$2.50

REMark

Issue 34 • November 1982

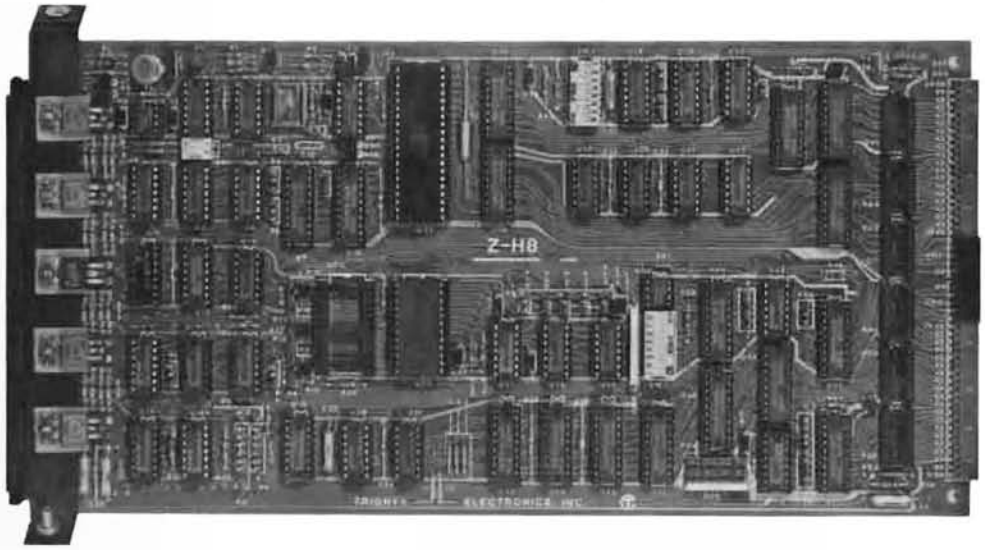
An Interview With Bill Johnson President of Heath Company

see page 10

Official magazine for users of HEATH ZENITH computer equipment.

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 Compatable 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

REMark Issue 34 • November 1982

on the stack

ON THE COVER: Bill Johnson, President of Heath Company, was interviewed by the Heath Users' Group. See page 10.

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

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

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

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

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

HUG Manager . . . . . Bob Ellerton
Software Coordinator and Developer . . . . . Jim Blake
Software Engineer . . . . . Pat Swayne
HUG Bulletin Board and Software Developer . . Terry Jensen
HUG Secretary . . . . . Margaret Bacon

REMark Editor . . . . . Walt Gillespie
Assistant Editor . . . . . Nancy Strunk

SMASH THE '89's 64K RAM LIMIT!

128K RAM BOARD — $595

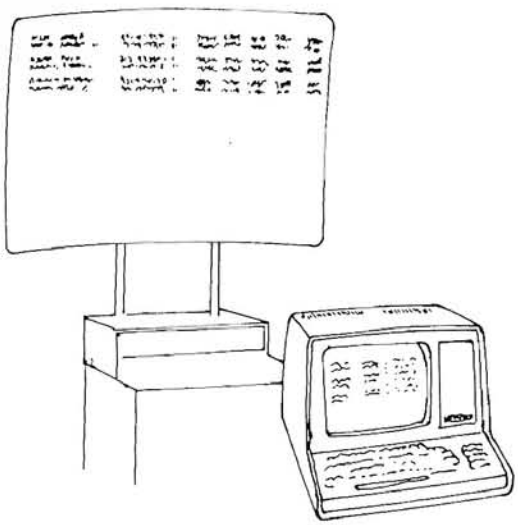
order no. 77318

Now have 176K of fast dynamic RAM at your disposal in your Heath/Zenith '88, '89, or '90.

Featuring:

- 128K of 200nSec Dynamic RAM, added to the 48K on your CPU board, for a total of 176K.
- Full compatability is retained for MMS and Zenith CP/M® (64K), as well as HDOS (56K).
- Versatile bank switching technique supports three full MP/M II® compatible banks.
- 112K "Electronic Disk" BIOS module for MMS CP/M included, complete with source code.
- Ultimeth Corp. supplies HDOS support.

Delivery beginning March 15, 1982.



VIDEO OUTPUT — $79

order no. 77319

Add an industry standard video output to allow reproduction of the CRT image on another monitor or projection TV system to enhance the usefulness of your terminal in classroom and other educational applications. Allows simultaneous viewing of the display in group situations.

The auxillary display unit should be capable of high resolution for satisfactory performance.

Delivery beginning March 15, 1982.

LOWER PRICE! 16K RAM — $125

order no. 77311

Magnolia's 16K Add-On RAM board has been so popular we lowered the price.

DDUBLE DENSITY DISK CONTROLLER — $595

order no. 77316

Complete hardware and software support for FOUR 8" single or double sided drives and FOUR 5" Single or Double sided, 48tpi (40 track) or 96tpi (80 track) drives, in addition to the three 5" drives supported by your Heath/Zenith controller.

Plus, the obvious advantage of being able to use Single Density 8" media for program and data interchange.

Full compatability is retained with MMS support of the '89s built-in 5" floppy, as well as several Winchester hard disk subsystems.

The package includes:

- Double Density Controller Card.
- Cables for both 5" and 8" disk drives
- CP/M 2.2 on both 5" and 8" media
- New I/O Decoder and Monitor PROMs
- Ultimeth Corp. supplies HDOS support.

If your '89 isn't ORG-0 CP/M compatible yet, our modification is available for $50 additional.

