACES to determine baud rate".

EOF

Cut Down Drive Run Time

When you do not access the drives in your H17 or H89 for a while, the motors shut down after about 30 seconds of disuse. The following program will allow you to shorten the timeout period to about 5 seconds.

```
USERFWA EQU     042200A USER PGM START
TIMCON  EQU     040110A TIME CONTROL BYTE
.EXIT   EQU     0       EXIT TO HDOS
        ORG     USERFWA
START   MVI     A,10Q   GET TIME VALUE
        STA     TIMCON   SET IT
        XRA     A       CLEAR A
        SCALL   .EXIT   EXIT TO HDOS
        END     START
```

Apparantly the value in 040.110 determines the drive timeout period. This value is normally 74Q. You could include this routine in your PROLOGUE.SYS to change the time automatically, or make it a separate file to run when you want.

Thanks to Bill Rosen
c/o the Metro-Detroit Area
Heathkit Computer Users' Group

FORTRAN Corner

by James G. Jerling
Heath Software Developer

EDITOR'S NOTE: The following is the first installment in a series on the FORTRAN language. We hope you will find it helpful in learning this powerful software tool.

FORTRAN is one of the more versatile languages available to Heath computer users. Even though it does not have as many commands as an extended BASIC, it is considered more powerful in many respects. In this column, we will learn how to use the power of FORTRAN.

Fortran stands for FORmula TRANslation. It was designed primarily as a scientific number cruncher, but it is useful for business applications as well. It does not handle strings as easily as with BASIC, but it can be done. In later articles in this series, we will explore some business and scientific applications of FORTRAN, along with techniques such as string handling.

For most of you, FORTRAN will be easier to understand if compared to BASIC. One of the main differences between them is in the way they execute. BASIC is usually an interpreter. When you want to run a BASIC program, you must first load in the interpreter, then the program you have written. The BASIC interpreter takes each command word in your program (such as PRINT), looks it up in a table, finds the address of a routine that can perform the desired job, and executes that routine. BASIC must repeat this process for each command word in your program, and, in addition, it has to interpret each number from ASCII (text) to binary as it comes to it. Most BASIC's speed the process up a little by pre-translating command words to single-byte tokens, but it is still a very slow process when compared to machine language.

Let's look at an analogy. Suppose you decided to buy and consume food the way a BASIC interpreter runs a program. First you would drive to the store and by one item. The items are already arranged on a shelf in the order in which you need them. When you get the first one, you drive home, unwrap the food, prepare it, and eat it. Then you drive back to the store for the next item. You repeat this process until you reach the end of the shelf. Now here is how machine language compares to eating. The food is already bought, prepared, and on the table. All you have to do is eat it. But who did the buying and preparing? That is where FORTRAN comes in.

FORTRAN is a compiler. Here is how it works. First, you must write your program with an editor. When it is finished, you load in the FORTRAN compiler and instruct it to compile your program. It converts your program, which is called the source, into machine code or into assembler mnemonics that you can later assemble into machine code. The compiler does not execute your program directly as does the BASIC interpreter. When you want to run your program, you load in the machine code the compiler (and assembler) made, and run that. You don't even need the compiler at this point. It has prepared everything for your computer, so all it has to do is "eat" the program. Your program takes less memory with no interpreter tagging along, and it runs faster.

You may think that this sounds too good to be true. If compilers are so great, why are there interpreters? One reason is that every time you want to change your program (for example, to correct the ever-present bug), you have to edit the source and re-compile it. It is much easier to make changes in a BASIC program. Another reason is that compilers are usually large and eat memory. You can run Extended Benton Harbor BASIC in a 16k computer, but you need 40k to run the version of FORTRAN that Heath sells. A compiler usually requires a disk system to move files and parts of itself in and out of memory. An interpreter can use cassettes or paper tape for mass storage.

There are different kinds of compilers. Not all of them produce binary (machine code) or assembler mnemonics. Some of them produce what is called P-code, which is the machine code of an imaginary universal computer. It must be interpreted by a P-code interpreter when the program is run. It is easier and faster to interpret this P-code than BASIC, but it is still slower than machine code. The FORTRAN that Heath sells produces relocatable binary code, which is machine code tagged with special instructions. These instructions tell a program called the Linking Loader how to move the code around in memory, so you can put it where you want it, or combine it with other machine code programs. Little programs combined this way are called modules, and when you make up a program with modules it is called modular programming. Most compilers come with a set of modules already made that do certain common jobs. These sets are called libraries.

In addition to producing compact programs that run fast, another advantage to

(vectored to page 31)

Tiny BASIC Tricks

This is the first of a new column in which we will present programming tricks for ETA-3400 users. I will be presenting the tricks that I know, and I hope you will be inspired enough by them to send in your own tricks.

L-O-N-G DIVISION

As you know, tiny BASIC can only work with integer numbers. This is fine when you add, subtract, or multiply, because when you put integers into these operations, you get integers out. But division is a different story. If you divide 10 by 3, you get 3.33333... (decimals ad infinitum), but tiny BASIC only gives you 3. And if you do 2/8, the answer should be .25, but you get 0 with tiny BASIC.

The solution to this problem is to teach tiny BASIC to do long division the same way you were taught in school. Let's take the second example above and work it out. On your paper, you write

```
8)2
```

In long division, you divide one number at a time until you finish the problem. You know that 8 will not go into 2, so you write a zero above the 2. Then you calculate the remainder by multiplying your partial quotient (zero) by the divisor (8) and subtracting the result from the dividend (2). Then you "bring down" the next number if there is one, or else put a zero after the remainder and divide again. You repeat the process until you get an even division with no remainder. When you are finished, it looks like this:

```
  0.25
8)2
  0
  20
  16
   40
   40
    0
```

Of course, we did not cover all the rules, such as placement of the decimal, but you get the idea. To implement long division in tiny BASIC, we need a way to get the remainder after each step in the division. If we have the expression Q=X/Y, we can find the remainder by saying X-(Q*Y), because the quotient Q will always be truncated to an integer in tiny BASIC. There will never be a number to "bring down" since we started with integers, so we can add a zero to the the remainder by multiplying by 10. Now, we can do the next step of the division by dividing Q into the remainder with a zero added. If we use the same variable X to hold the new dividend, it becomes easy to repeat the process for as many decimal places as we want to. After the initial division Q=X/Y, we say X=(X-Q*Y)*10, then Q=X/Y again, and just keep repeating the last two steps for as places as we need. Below is a program using this method for division. Note that we have added a counter to limit the number of decimal places to which we carry the division. If we didn't do that, division would continue for ever even if it came out even (you would see a lot of zeros in that case).

```
10 PRINT "L-O-N-G DIVISION"
20 PRINT "ENTER THE NUMBER TO BE DIVIDED";
30 INPUT X
40 PRINT "ENTER THE NUMBER TO DIVIDE BY";
50 INPUT Y
60 PRINT "CARRIED TO HOW MANY PLACES";
70 INPUT N
80 REM    GET INTEGER PORTION
90 Q=X/Y
100 PRINT "THE ANSWER IS ";Q;".";
110 REM    CALCULATE DECIMAL PORTION
120 X=(X-Q*Y)*10
130 Q=X/Y
140 PRINT Q;
150 REM    PRINT N DECIMALS
160 N=N-1
170 IF N>0 GOTO 120
180 PRINT
190 PRINT
200 GOTO 20
210 END
```

Here is a sample run of the program. It illustrates a way to get the value of pi accurate to 7 digits (6 decimal places). You might want to experiment with long chains of non-repeating decimals. Dividing numbers that are close together will often produce such chains, for example 437 divided by 436.

```
:RUN
L-O-N-G DIVISION
ENTER THE NUMBER TO BE DIVIDED? 355
ENTER THE NUMBER TO DIVIDE BY? 113
CARRIED TO HOW MANY PLACES? 6
THE ANSWER IS 3.141592

ENTER THE NUMBER TO BE DIVIDED?
```

If you try very long divisions, you will have to set your terminal for automatic line carry-over. If you can't do that, put a second counter in the program to issue a PRINT statement (with no following semicolon) after a number of iterations. Now for those of you with floating point BASIC who are reading this, here is a test question: Why won't this program work in your BASIC?

