

\$2.50

REMark®

Issue 39 • April 1983



Official magazine for users of **HEATH ZENITH** computer equipment.



THE H-100 SERIES: THE WORLD'S FIRST 16-BIT COMPUTER KITS.

with authorized sales and service
only through the Heath Company and
Heathkit® Electronic Centers.*

The world-famous H-100 Series Heath/Zenith computers are the first to give you advanced 16-bit computing at a kit price. Many who have never considered kitbuilding are now interested because most circuit boards are prewired, making the H-100 Series our simplest computer kits. Dual microprocessors deliver 16-bit speed and 8-bit compatibility. The industry standard S-100 card slots allow a host of peripherals and memory expansion to 768K RAM. And our stores and catalogs have the software to help you take full advantage of the faster 16-bit operating speed.

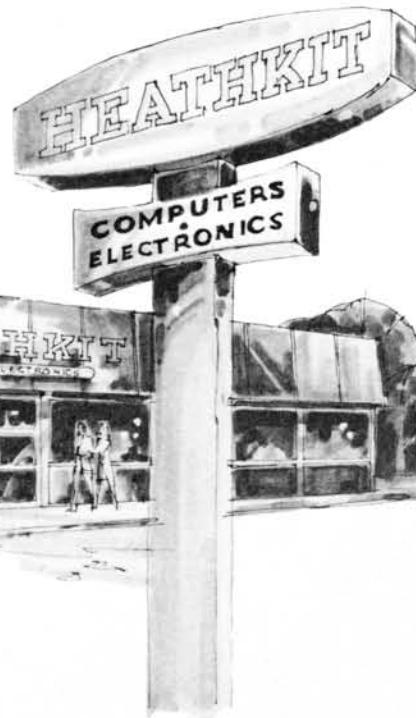
When you're ready to move up to H-100 Series,

remember: these computers are available *only* through the companies that give you solid service and a trustworthy warranty...Heath and Heathkit® Electronic Centers.* The companies that stand by their promise of support to kit-builders. The companies with the pledge: "We won't let you fail."

Many companies would like to sell our product but no other dealer or vendor can resell the H-100 Series without voiding the original factory warranty. When buying the world's first 16-bit computer kit, buy only from the world leader in electronic kits.

**Authorized sales and service available
only through the Heath Company
and your...**

Heathkit®
ELECTRONIC CENTER*



*Heathkit Electronic Centers
are units of Veritechnology
Electronics Corporation.

VEC-770

HUG Manager Bob Ellerton
Software Engineer Pat Swayne
HUG Bulletin Board

and Software Developer Terry Jensen
Software Coordinator Nancy Strunk
HUG Secretary Margaret Bacon

REMark Editor Walt Gillespie
Assistant Editor Donna Melland

Printers Imperial Printing
St. Joseph, MI

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

| | U.S. Domestic | Canada & Mexico | International |
|---------|------------------|--------------------|---------------|
| Initial | \$18 | \$20* | \$28* |
| Renewal | \$15 | \$17* | \$22* |

*U.S. Funds.

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

Limited back issues are available at \$2.50 plus 10% handling and shipping. Check HUG Product List for availability of bound volumes of past issues. Requests for magazines mailed to foreign countries should specify mailing method and appropriate added cost.

Send Payment to: Heath Users' Group
Hilltop Road
St. Joseph, MI 49085

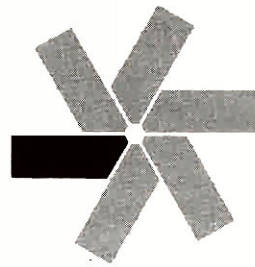
Although it is a policy to check material placed in RE-Mark for accuracy, HUG offers no warranty, either expressed or implied, and is not responsible for any losses due to the use of any material in this magazine.

Articles submitted by users and published in RE-Mark, which describe hardware modifications, are not supported by Heathkit Electronic Centers or Heath Technical Consultation.

HUG is provided as a service to its members for the purpose of fostering the exchange of ideas to enhance their usage of Heath equipment. As such, little or no evaluation of the programs or products advertised in RE-Mark, the Software Catalog or other HUG publications is performed by Heath Company, in general and HUG in particular. The prospective user is hereby put on notice that the programs may contain faults the consequence of which Heath Company in general and HUG in particular cannot be held responsible. The prospective user is, by virtue of obtaining and using these programs, assuming full risk for all consequences.

REMark is a registered trademark of the Heath Users' Group, St. Joseph, Michigan

Copyright © 1983, Heath Users' Group



on the stack

| | |
|---|----|
| Buggin' HUG | 7 |
| Conference Registration | 7 |
| A Faster Benton Harbor BASIC <i>Dahl B. Metters</i> | 11 |
| A Tutorial For Better Screen Control <i>David E. Warnick</i> | 19 |
| H/Z-89 Memory Replacement With 64K Chips <i>J. D. Ross</i> | 23 |
| Introduction to Z-BASIC Part V <i>Gerry Kabelman</i> | 24 |
| New HUG Products | 26 |
| An Educators Best Friend, P.I.L.O.T. <i>Kurt Albrecht</i> | 28 |
| Getting Started With Assembly Language <i>Pat Swayne</i> | 35 |
| Z-BASIC Power to the Rescue <i>Terry Jensen</i> | 39 |
| HERO-I Chats with the H89 | 41 |
| Local HUG Clubs | 48 |
| Heath Related Products | 49 |
| NAVPROGseven Updates <i>Alan Bose</i> | 50 |

ON THE COVER: Shown is Donna Melland, new RE-Mark Assistant Editor with the new Z-29 terminal, Zenith's latest release in it's terminal family that began shipping to the customer in late February. Also shown is the new dual 8" half height disk drive system, the Z-207-42 which was introduced at the West Coast Computer Fair last month. It is scheduled to be available sometime in May with the kit version, H-207-40 to follow.

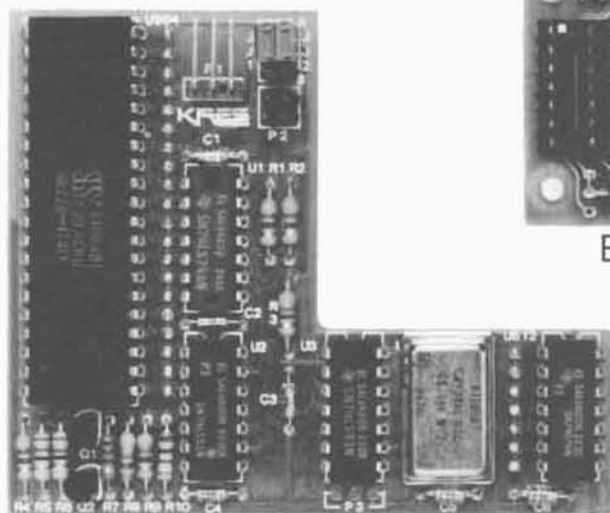
MORE SOLUTIONS

from KRES

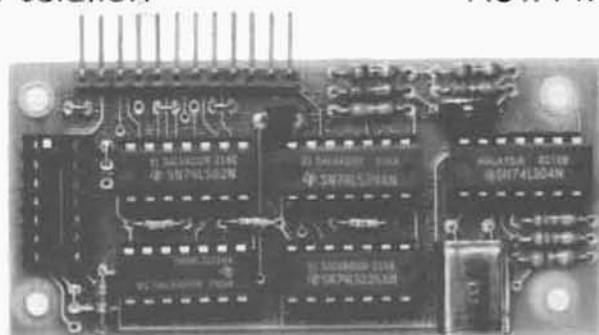
the DUAL SPEED MODULE

a faster solution

Now Available in Two Versions



DSM-240 for the H/Z 89-90



ESM-240 for the Expansion

Why run half speed when you can run full speed? Install one of these KRES Modules in your H/Z 89-90 and cut computation time in half. The supplied HDOS or CP/M software will allow you to operate at regular or double speed (2 or 4 MHz).

Either version with software **\$79.95**

the BACKPLATE

a neater solution

Are you tired of having wires and cables stuffed out the holes of your computer when they should be mounted? The KRES Backplate is an upgrade replacement for the existing rear panel on your H/Z 89-90.

With this KRES solution, you should be able to run every cable you will ever need out the back of your computer cleanly and neatly.

RBP-200
\$35.00



All prices FOB Irvine, CA
California Residents add 6% tax

H/Z 89 is a Registered Trademark of the Heath Company
CP/M is a registered trademark of Digital Research

KRES

ENGINEERING

P.O. Box 17328, Irvine, CA 92713

(714) 559-1047

or (213) 957-6322

Bulletin Board: (714) 559-8579



Moving Up In 83

A few months past I produced an Editorial on the fact that "Nothing remains the same but change". Well, here at HUG, that seems to hold most true. If you look close you will notice a change in the Mast Head of REMark on Page 3, also, REMark is now a registered trademark of the Heath Users' Group and should carry the appropriate designation when it is referred to in other publications

Due to increased pressures in our software sales Nancy Strunk was moved to the full-time duties of Software Coordinator, this she has been doing on a part-time basis for quite a while. This move also brought about changes here in the REMark office. Because Bob Ellerton, HUG Manager, wanted this magazine to grow it was decided that it was time for a full-time Assistant Editor, so I would like to introduce Donna Melland.

Donna is a native of North Dakota having lived in Michigan for the past three years. She comes to HUG after two and a half years as a Senior Clerk for the ZDS Hardware Engineering Group where she was responsible for handling all orders for prototype parts. Previously she had experience in professional photo-finishing ranging from negative development to printing of finished photographs. Donna has been using computers for the past two and a half years mainly for word processing and has taken computer courses including BASIC programming. Her hobbies include biking, camping and sailing. I'm sure that Donna will be a great asset here at HUG and we would like to give her a warm welcome.

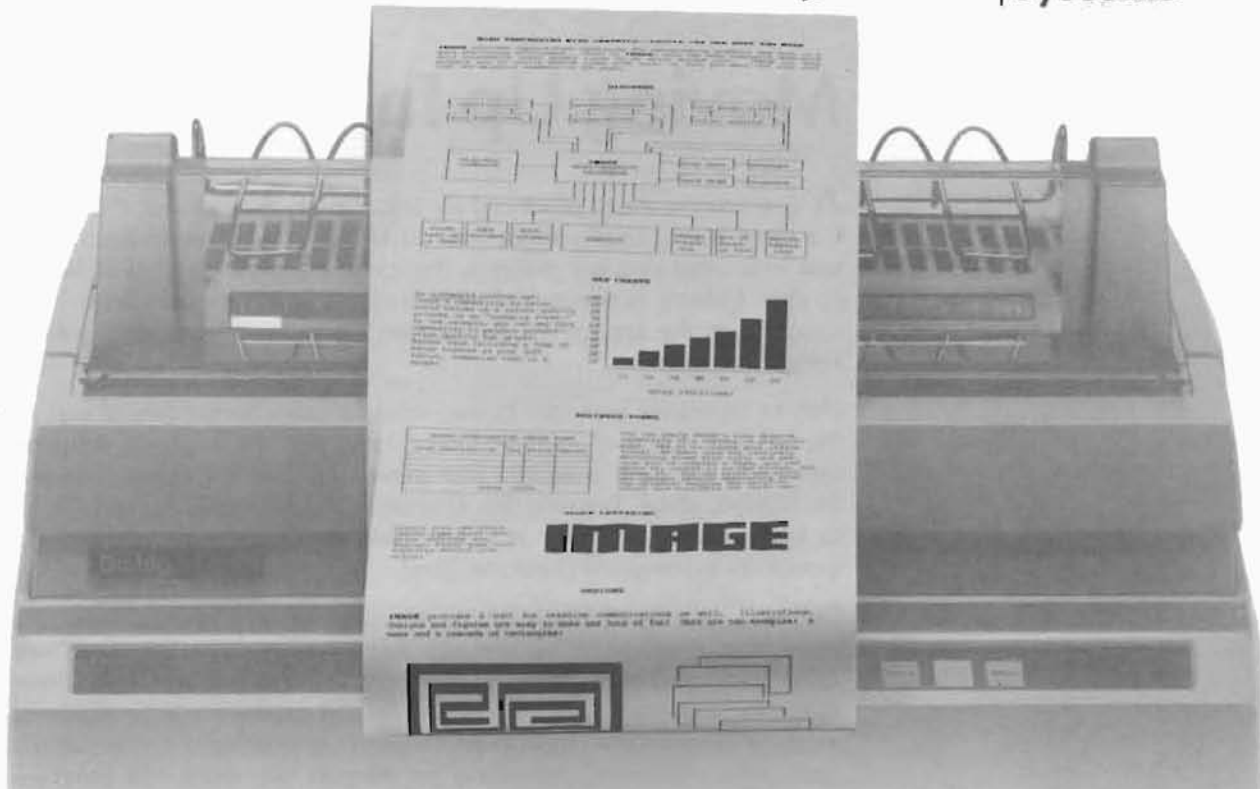
We hope you have enjoyed the changes and improvements both here with REMark but also with HUG. All the staff has been digging in deep to make improvements at what ever can be found, not all these changes will be obvious at first, but in time the changes will become clear.

If you haven't yet joined the Heath Users' Group, we know that there are some who borrow a copy from a friend, now is the time. With the up coming HUG Conference II in Chicago, plus many new bonuses here in REMark such as Cheap-Calc and PILOT, and the great software offers from the HUG library, a dollar goes a long way. Get your monies worth, join HUG.

Walt Gillespie
REMark Editor

WORD PROCESSING IS NOW MORE THAN PROCESSING WORDS.

ANNOUNCING **IMAGE™** FOR **ZENITH** data systems



WORD PROCESSING WITH GRAPHICS... PEOPLE CAN SEE WHAT YOU MEAN.

IMAGE adds a new dimension to word processing... graphics! Now, you can combine text and linear graphics to create **BAR CHARTS, FORMS, ORGANIZATIONAL CHARTS, FLOW CHARTS, BLOCK LETTERS**, and much more. The result? More powerful and more effective communication. People can *SEE* what you mean.

TOP MARKS FROM INFOWORLD

IMAGE won top marks from *INFOWORLD* for its innovation, quality, reliability, and ease of use:

"*IMAGE* certainly deserves accolades in the *performance* category."

"The *documentation* is simply superb. It is professionally done from cover to cover."

"Without a doubt, this program is *easy to use*."

"The program is *bombproofed* so well that I had trouble finding any errors."

IMAGE is a trademark of MicroArt Corporation.

InfoWorld
Software Report Card

Image

| | Poor | Fair | Good | Excellent |
|----------------|--------------------------|--------------------------|--------------------------|-------------------------------------|
| Performance | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| Documentation | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| Ease of Use | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| Error Handling | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> |

All quotes are from InfoWorld's *IMAGE* software review, by Marty Petersen, June 14, 1982.

Copyright 1982 by Popular Computing, Inc., a subsidiary of CW Communications, Inc., Framingham, MA—Reprinted from InfoWorld.

EXTRAORDINARY VALUE...

"The modest \$295 Price Tag is a Bargain."

IMAGE runs on any Zenith Z/89 or Z/90 computer, on any Heath H/89 or H/90 computer, or on any Z-80-based CP/M system linked to a Z/19 or H/19 terminal.

TO ORDER CALL TOLL FREE:
1-800-MICROART (1-800-642-7627)
in Oregon, call: **1-692-3950**

OR WRITE: MicroArt Corporation
200 Market Bldg.
Suite 961
Portland, OR 97201

Mastercard, Visa and COD orders accepted.

BUGGIN' HUG



Dear HUG,

Just finished reading Terry Jensen's article in REMARK Issue #35 on the MPI 99-G Printer. I purchased mine a couple of months ago and can readily agree with Terry's obvious satisfaction with his choice of printers. Even without Style Writer it would be a superior product; with Style Writer (and Calligrapher, the companion FONT creator), the 99-G really comes to life.

I have some additional comments that may be of interest to your readers who may be considering buying a new printer. Figure 1 in Mr. Jensen's article (page 27) which displays examples of all the supplied FONT styles shows a non-problem in the "MAS-SIVE" and "DEMO" fonts, i.e., several light horizontal lines that would seem to indicate that a row of printed dots was missed. These "light lines" occur intermittently only when paper is tractor fed through the printer. The tractor feed does not provide the tight graphics. However, by switching to the pressure roll feed used for single sheets, the "light lines" completely disappear, thus providing a very nice tight appearance.

Terry also correctly points out that the printer pauses for brief periods of time when printing special fonts. This is due to the serial 1200 baud rate which is the "standard" interface supplied by MPI for HEATH users. The printer is also capable of supporting a parallel interface or the serial interface can be speeded up by purchasing MPI's HI-SPEED Serial Adapter card which permits baud rates up to 9600 baud. At 9600 baud there is no noticeable pausing while printing special fonts or graphics.

The HI-SPEED Serial Adapter card (3" by 4") fits nicely in the right front corner of the printer. Some minor soldering is required on the "mother board" to convert from "character busy" to "buffer busy", but is overall very simple to install. In addition the card comes with a longer (round) cable to replace the shorter ribbon cable supplied with the printer. For those who decide to purchase the HI-SPEED Serial Adapter card, it works very nicely with CP/M and STYLE WRITER (just need to "CONFIGUR" the LST: device for 9600 baud). However, for HDOS users, the LP: device driver will need to be modified to handle 9600 baud correctly.

Vectored to 8 ☐



Heath
Users'
Group

NATIONAL HUG CONFERENCE II

Official Conference Registration Form
O'Hare Hyatt Regency Hotel, Chicago, Illinois
August 19, 20 and 21

Name(s) _____

Address _____

Company _____

City _____ State _____ Zip _____

Enclosed is \$20.00 per individual to attend The Second National HUG Conference to be held the weekend of August 19, 20 and 21 1983. Please send ticket(s) and information regarding hotel reservations.

AMOUNT ENCLOSED _____ NUMBER ATTENDING _____

For our information:

Which Heath/Zenith computer do you now operate? _____

Are you a Non-User-Attendee? YES NO

Are you a Heath/Zenith related vendor? YES NO

For your information:

Space limitations for the dinner to be held Saturday August 20, 1983, will restrict the number of attendees for that dinner to 1000. Therefore, it is important that you register as soon as possible. Visitor tickets for those of you simply attending and not planning to stay for the dinner and prize drawings will be available at the registration booth for \$10.00. Send your registration form or a suitable copy to:

Heath Users' Group
Attention: National HUG Conference Registration
Hilltop Road
Saint Joseph, Michigan 49085

Special Note to Vendors:

Vendor Information Packages will be made available to Heath/Zenith Related Vendors who are planning to exhibit their products while at the conference. Three times more space is available this year for the purpose of showing those products of interest to owners of Heath/Zenith computer products.

I spoke with the technical representative from MPI, Mr. Scott Anderson, at some length to nail down the problem. Scott was very patient and helpful. It seems that at 9600 baud, when the printer buffer (2K) cannot hold another complete line, the host is signalled to wait until there is available buffer space. That signal is sent at line termination time, if needed (CR, LF, or Form-Feed). But at 9600 baud, another character or two may already have sneaked over to the printer before the signal to wait is received by the host. Consequently, characters at the beginning of printed lines can be lost intermittently.

The fix to the above problem is for the LP: device driver to add a small delay after each line termination character is detected and before the next character is sent to the printer. The delay will allow the printer enough time to tell the host that it's buffer is full (busy) so the host won't overrun the printer.

Since the device driver already detects line termination characters for other reasons, it is only necessary to add a small subroutine that will implement the required delay and CALL the subroutine at the appropriate time. HEATH supplied Device Drivers for Line Printers (source available with HDOS 2.0) typically contain a subroutine labelled "LPOUTCH" - short for Line Printer Output Character. It is within this subroutine that tests are made for line termination characters. The following subroutine can be inserted after the end of "LPOUTCH" and CALLED when needed:

```

CHRDLY EQU *
        LHL D TLP,BAU pickup baud rate divisor
        DAD H
        DAD H multiply
        DAD H by/
        DAD H sixteen
LOOP1 DCX H
        MOV A,L delay for approximately
        ORA H 2 character
        JNZ LOOP1 times
        RET return to calling routine

```

The above subroutine obviously will work correctly for any baud rate. Adding this small delay at the end of each printed line is not discernible operationally, except that it does not correct the character loss problem for HDOS user.

But no matter which baud rate is used, in the final analysis, the MPI 99-G is a fine product. And so, for that matter, is REMark, which I've enjoyed for 3 years now. Keep up the good balanced coverage.

Warren Buss
3033 N. Homestead Pl.
Tucson, AZ 85749

Dear Walt,

I just read the article on Word-processing by Laura Sparrow in REMark Issue #37 and it brought to my mind my early attempts at Word-processing. I have been using my H-89 and Diablo printer for Word-processing for more than two years and have learned that it is the only way to write anything. I have found a solution to one of the problems she mentions in regard to Word-processing and I thought other readers might be interested in it. I was always unhappy with the ragged edged paper that emanated from the printer because it made the excellent quality print of the Diablo look rather shabby, so at one point I also resorted to feeding in single sheets of good quality paper and then later I also resorted to Xeroxing my originals, but both of these processes added an extra step and were time consuming. I then discovered that I could buy what is called laser cut paper, which when separated yields an almost perfectly smooth edge. I have no qualms about sending this paper to editors or using it for my correspondence and indeed send this letter to you on this paper. I assume this paper is readily available all over and for a price guideline I can say that I have been buying boxes of 3200 sheets for \$25.00 from J&B Data Processing at 2501 E. Cork St. in Kalamazoo, Michigan.

Robert I. Sundick
1732 Greenlawn
Kalamazoo, MI 49008

Dear HUG:

We had the requirement to patch a great number of disks on our SINGLE DRIVE system, making identical patches on all of the disks to speed up the default automatic boot time from 30 to 2 seconds (as per REMark 21, p. 11). After a little thought and experimentation we came up with the following procedure that eliminated errors and greatly speeded up the patching session.

A. Run DUMP and perform the patch on the disk currently in the drive. This becomes the "source disk. Stay in DUMP, do not exit or reboot as yet.