PS:

HUG Parts List

as of 01-Nov-80

Part Number	Description	Selling Price
885-0017	H8 Poster	$ 2.95
885-0018	H89 Poster	$ 2.95
885-0019	Color Graphics Poster	$ 2.95
885-1008	Volume I Documentation	$ 9.00
885-1009	Tape I Cassette	$ 7.00
885-1010	Adventure Disk H8/H89	$ 10.00
885-1012	Tape II BASIC Cassette	$ 9.00
885-1013	Volume II Documentation	$ 12.00
885-1014	Tape II ASM Cassette H8 Only	$ 9.00
885-1015	Volume III Documentation	$ 12.00
885-1019	Device Driver Disk H8/H89	$ 10.00
885-1022	HUG Editor (ED) Disk H8/H89	$ 15.00
885-1023	RTTY Disk H8 Only	$ 22.00
885-1024	Disk I H8/H89	$ 18.00
885-1025	Runoff Disk H8/H89	$ 35.00
885-1026	Tape III Cassette	$ 9.00
885-1027	Morse8 Cassette H8 Only	$ 14.00
885-1028	RTTY Cassette H8 Only	$ 11.00
885-1029	Disk II Games 1 H8/H89	$ 18.00
885-1030	Disk III Games 2 H8/H89	$ 18.00
885-1031	Disk IV MUSIC H8 Only	$ 23.00
885-1032	Disk V H8/H89	$ 18.00
885-1033	HT-11 Disk I	$ 19.00
885-1034	Character Ed Cassette H8 Only	$ 11.00
885-1035	ED/ASM/DEBUG Cassette H8 Only	$ 11.00
885-1036	Tape IV Cassette	$ 9.00
885-1037	Volume IV Documentation	$ 12.00
885-1038	Wise on Disk H8/H89	$ 18.00
885-1039	WISE on Cassette H8 Only	$ 9.00
885-1040	PILOT On Cassette H8 Only	$ 11.00
885-1042	PILOT on Disk H8/H89	$ 19.00
885-1043	MODEM Heath to Heath H8/H89	$ 21.00
885-1044	Disk VI H8/H89	$ 18.00
885-1045	FOCAL Cassette H8 Only	$ 11.00
885-1047	Stocks H8/H89 Disk	$ 18.00
885-1048	Personal Account H8/H89 Disk	$ 18.00
885-1049	Income Tax Records H8/H89 Disk	$ 18.00
885-1050	M.C.S. Modem for H8/H89	$ 18.00
885-1051	Payroll H8/H89 Disk	$ 50.00
885-1052	Morse8 Disk H8 Only	$ 18.00
885-1054 *	SmBusPkg II 3 Disk H8/H19/H89	$ 60.00
885-1055	MBASIC Inventory Disk H8/H89	$ 30.00
885-1056	MBASIC Mail List H8/H89 Disk	$ 30.00
885-1057	Tape V Cassette	$ 9.00
885-1058	Volume V Documentation	$ 12.00
885-1060	Disk VII H8/H89	$ 18.00
885-1061	TMI Load H8 ONLY Disk	$ 18.00
885-1062	Disk VIII H8/H89 (2 Disks)	$ 25.00
885-1063	Floating Point Disk H8/H89	$ 18.00
885-1064	Disk IX H8/H89 Disk	$ 18.00
885-1065	Fix Point Package H8/H89 Disk	$ 18.00
885-1066	Disk X H8/H89	$ 18.00
885-1067 -	Disk XI H8/H19/H89 Games	$ 18.00
885-1068 .	Disk XII MBASIC Graphic Games	$ 18.00
885-1069	Disk XIII Misc H8/H89	$ 18.00
885-1070	Disk XIV Home Fin H8/H89	$ 18.00
885-1075	HDOS Support Package H8/H89	$ 60.00
885-1077	TXTCON/BASCON H8/H89 Disk	$ 18.00
885-1078	HDOS Z80 Assembler	$ 25.00
885-1079	HDOS Page Editor	$ 25.00
885-1080	EDITX H8/H19/H89 Disk	$ 20.00
885-1082	Programs for Printers H8/H89	$ 20.00
885-1083	Disk XVI Misc H8/H89	$ 20.00
885-1085	PILOT Documentation Manual	$ 9.00
885-1201	CP/M (TM) Volumes H1 & H2	$ 21.00
885-1202	CP/M Volumes 4 & 21 C	$ 21.00
885-1203	CP/M Volumes 21 A & B	$ 21.00
885-1204	CP/M Volumes 26/27 A & B	$ 21.00
885-1205	CP/M Volumes 26/27 C & D	$ 21.00
885-1206	CP/M Games Disk	$ 21.00

CP/M a registered trademark of Digital Research
* Supplied on 3 disks, requires only 2 drives.

New HUG Products

HUG P/N 885-1067 Games Disk
H8/H19//H89/48k Disk Software $18.00

PINBALL is a graphic representation of a pinball game. The "ball" actually moves and your H19/H89 speaker beeps each time it hits a bumper. You "shoot" the ball by entering a number (a higher number = a harder shot), then work the flippers by hitting any key. This machine language program was written by an experienced pinball player.

REVERSI is another machine language game with graphics. This game is similar to Othello.

DIET is a BASIC program that can help you plan your weight loss program. It was previously released on cassette.

OTHELLO is BASIC language version of the game with excellent graphics.

BIORYTHM is a disk version of Heath's cassette Biorhythm program. It includes an option to have the chart printed on your LP:.

VEGAS is a blackjack game in BASIC with graphics.

GRWUMPUS is a graphics Wumpus game in BASIC. It is a little easier than some because it shows you where you are and where you have been.

All BASIC programs are in B H BASIC.

885-1085 PILOT Documentation Disk $9.00

This manual is intended for use with the two versions of PILOT offered by the Heath Users' Group. Information on how to use the commands and examples of each command are included for the first time user. A couple of sample programs are also included at the end of the manual.

MORE TO COME....

The software explosion has caught on here at HUG as well as with Heath and Softstuff. In the near future you can expect to see more games, utilities, instructional programs, and others. We really appreciate the response from members. You have helped HUG to become, I feel, the best microcomputer users' group.

PS:

Type My Name Ten Times

I have heard from people here and there that the program "Type my name ten times" in section 3 of the Heath Assembly Language Programming Course does not work. This article will explain why it doesn't work, and present a working version.

First, you should know that the course was written for 8080 computers in general, not a particular machine. Whenever you work with ports, as this program does, you work with hardware that is separate from the processor (8080). This hardware has to be initialized by sending it some data, which will be different on different computers. So the course assumes that your port is already initialized. We should also note at this point that different computers don't always use the same port addresses. The ones in the program (0FAH and 0FBH) just happen to be the ones normally used in an H8 computer with the H8-5 interface card. If you use the H8-4 card, or have an H88/H89, your ports will be different.

The first thing we have to do, then, is initialize the ports. In the "working" version of the program, I have added a routine to initialize the ports on an H8-5 card, with routines for other situations at the end of the program. I have used hex numbers in the main program to conform with the course. To point out clearly my changes to the program, I have placed a dollar sign ($) in the left margin to denote a line added, and a percent sign (%) to denote a line changed.

The next thing we need to do is reserve more stack space. In Heath computers, the system interrupts the processor every 2 ms so the monitor can do its thing, and that uses up stack space. So if we change the DS 014H in the original to a DS 05FH, that should give us plenty of stack space. Without this extra stack space, the program would crash when the stack ate it up. Actually, since the monitors in H8's and H88/89's put the stack at the top of RAM when the computer is turned on or reset, we could leave out the part of the program that assigns the stack (and the reserved space) and use it where it is.

If we ran the program at this point it would sort of work, but it would write each name over the previous one, so we need to make it add a line feed after each carriage return. I added the code to do that at the label "PRINT". It is not done the best way, but I would have had to use opcodes not yet introduced in the course to make it better. I also added code to print a line feed at the end of the program.

I made two more changes to the program. One was to assign the port addresses to labels with the EQU pseudo, instead of using the numbers directly in the program. That makes it easier to make changes. The other was to put a NOP instruction above and below the HLT instruction, to make sure the computer's monitor doesn't see it until we want it to.

If you would like to run this program under HDOS, the easiest way to do it would be to use the procedure described in my article "Do Your Own Thing" in REMark Issue #11. Remove the port intiialization code and replace it with the code to kill console interrupts. Then replace the NOP, HLT, NOP sequence with the exit routine from that article.

```
$       *       "TYPE MY NAME TEN TIMES"
                ORG     2280H           *START ADDRESS OF OBJECT CODE
$       *                               *THIS ADDRESS IS GOOD FOR HDOS
$       *                               *AS WELL AS CASSETTE USERS
$       SPORT   EQU     0FBH            *CONSOLE STATUS PORT
$       DPORT   EQU     0FAH            *CONSOLE DATA PORT
$       DRB     EQU     002H            *DATA READY BIT
$       CRB     EQU     001H            *CRT READY BIT
        BEGIN   LXI     SP,STACK        *DEFINE TOP OF STACK
$       *       INITIALIZE PORTS
$               MVI     A,081H          *REMOVE PREVIOUS INITIALIZATION
$               OUT     0FBH
$               MVI     A,040H
$               OUT     0FBH
$               MVI     A,04EH          *SET UP WORD LENGTH, ETC.
$               OUT     0FBH
$               MVI     A,015H              *ENABLE TRANSMIT AND RECIEVE
$               OUT     0FBH
                LXI     H,MESAG             *SET H-L TO MESSAGE
        LOOP1   CALL    WAIT                *OUTPUT A CHARACTER
                ANI     0FFH                *CHECK IF CHARACTER IS 00 HEX
                JNZ     LOOP1               *DO NEXT CHARACTER IF NOT
                LXI     H,BUFFR             *SET H-L ADDRESS TO STORE NAME
%       STAT    IN      SPORT               *INPUT DATA READY BIT
%               ANI     DRB                 *ISOLATE READY BIT
                JZ      STAT                *LOOP UNTIL KEY STROKE
%               IN      DPORT               *INPUT CHARACTER FROM KEYBOARD
                ANI     07FH                *STRIP PARITY
                MOV     M,A                 *SAVE CHARACTER IN MEMORY
                CALL    WAIT                *OUTPUT THE CHARACTER
                CPI     00DH                *IS CHARACTER 'CR'?
                JNZ     STAT                *GET NEXT CHARACTER IF NOT 'CR'
                MVI     B,00AH              *ESTABLISH A COUNT OF TEN
        PRINT   LXI     H,BUFFR             *SET H-L TO BEGINNING OF NAME
$       OUTLF   IN      SPORT               *INPUT DATA IN STATUS PORT
$               ANI     CRB                 *ISOLATE CRT READY BIT
$               JZ      OUTLF               *LOOP UNTIL CRT READY
$               MVI     A,00AH              *PUT LINE-FEED IN A
$               OUT     DPORT               *TRANSMIT IT
        LOOP2   CALL    WAIT                *OUTPUT A CHARACTER
                CPI     00DH                *WAS LAST CHARACTER A 'CR'?
                JNZ     LOOP2               *DO NEXT CHARACTER IF NOT
                DCR     B                   *COUNT ONE NAME
                JNZ     PRINT               *DO ANOTHER IF B NOT ZERO
$               LXI     H,LF                *POINT TO LINE-FEED
$               CALL    WAIT                *PRINT IT
$               NOP                         *ISOLATE 'HLT'
                HLT                         *STOP HERE
$               NOP                         *ISOLATE 'HLT'
%       WAIT    IN      SPORT               *INPUT DATA IN STATUS PORT
%               ANI     CRB                 *ISOLATE CRT READY BIT
                JZ      WAIT                *LOOP UNTIL CRT READY
                MOV     A,M                 *GET CHARACTER FROM MEMORY
%               OUT     DPORT               *OUTPUT CHARACTER TO CRT
                INX     H                   *SET H-L TO NEXT MEMORY ADDRESS
                RET                         *RETURN TO MAIN PROGRAM
        MESAG   DB      'HI, WHATS YOUR NAME?  ',000H
$       LF      DB      00AH                *LINE FEED CHARACTER
%               DS      05FH                *RESERVE SPACE FOR STACK
        STACK   DS      001H                *TOP OF STACK IS HERE
        BUFFR   EQU     *                   *STORAGE AREA STARTS HERE
                END     BEGIN               *END OF PROGRAM

        *       IF YOU USE AN H8-4 CARD FOR YOUR CONSOLE INTERFACE
        *       OR HAVE AN H88/H89, REPLACE THE DEFINITIONS AND
        *       PORT INITIALIZATION WITH THE FOLLOWING

        SPORT   EQU     355Q                *CONSOLE STATUS PORT
        DPORT   EQU     350Q                *CONSOLE DATA PORT
        DRB     EQU     1                   *DATA READY BIT
        CRB     EQU     40Q                 *CRT READY BIT
        *       INITIALIZE PORTS
                MVI     A,200Q              *ENABLE DIVISOR LATCH
                OUT     353Q
                MVI     A,LS                *GET BAUD RATE LOW
                OUT     350Q
                MVI     A,MS                *GET BAUD RATE HIGH
                OUT     351Q
        *       NOTE:  YOU WILL HAVE TO GET THE VALUES FOR 'LS' AND
        *       'MS' FROM YOUR H88/H89 OPERATION MANUAL OR YOUR H8-4
        *       OPERATION MANUAL.  SEE THE 'INS8250' SECTION
                MVI     A,3                 *SET WORD LENGTH
                OUT     353Q
                MVI     A,0                 *SET NO INTERRUPTS
                OUT     351Q
```

```
*       AFTER YOU PROGRAM THE 8250 (THAT'S WHAT YOU JUST DID),
*       YOU HAVE TO GIVE IT TIME TO DIGEST THE INFORMATION,
*       SO WE PUT IN A DELAY

        LXI     H,65000A        *GET DELAY VALUE
DELAY   DCR     L               *THIS TAKES ABOUT
        JNZ     DELAY           *100 MS
        DCR     H
        JNZ     DELAY

*       AT THIS POINT, THE PROGRAM CONTINUES WITH
*       LXI     H,MESAG
```

PS:

File Catalogs in MBASIC

Here is a way you can print a directory of your disk files while running MBASIC. First, you make ASCII directory files with pip, as follows:

```
>PIP
:P:  <dev:>DIRECT.L= <dev:>/L     (list non-system files)
:P:  <dev:>DIRECT.B= <dev:>/L     (list non-system files in brief format)
:P:  <dev:>DIRECT.BS= <dev:>/B/S  (list all files in brief format)
:P: cntrl-D
```

You will have to repeat the above each time you add or delete files from your disk. To list these files on your console while in MBASIC, use a routine like this one:

```
100 PRINT CHR$(27);CHR$(69)
120 PRINT "DISK DIRECTORY"
130 INPUT "Enter device to list (SY0:, etc.)";A$
140 PRINT "Enter type of directory as follows:"
150 PRINT " L   (for listing of non-system files)"
160 PRINT " B   (for brief listing of non-system files)"
170 PRINT " BS  (for brief listing of all files)"
180 INPUT F$
190 OPEN "I",1,A$+"DIRECT."+F$
200 IF EOF(1) GOTO 240
210 INPUT #1,A$
220 PRINT A$
230 GOTO 200
240 CLOSE
```

Don't forget to update your DIRECT files as necessary. With the RESET command, you can get catalogues of your entire disk library while in MBASIC. No need to return to HDOS to find that game you wanted to show to your kid brother.

Thanks to   Ben Heady
            27 Belhurst Lane
            Willingboro NJ 08046

Voice for the H8

M. I. - 8 (Micro interface-8) of Ft. Collins, Colo. has announced the V-8 voice synthesizer for the H8. It is available in kit form and comes with a cable to connect it directly to the H8-2 parallel interface card. M. I. - 8 says it also can be used on an H89 with a serial to parallel converter. It is described as a true voice systhesizer that can be programmed to speak in any language. The voice clarity is not as good as with "canned" synthesizers such as the TI toys, but you are not limited to a firmware vocabularity. Software available includes a BASIC USR program that allows you to add voice to BASIC programs with PRINT statements. For more details or ordering information write to M. I. - 8, 822 E. County Rd. 30, Ft. Collins, CO 80525.

Build Your Own EPROM Programmer

To most people the only advantage of having software in ROM (Read Only Memory) is that it is there as soon as the computer is turned on. With a disk system you can get the software up almost as fast, but there are still some advantages to being able to make your own ROM software. For example, you could make a debug program that could not be crashed by a run-away machine language program. You could make a new character ROM for the H19 and add those missing opposite corner graphics. You could even make changes to the H17 (HDOS) ROM, but I would not recomend it.