B. Answer the drive, track, sector questions with the same sector you just patched. Answer YES to the question "modify this sector?". Enter 00 for the "starting with?" question, and then hit SPACE and a carriage return.

C. The sector will be redisplayed (in unchanged form) and you will be asked "Write this sector back to disk?". We want to do this, but not on this disk, so just remove the "source" disk, and insert a "destination" disk. Since this disk now contains the correct sector, it can now become a "source" disk, and the procedure can loop back to Step B (above) without further disk handling.

Please note that this works only when an identical sector is to be patched on each disk. We used it to modify the boot track (track 0), but it should be safe on any other disk where you are sure the track/sectors are identical (such as SYSGENed disks, etc.).

Charles F. Santose
4630 Allen Road
Stow, OH 44224

Dear HUG,

Here is something I thought might be of some interest.

In the 33rd issue, I noticed the article 'Base Conversions for MBASIC' which described a similar problem of mine, but I don't have MBASIC. So, after attempting to use tables, I sat down to write a routine to do the conversions. The result was a Benton Harbor BASIC program which is an application very similar to the MBASIC conversion program. Although, this program also does conversions in binary, and other bases to be converted may be easily added.

This program basically utilizes two subroutines, one to convert Decimal to any base and the other to convert any base to Decimal. Through these two routines it was possible to condense this program and has made the addition of other bases to be converted very simple. This program can easily be added to the end of any program as to prevent the annoying switching of programs, or can be run alone. When this program is merged with a program care should be taken to prevent one program from running into the other, and to avoid stray effects not to use the variables.

Program description:

Lines 65010 to 65040 just determine which base to convert. Line 65030 determines if the requested base is legal, if it is not, the program stops (if the program is used as a routine, a RETURN can be substituted).

Both this line and the next must also be altered if other bases are added.

Lines 65050 to 65085 use the subroutines at line 65090 to first get the number to be converted, any illegal numerals are treated as zeros, and check to see if a different base conversion is wanted. They then convert that number to base ten and use the subroutines at 65110 to 65135 to determine which base to convert to and to go to the routine which does the conversion.

Lines 65140 to 65170 is the subroutine that converts decimal to any base, which is set by the value of B. Line 65150 determines the largest power of the base that goes into the decimal value. Lines 65160 to 65170 then determine the value of the conversion and print it, and then return for the next conversion.

Lines 65190 to 65220 is the subroutine that converts from any base to decimal which is done by breaking down the number to get the relative decimal value of each numeral, which is then totaled. The decimal equivalent is then printed, and execution proceeds to the next routine.

While writing this program I found an unexplainable error which causes the conversions to binary over 511 base ten to give random results. The test in line 65130 was the only way I could manage the error. I recommend trying it without the test before you set a limit on the binary conversions.

Louis Berger
2929 Greenvale Rd.
Chevy Chase, MD 20815

Software Editor, HUG

This program is designed for use with a Heathkit ET-3400 microprocessor trainer, ETA-3400 I/O memory accessory and H-9 CRT. It can be used on other fixed point or floating point micro-computers with little or no changes.

The data interpretation program calculates and displays OSCAR beacon Morse Code telemetry data. This data received from the satellite, after calculation, tells the status of the satellite. The calculations, although not difficult, involve a lot of "busy work" and are time consuming. They are a natural for a computer.

Data inputs are the 6 two digit numbers (A,B,C,D,E,F) following HI 1(A), 2(B), 3(C), 4(D), 5(E), 6(F) or the OSCAR beacon at 29.402 MHz. Outputs are the solar array current (ma.), battery current (ma.), battery voltage (mv.), baseplate temperature (°C), battery temperature (°C) and mode J power output (mw.).

```
65000 REM BASE CONVERSION ROUTINE WRITTEN BY LOUIS BERGER 20-OCT-82.
65010 A$="0123456789ABCDEF":PRINT CHR$(27);"E";
65020 PRINT :LINE INPUT "Which conversion (1-Hex, 2-Dec, 3-Oct, 4-Bin)? ";C$
65030 C=ASC(C$)-48:IF C<1 OR C>4 THEN STOP
65040 ON C GOTO 65050,65060,65070,65080
65050 B$="Hex":GOSUB 65090:B=16:GOSUB 65190:GOSUB 65110:GOSUB 65130:GOTO 65050
65060 B$="Decimal":GOSUB 65090:D=VAL(B$):GOSUB 65120:GOSUB 65110:GOSUB 65130
65065 GOTO 65060
65070 B$="Octal":GOSUB 65090:B=8:GOSUB 65190:GOSUB 65120:GOSUB 65130:GOTO 65070
65080 B$="Binary":GOSUB 65090:B=2:GOSUB 65190:GOSUB 65120:GOSUB 65110
65085 GOTO 65080
65090 PRINT :PRINT B$;" #";:LINE INPUT " ";B$:IF B$="" THEN 65020
65100 RETURN
65110 B=8:B$="Octal":GOTO 65140
65120 B=16:B$="Hex":GOTO 65140
65130 IF D<512 THEN B=2:B$="Binary":GOTO 65140
65135 RETURN
65140 T=0:D1=0:T$="":REM * * DEC - XXX * *
65150 IF B^T<=D THEN T=T+1:GOTO 65150
65160 FOR X=T-1 TO 0 STEP -1:Y=INT(D/B^X):T$=T$+MID$(A$,Y+1,1)
65170 D=D-(Y*B^X):NEXT X:PRINT B$;" equivalent: ";T$:D=D1:RETURN
65180 REM * * XXX - DEC * *
65190 Q=1:D=0:FOR X=LEN(B$) TO 1 STEP -1:C$=MID$(B$,X,1)
65200 FOR T=0 TO B-1:IF C$<>MID$(A$,T+1,1) THEN NEXT T:GOTO 65220
65210 D=D+(T*Q)
65220 Q=Q*B:NEXT X:PRINT "Decimal equivalent: ";D:RETURN
```

OSCAR DATA CALCULATION FROM TELEMETRY
BY JOHN GALLAGHER

PROGRAM LISTING

```
10 INPUT A,B,C,D,E,F
20 J=(72*(101-A)-5)/10
30 K=57*(B-50)
40 L=100*C+8250
50 M=((9580-148*D)+50)/100
60 N=((9580-148*E)+50)/100
70 P=23*F
80 PR"SOLAR I",J,"MA.", "BATT. I",K,"MA.", "BATT V",L, "MV."
85 PR"BASE T",M,"DEG.C", "BATT T",N,"DEG.C", "MODE J OUTPUT",P, "MW."
90 GOTO 10
100 END
```

EXAMPLE RUN - KEYBOARD ENTRIES ARE UNDERLINED

```
:RUN
?71,48,73,34,36,8(CR) CR: CARRIAGE RETURN
```

OUTPUT ON CRT

```
SOLAR I 215 MA. BATT I -114 MA. BATT V 15550 MV.
BASE T 45 DEG.C BATT T 43 DEG.C MODE J OUTPUT 184 MW.
```

John Gallagher
411 S. Elm Rd.
Lakeland, FL 33801

A Faster Benton Harbor BASIC



Dahl B. Metters
DCA-OKI Box 959
FPO Seattle, WA 98773

Benton Harbor BASIC is a good implementation of the BASIC language. It contains all the features that most 'standard' BASIC's contain and, in addition, offers other useful features not found in other implementations, e.g. features such as CHAIN, FREEZE, UNFREEZE, and others. Of course, another significant feature is its price. While BH BASIC is fast enough for most applications, a relatively simple modification, given below as FBASIC (Faster BASIC), can increase its speed of execution by a factor of two to three for long programs.

Ideas for implementing FBASIC came from the July 82 issue of REMark, "An Editor for BH BASIC", by Patrick Swayne. This was, in turn, partially based upon T. J. Eitel's RELOC program. ELOADR here is identical to that published in July 82 except for the size parameter (14 lines from the bottom of page 31 of the July 82 issue).

BH BASIC stores BASIC program text lines internally as follows:

- (1) A 16-bit binary line number,
- (2) Program text, where all BASIC reserved words are stored as one-byte tokens, and
- (3) a one-byte end of line token (0).

NOTE: The end of program is denoted by a line number equal to 65535.

An important factor in a BASIC interpreter that can affect execution speed is the strategy used to process GOTOs and GOSUBs. The strategy used by BH BASIC is:

- (1) the destination, stored as a string of ASCII characters following the one-byte token for GOTO or GOSUB, is converted to a 16-bit binary number.
- (2) a character pointer is set to the beginning of the user program.
- (3) The character pointer points at a 16-bit binary representation of the current line number. The current line number is compared to the destination line number.

(a) If equal, this is the correct line number. End of search.

(b) If the destination line number is great-

```
* FBASIC - A program to speed-up B H BASIC
*
*
* This program must be combined with ELOADR.ABS.
* See instructions in article text.
*
*
*
USERFWA EQU 42200A
$MOVE EQU 30252A
$TYPTX EQU 31136A
S.SYSM EQU 40320A
```

* Use this program with HDOS 2.0 only!

```
CODE PIC POSITION INDEPENDENT CODE
DW START
```

* LOAD BASIC.ABS INTO MEMORY
* ADAPTED FROM P. SWAYNE'S BEDIT PROGRAM

```
START LXI H,0
      DAD SP FIND STACK
      MOV A,L
      CPI 2000 HAS IT MOVED?
      JZ NONUM NO, NO ARGUMENT ENTERED
      CALL ATOB CONVERT NUMBER TO BINARY
      XCHG
      SHLD MEMTOP SAVE USER'S MEMORY TOP
NONUM LXI SP,USERFWA-8 LOWER STACK A BIT
      MVI A,3
      LXI B,0FFFFH
      SCALL .CONSL SET FULL CONSOLE WIDTH
      LXI D,DEFAULT
      LXI H,NAMRET POINT TO .NAME BUFFERS
      MVI A,-1 USE SYSTEM CHANEL (-1)
      SCALL .NAME GET FBASIC DEVICE NAME
      LXI H,BASIC POINT TO BASIC
      LXI D,DEFAULT AND DEFAULTS
      XRA A
      SCALL .OPENR OPEN FOR READ
      JC ERR COULDN'T DO IT
      CALL HERE PUSH PC ONTO STACK
      POP H GET CURRENT PC
      SHLD LIMIT SAVE MEMORY LIMIT
      LXI D,-USERFWA
      DAD D SUBTRACT MEMORY START
```


er than the current line number, the desired line has not yet been reached. The character pointer is incremented until it points one location past the end of line marker (zero byte). The search continues at step (3).

(c) If the current line number is larger than the destination line number, the required line was not found. No such line number exists in the program - BASIC prints, "Illegal or unknown statement number at line #xxx".

Clearly, if a long user program is being run with many references to line numbers near the end of the program, this searching process will be extremely time consuming.

A different strategy for long programs, used by FBASIC given below, is:

(1) Whenever a RUN or CONTINUE command is detected, the resident BASIC program is searched for all occurrences of GOTO and GOSUB. The destination line numbers are put into a table with a pointer to the beginning of that line in the BASIC text.

(2) Whenever a GOTO or GOSUB is encountered in a user program, the line number table is searched for the destination line number and its physical address is retrieved.

The increase in speed obtained by using FBASIC depends on the program being run. Our test program, a super-inefficient BASIC line renumbering program, takes 1 hour and 34 minutes when run under standard BH BASIC. FBASIC takes 'only' 37 minutes. This is an effective increase in speed of 2.5 times. Different programs will be affected differently depending on the number of GOTOs and GOSUBs and the location of the destination lines. Do not expect benchmark programs with only a few GOTOs and GOSUBs to be speeded up at all.

The FBASIC patch will cause the following actions:

(1) When RUN, CONTINUE, or STEP commands are given, BLDTBL (Build Table) is called from BH BASIC. BLDTBL builds the line number table and assigns a physical address to every destination line number. Note that the line number table is built with the largest numbers at the beginning of the table. Put the most frequently called sub-routines at the end of your programs with FBASIC. If a GOTO or GOSUB references a non-existent line, the physical address is set to zero. The cases of ON ... GOTO and ON ... GOSUB are handled by looking for a comma after each referenced line number. The case of IF...THEN... is handled by treating THEN the same as GOTO or GOSUB.

| | | | |
|--------|-------|--|----------------------------|
| | MOV | B,H | |
| | MVI | C,0 | (BC) = BUFFER SIZE |
| | LXI | D,USERFWA-8 | PUT BASIC.ABS HERE |
| | XRA | A | |
| | SCALL | .READ | READ IN BASIC.ABS |
| | CPI | 1 | GOOD READ (MUST BE EOF) |
| | JNZ | ERR | NO |
| | XRA | A | |
| | SCALL | .CLOSE | CLOSE FILE |
| * | | FIX USER MEMORY LIMIT | |
| | LHLD | LIMIT | GET MEMORY LIMIT |
| | LDA | MENTOP+1 | GET USER'S MEMORY TOP PAGE |
| | ORA | A | ANYTHING ENTERED? |
| | JZ | NOUSRM | NO, USE LIMIT |
| | CPI | 1320 | ADDRESS TOO LOW? |
| | JC | NOUSRM | YES, USE LIMIT |
| | MOV | B,A | SAVE PAGE IN B |
| | LDA | LIMIT+1 | GET PAGE LIMIT |
| | CMP | B | ADDRESS TOO HIGH? |
| | JC | NOUSRM | YES, USE LIMIT |
| | LHLD | MENTOP | ADDRESS OK, USE IT |
| NOUSRM | SHLD | S.SYSM | SET MEMORY LIMIT |
| * | | INSERT CALL TO BLDTBL | (INTO CONTINUE SECTION) |
| | MVI | A,3150 | CALL |
| | STA | 45165A | |
| | LXI | H,BLDTBL | ADDRESS OF BLDTBL |
| | SHLD | 45166A | |
| * | | INSERT CALL TO NFLN | (INTO GOTO-GOSUB SECTION) |
| | LXI | H,NFLN | REPLACES CALL TO FLN |
| | SHLD | 50035A | |
| * | | INSERT USR FUNCTION ROUTINE | |
| | LXI | H,USR | |
| | SHLD | 56364A | |
| * | | FIXUP BASIC KEYWORD TABLE (REMOVES 'SEG', MOVES PART OF TABLE, | |
| * | | AND INSERTS 'USR') | |
| | LXI | B,70063A-67355A | DELETE 'SEG' |
| | LXI | D,67355A | FROM ADDRESS |
| | LXI | H,67351A | TO ADDRESS |
| | CALL | \$MOVE | |
| | LXI | H,'U'*256+3400 | INSERT USR FUNCTION |
| | SHLD | 70057A | |
| | LXI | H,'R'*256+'S' | |
| | SHLD | 70061A | |
| * | | JMP | 112252A |
| | | | JMP TO BASIC |
| ERR | PUSH | PSW | SAVE ERROR CODE |
| | CALL | \$TYPTX | |
| | DB | 120,7,'ERROR -',2400 | |
| | POP | PSW | RESTORE ERROR CODE |
| | MVI | H,120 | END ERROR WITH NEWLINE |
| | SCALL | .ERROR | |
| | XRA | A | |

```

SCALL .EXIT          RETURN TO HDOS

**** BUILD TABLE OF LINE NUMBERS REFERENCED ****
THEN EQU 206        CODES FOR BASIC KEYWORDS
GOSUB EQU 148
GOTO EQU 149
MTAREA EQU 115010A  START OF BASIC TEXT

BLDTBL PUSH PSW
          PUSH D
          LXI H,MTAREA  HL = START OF BASIC TEXT
          MVI A,0
          STA NENTRIS   SET NO OF ENTRIES = 0
* BEGIN NEW LINE
ABLD MVI C,0         HAVEN'T SEEN LINE NO YET
      MOV E,M
      INX H
      MOV D,M        DE = CURRENT LINE NO
      INX H          HL -> 1ST CHARACTER
* IF DE = 65535 THEN BEGIN NEXT SECTION OF PROGRAM
      MOV A,E
      ANA D
      INR A
      JZ CBLD
BBLD MOV A,M         GET NEXT CHAR
      INX H
      ORA A
      JZ ABLD       IF A=0 THEN END OF LINE

* IF A IS TOKEN FOR THEN, GOTO, OR GOSUB THEN LOOK FOR LINE NUMBER

      CPI THEN
      JZ A1BLD
      CPI GOTO
      JZ A1BLD
      CPI GOSUB
      JZ A1BLD
      CMP C          C = COMMA IF WE SAW LINE NO
      JZ A1BLD
      MVI C,0
      JMP BBLD      GO ON TO NEXT CHAR

* FOUND THEN, GOTO, GOSUB, OR COMMA
A1BLD CALL AT0B     DECODE LINE NUMBER
      DCX H
      PUSH H        SAVE CHAR POINTER
      CALL SRCHTBL
      JC B1BLD     IF LINE NO IN TBL DON'T ADD IT
      LDA NENTRIS
      CPI 255
      JZ B1BLD     IF NO OF ENTRIES IS 255 DON'T ADD
* ANY MORE
      INR A
      STA NENTRIS  UPDATE NO OF ENTRIES
* PUT LINE NO INTO TABLE
      PUSH D        SAVE LINE NO TO PUT IN TBL
      LXI D,4
      XCHG

```

(2) BH BASIC contains a subroutine, FLN (Find Line Number). FLN is called whenever a GOTO or GOSUB is encountered; after evaluation of the destination line number. FBASIC patches Benton Harbor BASIC to call NFLN (New Find Line Number) rather than FLN. If NFLN is successful and finds the destination line number address, then NFLN returns. If NFLN cannot find the line number in its table or if the line number was never used in the program (address = 0), NFLN jumps to FLN which scans through the program again and will normally issue an error message.

An added feature provided by FBASIC is the capability to call machine language functions. A machine language function can be the ultimate method of speeding up a BASIC program. Machine language functions are invoked by Y = USR(A,V); where A is the address of the function, V is the value passed to the user function, and Y is the returned value. Values are passed to the function in the DE register pair and the machine language function returns a value by placing it in the DE before the return statement. Since space should be reserved for the function in memory that is separate from BASIC, FBASIC also provides for reserving a block of memory. When FBASIC is run, it can be run with a parameter that sets the top of memory for BASIC, e.g.

```
>FBASIC 40000
```

This will limit BASIC to memory below address 40000 decimal. To reserve space for a machine language subroutine, run FBASIC. At the * prompt, type:

```
PRINT PEEK(9590) + 256*PEEK(9591)
```

The number printed will be the address of the beginning of FBASIC which resides just below the HDOS system area. If you have a machine language function that occupies 100 bytes, then subtract 100 from the printed number to get n for the 'FBASIC n' command. The machine language function can then be POKE'd into the 100 byte reserved area starting at location n.

FBASIC provides space for the USR function by replacing the SEG function which can no longer be used if this patch is made.

Steps required to get FBASIC up and running are as follows:

(1) Using the HDOS editor, EDIT, type in ELOADR. Save it as ELOADR.ASM

(2) Again using EDIT, type in FBASIC. Save it as FBASIC0.ASM

(3) Assemble both programs separately, i.e.

```
>ASM ELOADR=ELOADR
```

```
>ASM FBASIC0=FBASIC0
```


(4) Correct any errors with EDIT and reassemble.

(5) Combine both ABS files as follows:

>COPY

FBASIC.ABS=ELOADR.ABS,FBASIC0.ABS

(6) That is all there is to it. Remember that to run FBASIC, FBASIC.ABS and BASIC.ABS must be on the same diskette.

FBASIC may be invoked by either -

>FBASIC

or >FBASIC n where n is the desired memory limit for BASIC.

It is not necessary to understand how FBASIC works to use it. The only difficulty with understanding FBASIC is understanding how things get started. It is a little complicated. Here is how it works:

(1) After Booting the system with a diskette containing both FBASIC.ABS and BASIC.ABS the user types 'FBASIC'. FBASIC is an absolute file which has ELOADR at the beginning of the file.

(2) ELOADR causes both HDOS overlays to be loaded, reserves room for FBASIC in high memory, and sets the user memory limit below itself.

(3) ELOADR then loads in FBASIC and transfers control to it.

(4) FBASIC then loads in BH BASIC and patches it so that the calls to BLDTBL and NFLN are made at the proper times.

(5) FBASIC then jumps to BH BASIC.

I hope you enjoy Fast BASIC.



Dahl Metters is an electronics engineer who will be completing a 20 year tour with the U.S. Air Force in May of this year. He is currently head of the Defense Communications Agency Field Office in Okinawa, Japan. Dahl's main interests are in digital hardware design and software engineering.

```

DAD    D                HL=DEST + 4
PUSH  D                SAVE DEST
PUSH  D                SAVE DEST
XCHG  DE=DEST+4
LXI   H, LASTADD + 1
MOV   A, L
SUB   E
MOV   C, A            BC=HL-DE
MOV   A, H            BC = LASTADD+1-(DEST+4)
SRB   D
MOV   B, A
POP   H
XCHG  HL=DEST+4; DE=DEST
CALL  *MOVE
POP   H                HL=DEST
POP   D                DE=LINE NO
*     STORE LINE NO IN TABLE WITH ZERO ADDRESS
MOV   M, E
INX   H
MOV   M, D
INX   H                DE IS NOW IN TABLE
MVI   A, 0
MOV   M, A
INX   H
MOV   M, A            ADDRESS PART IS NOW ZERO
BIBLD POP   H                RECALL CHAR POINTER
MVI   C, ', '
JMP   BBLD            SAW LINE NO, NOW LOOK FOR COMMA TOO
CBLD  LXI   H, MTAREA   HL=START OF TEXT
DBLD  PUSH  H                SAVE START OF LINE ADDRESS
MOV   E, M
INX   H
MOV   D, M            DE=LINE NO OF TEXT
*     IF LINE NO IS 65535 THEN GOTO FINISHED
MOV   A, E
ANA   D
INR   A
JZ    EBLD            WAS 65535, NOW QUIT
CALL  SRCHTEL
JNC   D0BLD
*     FOUND LINE NO IN TABLE STORE START OF THIS LINE
MOV   E, L
MOV   D, H
POP   H
MOV   A, L
STAX  D
INX   D
MOV   A, H
STAX  D
PUSH  H
*     DIDN'T FIND THIS LINE IN TABLE, GET RID OF SAVED ADDRESS
D0BLD POP   H
INX   H
INX   H
*     SKIP REST OF LINE
D1BLD MOV   A, M
INX   H

```

```

ORA  A
JNZ  DIBLD
JMP  DIBLD
FINISHED - NOW CLEANUP AND EXIT
POP  H
POP  D
POP  PSW
LHLD 112177A
RET

*
*
*
* ASCII TO BINARY -- CONVERT ASCII LINE NUMBER TO BINARY IN DE
* ON ENTRY - HL -> POSSIBLE NUMBER
*
* ON EXIT - HL IS UPDATED
* DE = LINE NUMBER
*
*
*
* ATOB  I,0
* AATOB A,M
* INX  H
*
* SKIP LEADING SPACES
* CPI  ' '
* JZ  AATOB
* SUI  '0'
*
* IF NOT A DECIMAL DIGIT THEN RETURN
*
*
*
* DE = DE*10 + A
*
* PUSH  H
* MOV  L,E
* MOV  H,D
* DAD  H
* DAD  H
* DAD  D
* DAD  H
* ADD  L
* MOV  E,A
* MOV  D,H
* JNC  CATOB
* INR  D
* POP  H
* MOV  A,M

```

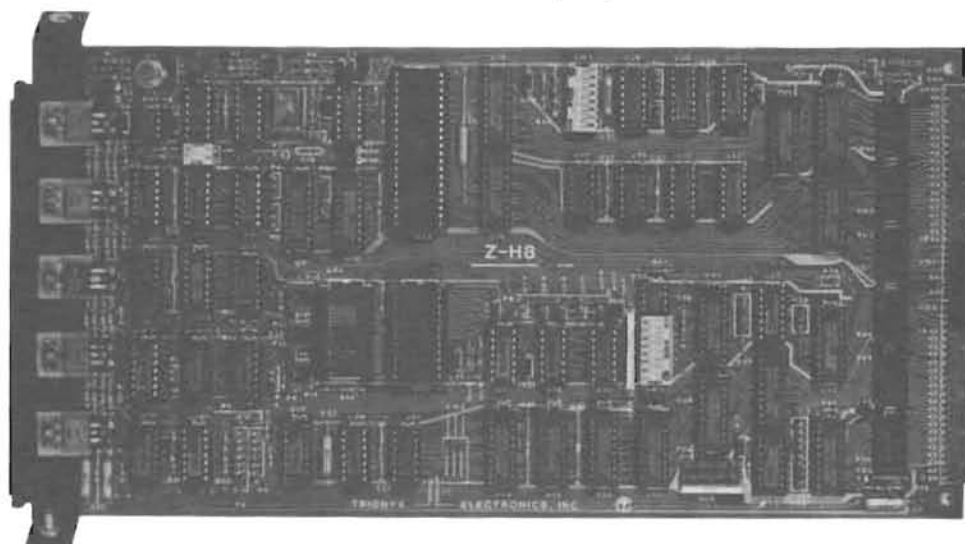
```

INX  H
JMP  BATOB
*
*
*
*SEARCHTEL -- SEARCH LINE NUMBER TABLE FOR DE
* ON ENTRY - DE = LINE NUMBER TO FIND
*
* ON EXIT - IF CY=1 HL -> DATA AREA FOUND
* IF CY=0 HL -> PLACE TO PUT NOT FOUND
*
*
*
* SRCHTBL LXI  H, TABLE
* LDA  NENTRIS
* MOV  B,A
* ORA  A
* RZ
*
* IF TABLE IS EMPTY RETURN
*
*
*
* COMPARE (HL) WITH DESIRED LINE NUMBER IN DE
* HL -> HIGH BYTE OF LINE NO
* HL -> LOW BYTE OF LINE NO
* CY=1 IF (HL)< DE
*
* ASRH  INX  H
* MOV  A,M
* DCX  H
* CMP  D
* JC  A1SRH
* JNZ  A2SRH
* MOV  A,M
* CMP  E
* JC  A1SRH
* JNZ  A2SRH
*
* *** FOUND MATCH *** GOT HERE IF (HL)= DE
*
* INX  H
* INX  H
* STC
* RET
*
* *** NOT THERE YET *** GOT HERE IF (HL)> DE
*
* A2SRH INX  H
* INX  H
* INX  H
* INX  H
* DCR  B
* JNZ  ASRH
*
* ***LINE NUMBER NOT PRESENT*** RAN PAST NUMBER OR RAN OUT OF TABLE
*
* A1SRH ORA  A
* RET

```


Z80 CPU BOARD for the H8*

* H8 is a Registered Trademark of the Heath Company



Model Z-H8

\$ 300.00 Assembled

\$ 250.00 Kit

Programmable Clock Rate — 2 MHz or 4 MHz Operation

Clock Rate under Software Control — May Be Changed at Any Time

Selection of Wait States (None, 1/2, 1 or 2) at 4 MHz

Buss Termination Network • 51 Integrated Circuits • Transistor Power Supply

Includes Heath Extended Configuration (ORG ZERO) Circuits

Exclusive Z80 Front Panel Monitor ROM — Based Upon Heath XCON-8 ROM

Exclusive Software Speed Utilities (For Both HDOS* and CP/M**) on Diskette

Fully Comptable With All Heath H8* Hardware and Software

* H8 and HDOS are Registered Trademarks of the Heath Company.

** CP/M is a Registered Trademark of Digital Research.

Check • Money Order • VISA • MASTERCARD • C.O.D.

Phone Orders Welcome (714) 830-2092 - Send For Free Brochure

TRIONYX ELECTRONICS, INC.

P.O. BOX 5131, SANTA ANA, CA 92704

A Tutorial For Better Screen Control

David E. Warnick
RD#2 Box 2484
Spring Grove, Pa. 17362

A computer is only a tool. Just as a saw is a tool designed to cut wood, a computer is designed for certain tasks. And, just as learning to use the saw correctly permits us to get the most from it, so it is with the computer. I've begun a series of articles on different areas of operation and programming which presented hurdles for me, and on which I get the most questions when discussing these fantastic machines. Each article is written to be understood by the newest of owners, yet hopefully has some real meat in it for those with lots of experience.

My first effort was the article on the special function keys in issue 28. This time I'll try to shed some light on screen control. I've included some sample programs which will demonstrate what we're trying to accomplish, and which should lead you in the right direction when you apply this information to your own programs. I'll be writing in MBASIC under the CP/M operating system, so will not use CLEAR or WIDTH statements. If your version of BASIC requires these, please add them, your BASIC manual will make it clear if you must. Should you have any problem with any program in this article, feel free to write me, but please include a stamped envelope for the answer.

As always, when you read any kind of tutorial or do-it-yourself article read it the first time lightly. Then go back with your operating manual at your side and really dig in. Also, please take the time to try the examples and sample programs I've included. This way you'll get the most out of this material and it will sink in better and stay longer.

Ever notice how all those neat programs you buy and the ones advertised in the magazines lay out the screen neatly? When you want info, or when you play a game, there's no scrolling. The cursor goes to the right place, prints or erases what it is supposed to, and is ready for its next assignment. That is screen control. It's much easier than it looks. (Much, much easier than the operating manual lets on.) So, let's get started.

The terminal logic board controls the screen in our computers. In order for us to control the screen, we've got to give the correct instructions to the terminal logic board. Remember that the screen is controlled by a microprocessor separate from the one that does the computing function. That's one of the beauties of the Heath system. We give our instruction to this microprocessor the same way we tell it to type letters on the screen. In MBASIC we use the PRINT statement.

Now that we know how to send instructions to the screen, how do we know how to tell it what we want it to do. I told you we'd need our operating manual, and now is the time to unlock some of its secrets. Turn to the appendix. In my manual it's section 12. Look up the Heath ESCape Sequences. Everything that's listed on that table

can be done by you easily in MBASIC. Now lay this article aside for a moment and look over all the things we'll learn to do. You will probably want to read the section on Heath ESCape Sequences Defined. That will make all the functions clear.

Well, you came back. That stuff must have whetted your appetite for a go at programming. To perform any of the functions listed, all we have to do is send ESCape and the appropriate letter or letter and number combination with a PRINT statement. However, we can't just write the program line:

```
10 PRINT ESC H
```

and expect the cursor to go to its home position. To send ESC to the terminal we must send it as a single character. Time for another quick look at the appendix. This time look at the table of ASCII Characters. Look down the CHARACTERS column until you find ESC. Note that it is defined as the escape function. Note also that its decimal code is 27. An ASCII character can be sent by the CHR\$ function in MBASIC. To send ESCape we program the line:

```
10 PRINT CHR$(27)
```

All we have to do now is add the letter for the function we want and we're off and running. Let's set up our first sample program and see how it works. For our example I've chosen something that is quite visible. In our manual under "Modes of Operation" we see:

```
ESC p   Enter Reverse Video
ESC q   Exit Reverse Video
```

We'll begin our program with:

```
10 PRINT CHR$(27); 'Send ESC--Don't forget the semi-colon
20 PRINT "p"      'Reverse Video--Lower case "p"
30 PRINT "THIS LINE IS IN REVERSE VIDEO"
```

We've just sent a control sequence and used it to perform a practical function. Lines 10 and 20 told the terminal logic board that anything we print from now on is to be in reverse video. We're not finished though, and this is important. **Any Time You Change The Terminal Operating Conditions With Your Program, You Should RESET The Terminal To Its Original Conditions Before Ending.** In our case, that means we should exit reverse video and requires that we send the terminal an ESC q. Our program continues.

```
40 PRINT CHR$(27); 'Send ESC--Don't forget the semi-colon
50 PRINT "q"      'Exit Reverse Video--Lower case "q"
60 PRINT "THIS LINE IS IN NORMAL VIDEO"
70 END
```

Now take a break from this article. Turn on your computer and run your basic. Type in the 7-line program we just wrote and run it. Neat isn't it. If it worked as it should have, when you ran it, you got one line printed in reverse video and one line printed in regular video.

Before we go farther with the features we program, there are some shortcuts to learn. They'll be easier to see if we use them in the sim-

ple program we've already proven to work. Then when we get into bigger and better things we can incorporate them into our programs with confidence. First, we could write lines 10 and 20 as a single line like this.

```
10 PRINT CHR$(27);"p"
```

The same thing applies to lines 40 and 50. When we only have one or two ESCape functions that's a good way to do it. But what do we do if we want to use many functions or to enter and exit a mode several times in a program? If there's a shorter way to write the program then we'll save typing time and reduce the amount of memory required to run that program. The answer is to use string variables. Each function we want to use needs merely to be assigned a string name and when we want that function we can just PRINT the appropriate string.

We'll do that now. The program we're about to write will first assign ESCape as variable E\$. Then lines 20 and 30 will build the functions we want. Finally, we'll print the 26 letters of the alphabet alternating between reverse and regular video. This will require that we repeat our functions between each letter, and that we use semi-colons often so we stay on the same line. Our program looks like this:

```
10 E$=CHR$(27)           'Set E$ to ESCape
20 E1$=E$+"p"           'ESC p--Enter Reverse Video
30 E2$=E$+"q"           'ESC q--Exit Reverse Video
40 PRINT E1$;"A";       'A in Reverse Video
50 PRINT E2$;"B";       'B in Regular Video
60 REM From here on we'll put several functions
                        on a single line
70 PRINT E1$;"C";E2$;"D";E1$;"E";E2$;"F";
80 PRINT E1$;"G";E2$;"H";E1$;"I";E2$;"J";
90 PRINT E1$;"K";E2$;"L";E1$;"M";E2$;"N";
100 PRINT E1$;"O";E2$;"P";E1$;"Q";E2$;"R";
110 PRINT E1$;"S";E2$;"T";E1$;"U";E2$;"V";
120 PRINT E1$;"W";E2$;"X";E1$;"Y";E2$;"Z"
130 END
```

Now it's time to take another break, type this program, and run it. Be sure to get the semi-colons at the ends of lines 40 thru 110. They will keep your output on the same line. Notice that we could perform any function the ESCape codes permit by just assigning the right code to a string variable and printing that variable. Also, notice that we ended with E2\$ as our last function. That left our terminal in its original normal video condition.

Now it's time to get into a more involved program. We've seen a simple program with one function, and have expanded that to make several changes. Our final demonstration will involve several functions and will show some true screen control. After you've typed and run this one you'll be ready to experiment on your own and develop any screen you want. I'll expect you to look up these ESC sequences in your manual to really understand them, and will only explain in detail the shortcuts and special controls we'll use from here on.

This last program will:

- 1) Freeze the screen and erase everything
- 2) Turn the cursor off
- 3) Move the cursor to any spot we want and type there
- 4) Use Reverse Video
- 5) Use Graphics

Our first step must be to assign the required ESCape sequences to string variables so we can PRINT them when needed, so here goes.

| | |
|-------------------|---------------------------------|
| 10 E\$=CHR\$(27) | 'ESCape |
| 20 E1\$=E\$+"Y" | 'ESC Y--Direct Cursor Address |
| 30 E2\$=E\$+"E" | 'ESC E--Clear Display |
| 40 E3\$=E\$+"y5" | 'ESC y5--Cursor On |
| 50 E4\$=E\$+"x5" | 'ESC x5--Cursor Off |
| 60 E5\$=E\$+"[" | 'ESC [--Enter Hold Screen Mode |
| 70 E6\$=E\$+"\ " | 'ESC \ --Exit Hold Screen Mode |
| 80 E7\$=E\$+"F" | 'ESC F --Enter Graphics Mode |
| 90 E8\$=E\$+"G" | 'ESC G --Exit Graphics Mode |
| 100 E9\$=E\$+"p" | 'ESC p --Enter Reverse Video |
| 110 E10\$=E\$+"q" | 'ESC q --Exit Reverse Video |

Lines 10 through 110 complete all the ESCape functions we'll need to run our sample program. We could have included any of the functions listed in our operating manual and written a program to use them too. Two of the functions above require some additional explanation. The first is Direct Cursor Addressing, ESC Y. We've assigned it to E1\$ in our program.

This is the function which permits us to direct the cursor to any spot on the screen and write to or erase from that spot. Note that even though we may turn the cursor off, it is still there. We just can't see it till we turn it back on. To send the cursor where we want it, we must tell it the line number and the column number of the place we want it to go. To do this, we have to add 31 to each of the line and column numbers. If we want it to go to line 10 and column 20 we have to send ESC Y followed by line 10+31 and column 20+31. The numbers are decimal equivalents from the ASCII table so they are sent by the CHR\$(##) function of BASIC. Our instruction thus becomes:

```
PRINT E1$;CHR$(41);CHR$(51)
```

Looking at our table of ASCII characters, we can substitute the actual characters for the CHR\$(##) function. The character shown for #41 is '!' (exclamation point) and for #51 is ')' (close parenthesis). To write the line above in shorter terms, we could write:

```
PRINT E1$;"!)";
```

That statement or the longer one above when in a BASIC program would send the cursor to line 10 and column 20. The semi-colon at the end prevents the cursor from moving to a new line when the next PRINT statement is encountered. Now we can put the cursor where we want it and add a PRINT statement to type a message at that spot.

There is just one exception to the use of actual characters in the shortened form above. In BASIC we can't enclose a quote mark " in quotes. BASIC will become confused and think it is the close quote. The quote is character #34. Since 34-31=3, any time we want to send the cursor to line #3 or column #3, we'll have to use CHR\$(34), not a quote mark.

The second item which needs a few words of explanation is the use of Graphics. Referring to your operating manual, you can find that Heath provides 33 different graphic symbols. To use them we must put the terminal in the graphics mode and send an instruction to PRINT the particular symbol we want. Checking your manual in the chart of GRAPHIC SYMBOLS you will find that each symbol has three pieces of information next to it. These are KEY, OCTAL, and DECIMAL. We can PRINT the desired symbol by sending the decimal value in the CHR\$(##) form, or by sending the KEY character. Thus to put a solid block the size of a single character on the screen, either of the following lines of programming would work after we entered the graphics mode.

```
PRINT CHR$(98)
PRINT "b"
```

The number of symbols can be doubled to 66 by using reverse video. The important thing to remember is that you must tell the terminal what mode you want it to work in with ESCape sequences, do your work in that mode, and exit that mode when it is no longer needed.

The final part of this article will complete our demo program. The delays I've put in should give you time to follow the program in this magazine as it executes and learn more from it. Use lines 10 through 110 above and the following:

```

120 REM The next line will
130 REM 1)Turn off the cursor
140 REM 2)Enter hold screen mode (no scrolling)
150 REM 3)Erase the screen
160 PRINT E4$;E5$;E2$
170 REM Now we'll send the cursor to line 5,
180 REM column 10 and print AAA
190 PRINT E1$;"*");
200 PRINT "AAA"
210 FOR X=1 TO 5000:NEXT X      'Delay
220 REM Now show the same operation on line 6,
230 REM column 10 by adding AAA on the same line
240 REM as the position control characters.
250 PRINT E1$;"%JAAA"
260 FOR X=1 TO 5000:NEXT X      'Delay
270 REM Now let's go to line 12 column 4,
280 REM then enter Graphics Mode and print
290 REM some symbols
300 PRINT E1$;"*#";           'That's line 12 column 4
310 PRINT E7$;                'Enter Graphics Mode
320 PRINT "aaabbbwwxxxxyyy"   'Those are the graphic symbols
330 FOR X=1 TO 5000:NEXT X      'Delay
340 REM Now lets go to line 13 and do the
350 REM same thing with a single line of instructions
360 PRINT E1$;"*#";E7$;"aaabbbwwxxxxyyy"
370 FOR X=1 TO 5000:NEXT X      'Delay
380 REM Before we go farther, don't forget
390 REM to get out of the Graphics Mode
400 PRINT E8$                  'Exit Graphics Mode
410 REM We can also use Reverse Video
420 REM with graphics.
430 PRINT E1$;"*#";E7$;E9$;"aaabbbwwxxxxyyy"
440 REM Now remember to get out of
450 REM Graphics Mode and Reverse Video
460 PRINT E8$;F1$
470 FOR X=1 TO 5000:NEXT X      'Delay
480 REM Before we end, we must
490 REM 1)Turn the cursor back on
500 REM 2)Exit Hold Screen Mode
510 PRINT E3$;E6$
520 END

```

As you run this program, the delays should give you time to think about why it's working as it is. Go to chapter 12 of your operating manual and look up other ESCape sequences and send them too. Experiment. You've learned screen control and some short cuts today. Use them and grow with your computer.

Be sure to follow the "Letters to the Editor" in future issues of "RE-Mark". There is bound to be some feedback on this article from some very knowledgeable programmers and their hints will build on what you've learned. See you next article. ✱



A Scene from Y-WING FIGHTER

"EXPERIENCE THE FUN AND EXCITEMENT OF THESE FAST ACTION GRAPHIC GAMES!!"

★ ★ ★ NEW ★ ★ ★

ZEETLE-DEET

Become the good wizard, Algernon, and tunnel deep in the earth to save the villagers trapped by the evil wizard, Dante! Dodge demons, cast spells and find secret messages with your companion, Prince Jason as you search for Dante's stronghold and the captive Princess of Zeetle-Deet. Strategy and skill will send Dante and his demon horde back to the netherworld that spawned them. Then witness the touching reunion of Prince Jason and his Princess! For one or two players.... **\$19.50**

Y-WING II

If you thought Y-wing fighter was the best thing this side of an arcade, wait until you see this sequel! Each quest takes you to a different region in the undersea world of Tazgard. Armed with more powerful lasers, bombs and a matter zapper, you will encounter nearly a dozen different types of dangerous creatures on your 3000 mile journey to confront the cruel sea-giant, Stormslayer! Many have tried to find his hidden weakness but only the brave and cunning have succeeded! **\$21.50**

★ ★ ★ BEST SELLERS ★ ★ ★

EXTERMINATOR ... Protect the outpost on Yargon from an attack of Anthropods before they turn into giant carnivorous critters that will eat you alive! **\$19.50**

SPACE ODYSSEY I ... Experience the excitement of space travel with this three dimensional space flight simulation! **\$21.50**

Y-WING FIGHTER ... Giant birds, spiders and volcanoes are just some of the challenges you will meet on your way to the enemy's mountain hideout! **\$19.50**

GALACTIC WARRIOR ... Battle star cruisers and the Evil Space Station on your mission to save the galaxy! New two player version **\$19.50**

MISSILE CONTROL ... Defend your cities against guided missiles and bombers by launching missiles that can be set to explode anywhere on the screen! ... **\$17.50**

★ ★ ★ COMING SOON ★ ★ ★

EVRYDIET: A Nutrition and Diet Guide

WRITE FOR A FREE CATALOG!

AVAILABLE FOR: H/Z89, Z90, Z100, and H8/19 systems at most Heath and Zenith retailers or order direct from us. Specify HDOS or CP/M, Hard or Soft Sector. All games require 48K. Please add \$1.00 per order for shipping and handling. Ca. residents add sales tax. For information or questions, call (415) 321-2708. Thank you.

EURUWARE

DEPT. R3, P.O. Box 60802, Sunnyvale, Ca. 94088

AT HEATH AND ZENITH DATA SYSTEMS WE'RE TAKING QUALITY TO NEW DIMENSIONS

Our Engineers, Committed To Excellence, Insist On It!

If you're looking for a sophisticated technical environment with high visibility and a tradition of engineering excellence, look to Heath and Zenith Data Systems.



Heath Company is a recognized leader in the design, production and sale of over 400 electronic kits and manufacturer of Zenith Data Systems' line of computer products. As a result of the recent introduction of our new ET-18 Robot Trainer and expansion of our instrumentation product line, we are preparing for growth in 1983 and beyond.

Innovative technological developments and a heritage of excellence in engineering have established Zenith Data Systems as a leader in the design of microcomputer hardware and software. It's these same qualities that led to the creation of our new Z-100 series of desktop computers and the ZT-1 Personal Terminal.