An EPROM is a ROM that can be programmed easily and erased if you want to change it. Fig. 1 is a simple circuit that will program 5-volt only 2716 EPROMS. It is built on an H8-1 8k RAM card, and uses the address decoding and buffer chips on that board, which makes it possible to build it with only three additional chips. It could probably also be built on a 16k ram card.

The 5-volt only family of EPROMS, including the 2716, are much easier to program than the earlier types were. All it takes is a 50ms pulse on the PGM pin while the Vpp pin is held at 25 volts to program a byte in the device. The 2716 holds 2k bytes, and you can program any portion of them in any order. If you could somehow lengthen the Memory Write pusle in your computer to 50ms, the only software you would need to program your eprom is a simple block move to transfer the data from some temporary storage buffer to the address of your EPROM. My circuit does the Write pulse lengthening for you by putting the processor in a WAIT state each time the EPROM is written to. Here is how it does it.

The H8 has a line on the bus called RDYIN that will put the processor in a wait state if it is brought low. But to put the processor into a wait state during a given cycle, such as the Memory Write cycle, you have to bring RDYIN low at the beginning of that cycle, so you can't use the Write pulse itself. On my programmer, I used the Chip Enable signal that normally enables a 4k block on the RAM card to start a 50ms timer (the upper 74121) which is connected to the RDYIN line. The Chip Enable signal is decoded from the address lines, which are active at the beginning of a memory cycle, so the RDYIN line is brought low in plenty of time. Unfortuantely, this means that both Read and Write memory cycles will be delayed, so I had to AND the 50ms pulse with the Memory Write pulse from the bus before I applied it to the EPROM. I also added a short delay (the lower 74121) to ensure that the address and data lines are steady before programming starts. I put a diode (1N914 or equivalent) in series with the output to the RDYIN line to allow other cards on the bus to use it, if necessary.

Switches no. 1 and no. 2 control the power to the EPROM. No. 3 disconnects the timer from the RDYIN line, and when it is off, the RAM board will function normally. When you are programming, you should turn the switches on in numerical order (1, 2, 3), and off in decending order when you are done (3, 2, 1).

When I built my programmer, I used "rework" sockets, which are wire wrap sockets with their pins sticking up, to hold the IC's, but you could simply glue the IC's upside down to the board. However you do it, you will need some kind of socket for the 2716's to be programmed. I ran a flat cable from the board to the side of my H8, where I mounted a zero insertion force socket.

The 2716 is designed so that data can be read from it while the Vpp is still at 25 volts, so a programmer can be built to do a read after each write to verify the programming. My simple design does not allow that, so you must verify the EPROM by putting it in a ROM board and executing a compare routine. In the next issue, I will present a design of a ROM board that holds 8 2716's for a total of 16k of data.

As I stated before, a simple block move routine is all that you should need to run my programmer. In actual practice, however, the 8080 runs so fast that the capacitor on the 50ms timer does not have time to discharge, so you will need a little software delay after each byte is moved. Below is an example of a block move routine that will work.

```
DADDR   EQU     (address of data)
EADDR   EQU     (end of data)
PADDR   EQU     (address of programmer)

        LXI     H,DADDR Get data address
        LXI     D,EADDR  and end address
        LXI     B,PADDR Programmer address
PGLOOP  MOV     A,M     Get a byte
        STAX    B       Program it in
        PUSH    D       Save DE
        LXI     D,4000A Delay value
TIMER   DCR     E       Wait a while
        JNZ     TIMER
        DCR     D
        JNZ     TIMER
        POP     D        Restore DE
        INX     B        Inc. pointers
        INX     H
        CALL    CPHD     End of block?
        JNC     PGLOOP   Loop until done

* Compare HL with DE
* If HL <= DE then carry = 0
* If HL > DE then carry = 1

CPHD    MOV     A,H      If HL is zero now,
        ORA     L        it was 377377A
        STC              Through the roof!
        RZ               Return with carry
        MOV     A,E      Subtract HL
        SUB     L         from DE
        MOV     A,D      Subtraction
        SBB     H         determines carry
        RET
```

You would, of course, want a bit more elaborate software package that would allow you to input the addresses, load data from cassettes or disks to be programmed, and perform the compare routine for varification. You might also want to be able to read in data from EPROMS so you can copy them or make small patches to the software before "burning" a new EPROM.

A few final considerations: This programmer is for 5-volt only 2716's. It could be modified for other 5-volt EPROMS, but it will not work with the TMS 2716, which uses three voltages. The TMS 5-volt only part is called the 2516. To erase EPROMS, you will need an ultra violet light EPROM eraser. The back pages of magazines such as Kilobaud MICROCOMPUTING and BYTE will have ads for them.

PS:

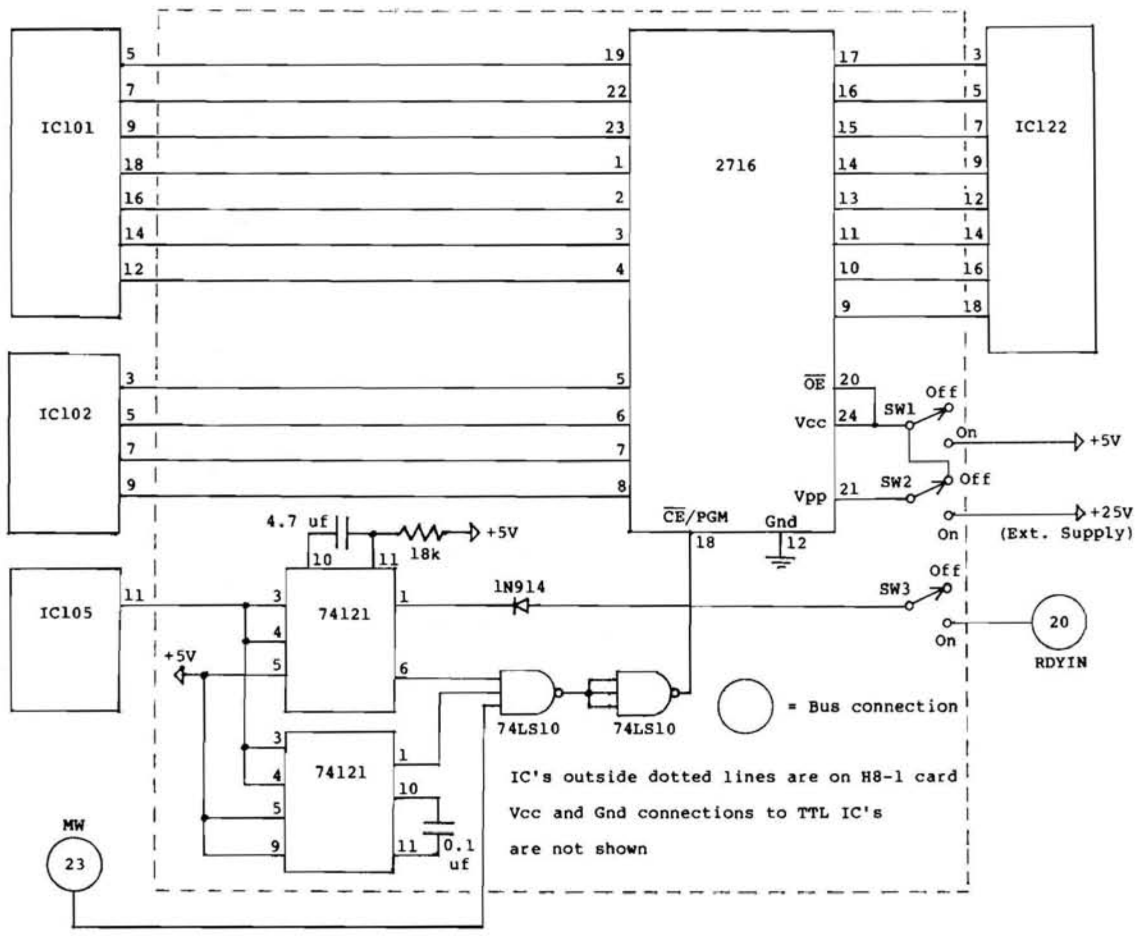


Fig. 1 RAM BOARD EPROM PROGRAMMER

Screen Formatting in BASIC

If you are planning to do extensive screen formatting in BASIC, you may find the two subroutines presented here useful. They make it easy to position the cursor, print titles, and use the 25th line of your H19/H89.