Discover Zenith's quality for yourself. Find out what a difference hands-on control, high visibility and excellence in engineering can mean to your career with these opportunities:

Computer Design

BS/MSEE and 3+ years design experience using 8-bit and 16-bit technology.

Software Design

BS/MSCS and experience enhancing operating systems for microcomputer products.

Diagnostics Software

BSME and one year experience developing diagnostics for digital electronics products.

Staff Engineering

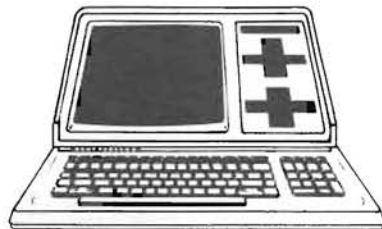
BSEE and 5 to 8 years experience as technical specialist designing analog and digital products and interfaces.

Project Engineering

BSEE and 3 to 5 years experience in analog and digital circuitry design. Project management or test instrument design experience desirable.

While experiencing a satisfying professional career, you will also enjoy the affordable lifestyles of St. Joseph, Michigan. Only 90 minutes from Chicago, St. Joseph is a picturesque community on the shores of Lake Michigan. In addition we offer competitive salaries and excellent benefits including relocation assistance and the opportunity to work on your Master's Degree through an on-site University extension program.

For immediate consideration, call collect: Roger Fitzwater, 616/982-3504, or send your resume to Zenith Data Systems, Dept. REM, Hilltop Road, St. Joseph, Michigan 49085.



Equal Opportunity Employer M/F/H



H/Z-89 Memory Replacement with 64K Chips

J. D. Ross
 Ross Custom Electronics
 1307 Darlene Way — Suite A12
 Boulder City, NE 98005

Many REMark readers may have read with interest Pat Swayne's article on the H/Z-89 4MHz modification in Issue 34, but were not thrilled with the prospect of replacing all the RAM with 150ns parts. With 64K chips now readily available, and the price dropping rapidly, it seemed reasonable to use 8 64K chips as a replacement for the 32 16K chips. It requires the addition of one chip to decode address bit A14 and A15. The schematic is shown in figure 1. This chip is easily mounted on the same PC board that contains the 3 chips for the 4 MHz modification.

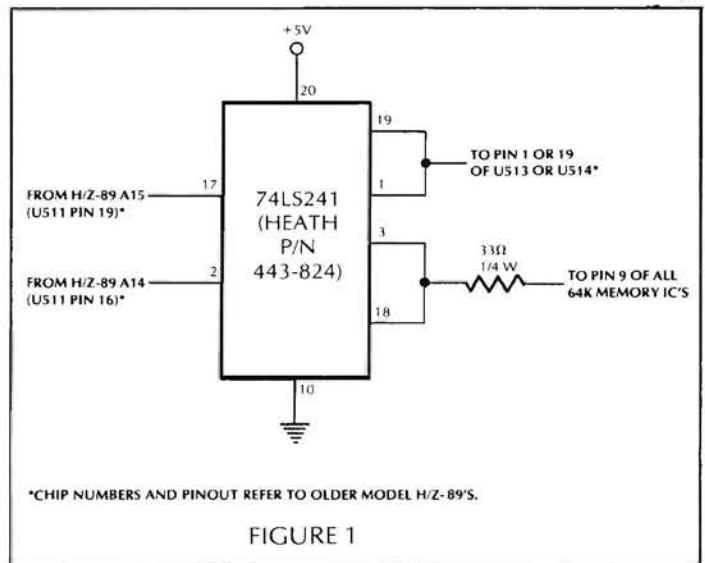
The pinout of the 4164 dynamic RAM is very close to that of the 4116. However, several pins, including the power supply pins, are different, and plugging them into the H/Z-89 sockets for the 4116 will cause irreversible damage. The easiest way to mount them is as follows:

1. Disconnect all power to the unit. Disconnect all power connections and peripheral boards from the CPU board. Remove the CPU board from the unit.
2. Remove all 16K memory chips from the system, and unplug the 64K memory expansion module (if installed).
3. Use 8 standard or low profile solder leg IC sockets (16 pin).
4. Bend pins 1, 8 and 9 straight out to the side on all sockets.
5. Plug the sockets into any one of the three rows available for memory chips on the H/Z-89 CPU board. Pins 1, 8 or 9 must not seat in the H/Z-89 sockets.
6. Solder all pin 8's of the new sockets together and connect this to +5 volts. (P509 pin 23 is a good point to pick up the +5V).
7. Solder all pin 9's of the new sockets together and connect this buss to the point indicated on Figure 1.
8. On the back of the H/Z-89 CPU board, solder a 3 point jumper from U549 pin 4 to U541 pin 4 to U533 pin 4. If you have installed the zero origin memory decoding PROM (Heath p/n 444-66) then continue the jumper on to P509 pin 19. These chip numbers are for the older H/Z-89's, and may be different on the newer versions. What is being done here is bussing together the CAS (column address strobe) signal for each 16K bank of RAM.
9. As indicated on Figure 1, connect pin 17 of the new IC to A15 of the H/Z-89 address buss. On older H/Z-89's A15 is available at U511 pin 19. It may be different on newer machines.
10. Also as indicated on Figure 1, connect pin 2 of the new IC to A14 of the H/Z-89 address buss. On older H/Z-89's A14 is available at U511 pin 16. It may be different on newer machines.
11. As indicated on Figure 1, tie pins 1 and 19 of the new IC together, and connect them to pin 1 or 19 of either of the H/Z-89's address strobe IC's. These are U513 and U514 on older 89's.
12. Don't forget to connect pin 10 of the new IC to ground, and pin 20 to +5 volts.
13. Set jumpers JJ501 and JJ502 to reflect the proper amount of memory in your system. This will usually be 64K (i.e. both

jumpers "1"), but must be set to 48K if the zero origin memory decoding PROM is not installed. You may wish to set the jumpers for 16K (both jumpers "0") to test the new memory initially.

14. Now plug the new memory chips into the 8 sockets just installed and re-install the CPU board and peripheral boards.

Testing of the memory should be done first in the 2MHz mode, by using a "GO 7375" from the H/Z-89 monitor. Then it should be tested at 4MHz according to Pat Swayne's procedure in REMark 34.



SDUP PATCH

There is a "bug" in the SDUP program on HUG disk 885-1121. It can be corrected using PATCH as shown below:

```
>PATCH
PATCH Issue #50.06.00
File Name? SDUP.A65
Address? 047364
047364 = 016/006
047365 = 120/^D      (Control-D)
Address? ^D

PATCH ISSUE #50.06.00
File Name? ^D

If you want to patch the source, locate the line containing:

      MVI      C, MAXTRKD
and change it to:

      MVI      B, MAXTRKD
```



Introduction To Z-BASIC Part V

Gerry Kabelman, C.E.T.
Zenith Data Systems

This is the fifth article in a series of articles dealing with the new commands of the Z-100's Z-BASIC over BASIC- 80. Previous articles dealt with the CLS, LINE, LOCATE, KEY, PSET, GET/PUT, PAINT and DRAW commands, plus using the 25th line. This month we will look at the CIRCLE command and try some additional uses for the GET and PUT commands to create objects on the screen.

The CIRCLE command is a very powerful command and has a very complex looking syntax.

CIRCLE(X,Y),R,C,S,E,A

The X and Y are the coordinates where the center of the circle will be. The R is the radius of the circle and must be a positive number. The X, Y, and R are all required when using the CIRCLE command. The other variables are not required, however, the other variables add the real power to the CIRCLE command.

EXAMPLE: CIRCLE(300,100),50

The C variable is the color or attribute of the circle to be drawn. If the color variable is left out, as in the above example, the foreground color will be used for the color of the circle.

EXAMPLE: CIRCLE(300,100),50,2

The above example will draw a circle at location 300 dots from the left edge of the screen and 100 dots down from the top of the screen. The size will be 100 dots wide (50 is the radius) and 45 dots high. The height is only 45 because of the screen aspect ratio is 7/16 not 1/1.

The S and E are the starting and ending locations of the circle. That is to say that the circle does not have to be a complete circle. When using the starting location only, the ending location is assumed zero (0) unless told different. The starting and ending locations may be any positive number between 0 and 2 X PI (6.28318 approximately).

EXAMPLE: CIRCLE(300,100),50,2,3.14159

The above example draws the lower half of the previously drawn circle. By adding an ending location to the CIRCLE command we could have drawn only the lower left quarter of the circle.

EXAMPLE: CIRCLE(300,100),50,2,3.14159,4.712385

When using the value of PI (3.14159) it is best to set a variable to the value of PI and then use that variable in the CIRCLE command.

EXAMPLE: PI=3.14159:CIRCLE(300,100),50,2,PI,PI*1.5

The 'A' is the aspect ratio which may be any positive number, however, the larger the number the less the width of circle or ellipse. The normal aspect ratio is .4375 when using a standard monitor. Try different aspect ratios from .01 to 10.

EXAMPLE: FOR I=.01 TO 10:CIRCLE(300,100),50,2,,,I:NEXT I

Or Try:

```
10 FOR C=1 TO 7
20 FOR I=.01 TO 10 STEP .1
30 CIRCLE(300,100),50,C,,,I
40 NEXT I,C
```

Using variables within the CIRCLE command allows many different patterns to be created using limited code.

EXAMPLE: CLS:FOR X=1 TO 20:CIRCLE(X*50,X*15),20,2:NEXT X

Try playing with the CIRCLE command, using different variables for all the different parts to the command and see how many unusual shapes or designs can be created.

Let's take last month's code, used to create stars and change the way the PUT command is used and see what happens to the stars.

```
10 CLS: STARS.BAS Version 01.21.83 GK:
:
20 C6=6:PSET(26,3),C6: Create Star
30 DRAW"F3R3G3D3H3G3U3H3R3E3
B40 PAINT(26,4),C6
50 DIM A$(100):GET(20,3)-(32,12),A#
:
60 CLS:LOCATE 25,16:PRINT"Press any key to stop"
:
70 R=35:FOR I=1 TO 13: 13 Stars
80 C=(I*(360/13))*3.14159/180
90 PUT(135+(INT(COS(C)*R))*2,63+INT(SIN(C)*R)),A#
100 NEXT I
110 A$=INKEY$:IF A$="" THEN 70 ELSE CLS:LIST
```

And add this line:

```
68 LINE(40,20)-(250,115),1,BF: Blue Field
:
```

Be sure to follow the line numbers as they are shown or your program will not work properly when run.

If only one bank of video memory has been installed use white (7) for the color of the square and don't fill the box (B ONLY).

The stars should now display on a blue background and appear white instead of yellow. (If only one color bank, a square with stars inside will appear.)

Try adding the following action verbs to the end of line 90. Be sure to add a comma to the line before adding an action verb.

Action verbs: PRESET, AND, OR, XOR and PSET

To fully understand the action verbs, add another line to our program that is identical to line 90 at line 95 except it will have an action verb at the end of the line.

```
95 PUT(135+(INT(COS(C)*R))*2,63+INT(SIN(C)*R)),A#[,Action Verb]
```

Try using all the action verbs in the end of line 95 and note how the background around the star will change as the different action verbs are used. By using the XOR action verb the star appears to move around in a circle. The XOR is used for motion in games.

Now let's use the CIRCLE command and add two additional lines to the code:

```
61 CIRCLE(25,7),15,6:PAINT(30,9),6: Top Of Pole
62 LINE(20,15)-(30,225),6,BF: Pole
```


With addition of these two simple lines we have added a new dimension to the display on the screen. The above two lines will add the flag pole to the left side of the display with a round ball at the top of the pole.

After running the program above add the following line:

```
63 PSET(30,15),7:DRAW"M39,20D155M37,233":' Rope pe
:
```

This line adds a rope to keep the blue flag with stars on it from flying away. Line 63 used the PSET command to set the dot pointer to the location 30,15 and turn that location to white. The DRAW command draws a line from the present dot pointer to the location 39,20, then draws a line straight down from there 155 dots and continues drawing to location 37,233.

Here is the complete listing as modified by adding the above lines and making the required changes.

```
10 CLS:' STARS.BAS Version 02.24.83 GK:
:
20 C6=6:PSET(26,3),C6:' Create Star
30 DRAW"F3R3G3D3H3G3U3H3R3E3
40 PAINT(26,4),C6
50 DIM A$(100):GET(20,3)-(32,12),A#
:
60 CLS:LOCATE 25,16:PRINT"Press any key to stop"
:
61 CIRCLE(25,7),15,6:PAINT(30,9),6:' Top Of Pole
62 LINE(20,15)-(30,225),6,BF:' Pole
:
68 LINE(40,20)-(250,115),1,BF:' Blue Field
:
70 R=35:FOR I=1 TO 13:' 13 Stars
80 C=(I*(360/13))*3.14159/180
90 PUT(135+(INT(COS(C)*R))*2,63+INT(SIN(C)*R)),A#
100 NEXT I
:
110 A$=INKEY$:IF A$="" THEN 70 ELSE CLS:LIST
```

Next month we will finish the flag, but in the meantime try finishing it yourself. Hint: It should take only three lines of code using the LINE command in a FOR-LOOP loop.

✱

H8 3MHz Bugs

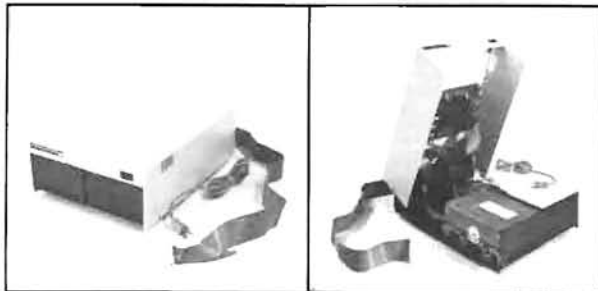
The following bugs have shown up in the 3MHz H8 modification article in REMark #38, page 38. In the paragraph just before the heading "Testing at 3MHz", change "and a 74S132 at U32" to "and a 74S132 at U34". Also, make the following additional connection on the CPU board. Connect pins 1 and 2 of U23 by bridging the pins with a 1/8" (.3 cm) piece of wire on the foil side of the board.

Join with other
HUGgies

at the

1983 NATIONAL

HUG CONFERENCE II



NOW 12 MEGABYTE (CDR-10M) \$3195

and 6 MEGABYTE (CDR-5M) \$2495

WINCHESTER SYSTEM

For the Heath/Zenith Computer

Systems complete with software case, power supply & signal cable.

Runs with CP/M, on the H/Z89, Z90 & H8 (with Z80 card).

- Switching power supply
- Expansion for backup installations
- Auto attach BIOS
- Hard disk utilities
- Formatting program
- 1 year parts & workmanship warranty

CP/M is a trademark of Digital Research. Heath, H8, H89 are trademarks of Heath Corporation. Zenith, Z89, Z90 are trademarks of Zenith Data Systems.

5-20 day delivery-pay by check, C.O.D., Visa, or M/C.

Contact:



C. D. R. Systems Inc.
7210 Clairemont Mesa Blvd.
San Diego, CA 92111
Tel. (714) 560-1272

23 PROGRAMS FOR HEATH/ZENITH COMPUTERS

Text processing Languages Utilities Games

ZENCALC—Spreadsheet calculator tailored to

Heath/Zenith display.

SPELL—50,000 word proofreader

C/80—Full-featured C compiler

PIE—Editor makes file changes easy

MUNCHKIN—Action arcade game

REACH—Access remote timesharing

MYCHESS—Play championship chess

ED-A-SKETCH—Create graphics displays

SPOOL-N-GO—Continue working while printing

Intro to BASIC—Learn BASIC by computer

ADVENTURE—Dare Colossal Cave

TEXT—Format your documents

LISP—Artificial intelligence language

ELIZA—Hilarious pseudo-therapy

RATFOR—structured FORTRAN

INVADERS—Video game

SPACE PIRATES—Space battle game

Catalog/Utilities—Helps track, process your files

SUPER ZAP—Disk dump and patch

UVMAC—Macro assemblers for Z80, 8080

PACK/CRYPT—Compress files, protect data

AIRPORT—Air Traffic Controller game

FREE CATALOG



Available from your local Heath/Zenith dealer or:

The Software Toolworks™



14478 Glorietta Drive,
Sherman Oaks, CA 91423
(213) 986-4885



Call or write for your free Software Toolworks catalog.
Dealer inquiries invited.



NEW HUG PRODUCTS

P/N 885-1228[-37]

CP/M Fast Action Games \$20.00

Introduction: This disk contains disks that will test the users reaction speed. The games are modifications to games already available in HDOS.

Requirements: This disk requires the CP/M operating system version 2.0 or later on an H19/H8/H17 or H/Z89, 90 or H/Z100 with at least 48K of memory.

All the programs are in executable form and do not need any language to run the programs. The source codes for all but one program are included. Refer to the documentation for details on the languages used.

Note: *The H/Z19 terminal graphics are used.*

The following programs are included on the HUG P/N 885-1228[-37] Action Games disk:

| | |
|---------|------|
| README | .DOC |
| BREAK19 | .COM |
| BREAK19 | .PAS |
| SNAKE | .COM |
| SNAKE | .ASM |
| BOB | .SNK |
| ACKACK | .COM |
| ACKACK | .ASM |
| SKI | .COM |

Authors:

BREAK19 - Lyle E. Wilkinson
 SNAKE - Michael Paabo
 ACKACK - Frank Bielsik II
 SKI - P. J. O'Connor

BREAK19 — This is a variation of the "Breakout" TV game. The player is served a "ball", which he must hit with a "paddle". The ball strikes a barrier, breaking out part of it, then it returns for the player to hit again. The object of the game is to break out as many blocks in the barriers before all the balls are used up.

This program was written in HUG Tiny Pascal (885-1085), and translated into SuperSoft Tiny Pascal for the CP/M version (P.O. Box 1628, Champaign, IL 61820). SuperSoft Tiny Pascal is not needed unless the player wants to make changes and re-compile the program.

SNAKE — In this game, the player controls the movements of a "snake" as it crawls around the screen. The player guides the snake to a block of "food" without hitting the barrier wall or looping back on the growing snake. Each time the food is hit, it jumps to a new

place on the screen. The object of the game is to hit the food as many time as possible.

This version of SNAKE has the ability to save games which have been played on a disk for re-loading and reviewing at a later date. The file BOB.SNK is a game that was actually played by HUG manager Bob Ellerton.

ACKACK — This is an anti-aircraft artillery game. The object is to shoot down the airplanes before they bomb the players gun emplacements. Additional guns are awarded for each hundred points of score achieved.

SKI — This game simulates a slalom ski race. The object of the game is to stay within the slalom course.

Comments: No comments.

P/N 885-3005-37

ZDOS ETCHDUMP \$20.00

Introduction: The ETCHDUMP programs, Etch-a-Sketch and Screen Dump, are paired into a complete graphics package for the Z-100 (ZDOS) and the MX-80. The ETCH program facilitates the quick and easy drawing of all types of complex designs on the screen of the Z-100, which can then be saved to a disk file. The SCREEN program reads a disk file created by the ETCH program and transfers the contents to an MX-80 printer, with GrafTrax, allowing very precise and detailed hardcopy records.

Requirements: ETCHDUMP require the ZDOS operating system for the H/Z-100 computers. Etch-a-Sketch is written in ZBASIC, while Screen Dump is written in assembly language. Only one drive is required.

The following files are contained on the HUG P/N 885-3005-37 ZDOS ETCHDUMP programs:

| | |
|----------|------|
| README | .DOC |
| ETCHDUMP | .DOC |
| ETCH | .BAS |
| SCREEN | .COM |
| SCREEN | .ASM |
| CONVERT | .BAS |
| READ | .BAS |
| WRITE | .BAS |
| PIC | .SCN |

Authors:

ETCH, CONVERT, READ, WRITE - Frank T. Clark
 SCREEN - Scott Cutshall

ETCH — With simple one-letter commands the ETCH program uses the graphics of ZBASIC to construct and combine detailed drawings. The screen contents of any video plane can be saved to a file and restored for later editing or for printing with the SCREEN program.

The program has an alphanumeric and graphics mode and three types of commands. The following is a list of the mode commands:

Graphic Commands -

NUMERIC KEYPAD - entering a number moves the cursor one pixel in one of the 8 possible directions relative to the "5" key as the center.

A(lphanumeric) - enter the alphanumeric mode.

B(ox) - draw a box

C(ircle) - draw a circle

D(ot) - place a dot at the current position

E(llipse) - draw an ellipse

F(ill) - fill an area

H(idden) - toggle hidden cursor mode

I(ndex) - change the horizontal motion index (pitch)

K(olor) - change the default foreground and background color

L(ine) - draw a line

M(ark) - remember the current cursor position for subsequent commands

P(oint) - remember the current cursor position for subsequent commands

Q(uit) - clear the screen and quit

R(ead) - read the contents of a file into a video plane

S(ize) - input the aspect ratio for drawing circles and ellipses

T(rack) - toggle tracking mode

W(rite) - write the contents of a video plane into a file

X(Y-addressing) - input a specific x-y coordinate for new cursor location

Alphanumeric mode -

LINE FEED - return to the graphics mode

RETURN - move the cursor to the beginning of the next row

DEL or **BS** - erase the previous character

CONVERT, READ and WRITE — These three subroutines may be merged into a ZBASIC program for working with a screen file created with ETCH.

PIC.SRN — an example screen file created with ETCH.

SCREEN — The SCREEN program can take a file image of a video plane and print it to the MX-80 printer using GrafTrax with very high resolution matching that of the Z-100 monitor. A psuedo gray scale even simulates the color monitor or monochrome gray scale.

The command line can contain up to three file names which will be merged into one picture. The three files can be either separate video planes or the same file to make a darker picture.

Command line arguments or switches allow the user to select a border to appear around the picture plus the type of border. Another option allows for multiple drawings to be combined on a single page.

The aspect ratio for the MX-80 is different than that of the monitor. A circle that appears visually correct on the monitor will appear oblong on the printer. Changing the aspect ratio will produce circles that appear visually correct on the MX-80 but oblong on the monitor.

Comments: This ETCHDUMP utility can be a versatile program even if the user does not have an MX-80 printer.

HUG Price List

The following HUG Price List contains a list of all products not included in the HUG Software Catalog. For a detailed abstract of these products refer to the issue of REMark specified.

| Part Number | Description of Product | Selling Price | REMark Issue |
|----------------------|--------------------------------------|---------------|--------------|
| HDOS | | | |
| 885-1121 | Hard Sectored Support Package | \$ 30.00 | 37 |
| 885-1122 | MicroNET Connection | \$ 16.00 | 37 |
| CP/M | | | |
| 885-1211 [-37] | Sea Battle | \$ 20.00 | 36 |
| 885-1222 [-37] | Adventure | \$ 10.00 | 36 |
| 885-1223 [-37] | HRUN HDOS Emulator | \$ 40.00 | 37 |
| 885-1224 [-37] | MicroNET Connection | \$ 16.00 | 37 |
| 885-1225 [-37] | Disk Dump and Edit Utility (DDEU) .. | \$ 30.00 | 38 |
| 885-1226 [-37] | CP/M Utilities by PS: | \$ 20.00 | 38 |
| 885-1227 [-37] | CP/M Cassino Graphic Games | \$ 20.00 | 38 |
| 885-3003 [-37] | ZTERM Modem Package | \$ 20.00 | 36 |
| 885-8012 [-37] | Modem Appl. Effector (MAPLE) | \$ 35.00 | 36 |
| ZDOS | | | |
| 885-3004-37 | ZBASIC Graphic Games Disk | \$ 20.00 | 37 |
| MISCELLANEOUS | | | |
| 885-0004 | HUG 3-Ring Binder | \$ 5.75 | |
| 885-4001 | REMark VOLUME 1, issues 1-13 ... | \$ 20.00 | |
| 885-4002 | REMark VOLUME 2, issues 14-23 .. | \$ 20.00 | |
| 885-4003 | REMark VOLUME 3, issues 24-35 .. | \$ 20.00 | |

NOTE: The [-37] means the product is available in hard-sectored or soft-sectored. Remember, when ordering the soft-sectored format, you must include the "-37" after the part number; e.g. 885-1223-37.

Attention Users of 885-1118 & 885-1218 HUG Payroll Packages.

OOPS!!

PAYROLL for both HDOS and CP/M have a bug in the PAYCAL.BAS programs. The bug is that when an employee's gross amount reaches or exceeds the maximum for FICA tax the program does not deduct the last FICA payment. A two line fix will correct the problem in the PAYCAL.BAS program.

```
470 IF CVD(Y$(1))>T2
      THEN P3=0:GOTO 480:' Max FICA
475 IF P1+CVD(Y$(1))=>T2
      THEN P3=((CVD(Y$(1))+P1)-T2)*T1:
      P3=INT(P3*100+.5)/100
```

The line 470 is the same as the current line except that the **P1** + is removed after the **IF** command. The entire line 475 is a new line. The above lines apply to both the HDOS and CP/M versions.

An Educators



PILOT/80 ©Copyright 1983, Kurt Albercht

The Background of PILOT

PILOT (Programmed Inquiry, Learning, Or Teaching) was the first computer language dedicated to Computer-Aided Instruction (CAI). Developed in 1969 by John Starkweather, it has been implemented on mainframes down to micros. This interactive language enables a person with very little previous computer experience to construct and run simple computer programs. Because of its simple structure and syntax, it has proven to be an excellent starter language for the novice programmer. The average person can be writing functional programs in about an hour after learning the language.

Using PILOT, a teacher can present a student with a reading passage, give him time to study it, and then ask him multiple-choice, fill-in-the-blank questions, or true-false questions. The program can include responses keyed to the student's answer, and thus it can give comment or advice to the student. It can introduce a mathematical problem and offer the solution on a step-by-step basis or give the student an opportunity to discover as many of the steps as he can, with hints from the computer if needed. However, PILOT is a word and letter language, not a number cruncher.

A Few Notes About PILOT/80

PILOT/80 is a full implementation of the PILOT language. PILOT/80 uses line numbers to identify individual program lines. Not all versions of PILOT do however. Line numbers are used for editing purposes only and are never used to branch within a program. They also painlessly prepare the novice programmer for languages such as BASIC.

Some versions of PILOT require the programmer to write his/her program with a text editor. But PILOT/80 contains its own editor, as well as disk handling functions, and full error reporting capabilities that explain in plain English what the user did wrong.

A text editor may be used to write PILOT programs. But remember, the PILOT/80 editor will catch most all errors as the program is

```

10 CLEAR 10000:WIDTH 80
20 PRINT CHR$(27)"PILOT/80 Version 83.03.06"
30 PRINT "By Kurt Albercht Copyright (c) 1983"
40 REM
50 REM *** Initialization ***
51 REM
60 ML%=100:MV%=10:MS%=10:MIX%=70:DIM P$(ML%), V$(MV%,2), SZ(MS%)
70 ER$="Error - ":ON ERROR GOTO 490
80 REM
90 REM *** Command Level ***
91 REM
100 EF%=0:PRINT CHR$(27)"q"CHR$(27)"G":PRINT "Command? ";:G
  GOSUB 170:IF B$="" THEN 100
110 IF B$="NEW" THEN 180 ELSE @
  IF LEFT$(B$,4)="EDIT" THEN 620 ELSE @
  IF LEFT$(B$,6)="INSERT" THEN 660 ELSE @
  IF LEFT$(B$,6)="DELETE" THEN 720 ELSE @
  IF LEFT$(B$,4)="LIST" THEN 780 @
120 IF LEFT$(B$,4)="SAVE" THEN 830 ELSE @
  IF LEFT$(B$,4)="LOAD" THEN 880 ELSE @
  IF LEFT$(B$,3)="RUN" THEN 940 ELSE @
  IF B$="LLIST" THEN B$="SAVELP":GOTO 830 ELSE @
  IF B$="BYE" THEN 140
130 PRINT ER$"I don't recognize that command."
131 PRINT "Perhaps you made a typing mistake.":GOTO 100
140 PRINT "Are you sure you want to return to HDOS? (Y/N)":@
  C$=INPUT$(1):IF C$="N" OR C$="n" THEN 100 @
  ELSE IF C$(">")Y" AND C$(">")y" THEN 140 ELSE PRINT "bye!":SYSTEM
150 REM
160 REM *** String Input ***
161 REM
170 A$="":B$=""
180 C$=INPUT$(1):IF A$="" AND (C$=" " OR C$=CHR$(9)) THEN 180
190 IF C$=CHR$(13) THEN PRINT:GOTO 270
200 IF C$=CHR$(8) THEN IF LEN(A$)=0 THEN 180 @
  ELSE A$=LEFT$(A$,LEN(A$)-1):B$=LEFT$(B$,LEN(B$)-1):@
  PRINT C$ " "C$:GOTO 180
210 IF C$=CHR$(9) THEN IF LEN(A$)+8>MIX THEN 180 @
  ELSE A$=A$+SPACE$(8):B$=B$+SPACE$(8):PRINT SPC(8):GOTO 180
220 IF LEN(A$)=MIX OR C$=CHR$(164) OR C$=CHR$(127) THEN 180
230 A$=A$+C$:PRINT C$:IF C$="a" AND C$("<")z" THEN C$=CHR$(ASC(C$)-32)
240 B$=B$+C$:GOTO 180
250 REM
260 REM *** Remove Spaces ***
261 REM
270 I%=INSTR(B$," "):IF I%=0 THEN RETURN @
  ELSE B$=LEFT$(B$,I%-1)+RIGHT$(B$,LEN(B$)-I%):GOTO 270
280 REM

```

Best Friend Programmed Inquiry, Learning Or Teaching

Kurt Albrecht
1061 Boot Rd.
Downingtown, PA 19335

```

290 REM *** Input Program Line ***
291 REM
300 IF LN%>ML% THEN @
    PRINT ER$"You have entered the maximum number of "ML%"program lines.":@
    EF%=1:RETURN
310 PRINT LN%;:GOSUB 170:C%=LEFT$(B$,1)
320 IF B$="DONE" THEN EF%=1:RETURN
330 IF B$="" OR C$="*" THEN A$=B$:RETURN @
    ELSE IF C$=":" THEN A$=C$+RIGHT$(A$,LEN(A$)-1):RETURN
340 IF INSTR("TAIM,UESR",C$)=0 THEN @
    PRINT ER$"I don't know the program command in line"LN%":GOTO 310
350 IF MID$(B$,2,1)<>":" AND MID$(B$,2,2)<>"Y:" AND MID$(B$,2,2)<>"N:" THEN @
    PRINT ER$"The program command in line"LN%"must be followed by either @
    ;, Y:, or N:":GOTO 310
360 IF C$="T" OR C$="R" THEN A$=LEFT$(B$,2)+RIGHT$(A$,LEN(A$)-2) ELSE A$=B$
370 RETURN
380 REM
390 REM *** Check Line Number Validity ***
391 REM
400 IF LL%=0 THEN PRINT ER$"There isn't a program presently in memory.":@
    EF%=1:RETURN
410 FOR I%=1 TO LEN(B$):IF INSTR("0123456789-",MID$(B$,I%,1)) THEN NEXT @
    ELSE PRINT ER$"A line number from 1 to"ML%"must follow that command.":@
    EF%=1:FOR J%=1 TO I%:NEXT:RETURN
420 BL%=VAL(B$):LN%=BL%:GOSUB 450:IF EF%=1 THEN RETURN
430 IF INSTR(B$,"-")=0 THEN EL%=0 ELSE IF RIGHT$(B$,1)="-" THEN EL%=LL% @
    ELSE EL%=VAL(RIGHT$(B$,LEN(B$)-INSTR(B$,"-"))):LN%=EL%:GOSUB 450:@
    IF EF%=1 THEN RETURN ELSE IF BL%>EL% @
    THEN PRINT ER$"The line numbers must be in ascending order.":@EF%=1
440 RETURN
450 IF LN%<1 OR LN%>ML% THEN PRINT ER$"A line number from 1 to"ML%"must @
    follow that command.":@EF%=1:RETURN ELSE IF LN%>LL% AND EF%=0 THEN @
    PRINT ER$"Line"LN%"does not exist.":@EF%=1:RETURN ELSE EF%=0:RETURN
460 REM
470 REM *** Error Trapping ***
471 REM
480 IF ERR=6 THEN PRINT ER$"Numbers must be in the range of -32767 to 32767.":@
    RESUME 100
490 IF ERR=65 THEN PRINT ER$"A file name must follow this command. Basically, @
    a file name may not"
500 IF ERR=65 THEN PRINT TAB(8) "exceed 8 characters in length, begin with a @
    number, or contain special":PRINT TAB(8) "characters. See the PILOT/80 @
    documentation for a more detailed":PRINT TAB(8) "explanation of file @
    names.":RESUME 100
510 IF ERR=53 THEN PRINT ER$B$" cannot be found on the indicated disk drive.":@
    PRINT TAB(8) "Check your spelling, make sure you used the correct @
    extension, and that":PRINT TAB(8) "you indicated the correct disk drive.":@
    RESUME 100

```

being typed in. If you use a text editor, do not use line numbers (PILOT/80 automatically numbers a program upon loading), type all program lines in uppercase except those with the command T: or R:, and use the filename extension .PIL.

The Documentation

The PILOT/80 documentation is divided into three sections; Introduction To Computer Programming, PILOT/80 Commands, and Implementation Notes. If you're already familiar with the art of computer programming proceed directly to PILOT/80 Commands. However, if you're a computer neophyte you should definitely start with 'Introduction To Computer Programming'. Only a basic knowledge of the computer's parts and a desire to learn is required. So without further interruption lets learn PILOT!

Computer Introduction

What is a Program Any Way???

First off we should define what a computer program is. My dictionary defines a program as,

"A list of instructions given in a certain order so as to produce a desired result."

You're probably saying, "What the hex is that supposed to mean!" Well lets take a closer look at it and see if we can figure out what the term program means. Now, "A list of instructions..." is pretty easy, and "...given in a certain order..." just means that those instructions are in a particular order, not just randomly put together. "...so as to produce a desired result." is a lot of words used to simply mean, what you, the programmer, want the program to accomplish.

Now that we've defined what a program is lets take a look at one. It may surprise you, but it takes a lot of instructions to produce even the simplest results. For example, suppose we wanted to list all the instructions needed for a baby to place its thumb in its mouth. They would probably look something like this:

1. Choose the left or right hand randomly or by preference.
2. Locate the desired hand visually.

3. Extend the thumb of the desired hand.
4. Bend the arm of the desired hand inward at the elbow joint.
5. Align the extended thumb with the approximate location of the mouth.
6. Open the mouth
7. Bend the the arm of the desired hand until the thumb of the desired hand enters the mouth.
8. Close the mouth.

As you can see, it takes a lot of instructions given in a certain order to produce the desired result.

Speaking a Common Language

Computer programs rely on a language that both the programmer and computer will understand. This language is composed of certain words and characters that are interpreted by the language so that the programmer and the computer can communicate. Thus, the reason why some computer languages are referred to as interpreters. If the programmer uses a word that the computer doesn't recognize, an error will result.

The words and characters used vary among the many computer languages. For example, a person who knows only the English language will be able to read and understand only a book written in English. The same is true with computers. A computer that 'knows' only the PILOT language will ONLY be able to properly execute a program written in PILOT.

Fortunately though, most computers have the ability to use a wide variety of languages such as BASIC, Pascal, and FORTRAN. These multi-lingual computers have a definite advantage over those that 'speak' only one language. Because there are a great number of programs written in many different languages that could be of value to the user, his/her computer should be able to 'speak' most of these languages. Otherwise, the number of programs available to the user is sharply diminished. This in turn severely lowers the value of the computer to the user.

Line Numbers

Most computer languages require the program instructions be written line by line, like the baby program above. PILOT/80 is no exception. In this case lines are numbered for easy reference. Not all versions of PILOT number the program lines. Because the program lines are numbered, they are usually referred to by their line numbers, the number associated with an individual program line.

Variables

Variables are something most people run into in high school algebra and then forget.

```

520 IF ERR=62 THEN PRINT ER$"There is no space available on the designated @
    disk drive.":CLOSE:RESUME 100
530 PRINT ER$"A Microsoft Basic error has occurred. See appendix A of the @
    MBasic":PRINT TAB(8) "manual for an explanation of error"ERR".";
540 IF LLZ>0 THEN PRINT " I strongly suggest that you":@
    PRINT TAB(8) "first LLI$ your program in case you are unable to recover @
    it. ";
550 PRINT:CLOSE:RESUME 100
560 REM
570 REM *** New ***
571 REM
580 ERASE P$:DIM P$(MLZ):LNZ=1:LLZ=0
590 GOSUB 300:IF EFZ=1 THEN 100 ELSE P$(LNZ)=A$:LLZ=LNZ:LNZ=LNZ+1:GOTO 590
600 REM
610 REM *** Edit ***
611 REM
620 B$=RIGHT$(B$,LEN(B$)-4):GOSUB 400:LNZ=BLZ:IF EFZ=1 THEN 100
630 PRINT LNZ:P$(LNZ):GOSUB 300:IF EFZ=1 THEN 100 ELSE P$(LNZ)=A$:GOTO 100
640 REM
650 REM *** Insert ***
651 REM
660 IF LLZ=MLZ THEN PRINT ER$"You have entered the maximum number of"MLZ"@
    program lines.":GOTO 100
670 B$=RIGHT$(B$,LEN(B$)-6):EFZ=1:GOSUB 400:LNZ=BLZ:IF EFZ=1 THEN 100 @
    ELSE IF LNZ>LLZ+1 THEN PRINT ER$"Line"LNZ"does not exist.":GOTO 100
680 GOSUB 300:IF EFZ=1 THEN 100
690 FOR IZ=LLZ TO LNZ STEP -1:P$(IZ+1)=P$(IZ):NEXT:P$(LNZ)=A$:LLZ=LLZ+1:@
    LNZ=LNZ+1:GOTO 680
700 REM
710 REM *** Delete ***
711 REM
720 B$=RIGHT$(B$,LEN(B$)-6):GOSUB 400:IF EFZ=1 THEN 100
730 IF BLZ=MLZ OR ELZ=LLZ THEN LLZ=BLZ-1:GOTO 100
740 IF ELZ-BLZ<1 THEN DFZ=1 ELSE DFZ=ELZ-BLZ+1
750 LLZ=LLZ-DFZ:FOR IZ=BLZ TO LLZ:P$(IZ)=P$(IZ+DFZ):NEXT:GOTO 100
760 REM
770 REM *** List ***
771 REM
780 IF B$="LIST" AND LLZ>0 THEN BLZ=1:ELZ=LLZ:GOTO 800
790 B$=RIGHT$(B$,LEN(B$)-4):GOSUB 400:IF EFZ=1 THEN 100
800 FOR IZ=BLZ TO ELZ:PRINT IZ:P$(IZ):NEXT:GOTO 100
810 REM
820 REM *** Save ***
821 REM
830 IF LLZ=0 THEN PRINT ER$"There isn't a program presently in memory.":@
    GOTO 100
840 B$=RIGHT$(B$,LEN(B$)-4):GOSUB 270:IF RIGHT$(B$,4)<>".PIL" THEN B$=B$+".PIL"
850 OPEN "0",1,B$:FOR LNZ=1 TO LLZ:PRINT #1,P$(LNZ):NEXT:CLOSE:GOTO 100
860 REM
870 REM *** Load ***
871 REM
880 B$=RIGHT$(B$,LEN(B$)-4):GOSUB 270:IF RIGHT$(B$,4)<>".PIL" THEN B$=B$+".PIL"
890 OPEN "1",1,B$:LNZ=0
900 LNZ=LNZ+1:LINE INPUT #1,P$(LNZ):IF EOF(1) OR LNZ=MLZ THEN LLZ=LNZ:CLOSE:@
    GOTO 100
910 GOTO 900
920 REM

```



```

930 REM *** Run ***
931 REM
940 VP%=0:SP%=0:C%=0:LF%=0
950 IF B$="RUN" AND LL%>0 THEN LN%=1 ELSE B$=RIGHT$(B$,LEN(B$)-3):GOSUB 400:@
  IF EF%=1 THEN 100 ELSE LN%=BL%
960 P$=P$(LN%):C$=LEFT$(P$,1)
970 IF P$="" OR C$="*" THEN 1020 ELSE IF C$=":" THEN P$=RIGHT$(P$,LEN(P$)-1):@
  GOTO 1010
980 C%=INSTR("TAINJUESR",C$):IF C%=0 THEN PRINT ER$"I don't know the program @
  command in line"LN%":GOTO 100
990 IF MID$(P$,2,1)<>":" AND MID$(P$,2,2)<>"Y:" AND MID$(P$,2,2)<>"N:" THEN @
  PRINT ER$"The program command in line"LN%"must be followed by @
  either :, Y:, or N:":GOTO 100
1000 IF MID$(P$,2,1)=":" THEN P$=RIGHT$(P$,LEN(P$)-2) @
  ELSE IF MID$(P$,2,2)="Y:" AND LF%=1 OR MID$(P$,2,2)="N:" @
  AND LF%=0 THEN P$=RIGHT$(P$,LEN(P$)-3) ELSE 1020
1010 ON C% GOTO 1050,1140,1140,1210,1260,1290,1330,1360,1020
1020 IF LN%>=LL% THEN 100 ELSE LN%=LN%+1:GOTO 960
1030 REM
1040 REM *** Type ***
1051 REM
1050 FOR I%=1 TO LEN(P$):C%=MID$(P$,I%,1):IF C$="*" AND VP%>0 THEN 1060
1060 PRINT C%;
1070 NEXT I%:PRINT:GOTO 1020
1080 A$=MID$(P$,I%+1,10):B$="":FOR J%=1 TO LEN(A$):C1$=MID$(A$,J%,1)
1090 IF C1$>"a" AND C1$<"z" THEN C1$=CHR$(ASC(C1$)-32)
1100 IF C1$>"A" AND C1$<"Z" THEN B$=B$+C1$:NEXT J%
1110 FOR K%=1 TO VP%:IF B$=V$(K%,1) THEN PRINT V$(K%,2):I%=I%+LEN(V$(K%,1)):@
  GOTO 1070 ELSE NEXT K%:GOTO 1060
1120 REM
1130 REM *** Ask ***
1131 REM
1140 IF C%=2 THEN GOSUB 170 ELSE A$=INPUT$(1):IF A$>"a" AND A$<"z" THEN @
  B$=CHR$(ASC(A$)-32)
1150 V$(0,2)=B$:IF LEFT$(P$,1)<>"$" THEN 1020 ELSE P$=MID$(P$,2,10):@
  IF VP%=0 THEN 1180
1160 FOR J%=1 TO VP%:IF V$(J%,1)=P$ THEN V$(J%,2)=A$:GOTO 1020 ELSE NEXT
1170 IF VP%=10 THEN PRINT ER$"Line"LN%"may not define a new variable, the @
  maximum of"MV%variables has":PRINT TAB(8) "already been defined. @
  However, one of these may be re-defined.":GOTO 100
1180 VP%=VP%+1:V$(VP%,1)=P$:V$(VP%,2)=A$:GOTO 1020
1190 REM
1200 REM *** Match ***
1201 REM
1210 I%=INSTR(P$,""):IF I%=0 THEN A$=P$ ELSE A$=LEFT$(P$,I%-1)
1220 IF A$<>"" AND INSTR(V$(0,2),A$) THEN LF%=1:GOTO 1020
1230 IF I%=0 THEN LF%=0:GOTO 1020 ELSE P$=RIGHT$(P$,LEN(P$)-I%):GOTO 1210
1240 REM
1250 REM *** Jump ***
1251 REM
1260 A$="*" + LEFT$(P$,10):FOR I%=1 TO LL%:IF LEFT$(P$(I%),11)=A$ THEN LN%=I%:@
  FOR J%=1 TO I%:NEXT:GOTO 1020 ELSE NEXT:PRINT ER$"Line"LN%"jumps to a @
  label that does not exist.":GOTO 100
1270 REM
1280 REM *** User Subroutine ***
1281 REM
1290 IF SP%=MS% THEN PRINT ER$"Before line"LN%"may jump to a subroutine, one @
  of the"MS%"presently being":PRINT TAB(8) "used must be ended.":GOTO 100

```

But computer programmers use them all the time. Just as a quick refresher, a variable is a word, letter, or special character used to represent a piece of information. For example, the Greek letter pi is used to represent the number 3.14. Variables can also represent alphabetical information as well. \$NAME="Kurt" means that the variable \$NAME equals the word Kurt. These variables are used to represent words, letters, or special characters including numbers.