The first of these routines is H89INIT.BAS. It contains a routine at line 65000 that must be called (GOSUB 65000) before any of the other routines can be used. This initializes a function FN C$(x,y) that you can use to place the cursor at line x, column y. The statement

```
PRINT FN C$(5,25);
```

moves the cursor to line 5, column 25. A variable D1$ is also initialized to contain the current system date (entered when you booted up). H89INIT.BAS also contains a routine at line 65300 that prints a title in the variable T1$ centered at the top of the page in reverse video. The cursor is then moved to the first column of the next line. To use the word "TITLE" as a title, you would do the following:

```
T1$ = "TITLE":GOSUB 65300
```

The second routine is MSG25.BAS. Line 64000 starts a routine that will print a message contained in M$ on line 25 and return the cursor to the starting point. The statement

```
M$ = "HELLO, THERE":GOSUB 64000
```

will print "HELLO, THERE" on the 25th line. To clear the 25th line, use GOSUB 64200. As with the previous routines, you must initialize by calling the routine at line 65000 before you use the MSG25.BAS routines.

Thanks to Bob Thomas
2704 19th Ave.
Shawmut, AL 36876

```
65000 REM *************************************************************
65010 REM                              H89INIT.BAS
65020 REM Initialize string variables for easy screen control on the
65030 REM Heath H19, H88, or H89.  Escape sequences conform to those
65040 REM given in REMark Issue #9.
65050 E$=CHR$(27): REM      Escape character
65060 E1$=E$+"E":  REM      Erase page
65070 P$=E$+"p":   REM      Enter reverse video
65080 Q$=E$+"q":   REM      Exit reverse video
65090 Y$=E$+"Y":   REM      Direct cursor addressing
65100 L$=E$+"l":   REM      Erase line
65110 X1$=E$+"x1": REM      Enable 25th line
65120 J$=E$+"j":   REM      Save cursor position
65130 K$=E$+"k":   REM      Set cursor to saved position
65140 REM
65150 REM Put system date in D1$
65160 D1$="": FOR D1=8382 TO 8391: D1$=D1$+CHR$(PEEK(D1)): NEXT D1
65170 REM
65180 REM Define function FN C$(x,y) for cursor positioning
65190 DEF FN C$(C1,C2)=Y$+CHR$(C1+31)+CHR$(C2+31)
65200 RETURN
65300 REM *************************************************************
65310 REM Clear the screen, and center the value in string variable
65320 REM T1$ on line one in reverse video
65330 REM
65340 C1=1: C2=40-INT((4+LEN(T1$))/2)
65350 PRINT E1$; FN C$(C1,C2); P$; "  "; T1$; "  "; Q$
65360 RETURN
65370 REM *************************************************************
```

```
64000 REM ****************************************************************
64010 REM                          MSG25.BAS
64020 REM Issue a message to line 25 using the contents of M$.
64030 PRINT J$;X1$;
64040 PRINT FN C$(25,1);L$;
64050 IF M$="" THEN 64090
64060 PRINT "******  ";
64070 PRINT M$;
64080 PRINT "  ******";
64090 PRINT K$;
64100 M$=""
64110 RETURN
64200 REM ****************************************************************
64210 REM Set the string variable M$ to null and clear line 25.
64220 REM
64230 M$=""
64240 GOSUB 64000
64250 RETURN
64260 REM ****************************************************************
```

EOF

MOUNTALL

The following program will make it easier to mount your disks. It automatically determines how many drives you have, and checks if the drives are occupied and if the disks are INITed. If these conditions are met, the disks are mounted.

```
MOUNTALL mount all available disks by: /AIWZ/            HEATH ASM #104.05.00

   000.177                    DSTAT    EQU     177Q         DISK STATUS PORT
   040.033                    TICCNT   EQU     40033A       TIMER
   040.205                    R.SDP    EQU     40205A       SET DEVICE PARAMETERS
   041.061                    AIO.UNI  EQU     41061A       DRIVE UNIT NUMBER
   042.200                    USERFWA  EQU     42200A
   000.011                    .VERS    EQU     011Q
   000.041                    .CTLC    EQU     41Q
   000.000                    .EXIT    EQU     0
   000.002                    .SCOUT   EQU     2Q
   000.200                    .MOUNT   EQU     200Q
   000.003                    .PRINT   EQU     3Q
   000.060                    .ERROR   EQU     60Q


   042.200                             ORG     USERFWA
   042.200  257               ST       XRA     A
   042.201  377 011                    SCALL   .VERS        GET VERSION NUMBER
   042.203  332 227 042                JC      BYE          EXIT IF NONE
   042.206  076 003                    MVI     A,3           C
   042.210  041 265 042                LXI     H,CCTYP      DO NOTHING
   042.213  377 041                    SCALL   .CTLC        INSTALL VECTOR
   042.215  076 001                    MVI     A,1          TRY SY1: FIRST
   042.217  315 232 042                CALL    FMOUNT
   042.222  076 002                    MVI     A,2          NOW SY2:
   042.224  315 232 042                CALL    FMOUNT
   042.227  257               BYE      XRA     A
   042.230  377 000                    SCALL   .EXIT
```

```
042.232  062 061 041 FMOUNT  STA     AIO.UNI         STORE UNIT NUMBER
042.235  306 060             ADI     '0'             MAKE DIGIT
042.237  062 003 043         STA     DNUM            SAVE DRIVE NAME
042.242  315 205 040         CALL    R.SDP           SETUP DRIVE
042.245  315 266 042         CALL    DTEST           SEE IF THERE
042.250  330                 RC                      NOPE
042.251  076 012             MVI     A,12Q           IS THERE - SKIP LINE
042.253  377 002             SCALL   .SCOUT
042.255  041 001 043         LXI     H,DRIVE         POINT TO NAME
042.260  377 200             SCALL   .MOUNT          MOUNT IT
042.262  334 347 042         CC      ERR             PRINT ERROR
042.265  311         CCTYP   RET                     DONE

042.266  041 033 040 DTEST   LXI     H,TICCNT        POINT TO CLOCK
042.271  106                 MOV     B,M             GET CURRENT TIME
042.272  016 000             MVI     C,0             CLEAR COUNT
042.274  315 342 042 NOHOLE  CALL    CTIME           CHECK TIME
042.277  322 336 042         JNC     TALLY           TIME UP
042.302  333 177     W.HOLE  IN      DSTAT           GET STATUS
042.304  017                 RRC                     TEST HOLE BIT
042.305  332 321 042         JC      GOTHOL          GOT A HOLE
042.310  315 342 042         CALL    CTIME           ONE REVOLUTION?
042.313  332 302 042         JC      W.HOLE          NO, WAIT FOR HOLE
042.316  303 336 042         JMP     TALLY           YES, CHECK TALLY
                     *
042.321  014         GOTHOL  INR     C               BUMP HOLE COUNT
042.322  333 177     W.NOHO  IN      DSTAT           GET STATUS
042.324  017                 RRC                     TEST
042.325  322 274 042         JNC     NOHOLE          NO HOLE NOW

042.330  315 342 042         CALL    CTIME           TIME UP?
042.333  332 322 042         JC      W.NOHO          NO, KEEP WAITING
042.336  171         TALLY   MOV     A,C             TIME'S UP
042.337  376 012             CPI     10              GET 10 HOLES (AT LEAST)
042.341  311                 RET                     RETURN WITH ANSWER

042.342  176         CTIME   MOV     A,M             GET TIME NOW
042.343  220                 SUB     B               MEASURE ELAPSED
042.344  376 144             CPI     100             ONE REVOLUTION?
042.346  311                 RET                     ANSWER IN (C)

042.347  365         ERR     PUSH    PSW             SAVE ERROR CODE
042.350  041 364 042         LXI     H,EMSG          PRINT MESSAGE
042.353  377 003             SCALL   .PRINT
042.355  361                 POP     PSW
042.356  046 012     ERR1    MVI     H,12Q
042.360  377 060             SCALL   .ERROR          NOW HDOS'S MESSAGE
042.362  067                 STC                     LEAVE CARRY SET
042.363  311                 RET

042.364  077 052 052 EMSG    DB      '?** Error on '
043.001  123 131     DRIVE   DB      'SY'
043.003  130 072 212 DNUM    DB      'X:',212Q,0

043.007  000                 END     ST

 00083 Statements Assembled
 35259 Bytes Free
  No Errors Detected
```

EOF

HUG BUG: The article "The Secret HDOS" in REMark Issue #7 (page 12) was authored by David A. Wallace, 146 Westford St., Chelmsford, MA 01824.