I suppose you're now trying to figure out what that dollar sign (\$) is doing in front of the variable, NAME. The dollar sign is used as a marker. When PILOT/80 finds a word preceded by a dollar sign, it assumes that the word to be a variable. It will let that variable represent some piece of information or will retrieve the information associated with that variable. If the variable has not yet been assigned a value, PILOT/80 assumes that it is actually a word with a dollar sign preceding it and will act accordingly.

Memory

A program will always be a certain number of lines long, each line will fluctuate in its length, and a certain number of variables will be used. These three factors use the ever precious computer memory. So a very large program may not leave too much room for many variables and vice versa. To compensate for this, PILOT/80 has preset limits on the maximum number of program lines and variables. (See PILOT/80 Commands for a more detailed explanation.)

Review

Well what have we learned so far in our discussion of programs and computer languages?

1. A program is list of instructions given in a certain order so as to produce a desired result.
2. Computer languages act as a common ground between the programmer and computer. Sometimes languages are called interpreters. The language a program was written in must be present in the computer for the program to execute at all.
3. Instruction lines in a program are sometimes referred to by line numbers.
4. A variable is a word, letter, or special character used to represent a piece of information. Variables may represent words, letters, special characters, or numbers.
5. Memory constraints limit the size of a program. The number of program lines, the length of each line, and the number of variables used are directly related factors; each reduces the amount of memory available to

the others.

Well you learned more than you thought, huh? It's time to take a break now. Go raid the refridge or take a nap and come back in a little while. Give this information a little time to sink in and then move onto PILOT/80 Commands.

Command Level Commands

When PILOT/80.BAS is loaded and run a prompt will appear, (Request?). This prompt tells you that you are at the command level and that the computer is awaiting your instructions. Following is a summary of each command, a brief list is given near the end of this article.

NEW — Erases any program that might be in the program memory and prints the line number 1. The computer then waits for the programmer to input. Upon receipt of a carriage return, the computer will jump to the next line and print the next line number. This process continues until the programmer types the word 'DONE' following a line number.

LIST — Displays the program in the program memory, line by line. If there is no program in the program memory, the computer ignores this command. Sections of the program memory may be listed as may individual lines. To list from one line number to another, type the two line numbers after the command 'LIST' but make sure to separate them with a dash (-). To list from one line number to the end of the program, type the line number after the command 'LIST' but follow it with a dash. To list an individual line, type the line number following the command 'LIST'.

LIST = Lists the entire program

LIST 1-5 = Lists from line one to line five

LIST 5- = List from line five to the end of the program

LIST 5 = Lists just line five

EDIT — Gives the programmer the ability to re-write a program line that is already in the program memory. A line number follows the command 'EDIT'. The computer prints this line and then jumps to the next line where it prints the line number and waits for the programmer to type in the new program line. Upon receipt of a carriage return, the computer will jump to the next line where it prints the next line number. This process continues until the programmer types the word 'DONE' following a line number.

INSERT — Inserts a new program line into the program memory. The new line number should follow the command 'INSERT'. All

```
1300 A$="*" + LEFT$(P$,10):FOR IX=1 TO LLX:IF LEFT$(P$(IX),10)=A$ THEN @
    SPX=SPX+1:S$(SPX)=LNZ+1:LNZ=IX:FOR JX=1 TO 1:NEXT:GOTO 1020 @
    ELSE NEXT:PRINT ER$"Line"LNZ,"jumps to a label that does not exist.":@
    GOTO 100
1310 REM
1320 REM *** End User Subroutine / End Program ***
1321 REM
1330 IF SPX=0 THEN 100 ELSE LNZ=S$(SPX):SPX=SPX-1:GOTO 1020
1340 REM
1350 REM *** Screen ***
1351 REM
1360 IF P$="CLEAR" THEN PRINT CHR$(27)"E";:GOTO 1020
1370 IF P$="REVERSE" THEN PRINT CHR$(27)"p";:GOTO 1020
1380 IF P$="GRAPHICS" THEN PRINT CHR$(27)"F";:GOTO 1020
1390 IF P$="NORMAL" THEN PRINT CHR$(27)"q";CHR$(27)"G":GOTO 1020
1400 IF P$="BEEP" THEN PRINT "":GOTO 1020
1410 PRINT ER$"I don't recognize the screen command in line"LNZ."":GOTO 100
```

R:Vowels and Constant Version 83.03.07
R:By Kurt Albrecht

S: CLEAR

T: Vowels and Constanants

: By Kurt Albrecht

:

: My Name's Heath, what's yours?

A: \$NAME

T:

: \$name, I'm going to ask you to name some of the vowels and constants of
: the alphabet.

*BEGIN

T:

: Which would you like to try first; vowels or constanants?

: Just press either V or C.

I:

M: V,C

TN: I'm not sure I understand you. Please re-phrase your answer.

JN: BEGIN

M: V

JN: CONSTANANTS

*VOWELS

T:

: \$name, please name a vowel.

A: \$ANSWER

M: A,E,I,O,U,Y

UY: CORRECT

TN: I'm sorry, \$ANSWER is not a vowel.

J: CONTINUE?

*CONSTANANTS

T:

: Try and name a constanant \$name.

A: \$ANSWER

M: B, C, D, F, G, H, J, K, L, M, N, P, Q, R, S, T, V, W, X, Y, Z
 UY: CORRECT
 TN: Nope, \$ANSWER is not a constant.
 J: CONTINUE?

*CORRECT
 T: That's right \$name!!!
 E:

*CONTINUE?
 T: Do you want to continue (Just press either Y or N)?
 I:
 M: Y, N
 TN: I'm sorry \$name, But I don't understand you.
 JN: CONTINUE?
 M: Y
 JY: BEGIN
 T: Well I hope you enjoyed are little game \$name, so long.

lines after the inserted line are moved down one line in the program memory.

DELETE — Deletes a program line from the program memory. The line number to be deleted should follow the command 'DELETE'. All lines following the deleted line are moved up one line in the program memory.

RUN — Executes a program if there is one in the program memory. If there is not a program in the program memory, the computer will ignore the command. The program may be executed from any program line by typing the desired starting line number after the command 'RUN'.

SAVE — Saves the program presently in the program memory out onto a disk. A file name must follow the command 'SAVE'. This file name must consist of three things: A device specification, file name, and extension. The device specification must be used to indicate drives. The file name must be at least one character long and not greater than eight characters long. The extension of all PILOT files is .PIL. The extension is optional, if you don't type it, the computer will.

Special Note: If you wish, the program presently in the program memory can be outputted to any peripheral connected to the computer if it's device driver is loaded. This means that you can make hard copy listings of PILOT programs with a printer or speak the programs with a voice synthesizer (Though I don't know of what value this is!). To do this, the device specification must follow the command 'SAVE'.

LOAD — Loads a program from disk into the program memory. This command will erase whatever was in the program memory, so make sure this is what you want to do before you do something you'll regret! Everything

about file names applies to this command as it does to the 'SAVE' command.

BYE - Exits PILOT/80.BAS and returns to the operating system.

Concluding Remarks

By now you should have a good understanding of the PILOT language and the special commands available to you by PILOT.BAS.

But what better way to learn a language than to do a little programming in it? Although it is rather simple, the Demonstration program should give you a good idea of what a normal PILOT program looks like. Programs like it are very simple to write, that's what's so great about PILOT! You might consider writing a few programs like it for the kids. If you want anymore information about PILOT or come up with some modifications for PILOT.BAS, contact me. There is also a national organization for the advancement of PILOT known as (Are you ready for this?) PILOT Exchange, c/o Loop Center, 8099 La Plaza, Cotati, CA 45628.

Have fun with PILOT.BAS (its really fun to experiment with!), hang loose, and keep 'on hack'in!

Special Note: For CP/M implementation you should replace the @'s within the PILOT/80.BAS listing with a "Line Feeds".

(Note: You may obtain graphics, reverse video and other such special terminal functions by using the 'S' command. This adds a whole new dimension to PILOT/80.)



PILOT/80 Command Syntax

| PILOT/80 Command | Function | MBASIC Equivalent |
|--------------------|-----------------------------------|--------------------|
| NEW | Enter a new program | NEW |
| EDIT | Edit a program line | EDIT |
| INSERT | Insert a line(s) into the program | Line numbers |
| DELETE | Delete a line(s) from the program | DELETE |
| LIST | Display program | LIST |
| LLIST | Make a hardcopy listing | SAVE "LP:",A |
| SAVE | Store program on diskette | SAVE |
| LOAD | Retrieve program from diskette | LOAD |
| BYE | Return to HDOS | SYSTEM |
| T: | Type | PRINT |
| A:<Variable> | Ask for keyboard input | LINE INPUT |
| I:<Variable> | Input a single character | INPUT\$(1) |
| M:<List> | Match (Set Y/N logic flag) | IF -- THEN -- ELSE |
| J:<Label> | Jump | GOTO |
| U:<Label> | Jump to a User subroutine | GOSUB |
| E: | End user subroutine / program | RETURN / END |
| R: | Remark | REM |
| S: | Screen functions | H19 functions |
| <Op>Y: | Do operation if logic flag is Yes | |
| <Op>N: | Do operation if logic flag is No | |
| : | Repeat last operation | |
| &<1-10 letters> | String variable (10 max.) | <Variable># |
| *<1-10 characters> | Label for J: and U: | Line numbers |

BE A HAPPY HUGGER.

Z-100 Users! Call us for floppy & hard disk add-on prices. We have 8" & 5 1/4" drives for you!

Get to know Floppy Disk Services.

We are contracted dealers for Siemens, Shugart and Tandon disk drives. You may do a double take when you see our prices, but there's no catch—we simply buy in large quantities and sell at reasonable prices.

We've sold thousands of drives to our Heath/Zenith friends. And with our good service and fair guarantee, we keep our friends friendly.

Check our prices on these Heath/Zenith compatible drives . . .

| | |
|---|------------------------------|
| FDD-100-5 'flippy' (Just like your internal H89 drive) | \$215.00 |
| FDD-221-5 DS/DD 96tpi (80 track) same as TM-100-4 | 330.00 |
| FDD-200-5 same as 100-5 but double sided (2 heads) | 250.00 |
| FDD-100-8d5 SS/DD 8 inch with 3ms step! Sale! | 240.00 2 for \$450.00 ea. |
| FDD-200-8 DS/DD 8 inch 3ms step Sale! | 345.00 |
| New Shugart half height disk drives!! | |
| SA-455 double sided, double density, 48tpi half height 5 1/4" step time to 6ms! | 295.00 2 for \$245.00 ea. |
| SA-465 double sided, double density, 96tpi half height 5 1/4" step time to 6ms! | 350.00 2 for \$295.00 ea. |
| SA-860 double sided, double density, 8 inch half height. Step time to 3ms! | 495.00 2 for \$445.00 ea. |

PAYMENT POLICY We accept Mastercard, Visa, personal checks & M.O. We reserve the right to wait 10 working days for personal checks to clear your bank. All shipping standard UPS rates unless otherwise requested. New Jersey residents *must* add 6% sales tax. Due to production deadlines, prices in this ad could be as old as 2 months. If in doubt, call!

Prices and specs subject to change without notice.

Need an enclosure for 5 1/4 or 8"?

| | |
|---|----------|
| Single vertical case w/ps A&T styled to match Heath color | \$ 50.00 |
| Dual vertical case with power supply A&T | 75.00 |
| Dual horizontal case with dual floppy supply A&T | 125.00 |
| Dual horizontal 8" A&T | 275.00 |

Half Height Enclosures

Floppy Disk Services has designed these new half height enclosures from the ground up! We had you, the end user in mind and designed these 5 models:

| | |
|---|----------|
| Vertical dual 5 1/4 half height | \$ 75.00 |
| Single full size 5 1/4 or up to 2 half height | 125.00* |
| Dual side by side horizontal half height 5 1/4 | 125.00* |
| Dual side by side horizontal half height 8 inch | 295.00* |
| Single full size 8 inch or up to 2 half height | 295.00* |

* These enclosures are of professional design and have data cabling options available at extra charge.

** all of the above enclosures include A&T with power supply.

Magnolia 8" Controller

Magnolia Microsystems controller for the Heath H88, 89, 90 enables you to use any combination of 8 or 5 1/4 single or double sided, single or double density—up to 4 of each size drive! You even get CP/M in 8 and 5 1/4 format serialized for you. This controller is available at a special price of \$525.00.

Our pledge to you . . .

All Floppy Disk products are brand new and 100% warranted. If you are unhappy with our systems or components for any reason, we will refund your money promptly. There are two requirements: 1. the equipment must be in as good shape as you received it; and 2. you must call or write for a return authorization. This offer is good for 30 days, beginning with your invoice date. Feel free to call us with any questions.

We make Huggers happy. Call today!

Toll Free Order Line (800) 223-0306
Tech Help or Info. (609) 799-4440



Dual Heath Add-on
FDD-100-5



Dual Half Height
or Single Full Size 8"



741 Alexander Rd.
Princeton, NJ 08540

CP/M and H88-89-90 are registered trademarks of Digital Research and Heath Co., respectively.

Getting started with Assembly Language

(With Particular Reference to Writing Better Programs)



Pat Swayne
Software Engineer

This article is the first of a series I hope to do on assembly language programming. I do not intend to write a course on assembly language, but rather to help the person just getting started with helpful techniques and examples. I will not start at the very beginning, so readers who are at that point should read the series "A KISS for Assembly Programming" in REMark issues 15, 16, and 17. I will assume that the reader knows BASIC and how to create a file using a text editor. I also assume that the reader is capable of reading and following the instructions provided with the HDOS or CP/M assembler as to how to actually assemble a program, and that he/she is at least a little familiar with the 8080 instruction set.

This series will pay close attention to working within the operating system environment (HDOS or CP/M). The programs presented here will not be generalized examples, but will actually work on your computer. Each one will present basic techniques that you can incorporate into your own programs.

PART I — INTRODUCTION

There are some basic concepts that anyone attempting to become an assembly language programmer should know, which I will discuss here.

A computer **program** is a list of instructions for the computer. A **programming language** is itself a computer program that allows us to write our own programs in a way that we can understand, instead of having to work directly in the numbers that the computer understands.

BASIC is an example of a **high level** programming language. It is called high level because the programmer does not have to be concerned with what actually goes on in the computer. For example, if you say

```
LET A=100
```

BASIC takes the value 100 and stores it in a place called "A". You (the programmer) do not have to know where A is, or how big it is in order to use it.

Assembly language is a **low level** programming language. That means that you have to know something of what is actually going on in the computer when you write programs. If you want to store the value 100 somewhere, you have to provide the place to put it. That means more work for you, but it also means more efficiency, because you can, for example, reserve exactly the space required by 100 with none wasted, while BASIC may use the same amount of space for 100 that it uses for one million.

In assembly language, you can use the exact instructions necessary to do a particular job. If you want to add 100 to 100, you can use a simple but fast instruction capable of adding small numbers, while BASIC might add 100 to 100 using the same instructions capable of adding millions. So good assembly programs will always be smaller and faster than programs written with a higher level language. (When you compare the size of programs, be sure to include any in-

terpreter or run-time executive required to run them. Once assembled, an assembly language program will usually run by itself.)

The above statements may sound a bit elementary for a series that is not going to start at the very beginning, but some of the assembly programs that I have seen appear to have been written without a grasp of those basic concepts.

Throughout this series, I may present from time to time what I consider to be good rules for assembly programming. The rules may not apply to every situation, but in general they represent good practice learned through experience.

PART II — CONSOLE I/O IN HDOS

Console I/O means inputting information from your console keyboard or outputting information to your console screen. Nearly every program that you write will require some kind of console I/O, so most programming courses start out in this area. For example, a BASIC course may start with the PRINT statement.

The traditional assembly language course or discussion starts out in what I feel is the wrong way in this area. They usually deal with the hardware ports, and present routines to set them up and to send and receive characters via the ports. That is the wrong approach to take because, in the "real world", the operating system (HDOS or CP/M) already takes care of all of that and provides several ways to input and output that are built in. This brings up the first of my rules for assembly language programming.

Rule 1. If the operating system can do it, let it do it.

Sometimes we receive a program submission here at HUG that is an extremely good program written by someone who is obviously an accomplished programmer that, for example, will run on an H89 but not on an H8 because it breaks this rule.

Type My Name...

The first example program I will use is the famous (or infamous) "Type My Name Ten Times" program that is used in the Heath Assembly Language Course, and that appeared in REMark #12. But this time, we're going to do it right. The program will be presented in two versions, HDOS and CP/M, with each one using the operating system for all I/O functions. I will discuss the HDOS version first, and the CP/M version in next month's issue.

...in HDOS

Listing 1 is the HDOS version of the program. Notice that I have treated it as an assembly translation of a BASIC program, which is listed as comments at the beginning of the assembly source. The unusual syntax of line 20 in the BASIC program allows it to run in either Benton Harbor BASIC or MBASIC.

After the initial comments, the source listing contains 4 equate (EQU) statements that define the HDOS system calls that we will be using. The first of these, \$TYPTX, is actually a ROM subroutine, but

it may be considered part of the operating system as may all of the routines listed starting on page 2-99 of the HDOS 2.0 Software Reference Manual, or page 1-63 of the HDOS 1.6 manual. The other three are true system calls. In HDOS, a call to the system is made by executing a RST 7 instruction which is actually a one-byte CALL to address 70Q (octal), followed by a byte designating the function of the call. You could call the function labeled .SCIN in the program with

```
RST    7
DB     .SCIN
```

But the HDOS assembler makes it easier with a built-in pseudo opcode called SCALL, which replaces the RST 7 and allows the call function to be specified as an operand. So a call to the function .SCIN becomes

```
SCALL .SCIN
```

HDOS system calls are discussed in the HDOS System Programmer's Guide, which is supplied with HDOS 2.0. Users of other versions may get one by ordering 595-2553-01 from the Heath Parts Dept. This document discusses such things as which registers and flags are required or affected by the routine. Such things will not be discussed in detail for every system call in this article, but will be shown by example in the accompanying programs.

The HDOS assembler allows definitions such as the four in this program, or even actual program code, to be placed in separate files (which have .ACM as their extension) from the source file. The pseudo opcode XTEXT is used to bring these separate files into your program during its assembly. Several ready-made definition files are provided with HDOS version 2.0, including HOSDEF.ACM, which defines all of the system calls, and TYPTX.ACM, which defines the address of the \$TYPTX routine. So the four equate statements at the beginning of the program could be replaced with

```
XTEXT  HOSDEF
XTEXT  TYPTX
```

The ASM chapter of the HDOS Software Reference Manual explains the use of XTEXT in detail, so I will not cover it here.

An origin statement (ORG) follows the system definitions. The operand to it is the default address for loading programs into memory that run directly under HDOS. They may be started at a higher address (as long as there is free memory space), but cannot be started at a lower one. The assembler writes a header onto the executable binary file it produces that tells HDOS where to load the program.

The actual program begins with the label START. The purpose of this label will be explained when we get to the END statement. The first program lines translate line 10 of the BASIC program. PRINT statements in BASIC may be translated almost directly with calls to the \$TYPTX routine. \$TYPTX will cause any bytes following it to be sent to the console screen as text until a byte with the 8th bit set is reached. That byte will also be sent to the console after the 8th bit is removed, and then control will be returned to your program at the first address following the text. Notice the two apostrophes in the word WHAT'S. That tells the assembler not to use the apostrophe as a delimiter (its normal function), but to take the double apostrophe as a single literal one.

The next lines translate line 20 of the BASIC program, the LINE INPUT statement. As I mentioned in the Introduction, you must always provide places to put things in assembly language, so the first

line in this section points the HL register pair to a location (NAME) where we will store the input name. The next line sets up a counter that will limit the numbers actually accepted by the program. Then the program uses a system call to perform the actual input.

HDOS has only one system call for input, but it can be programmed to function in different ways through the use of another system call, .CONSL., which can also make changes to HDOS output functions. The use of .CONSL to affect input is discussed starting on page 5-65 in the HDOS 2.0 manual, or 4-65 in the HDOS 1.6 manual.

Our program does not use .CONSL because it uses only the default mode of operation of .SCIN, which is the line mode with echo. That means that .SCIN will not start delivering characters to our program until a RETURN is typed, and that each character typed at the keyboard will be echoed to the screen by HDOS. This mode allows HDOS to process the BACK SPACE or DELETE keys for correcting mistakes, or Control-U to abort the entry and start over. If these things were not handled by HDOS, our program would either have to take care of it, or there would be no way of correcting mistakes.

The .SCIN routine signals our program that it is not ready to deliver a character by setting the Carry flag, so the program will keep looping back to INPUT until the user hits RETURN. Characters returned by it are stored starting at NAME. I should point out here that when you press the RETURN key while using .SCIN for input, HDOS converts it to what is called the HDOS New-Line character, which is actually a line feed character. An HDOS output routine (such as \$TYPTX) will send both a carriage return and a line feed to the console when it encounters a New-Line character.

When our program detects the New-Line character (12Q), it terminates the entry by setting the 8th bit in the last character. If more than 30 characters have been typed before RETURN is hit, the counter (B register), which is decremented each time a character is accepted, will reach zero, and the program will "dump" the extra characters with calls to .SCIN (the program will jump back to DUMP until no character is available) and insert a New-Line character after the first 30 characters received. Note that we could have had the program print an error message or take some other course of action in the case of too many characters input.