LP: Listings with the H11A

With this program, an H11 or H11A user can make paged listings of any ASCII file on his disks, such as BASIC programs or files produced by an editor. It asks the user to input a file name, with the default extension set to .BAS. The program determines how many pages are in the file, and heads each page with the file name and "PAGE x OF y PAGE(S)", where x is the current page number and y is the total number of pages. The first page is also dated at the top. Pages are the standard size (66 lines total per page).

```
10 REM - PRINT.BAS          VERSION 1.3
20 REM - AUTHOR:    DANNY CHERRY, LITTON DATA SYSTEMS
30 REM -            NEW ORLEANS ENGINEERING CENTER
40 REM -            NEW ORLEANS, LA.
50 REM - PROGRAM TO PRINT ON THE LINE PRINTER FROM XBASIC
60 PRINT \PRINT "ENTER INPUT FILE- <DEV:>FILENAME<.EXT>  ";
70 D$=DAT$
80 INPUT A$
90 P=0
100 P=POS(A$,".",1)
110 IF P>0 THEN 130
120 A$=A$&".BAS"
130 OPEN A$ FOR INPUT AS FILE #1
140 C=0
150 INPUT #1:B$
160 IF END #1 THEN 190
170 C=C+1
180 GOTO 150
190 C=C+1
200 RESTORE #1
210 P1=INT(C/60)
220 IF P1*60=C THEN 240
230 P1=P1+1
240 P=1
250 OPEN "LP:" FOR OUTPUT AS FILE #2
260 PRINT #2:TAB(60);D$
270 PRINT #2:A$;TAB(50);"PAGE";P;"OF";P1;"PAGE(S)"
280 PRINT #2:
290 C=4
300 INPUT #1:B$
310 IF END #1 THEN 370
320 PRINT #2:B$
330 C=C+1
340 IF C<64 THEN 300
350 GOSUB 450
360 GOTO 300
370 PRINT #2:B$
380 C=C+1
390 FOR I=C TO 66
400 PRINT #2:
410 NEXT I
420 CLOSE #2
430 CLOSE #1
440 STOP
450 FOR I=C TO 66
460 PRINT #2:
470 NEXT I
480 P=P+1
490 PRINT #2:
500 PRINT #2:A$;TAB(50);"PAGE";P;"OF";P1;"PAGE(S)"
510 PRINT #2:
520 C=4
530 RETURN
540 END
```

EOF

BUGGIN' HUG

Dear HUG:

I am interrested in, and have enjoyed all the articles in REMark. Even the very basic articles lead me to picking up something that I have forgotton or never knew. I don't know about the rest of the Heath Users, but my method of learning how to use a new program or utility is to play with it until I am familiar with it, and then rarely do I return to the manual to pick up the finer points. But I will read REMark cover to cover, and when you have an article on something like this (for example, on SYSGEN) I will pick up those points. I believe this is one of the primary reasons, along with clarifying obscure (to me) parts of programs that people have asked for a "big dummy" column in REMark.

I want to congratulate you on the excellent job you and your associates are doing in REMark, and I hope you keep up the good work. Now if you could just go to monthly publication........!

Sincerely,

Errol E. Devore
3543 Park Avenue West
Mansfield OH 44906

Editors note: None of us here are experts when it comes to publishing a magazine, but I have made a publication schedule and will do my best to see that we stick to it. Increasing publication frequency means more articles, so we encourage you to send in anything you think other HUG members will enjoy. Send the articles on disk or tape, if possible. -- PS:

Dear HUG:

I recently bought the software disk IV (885-1044) and found the PUTCAT and SYSCAT programs to be just what I needed to keep track of all my programs. However, it appears to have been written for a terminal with greater than 80-character width. When I run the SYSCAT program on my H19 or H36 (set for 80), the listings of the disk contents wrap to the next line.

I'm just beginning to get into assembly language programming, but I was able to change the program for more legible output. These changes put the volume number and title on one line, the rest of the information on the next, then it leaves a blank line before the next listing. I hope these changes will be useful to others.

Here is the old code:

```
VOLNUM  DB      '000  '
BOOT    DS      4
        DB      ' '
VOLDATE DS      9
VOLFOOT DS      40
        DB      ' '
VOLLAB  DS      60
        DB      CRLF+200Q
CATFL   DB      'CAT'
```

And here is the new:

```
VOLNUM  DB      '000  ' OLD CODE
VOLLAB  DS      60      CHANGE STARTS HERE
        DB      CRLF
BOOT    DS      4
        DB      ' '
VOLDATE DS      9
VOLFOOT DS      40
        DB      CRLF,CRLF+200Q
CATFL   DB      'CAT'   OLD CODE RESUMES
```

William C Richter
1001-140 Evelyn Terrace East
Sunnyvale CA 94086

Dear Jim:

We discovered this random number generator for the ET-3400:

```
0000 C6 00      Load B with 0
0002 5C         Increment B
0003 BD FD BB   Jump to incode
0006 24 FA      Branch if C clear
0008 17         Transfer B to A
0009 BD FE 20   Display A at H and I
000C 3E         Stop
```

Regards,

Gary Hawthorne
c/o Heathkit Electronic Center
35-07 Broadway
Fairlawn NJ 07410

Dear HUG:

In REMark Magazine, Issue #10, you ran an add under Buggin' HUG on a 64k H8 dynamic RAM board sold by Tryonix Electronics. Also an article was published in Kilobaud MICROCOMPUTING, July and August issues, on this same board. I purchased one of these boards from them some time ago and have had very good service from it except for a bad 12-volt regulator (the problem was the regulator itself, not temperature or current).

When I read the documentation that came with the board, I learned that the mounting bail (used as a heat sink) can be expected to reach tempreatures up to 150 degrees Fahrenheit, so I decided to try to reduce that temperature. The board's timing and refresh circuits use standard 7400 series IC's, and there are pin for pin replacements for some of them, such as 74123's. I looked up the typical current drain on a 74123 and a 74LS123, and found a difference of approximately 30 mils per chip. Since there are six of them on the board, that makes a total of 360 mils. The others that can be replaced add up to about 40 mils more. Since the 74123 chips are one-shot IC's and the 74LS123 is a faster chip, some R-C timing changes are required to get board timing back into its proper perspective. I did not change one of the 7408's (U 50), due to the inductors connected to it for 50ns and 90ns timing delays. I measured the bail tempurature with a Techtronics model 465 scope with a digital time, temperature and voltage readout before and after these IC's were replaced with 32k of RAM chips installed. The temperature before replacement was 53.7 degrees Celcius (128 F.), and after it was 45.8 Celcius (114 F.), a difference of 14 degrees F.

The following is a list of the parts I replaced on my board.

Chips		Resistors	
U 36	74LS00	R8	6.8K
U 38	74LS123	R9	27K
U 42	74LS123	R16	12K
U 44	74LS74	R17	12K
U 45	74LS123	R19	6.8K
U 51	74LS08	R20	12K
U 52	74LS00	R28	22K
U 57	74LS123	R29	8.2K
U 58	74LS74	R30	10K
U 59	74LS123	R31	18K
U 60	74LS02	R32	12K
U 61	74LS123	R33	12K
U 64	74LS02		
U 65	74LS74		
U 66	74LS38		

I did not replace any others since they are line drivers, etc. I am passing this information on to anyone who may have or may in the future buy a Tryonix board and is interrested in keeping temperatures inside the H8 as low as possible.

Thank you,

La Verne R. Mantei
S. 7516 Carnation Rd.
Spokane WA 99204

Dear Hug:

Most users of the Heath ET 3400 and ETA 3400 are storing programs in tiny basic onto cassette at 300 baud. The users manual points out the procedure using the Dump and Load commands.

I was ready to sell my unit and buy a "stringy floppy" when I discovered the following procedure which allows me to keep the unit I dearly love and read/write to the cassette at 2400 baud.