The next BASIC line translated is the FOR statement in line 30. We translate it by putting the value 10 into the B register, which will be used as a counter.

The next line translated is PRINT N\$ (line 40). This line illustrates another method for outputting lines of text in HDOS. The system call .PRINT works like \$TYPTX except that the text printed starts where the HL register points, not immediately following the call. This routine allows us to print text located anywhere in memory, which in this case is the area labeled NAME, where we stored the user's input. As with \$TYPTX, a character with the 8th bit set signals the end of the text, which is why we had to terminate the user's entry with such a character. Note that we have placed the label PRINT at the beginning of this section. That is because the program is going to print the name 10 times by jumping back to this section 10 times.

The NEXT I statement is translated in the next section. Here, the program decrements the counter (B register) and jumps back to PRINT if it is not zero. We could have started out with the counter at zero and incremented it each time until it reached 10. But that would have required extra program steps to test the counter. The processor can detect when a register reaches zero without extra test steps, so it is usually best to decrement a counter in assembly language rather than increment one.

The END statement in BASIC is replaced with code that causes a return to HDOS via the .EXIT system call. There are two kinds of exit

available, called Normal and Abort. Putting a zero in the A register causes the normal exit, while a non-zero causes the abort exit. The abort exit re-programs the console and disk ports, in case a user program has tampered with them directly. Our program uses a normal exit.

The storage place for the user's name follows the exit code. The DS 31 statement reserves 31 bytes for storage, including 30 for the name and one for the ending New-Line character.

The operand to the END statement at the end of the program tells the assembler the address of the entry point to the program. That is the point where execution will start, and it does not necessarily have to be the first memory location used by the program. As with the first memory location, the entry point is written into a header on the assembled binary file by the assembler.

This completes the discussion of the HDOS version of the program. There is one HDOS system call for output that I did not use in the program, the .SCOUT system call. It is used for single character output to the console. If you wanted to print the character "X" on the screen, you could do it with

```
MVI    A, 'X'
SCALL  .SCOUT
```

This is the basic HDOS console output function, which is used by the other output routines.

Next month, in Part III of this series, I will discuss console I/O in CP/M, and present a CP/M version of the Type My Name program. Don't miss it!

```
* THIS PROGRAM IS AN ASSEMBLY VERSION OF THE
* FOLLOWING BASIC PROGRAM
```

```
10 PRINT "HELLO, WHAT'S YOUR NAME? ";
20 LINE INPUT " ";N$
30 FOR I=1 TO 10
40 PRINT N$
50 NEXT I
60 END
```

```
* THIS VERSION IS FOR HDOS, AND UTILIZES SOME
* BUILT-IN HDOS ROUTINES
```

```
* BY P. SWAYNE, HUG 20-MAY-82
```

```
* DEFINE HDOS ROUTINES
```

```
$TYPTX EQU 31136A TYPE THE TEXT THAT FOLLOWS CALL
.SCIN EQU 1 INPUT ONE CHARACTER FROM KEYBOARD
.PRINT EQU 3 PRINT TEXT POINTED TO BY HL
.EXIT EQU 0 EXIT TO HDOS FROM PROGRAM
```

```
ORG 42200A MOST HDOS PROGRAMS START HERE
```

```
* 10 PRINT "HELLO, WHAT'S YOUR NAME? ";
```

```
START CALL $TYPTX THIS IS AN HDOS RESIDENT ROUTINE
DB 'HELLO, WHAT'S YOUR NAME? ', '+2000
```

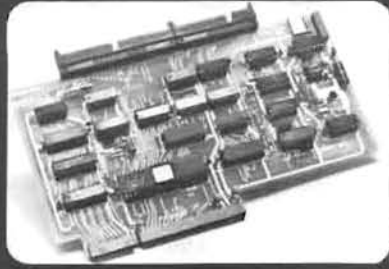
```
* 20 LINE INPUT " ";N$
```

```

INPUT
LXI H,NAME POINT TO WHERE NAME GOES
MVI B,31 ALLOW 30 CHARACTERS + RETURN
SCALL .SCIN CALL HDOS INPUT ROUTINE
JC INPUT WAIT FOR INPUT
MOV M,A STORE ONE CHARACTER
DCR B USED UP CHARACTER LIMIT?
JZ DUMP IF 0, DUMP THE EXTRAS
INX H MOVE TO NEXT LOCATION
CPI 120 END OF ENTRY?
JNZ INPUT IF NOT, GET MORE
DCX H BACK UP TO LAST ENTRY
SCALL .SCIN DUMP EXTRA CHARACTERS
JNC DUMP
```

CONTROLLER

1



FOR 8" & 5.25" DRIVES

Now be able to run standard 8" Shugart compatible drives and 5.25" drives (including the H37 type) in double and single density, automatically with one controller.

Your hard sectored 5.25" disks can be reformatted and used as soft sectored double density disks. The FDC-880H operates with or without the Heath hard sectored controller.

NEW PRICE \$495

Includes controller board CP/M boot prom, I/O decoder prom, hardware/software manuals BIOS source listing. HSOD driver now available for \$40.00.

5-20 day delivery—pay by check, C.O.D., Visa, or M/C.

R

Contact:
C.D.R. Systems Inc.
7210 Clairemont Mesa Blvd.
San Diego, CA 92111
Tel. (714) 560-1272

| | | | |
|-------|-------|-----------------------------------|---------------------------------|
| TERM | MVI | M, 120 | END ENTRY (IN CASE OF OVERFLOW) |
| | MOV | A, M | GET LAST CHARACTER |
| | ORI | 2000 | SET 8TH BIT (MARK END) |
| | MOV | M, A | REPLACE CHARACTER |
| * | | 30 FOR I=1 TO 10 | |
| | MVI | B, 10 | SET UP A COUNTER |
| * | | 40 PRINT N# | |
| PRINT | LXI | H, NAME | POINT TO NAME |
| | SCALL | .PRINT | PRINT IT |
| * | | 50 NEXT I | |
| | DCR | B | DECREMENT COUNTER |
| | JNZ | PRINT | LOOP UNTIL IT'S ZERO |
| * | | END | |
| | XRA | A | PUT ZERO IN THE A REGISTER |
| | SCALL | .EXIT | RETURN TO HDOS |
| * | | PLACE TO STORE NAME (VARIABLE N#) | |
| NAME | DS | 31 | ALLOW 30 CHARACTERS + END CHAR |
| | | | TELL ASSEMBLER WHERE |
| | END | START | PROGRAM STARTS * |



SJULSTAD ENGINEERING

BRAND NEW 256 K RAM — \$699.00 (Software Included)

Expand your computer memory beyond its present 64 K RAM limit. Inexpensive, easily installed, and completely compatible with Heath/Zenith 88, 89, and 90 Microcomputers. Nothing on your present system is wasted or thrown away as with other 256 K RAM's. Write for more information on these products:

| | | |
|-----------|-------|----------|
| 256 K RAM | | \$699.00 |
| 128 K RAM | | \$469.00 |

Both Memory Expansions complete with Instructions and Software.

REMOTE VIDEO OUTPUT — \$59.95

Provides NTSC Industry standard composite video to any monitor equipment including:

- Color or Black/White Television Monitors
- Large Screen Monitors
- Beta/VHS Recording Equipment

Especially suited for Educational and Business Applications!

16 K ADD-ON RAM — \$59.95

Increase your present 48 K system to its full capacity of 64 K. Required in order to run D-base II.

DISCOUNTS AVAILABLE FOR LARGER ORDERS

For more information write or phone:

SJULSTAD ENGINEERING

503 East Fremont • Northfield, MN 55057

(507) 663-3422

All boards are completely assembled and tested and are guaranteed for ninety days to be free from all defects.



We repair and service most microcomputers.



Software for HEATH/ZENITH Systems

UTILITIES

SUPER SYSMOD2—An HDOS enhancement program that abbreviates HDOS commands and adds many new ones.

UD.DVD—A truly universal driver for printers. Provides seventeen options including multiple copies and page formatting.

H25.DVD—A special driver for the H/Z-25 printer.

MX.DVD—A special driver for the MX-80 family of printers.

SPOOLER DRIVERS—Above drivers with built-in spooling.

BASEDIT—Adds a built-in editor to HDOS BASIC

EDGE—A powerful character oriented text editor

DISAS—A fast disassembler (includes TINY PASCAL)

LANGUAGES

HFOFTH79—fig FORTH plus the FORTH-79 standards

HFORTH—fig FORTH 1.1 with many other features

EDUCATIONAL

SPANISH, FRENCH, GERMAN, and ITALIAN HANGMAN

Each version contains over 2000 words presented as flashcards or in the popular game of HANGMAN

Call or write for FREE catalog.

Available at your local HEATH dealer or from:

35 Shadow Oak Dr.
Sudbury, MA 01776
(617) 443-9693

SoftShop
Jim Teixeira

Generic Software

QUALITY SOFTWARE FOR HEATH/ZENITH SYSTEMS

Generic's DATA ENTRY Utilities make getting information into your computer easier!

SFDVD (HDOS only!) \$19.95

SFDVD allows H19 screen form access to user programs written in Benton-Harbor BASIC, MBASIC, FORTRAN-80, and Assembly language. **SFDVD** will manage all of the terminal input and display functions during program execution. Supports numeric only, alpha only, upper case, and non-echo data input fields. With **SFDVD**, data collection and file updating can be more "user friendly."

FORM-IT (MBASIC) \$19.75

FORM-REL (FORTRAN) \$19.75

Data entry kits for your H19. Each kit includes sophisticated utilities for screen form editing and generation. A set of library routines are provided which allow you to easily utilize screen forms within your program! The "FORMS" kit can be used for both sequential and random access file structures. Screen form input can greatly enhance any database/data entry application, because it simplifies data input and modification. CP/M and HDOS.

Call or write for a free Spring 1983 catalog.

P.O. BOX 790 • MARQUETTE, MICHIGAN 49855

PHONE: (906) 475-7151 • 10 AM to 5 PM EST

ATTENTION SOFTWARE AUTHORS: GENERIC SOFTWARE is interested in high quality and well documented software for Heath Zenith computers. GENERIC offers professional packaging, international marketing, and high royalties. If you are interested in making money from the software you develop, then request our FREE booklet **Software Author's Kit**.

ZBASIC Power or To The Rescue

Terry Jensen
Software Developer

A few weeks ago, I had the experience of representing HUG at a state wide Mathematics Conference for teachers, which was held at the St. Joseph, Michigan area high school. Jim Bartley from the Education Department of Heath company and Andy Ealovega, the local sales rep, were there to present their offerings. Through the eight hour day affair, we had the opportunity of meeting many interesting teachers from all over Michigan (and as you will see, one of the local high school students).

Many of the teachers stood at awe watching the Z-100 demonstration programs. While the rest, thinking the machine was too complicated, continued on their way to view the programs by other vendors. Generally, the teachers were looking for CAI (Computer Aided Instruction) programs for their students.

Two days prior to the Conference, I was notified that I would be attending. Having never used my Z-100 computer (which had been setting in a corner of my office until the catalog was done), I began to scratch my head on how I could talk intelligently to the math instructors on the neat things they could do with the Heath/Zenith H/Z89 and Z-100 series computers in their classes.

After putting on my thinking cap for about two minutes, something came to mind: up until this point it has not been possible to plot graphs on the CRT screen of a micro-computer. I don't mean bar graphs either. What about those days back in Calculus and Analytic Geometry, where the instructor would quickly sketch on the black board the representation of a sine wave, a parabola, an ellipse, or how about a hyperbola? Most of the time the graphic representation was enough to give the student an idea of what a particular graph should look like.

The Z-100 can access each point on a 224 X 639 dot matrix, so why not let the Z-100 plot the graph so the student can see a fairly true representation of a particular graph. With this in mind, I headed to the conference with my low profile Z-100.

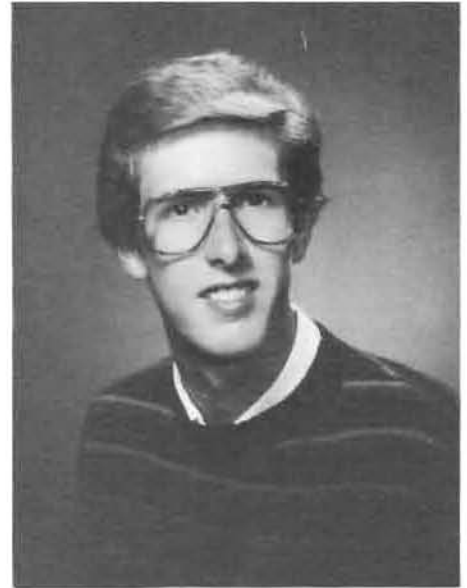
During the seminars the display area was free of teachers, so I spent that time sitting in front of the Z, frantically trying to set up in my mind how to go about creating a program which would plot a graph . . . any graph.

The normal (X,Y) axis should have the (0,0) coordinate in the center of the screen, with

the positive X and Y axis pointing to the right and top of the screen, respectively. The negative values would be the opposite directions. Here's the confusing part: the Z-100 plots from a (0,0) coordinate at the top left corner with the imaginary positive X,Y axis in the right and bottom directions on the screen.

While sitting and explaining my problem to a few people looking over my shoulder, one of the individuals asked if he could try. I said sure and let him sit down at the keyboard. An hour or so later, he had written a program which would plot a sine wave, in any amplitude or length. That is not all, by the end of the day he had written programs to plot a parabola, an ellipse, and a hyperbola!

Let me introduce this young gentleman to you. His name is Chris Linn. He is a senior at St. Joseph High School here in St. Joseph, Michigan, and a member of the National



Chris Linn

```

1 DEF FNE(X)=SIN (X)*25
2 DEFDBL Z
3 REM this program by Chris Linn
5 CLS
10 LINE (318,0)-(318,224),1:LINE (0,112)-(639,112),1
20 DEF FN(X)=(X*2)+318
30 DEF FNY(Y)=-Y+112
40 FOR A=-159 TO 159 STEP .2
45 X=A/10
47 Z=FNE(X)
48 IF ABS(Z)>.700 THEN 60
50 PSET (FN(X), FNY(Z)),6
60 NEXT

```

SIN.BAS will plot a sine wave.

Honor Society. He has been accepted at Massachusetts Institute of Technology (MIT) where he will major in Computer Science.

Chris has done programming for about four years on Radio Shack, Commodore, and Heath computers. However, this was the first day that he had ever programmed on a Z machine and therefore, the first time he had seen ZBASIC.

Chris solved the problem of locating the (0,0) coordinate of the axis at the center of the screen by defining X and Y functions, FN(X) and FNY(Y), as you will see in the programs. From there he was able to plot a graph by defining another function, FNE(X), as the quadratic equation. The most difficult part of the program was displaying the X and Y axis and their subsequent coordinat

values along the axis.

Here are his comments concerning his programs:

"The basic code in my programs is very poor — I just threw it together to make it work as quickly as I could. It works, but is inefficient."

"I found ZBASIC very easy to use and the graphic possibilities very exciting. With very little work, it is possible to create stunning displays on the Z-100."

The following programs are two of the programs he "threw" together on the day of the conference. The SIN.BAS is the first program he wrote. The QUIKLOT.BAS is a more sophisticated program, which includes a (X,Y) axis, where any quadratic equation my

be entered as the function (FNE).

I am sure that anyone who runs these pro-

grams will enjoy what appears on the screen of the H/Z-100 series computer.

```
5 REM
10 A1=20 : B1=100: REM A1= radius on x axis, B1= radius on y axis
15 REM***** Put function on line 20 in form f(x)=x (e.g. f(x)=2*x^2)
20 DEF FNE(X)=SOR((B1*B1)-(X*X*(B1*B1)))/(A1*A1)
21 REM *****
25 F1=1: REM flag for SOR function (so computer plots negative too)
30 DEFDBL Z
40 REM this program by Chris Linn
50 CLS
60 INPUT "accuracy (smaller=more accurate)";I
65 CLS
70 LINE (318,0)-(318,224),1:LINE (0,112)-(639,112),1
80 FOR S=94 TO 4 STEP -18
90 LINE (318,S)-(323,S),1: LINE (318,223-S)-(323,223-S),1
100 NEXT
110 COLOR 2
120 FOR S=1 TO 11 STEP 2: LOCATE S,42: PRINT MID$(STR$(13-S)*10),2):
LOCATE 26-S,42: PRINT STR$((-13+S)*10): NEXT
130 FOR S=278 TO 2 STEP -40
140 LINE (S,112)-(S,115),1: LINE (S39-S,112)-(639-S,115),1
150 NEXT
160 N=140: FOR S=4 TO 34 STEP 5: LOCATE 14,S: PRINT -N;
LOCATE 14,78-S: PRINT N; N=N-20: NEXT
170 DEF FN(X)=(X*2)+318 : REM mult for distortion, add for offset
180 DEF FNY(Y)=-(Y*.89)+112 : REM mult for distortion, add for offset
190 LOCATE 1,1
200 FOR P=-41 TO A1 STEP I
210 X=P
220 M=FNE(X)
230 IF ABS(M)>700 THEN 260
240 PSET (FN(X), FNY(M)),6: IF F2 THEN LINE (G,H)-(FN(X),FNY(M)),6
250 IF F1=1 THEN PSET (FN(X), FNY(-M)),6:
IF F2 THEN LINE (G,H1)-(FN(X),FNY(-M)),6
255 F2=1: G=FN(X): H=FNY(M): H1=FNY(-M)
260 NEXT
```

QUIKLOT.BAS will plot an ellipse.
Any quadratic equation may be replaced as
the FNE(X).

H-1000

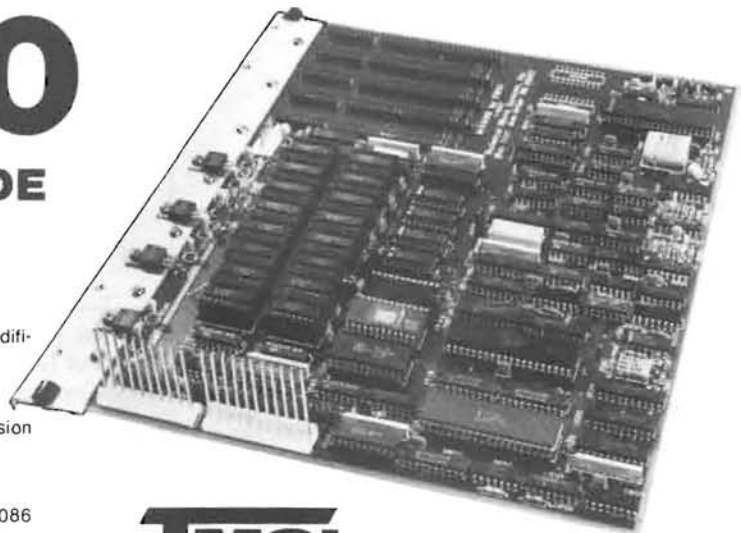
A Z80/8086 UPGRADE FOR THE H89/Z89

HARDWARE

- plug-in replacement for the H89/Z89 CPU board; no modifications required
- dual CPUs: Z80 and 8086
- 128K RAM standard; sockets for 256K, memory expansion bus for up to 1 megabyte RAM
- 5 I/O slots
- faster program execution: 2/4 MHz for Z80, 8MHz for 8086
- fully compatible with all Heath/Zenith peripherals

SOFTWARE

- runs all Heath/Zenith software without modification
- compatible with Zenith Z100 and IBM Personal Computer
- choice of MSDOS or CP/M-86 for the 8086
- supplied with diagnostic software package
- "soft disk" feature: copies an entire disk in RAM for instant disk access
- supports multi-user and multi-task operating systems



TMSI

Technical Micro Systems Inc.

P.O. Box 7227, Dept. H
366 Cloverdale • Ann Arbor, Michigan 48107
(313) 994-0784

We accept MasterCard and VISA.

The H-1000 is a quality product of Technical Micro Systems, Inc., manufacturers of innovative microcomputer systems since 1979.

H-1000 and TMSI are trademarks of Technical Micro Systems, Inc. H89, Z89, Z100 and HDOS are trademarks of Heath/Zenith Corp., Benton Harbor, Michigan. MSDOS is a trademark of Microsoft, Bellevue, Washington. CP/M and CP/M 86 are trademarks of Digital Research, Inc., Pacific Grove, California.

HERO I Chats with the H89

(ED: The following programs were developed within Heath Company and are in use daily to check the HERO I in various test phases as they are assembled. Kurt Teschendorf is the author of ETLOAD, the program written for HERO I. Kurt does a fine job of explaining the download procedure in the following article. Brent Miller is the author of DLOAD, the program used to interface with HERO I through the H89. Brent developed the hardware described in the article and used to complete the communications link. If you are using computers other than Heath/Zenith, the information given here will require slight modification for the port addressing used by the CP/M Operating System.)

Hardware Requirements

HERO I communicates with the H89 through the 8255 parallel port as described in Fig. 1. If you construct this project, a wire wrapping board for the H89 is suggested for best results (H-88-10 Wire Wrapping Board in kit form is available from Heath for \$30.00). Be sure to install the 1K (1000 Ohm) resistors from each of the lines indicated to the five volt supply (+5VDC) for proper operation of the 8255 chip. The source code for DLOAD is included with the article should you wish to interface HERO with the Z-89-11 serial/parallel I/O card available for the H89 from Heath/Zenith.

The Download Utility

The DLOAD program communicates with HERO by first transmitting the number of code bytes to be downloaded. DLOAD then sends the starting address followed by the actual code bytes. HERO starts storing the code bytes via ETLOAD in a contiguous fashion at the starting address until it has received the number of code bytes to be downloaded as originally communicated from the DLOAD program.

DLOAD is designed for the CP/M operating system. Enter the program using your favorite editor and assemble the program using normal CP/M conventions on the H89. ETLOAD is given in hex format for entry on the keypad of HERO. The Avocet Systems Inc. 6800 Cross-Assembler was used to create ETLOAD as a hex file. You may obtain information on the cross-assembler from:

Avocet Systems Inc. 804 South State Street Dover, Delaware 19901

(ED: Next month, HUG will introduce Pat Swayne's cross-assembler. Pat will describe the details when the project is completed.)