Store to Cassette - 2400 Baud

At the end of writing a program in tiny basic, type:

```
: BYE (CR)
MON> M24 (CR)
0024 XX (CR)
0025 YY (BREAK-KEY)
MON> (CNTRL-T) T1 : 10, XXYY Push record
then (CR)
```

When the program has finished transferring (recording) onto tape at 2400 baud the CRT will respond with:

MON> (Stop the cassette).

To load a program from cassette which has been recorded at 2400 baud (by the above procedure). Type the underlined human commands shown below.

MON> L1 Push play on the cassette, then (CR). If the volume is not set right or a read error occurs, the CRT responds with "error".

When the program has successfully loaded into memory the CRT responds:

MON> G1C03 (CR) (Basic warm start Addr).

: RUN (CR) (Program will execute).

Using this method I have several programs stored and haven't yet filled my first cassette. 4K loads in about 15 seconds. Who needs "Stringy Floppy" with ETA 3400? Not me.

Very truly yours,

Howard Cooper
Electrical Engineer
EIMCO PROCESS MACHINERY DIVISION
669 West Second South
Post Office Box 300
Salt Lake City, Utah 84110

Dear Mr. Blake:

I have found what appeared to be a flaw in the new edition of the HT-11 operating system sent to me. On review, however, this is a communication problem between the terminal and serial I/O board that you might wish to know about.

I modified my H-19 as described by George Roth in a recent edition of REMark to enable variable data transfer rates between the computer and terminal. This has been very helpful. I noticed, however, that a command in PIP would blow up the system. That command:

```
     TT:=fname.ext
```

would cause the file to fly by on the screen followed by a ceaseless bell [CHR$(7)] and a lot of @yyy@UUU@uuu in no particular pattern.

The key to the flaw, though, was the baud rate selected. This only occurred at 19,200 baud [ESC r M] and would not occur at any lower baud rate.

I mention this to you so you will not replace a lot of copies of the new DOS from those of us who are "listening" too fast.

Yours truly,

Douglass H. Mc Neill, M.D.
Poynette Family Practice Center
PO Box 115 -- 330 North Main St.
Poynette, WI 53955

Dear Jim:

About 6 months ago I had a conversation with you regarding the distribution of HUG software to our club members. You said you had no objections to what we did with it after we puchased it.

Due to recent articles published in some of the trade publications concerning the pirating of software, I would like to confirm this verbal approval by you.

Does MACO (Midwest Amateur Computer Organization) have the right to distribute non-licensed software purchased through HUG to its members for club and personal use?

At no time will MACO distribute any software to non club members or use HUG software to realize profits in any way, shape or form.

If you do not agree with the above, could you advise us of your policies concerning HUG software and its distrubtion to club members for personal use.

Midwest Amateur Computer Organization started approximately one year ago. We have 22 members from all fields. Our goal is to exchange information between H8 users and to provide members with information and support on new and existing H8 hardware and software. As a club we find HUG a valuable source of information. Because of HUG products we are able to influence prospective members to purchase Heath products and keep the H8 sales strong. This also provides incentive to H8 owners to upgrade thus further supporting the H8 product line.

Please put a note about our group in the next REMark and keep up the good work.

Sincerely,

Robert E. Bileski
Secretary MACO
(Midwest Amateur Computer Organization)
47 Winrock Road
Aurora, Illinois 60538

Bob... HUG has been successful, in part, because we have been able to distribute low cost software to the membership and make enough profit from that effort to hire more bodies to turn out even more software. I say again, what an individual does with HUG software that is purchased from HUG, is something over which I have no control. I recognize that when a new disk is released, everyone wants a copy as soon as possible regardless of the method of acquistion. I think the responsibilty lies with each group to decide if it is in the best interest of HUG to 'share' software.

Concerning proprietary software, anyone that obtains MBASIC, for example, without paying for it may see that day when his life becomes quite complicated. For instance, if your copy of HDOS suddenly broke, how would you like to have your $10,000 worth of hardware sit idle, while you waited for the postman to deliver your broke disk back to HEATH and then waited some more for someone to verify that fact, then waited some more for some clerk to process an order to ship you a new one. That day is probably not too far away. But it is at least one way for a software house to stay in business. What if it were possible to FTCOPY H89s??? How long do you think HEATH would be in the computer business?

JB:

A Text Pre-processor for RUNOFF

If you use RUNOFF (HUG part no. 885-1025) to do text processing, you may find the program listed below useful. It is a text pre-processor written in MBASIC that automatically inserts the more commonly used RUNOFF commands into your text. It puts in your margins, allows you to set justification or ragged right margins, sets up one or two spaces after periods, and allows you to indent paragraphs. It also puts an underscore before any of the six special characters that RUNOFF uses that occur in your text.

RNO.BAS does not handle the CENTERING, TITLE, LIT, or ELI commands, and you will have to add those later with an editor if you need them, but for many jobs it will eliminate coding by hand. This program was sent to us by Doc Campbell, who thought it would come in handy for processing his long articles. It has.

```
10 REM   RNO.BAS    PARTIALLY CODES A MANUSCRIPT FOR PROCESSING BY RUNOFF
20 REM              (ALSO INSERTS AN UNDERSCORE BEFORE ALL SPECIAL
30 REM               CHARACTERS). LEAVES SOME CODING TO BE DONE with EDITOR.
40 REM          %This is an MBASIC program.%
50 REM
60 CLEAR 300
70 PRINT
80 LINE INPUT "Enter name of input ASCII file (ex. SY1:TEST.DAT).... ";P$
90 PRINT
100 OPEN "I",1,P$
110 OPEN "O",2,"SY1:NEW.DAT"
120 PRINT
130 LINE INPUT "Enter desired left margin (default=4)... ";A$
140 IF LEN(A$)=0 THEN A$="4"
150 PRINT:LINE INPUT "Enter desired right margin (default=45)... ";B$
160 IF LEN(B$)=0 THEN B$="45"
170 PRINT
180 LINE INPUT "Enter J for justify, R for ragged rt margin (default=J)... ";C$
190 IF LEN(C$)=0 OR C$="J" THEN C$=".JUSTIFY;FILL" ELSE C$=".NOJUSTIFY;FILL"
200 PRINT
210 PRINT "Enter 1 for 1 space after '.', 2 for 2 spaces after '.'"
220 LINE INPUT "(default is for 2 spaces)... ";D$:PRINT
230 IF LEN(D$)=0 OR D$="2" THEN D$=".PERIOD" ELSE D$=".NOPERIOD"
240 LINE INPUT "Enter desired # of spaces for ppg. indent (default=0)... ";E$
250 IF LEN(E$)=0 OR E$="0" THEN E$=".SKIP" ELSE E$=".P "+E$
260 PRINT #2,"^^"
270 PRINT #2,".LM "+A$+";RM "+B$:PRINT #2,C$:PRINT #2,D$
280 REM     %  now process input file, line by line  %
290 IF EOF(1) THEN 480
300 LINE INPUT #1,X$
310 REM          %  add .SKIP between paragraphs or paragraph indent  %
320 IF LEN(X$)=0 THEN X$=E$:GOTO 460
330 N=1:REM      %  underscore handler  %
340 A=INSTR(N,X$,CHR$(95)):IF A=0 THEN 350 ELSE GOSUB 490:GOTO 340
350 N=1:REM      %  reverse slash handler  %
360 A=INSTR(N,X$,"\"):IF A=0 THEN 370 ELSE GOSUB 490:GOTO 360
370 N=1:REM      %  ampersand handler  %
380 A=INSTR(N,X$,"&"):IF A=0 THEN 390 ELSE GOSUB 490:GOTO 380
390 N=1:REM      %  number sign handler  %
400 A=INSTR(N,X$,"#"):IF A=0 THEN 410 ELSE GOSUB 490:GOTO 400
410 N=1:REM      %  at sign handler  %
420 A=INSTR(N,X$,CHR$(64)):IF A=0 THEN 430 ELSE GOSUB 490:GOTO 420
430 N=1:REM      %  up arrow handler  %
440 A=INSTR(N,X$,"^"):IF A=0 THEN 450 ELSE GOSUB 490:GOTO 440
450 REM     %  print out coded line to SY1:NEW.DAT file  %
460 PRINT #2,X$:GOTO 290
470 PRINT:PRINT "Partially coded raw RNO file now on disc called SY1:NEW.DAT."
480 PRINT:CLOSE:END
490 X$=MID$(X$,1,A-1)+CHR$(95)+MID$(X$,A,LEN(X$)-A+1):N=A+2:RETURN
```

MBASIC to Machine Code Link

Linking from Microsoft (MBASIC) to a machine code program is possible through the use of the program shown below. This subroutine maybe called at any time in a MBASIC program.

```
2000 CLOSE:DIM U0%(12)
2100 U0%(0)=&H3FE : U0%(1)=&HEBC0 : U0%(2)=&H237E : U0%(3)=&H235E
2300 U0%(4)=&H8356 : U0%(5)=&H7A4F : U0%(6)=&HCE : U0%(7)=&HB47
2500 U0%(8)=&HFE0A : U0%(9)=&HC020 : U0%(10)=&H2AF : U0%(11)=&HFFEB
2700 U0%(12)=&HC920 : DEF USR0=VARPTR(U0%(0))
2900 PRINT USR0("SY0:FNAME.ABS ");" CANNOT BE EXECUTED"
```

In line number 2900 the name of the program to be called up should be inserted between the quote marks. You must add a space after the file name for proper execution of this subroutine.