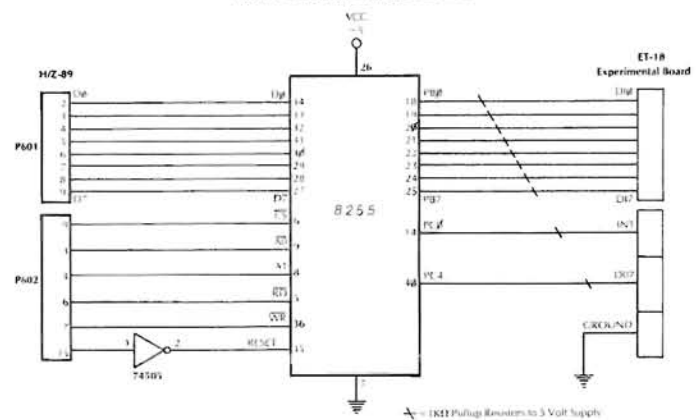
The Download Procedure

A. Entering ETLOAD for the first time.

1. Push the following keys on the ET-18 keypad: AA 003D
2. Enter the HEX file by line from left to right.

```
B6 0E E1 81 FF 26 01 83 BD F6 5B BD F7 E5 3D
0E 7E 77 BD 86 7E 97 2D CE 00 A3 DF 2E 8D 33
81 AA 26 FA 8D 2D B7 00 B5 8D 28 B7 00 B4 8D
23 B7 00 B7 8D 1E B7 00 B6 FE 00 B4 8C 00 00
26 02 3F 3A 09 FF 00 B4 8D 0B FE 00 B6 A7 00
08 FF 00 B6 20 E5 C6 00 F7 00 B8 86 80 B7 C2
20 F6 00 B8 C1 00 27 F9 B6 00 B7 39 86 00 B7
C2 20 B6 C2 A0 B7 00 B7 C6 01 F7 00 B8 39 00
00 00 00 00
```

H/Z-89 to ET-18 Download Hardware



3. Push RESET on the keypad.
 4. Push the following keys on the ET-18 keypad: 33 003D
 5. (Optional) Place your tape recorder in the record mode.
 6. Push the following keys on the ET-18 keypad: 00B9
 7. The robot will display HERO 1.0 when finished storing the program on tape.
- B. Loading ETLOAD from tape
1. Push the following keys on the ET-18 keypad: 34
 2. Play the recorded ETLOAD program back.
 3. The robot will display HERO 1.0 when finished loading the program from tape.
- C. Downloading from the H89 to the ET-18
1. Connect the cables from the H89 to the ET-18 experimental board.
 2. Type "DLOAD FNAME.HEX" on the H89. FNAME is any hex file you have created using a cross-assembler or other method of producing an Intel-Hex Format file.
 3. The DLOAD program will respond with: Ready to Download
 4. Assuming you have loaded ETLOAD as described previously into the memory of the ET-18, push the following keys on the ET-18: AD 003D
 5. The robot will display "dLOAD".
 6. Push the D key on the H89 keyboard to begin the download process. When the download procedure is complete, the H89 will return the the CP/M command prompt (e.g. A:). HERO will display HERO 1.0. The program is now ready to run.

SPECIAL NOTE: Since the ETLOAD program occupies a portion of HERO's memory, user programs should not be originated to an address of less than 00B9H or the ETLOAD program will be destroyed.

| | | | | | |
|----------|------|---------|--|---------|----------|
| Z | STA | COUNT | | SHLD | DECNUM |
| | STA | B,4 | | JMP | DECINI |
| | CALL | HEX#2D | | MVI | A,BEL |
| | MVI | B,2 | | CALL | OUT\$CHR |
| | CALL | HEX#2D | | JMP | DECINI |
| PAT#3: | MVI | B,2 | | DECOUT: | PUSH |
| | CALL | HEX#2D | | | PUSH |
| | MOV | A,L | | | PUSH |
| | CALL | PAT#4 | | | PUSH |
| | LIA | COUNT | | XRA | A |
| | DCR | A | | STA | DECFLG |
| | STA | COUNT | | LXI | B,10000 |
| | JNZ | PAT#3 | | CALL | DEC#1 |
| | MVI | B,2 | | LXI | B,1000 |
| | CALL | HEX#2D | | CALL | DEC#1 |
| | MVI | B,1 | | LXI | B,100 |
| | CALL | HEX#2D | | CALL | DEC#1 |
| | MVI | B,1 | | LXI | B,10 |
| | CALL | HEX#2D | | CALL | DEC#1 |
| | JMP | PAT#2 | | MOV | A,L |
| | | | | ADI | '0' |
| PAT#4: | PUSH | H | | CALL | OUT\$CHR |
| | PUSH | B | | POP | H |
| | CALL | OUTP | | POP | D |
| | LHLD | COUNT | | POP | B |
| | DCX | H | | POP | PSW |
| | SHLD | COUNT | | RET | |
| | MOV | A,L | | DEC#1: | MVI |
| | ORA | H | | | D,0FFH |
| | POP | B | | DEC#2: | MOV |
| | POP | H | | | A,L |
| | RNZ | | | | C |
| | POP | PSW | | | L,A |
| | RET | | | | A,H |
| | | | | | B |
| | | | | | H,A |
| | | | | | D |
| | | | | | DEC#2 |
| HEX#2D: | PUSH | D | | | B |
| | LXI | D,0 | | | A,D |
| HEX#2D1: | LHLD | SQUADD | | | '0' |
| | MOV | A,M | | | PSW |
| | INX | H | | | '0' |
| | SHLD | SQUADD | | | JZ |
| | SUI | '0' | | | DEC#3 |
| | CPI | ?+1 | | | PSW |
| | JC | HEX#2D2 | | | STA |
| | ADI | '0' | | | DECFLG |
| | | | | | CALL |

;Get Address and discard

;Get Null and discard

;Send Byte to ET-18

; (check remaining bytes to patch in line)

;Skip Remaining to Checksum bytes in Patch

;skip CR and LF

;Return after required number of bytes

;If counter=0 then Exit

;count=0, so exit

; Gets a "B" Digit Hex Number
; Returns Number in HL

Local HUG Clubs

The following notes concerning local HUG clubs have been received here at HUG and are placed here in REMark for your information and to help further the growth of Heath/Zenith computer interest around the world.

The following have recently formed or are looking to form a local HUG club:

MGHUG (Mid Georgia?)

Contact: Jerry Dalldorf
107 Cherokee Forest Trail
Warner Robins, GA 31093
Meets fourth Wednesday of the month. Phone Jerry Dalldorf at (912) 923-6962 or John King at (912) 923-1977 for time and place.

COMPUDRAGON (Hong Kong)

Contact: Mr. K.T. Lee
273 Prince Edward Rd.
11/C Kowloon, Hong Kong
Club is just organizing. Write to Mr. Lee or phone 3-7118904 after 8 PM (HK time) for additional information.

Las Vegas, Nevada

Contact: Anthony Marshall
4500 Draga Place
Las Vegas, NV 89115
Mr. Marshall is interested in making contact with anyone in his area that would like to help start a local HUG group. Phone (702) 644-5388.

Los Angeles, California

Contact: Gilbert Murillo
2309 So. Flower
Los Angeles, CA 90007
A ET/ETA-3400 interest group is forming in the area. Phone Gilbert at (213) 749-0261 or Charlie at (213) 443-2237 for more information concerning meetings.

Knoxville, Tennessee

Contact: Bruce Cliff
E. TN HUG
110 Northshore Drive
Knoxville, TN 37919
A new local HUG is forming in the East Tennessee area phone Bruce at (615) 588-0281 for more information on meetings, time & place.

Little Rock, Arkansas

Contact: David W. Schade
113 Dakota
Jacksonville, AR 72076
Anyone interested in forming a Heath/Zenith Users' Group in the Little Rock, North Little Rock and Jacksonville, Arkansas area please contact David at above address or phone (501) 988-5273.

Indianapolis, Indiana

Contact: Neil McCreary
563 Madison Ave.
Peru, IN 46970
Neil would be interested in making contact with anyone in his area interested in forming a Heath/ Zenith Users' Group. Write the above address or phone (317) 473-6160.

Central Pennsylvania

Contact: W. Douglas Wilkens
Dimension Five, Inc.
24 N. Third Street
Womelsdorf, PA 19567
Anyone interested in forming a local HUG in the central PA area please contact Doug at the above address or phone (800) 422-8582 (in PA. only) or (215) 589-2546. H-11 owner's — don't despair! This includes you too!

QUAD-Cities Area (Iowa)

Contact: Dennis C. Queal
RR 1 Box 149B
Princeton, IA 52768
Persons in the Quad-Cities Area (Princeton, LeClaire, Bettendorf, Davenport, Eldridge, Park View, Iowa or Moline, East Moline, Rock Island or Silvis Illinois please contact Dennis about forming a Heath/Zenith Users Group. Write above address or phone (319) 289-5856.

NORWEGIANS (Norway)

Contact: Siv. Ing. Arne Moestue
Gullregneveine 5
3150 Tolvsr0d
Telephone: 033 24460
Call: LA 8 NV
Present Norwegian members of HUG and other persons interested in forming a sub-group please contact as noted above. (number and distance from OSLO can be the limiting factor.)

The following information is changes, additions or corrections to previously published material on local HUG groups.

HUG of Central Florida

Contact: Joseph Walker
Phone: (805) 644-6848 (home)
Address: 121 Talmeda Trail, Maitland, FL 32751
All other info remains the same.

HUGRI (HUG of Rhode Island)

Contact: Leo Therrin (Treas.); Dave Haskell (Co-Chair)
All other info remains the same.

Capital HUG (CHUG)

Contact: Mike Cogswell (Pres.)
Phone: (703) 759-6176
Group Size: 600+, Library: 150+ disks
All other info remains the same.

FWHUG (Ft. Worth, Texas)

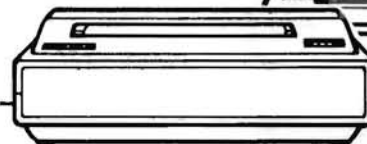
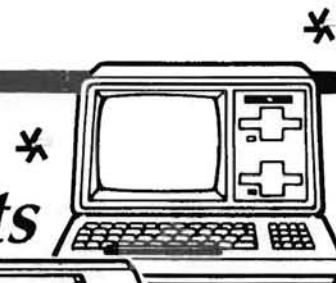
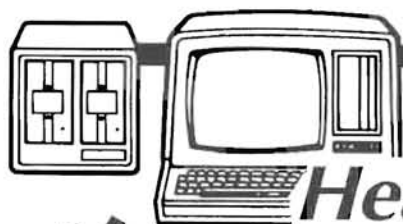
Contact: Don Murray (Temp. Sec.)
Meeting: 7:30, four Thursdays of each month.
Permanent election at September Meeting.
All other info remains the same.

Pomona HUG (Pomona, California)

Contact: Herb Friedman (Pres.)
Phone: (714) 985-5303, Group Size: 90
Meet 4th Thursday each month at 7:30 PM at HEC.
BB (714) 629-1943
All other info remains the same.

Jeri-HUG (Jericho, New York)
Contact: Alan Scott Dodge (Sec./Tres.)
Address: P.O. Box 78, Jericho, NY 11753
Phone: (516) 676-5616, Group Size: 65
Meets 2nd Thursday 8:00 PM Jericho Pub. Library.
Monthly newsletter, software library.
This is all new info.

VANHUG (Vancouver BC, Canada)
Contact: Eric Worthy
Address: 3058 Kingsway, Vancouver BC, Canada V5R 517
Phone: (604) 576-9842, Group Size: 35
Meets last Monday 7:30 PM at HEC
BB (604) 430-8233
This is a new club.



Heath Related Products

New Bits



Tom Huber
Related Products Editor

NOTE: The following information was gathered from vendors' material. The products have not been tested nor are they endorsed by HUG.

Ackerman Digital Systems Offers a Line of S-100 Cards

Ackerman Digital Systems has announced a new IEEE-696 S-100 Floppy Disk Controller that uses Western Digital's new WD2797 Controller and DM1883 DMA IC. The card will support two strings of 5.25" and/or 8" disk drives and systems with clock speeds up to 8MHz. ADS also sells a number of other IEEE-696 S-100 compatible cards, including the Synthetalker™, a speech synthesis board that includes waveform synthesis (via a D to A converter), the Votrax™ SC-01 synthesizer, two 8-bit I/O ports, and other features; a versatile PROM programmer card for 19 different devices; and CP/M compatible software for the Synthetalker. Inquiries should be directed to the vendor.

Vendor: Ackerman Digital Systems, Inc.
110 N. York Rd, Suite 208
Elmhurst, IL 60126
(312) 530-8992

System: S-100 bus (H/Z-100)

Prices: (Available from manufacturer)

TYCOON Simulates the Commodity Market

TYCOON is designed as a recreational/educational program and will allow the player to engage in a complex trading simulation of the commodities, gold, and foreign currency markets. It includes a program generator that allows the user (or educator) to create new scenarios at any time. Available in IBM PC, Apple (DOS), CP/M, and CP/M-86 versions.

Vendor: Blue Chip Software
19824 Ventura Blvd., Suite 125
Woodland Hills, CA 91364
(213) 881-8288

Price: \$79.95 for Apple
\$99.95 for all other versions

SUPER ZAP For CP/M

SUPER ZAP, a terminal-configurable disk dump and patch program, displays sectors from any CP/M disk format, in hexadecimal, octal, and ASCII formats. Sectors may be accessed by file or by absolute sector number on any disk, allowing changes to be made to system and directory areas as well as files. Displayed sectors may be dumped to the printer, or save on disk in a separate file, permitting dumping of arbitrary disk sectors for use by any program. Specify #216-C5 for Heath/Zenith or #216-C8 for 8" CP/M format.

Vendor: The Software Toolworks
15233 Ventura Blvd. #1118
Sherman Oaks, CA 91423
(213) 986-4885

Price: \$24.95 + \$2.00 S&H

Texteditor and
Line Count Programs
for CP/M and MBASIC

The Texteditor program, with an Epson Printer Controller, uses random access techniques to store data in ASCII. Texteditor can be learned in a half hour, and the package includes a Line Count program to indicate the length of a file before using Texteditor. Requires CP/M and MBASIC. Specify system when ordering.

Vendor: Tang Data Corporation
100 Eames Street
Framingham, MA 01701
(617) 872-7520

Price: \$9.95 on 5.25" diskettes
\$12.50 on 8" diskettes

TAXEZ 1982

TAXEZ 1982 is a series of income tax preparation worksheets that is designed to operate with Microsoft's Multiplan program. All of the worksheets are set up to take advantage of the Multiplan eXternal copy function. Forms 1040a and 1040 are fully supported, along with Schedules A, B, C, D, E, SE, and forms 2441 (Dependent Care), 4562 (Depreciation), and 5695 (Energy). Additional support is supplied with IRA, Earned Income, and All-Savers worksheets. Minimum system requirements are two 100K drives on the H/Z-89 series. Available on either hard- or soft-sectored CP/M format disks for the H/Z89, or soft-sectored Z-DOS format for the H/Z-100. Specify disk preference.

Vendor: LINKS COMM
P.O. Box 2028
Reston, VA 22090

Price: \$26.00

(VA residents add 4% sales tax)

NAVPROGseven UPDATES

Climb fuel modifications, usage notes and cruising Europe

Alan Bose
Taildragger Flyers
Ross Field
Benton Harbor, MI 49022

Climb Fuel Modifications

These patches will improve the accuracy of the climb fuel usage. In some situations, depending on the climb profile, a portion of the climb fuel would not be taken into account in the fuel synopsis & reserves.

```
NAVPROG7.BAS -- CP/M version
1490 ...:GOSUB 2810:K(I)=KU:KC=KU:CK=CK+KU:...
                                     ===== <-- insert this statement
1500 ...:GOSUB 2810:K(I)=K(I)+KU:K8(I)=K8(I)-KC:R=R+TT:...
                                     and this one here --> =====
```

```
NAVPROG7.BAS -- HDOS version
1780 K(I)=KU:KC=KU:CK=CK+KU
                                     ===== <-- insert this statement here,
1825 K(I)=K(I)+KU:K8(I)=K8(I)-KC
                                     ===== <-- and this statement
```

Short Distances and Other Notes

As mentioned in a previous article, plotting a great-circle route over short distances (less than a nautical mile or so) can exceed the limits of the trig functions available under MBASIC. I've talked with writers of other nav programs for other micros, and they've run into the same problem. When you're working on a global or continental scale, 1 or 2 nautical miles from checkpoint to checkpoint is very small indeed.

NAVPROGseven has a provision that seems to handle this dilemma quite nicely. You'll notice on the "Facility Code" there is space for two "co-located" facilities. In my database I have used AV to indicate a VOR located at an airport, or AN for an Airport & NDB, etc.

If the navaid is within 3 or 4 miles of the airport, I still consider them co-located, and enter the latitude & longitude of the navaid. At that distance, the airport should be in sight when flying VFR. On an IFR flight approach plates would be in use anyway for this final approach phase.

Also to be considered is the trig accuracy of MBASIC when heading, course or bearing is 90 or 270 degrees. Again it seems to be a limitation of working with a micro rather than a mainframe. A variance of 4 degrees is deemed acceptable in airborne nav receivers, and NAVPROGseven is well within these limits even when things read due east or due west (when trig accuracy is at its worst). Even so, it's good to know that these deviations exist since they affect general navigation & RNAV functions alike.

NAVPROGseven National Database

It has been suggested, by Bob Case of California and others, that it

would be nice if we could set up a nationwide database for users of the NAVPROGseven system. A formidable project to say the least, but work has begun on obtaining machine-readable VOR & airport data from the FAA and converting it to the NAVPROGseven file structure.

According to the Aircraft Owners & Association (AOPA) there are over 6,000 public use airfield in the country, along with over 1000 VORs, and 1100 Non-Directional Beacons. Quite a project to set up and maintain on a nation-wide basis. In addition all of the file handling routines in all the programs will have to be modified.

If you are interested in obtaining these modifications & upgrades as they become available, let us hear from you. Drop a line either to the HUG office, Hilltop Road; Saint Joseph, MI 49085, Attention: Flying Huggies. Or you can write me directly at 2514 Essex Court; Saint Joseph, MI 49085.

NAVPROGseven Goes to Europe

NAVPROGseven as originally released was confined to the Northern & Western hemispheres (ie North America). The following changes will allow the system to be used east of the Prime Meridian, which includes most of Europe. However, navigation is still limited to the Northern Hemisphere.

These changes allow the basic navigation east of, west of, or crossing the Prime Meridian. The automatic navigation also works properly with no further changes. The RNAV functions are NOT enabled east of the Prime Meridian at this time.

I should note that before you attempt to find the distance non-stop from Los Angeles to Paris that NAVPROGseven has an upper limit of 5400 miles for a single flight leg. Even Lindberg didn't fly that far without a positive position fix.

AIRINPUT.BAS (CP/M version)

Add the following lines:

```
1162 PRINT "East or West Longitude? <W> ";
      X%=INPUT$(1);PRINT X%
1164 IF X%="E" or X%="e" THEN EW=1 ELSE EW=0
1235 IF EW=1 THEN X=-X
1310 X=X+(Y/60)
1312 IF EW=1 THEN X=-X
1314 LSET M%=MKS$(X);GOTO 500
```

Also make the following changes:

```
1170 ....FNC$(7,31)... changes to FNC$(8,31)
1230 ....X<=0.... changes to X<0
1880 ... ;M5; changes to ;ABS(M5);
```

NAVPROG7.BAS (CP/M version)

Change line 610 ;M5; changes to ;ABS(M5);

AIRROUTE.BAS (CP/M version)

Change line 770 ;M5; changes to ;ABS(M5);

If you're using the HDOS version, the changes are the basically the same, but the line numbers are different.

AIRINPUT.BAS (HDOS version)

Add the following lines:

```
1362 PRINT "East or West Longitude? <W> ";
      X%=INPUT$(1);PRINT X%
1364 IF X%="E" or X%="e" THEN EW=1 ELSE EW=0
1435 IF EW=1 THEN X=-X
1525 IF EW=1 THEN X=-X
```

Also make the following changes:

```
1370 ....FNC$(7,31)... changes to FNC$(8,31)
1430 ....X<=0.... changes to X<0
2240 ... ;M5; changes to ;ABS(M5);
```

NAVPROG7.BAS (HDOS version)

Change line 750 ;M5; changes to ;ABS(M5);

AIRROUTE.BAS (HDOS version)

Change line 980 ;M5; changes to ;ABS(M5);

All these changes do is store longitudes East of the Prime Meridian (variables D5 & M5) as negative numbers. If you need to navigate the Southern Hemisphere simply make similar changes to the latitude sections of the same programs (affecting variables D6 & M6). If you intend to make a flight that crosses the Equator, you may not have one checkpoint north of the Equator and another on the south — you MUST have a checkpoint on the Equator itself. Fortunately for European pilots this restriction does not apply when crossing the Prime Meridian.



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

----- CUT ALONG THIS LINE -----

HUG MEMBERSHIP RENEWAL FORM

When was the last time you renewed?

Check your ID card for your expiration date.

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

Name _____

Address _____

City-State _____

Zip _____

REMEMBER — ENCLOSE CHECK OR MONEY ORDER

CHECK THE APPROPRIATE BOX AND RETURN TO HUG

NEW MEMBERSHIP
FEE IS: 

RENEWAL RATES

| | | |
|-------------|--|-------------------------------|
| US DOMESTIC | \$15 <input type="checkbox"/> | \$18 <input type="checkbox"/> |
| CANADA | \$17 <input type="checkbox"/> US FUNDS | \$20 <input type="checkbox"/> |
| INTERNAT'L* | \$22 <input type="checkbox"/> US FUNDS | \$28 <input type="checkbox"/> |

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

Superior Support

D·G



Heath
Users'
Group

Hilltop Road
Saint Joseph, Michigan 49085

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

POSTMASTER: If undeliverable,
please do not return.

Volume 4, Issue 4

885-2039