NOTE: The MBASIC link to a machine code subroutine destroys the present program and MBASIC. To return to MABSIC and the program exited from they must be reloaded.

The program listing below may be added to an assembly language program to link to a PROLOGUE.SYS or any other other program by placing the file name of the program in the **FNAME** statement at the end of the program.

```
 LINK TO PROLOGUE.SYS, 08-OCT-80, JWF                           HEATH ASM #104.05.00
                                                                08-Oct-80    Page   1


  000.000                  .EXIT    EQU     0Q
  000.040                  .LINK    EQU     40Q
  000.041                  .CTLC    EQU     41Q
  031.136                  $TYPTX   EQU     031136A

  042.200                           ORG     042200A

  042.200                  START    EQU     *
  042.200  041 341 042              LXI     H,ABORT
  042.203  076 003                  MVI     A,3
  042.205  377 041                  SCALL   .CTLC

  042.207                  LINK     EQU     *
  042.207  315 136 031              CALL    $TYPTX
  042.212  012 114 151              DB      12Q,'Linking to and Running'
  042.241  040 120 122              DB      ' PROLOGUE.SYS',212Q
  042.257  041 000 043              LXI     H,FNAME
  042.262  377 040                  SCALL   .LINK
  042.264  315 136 031              CALL    $TYPTX
  042.267  012 125 156              DB      12Q,'Unable to Execute, '
  042.313  157 162 040              DB      'or file not found.',212Q
  042.336  303 375 042              JMP     EXIT

  042.341                  ABORT    EQU     *
  042.341  315 136 031              CALL    $TYPTX
  042.344  012 120 162              DB      12Q,'Program User Aborted',212Q
  042.372  303 375 042              JMP     EXIT

  042.375                  EXIT     EQU     *
  042.375  257                      XRA     A
  042.376  377 000                  SCALL   .EXIT

  043.000  123 131 060 FNAME        DB      'SY0:PROLOGUE.SYS',0

  043.021  000                      END     START
```

GK:

(vectored from page 14)

compiling is that it protects your code. If you sell a BASIC program, you have to give the buyer the source, but with FORTRAN, all you have to give him is the machine code. That makes it harder for someone to copy your work.

This article has been a discussion of FORTRAN in general, but in future articles, we will narrow in on the Microsoft FORTRAN available for Heath 8-bit computers. Most of what is said will also apply to H11 FORTRAN. If you have any ideas, suggestions, or constructive criticism about this column, or any FORTRAN programs, I hope you will send them to me. I will try to answer some questions in this column. I hope we can learn more about FORTRAN together and benefit from its use.

EOF

Local HUG News

A new HUG group has been formed for Heath users in the Minneapolis-St. Paul area. Meatings will alternate between the Heath Electronic Centers in St. Paul and Hopkins, with a meeting scheduled for Dec. 7 in Hopkins. The proposed name of the group is SMUG (St. Paul-Minneapolis Users' Group). The St. Paul store is at 1645 White Bear Ave., St. Paul MN 55106, phone (612)-778-1211. The Hopkins store is at 101 Shady Oak Rd., Hopkins MN 55345, phone (612)-938-6371.

The Triad Heath Users' Group has been formed in Winston-Salem NC, with meatings scheduled for the second Saturday of each month at the Sears Activity Room at Hanes Mall at 1:00 pm. For more information, contact Huges Hoyle in Greensboro at (919) 378-1050, or Steve Minor in Winston-Salem at (919) 765-7717. Direct your correspondence to Steve Minor, 424 Cliffdale Drive, Winston-Salem NC 27104.

The San Diego Heath Users' Group meets on the first Wednesday of the month at the La Mesa Heath Electronics Center, 12202 Kingsford Ct., El Cajon CA 92021.

The Milwaukee Heath Users' Group (MHUG) holds meetings on the third Saturday of each month at 2:00 pm. They are held at the Heathkit Electronics Center, 5215 W. Fond du Lac Ave. Call Marvin Olson at 352-3346 for details.

The Hialeah users' group (Miami area) has a bulletin board on line. It is available after 6 pm Eastern time at (305)823-2282. A special thanks to Ralph Boyd of Miami for setting it up.

ECHUG (El Cerrito Heath Users' Group -- Oakland area) meets at 7:30 pm on the fourth Wednesday of every month at the El Cerrito CA store.

EOF

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

CUT ALONG THIS LINE

HUG MEMBERSHIP RENEWAL FORM

When was the last time you renewed?

Check your ID card for your expiration date.

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

Name ______________________

Address ______________________

City-State ______________________

Zip ______________________

REMEBER — ENCLOSE CHECK OR MONEY ORDER

CHECK THE APPROPIATE BOX AND RETURN TO HUG

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

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

Sunrise, Sunset

Ever wonder how the TV weather man knows just when sunrise and sunset will be? Well, I don't know if he has a computer, but if he did, he could use the following program. Just give it the date and your location (longitude, latitude), and it will tell you when sunrise and sunset are for that date. The equation used is an approximation, so there may be errors of up to 6 minutes.

```
00010 INPUT "ENTER THE MONTH (1-12) ";M
00020 INPUT "ENTER THE DAY (1-31) ";D
00030 INPUT "ENTER THE LATITUDE: DD.MM (N=+,S=-) ";L1
00040 INPUT "ENTER THE LONGITUDE: DD.MM (W=+,E=-) ";L2
00050 L1=(L1-INT(L1))/60+L1
00060 L2=(L2-INT(L2))/60+L2
00070 T=.988*(D+30.3*(M-1))
00080 D8=23.5*COS((T+10)/57.296)
00090 E=.123*COS((T+87)/57.296)-1/6*SIN((T+10)/57.296/.5)
00100 Z=TAN(D8/57.296)*TAN(L1/57.296)
00110 Z1=ATN(SQR(1-Z*Z)/Z)
00120 IF Z1<0 THEN Z1=Z1+3.14159
00130 G1=12-E+(L2+Z1*57.296)/15
00140 G2=12-E+(L2-Z1*57.296)/15
00150 K=G1-G2
00160 G4=INT(G2):G2=(G2-INT(G2))*60
00170 G3=INT(G1):G1=(G1-INT(G1))*60
00180 PRINT "THE DATE IS ";M;"/";D
00190 PRINT "SUNRISE GMT IS";G4;"HOURS";INT(G2);"MINUTES."
00200 PRINT "SUNSET GMT IS";G3;"HOURS";INT(G1);"MINUTES."
00210 PRINT "THERE ARE";INT(K);"HOURS AND";INT((K-INT(K))*60);"MINUTES OF DAYLIGHT."
00220 END

*RUN
ENTER THE MONTH (1-12) 3
ENTER THE DAY (1-31) 16
ENTER THE LATITUDE: DD.MM (N=+,S=-) 37.57
ENTER THE LONGITUDE: DD.MM (W=+,E=-) 91.46
THE DATE IS  3 / 16
SUNRISE GMT IS 12 HOURS 19 MINUTES.
SUNSET GMT IS 24 HOURS 8 MINUTES.
THERE ARE 11 HOURS AND 49 MINUTES OF DAYLIGHT.
End at line 220
```

Thanks to Laird D. Schearer
2311 Pimmit Dr. #915
Falls Church VA 22043

Heath Users' Group
Hilltop Road
St. Joseph MI 49085