DOUBLE DENSITY SUBSYSTEMS

Dual 8"	DS	48tpi (2.4M)	order no. D8DS	$2695
	SS	48tpi (1.2M)	order no. D8SS	$1995
Single 5"	SS	48tpi (162K)	order no. S540SS	$ 945
	DS	48tpi (343K)	order no. S540DS	$1095
	DS	96tpi (700K)	order no. S580DS	$1295
Dual 5"	DS	96tpi (1.4M)	order no. D580DS	$1995

COMPLETE SYSTEMS

As well as manufacturing enhancements for the '89 (also '88 and '90), we are a Zenith Data Systems OEM, and have all of their hardware and software products available as well. We can provide a completely integrated system, combining the best Zenith products with our own to provide the exact system capabilities to best satisfy your requirements.

ORDERING INFORMATION

Our products are available from many Heathkit Electronic Centers and independent computer stores throughout the United States. If your local dealer doesn't stock our products, you may order direct or request further information by calling our Sales Department on our toll-free number, (800) 426-2841.

MAGNOLIA MICROSYSTEMS
2264-15th Avenue W. • Seattle, WA 98119
(206) 285-7266 (800) 426-2841

Snuggling up to a warm computer!

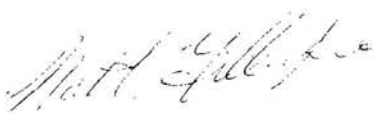
It's that time of year again, the boat has been put up in dry storage, the kids are settled back in school, local as well as national football rivalry is rampant, a certain smell of dry leaves and a snap of cold are in the air.

As the sure signs of Fall settle-in can Winter be far behind and that is definitily "computer season". Time to snuggle up to a warm computer and throw another log on the fire.

It's hard for a non-computer spouse to understand just why we, the enlightened, would want to steal away to a corner of the house to keep company with that green glowing, buzzing, grinding electronic-widow-maker than to cuddle up with them in front of the TV munching of all things pop-corn. But after all they did promise to take us for "better or worse".

Yes, it's time to dig out those forgotten program listings that you thought were such brain storms last winter and try and unravel that maze of "spaghetti" you found so easy to weave then. Or maybe it's a copy of Adventure that you now feel you can master and will be able to save the maiden from the wicked sorcerer.

But, after a busy night of "computer hacking" and the power switch is pushed to off, the warmth soon disappears from your friend, it's glowing face goes blank and it's nice to slip between the sheets and rub toes with someone who still wants us even if they can't figure out just what it is that we find so entrancing within that box of electronic parts.



Walt Gillespie
REMark Editor

BUGGIN' HUG

PLEASE NOTE: Alison Phillip's article "Moving on up to Soft-Sector Operation" page 13 Issue 32, has an error. The HS-89-2 INCLUDES H-88-1, HA-88-3, and H17-1. Anyone wishing to do a "kit" Z-90-82 only needs HS-89-2, WH-88-16, and Z-89-37.

Dear Walt,

I'm an original Heath H8/H9 owner too. I still have fun with my machine even though I never upgraded it with the H19 and/or disk drives. And this leads us to my reason for this letter...

Suggestion: I would like to see a section in REMark on interesting things that can be done with the H8. Could it be used as a printer controller/buffer in order to free up a separate processor to continue processing? Could it process analog to digital and DA signals direct from microphone to tape (or tape to audio output)? I'd like to know how people are still using their minimum configuration H8/H9 systems. I find myself using my H9 as a terminal while the H8 sits idly by.

Peter J. Terzian
Box 145
Alplaus, NY 12008

Dear Walt:

WOW! Looks Great!

Congratulations on your new assignment as REMark Editor. I am looking forward to more goodies in Issue 32. I particularly enjoyed the tutorial on escape codes. As a beginner in this "wizardry" I found it interesting and educational. Now if you would do the same type of article on the user defined function keys, I and other newcomers, would be delighted.

As an Amateur Radio operator, I would appreciate reading articles about Ham Radio applications for the H89. Best wishes for continued success.

Ben Reber
202 Hancock Blvd
Reading, PA 19611

Ben: In Issue 28 of REMark page 6 an article entitled "Oh Those Keys... They're Special... by David E. Warnick explains use of the function keys.

Dear Walt

The BIOS mod by Van C. Baker in REMark Issue 32 is an outstanding piece of work. Very well conceived and presented, even to the point of citing the proper pages in the BIOS listing. In fact, this excellent documentation helped me to find an error in the text on page 34. The statement "DW 4" must read "DB 4" or the DPB table will be skewed and the entire project will bomb. No doubt Mr. Baker has already pointed this out to you.

This mod worked just fine on my H8 with HA-8-6 Z80 board so it is not limited to only the H89.

Jim Meyers
13A Riggs Parkway
Las Vegas, NV 89115

(ED: Yes Jim, Van just called. I'm sorry for the typo but in trying to get materials ready for publication one sneaks by once in a while).

Dear HUG,

This is a note of Thank You to you and to Frederick Galloway on his timely letter in the August Issue concerning interfacing the H88 to the MX-80 printer. It helped a great deal. But, I would like to add a paragraph to it for all those others who might tend to hurry through the installation instructions for the H88-3 Serial Board. DO NOT disable the interrupts when installing this board in the H88. Unlike the H89, the cassette software uses these interrupts. Set the LP interrupt to 3 for the correct operation of the MX-80. Hope this helps.

Richard A. Schmidt
2004 Airfield
Kingman, AZ 86401

Gentlemen,

This is my first time writing to you and I wish to inform you and all other HUGgies on a problem that I encountered not only with my H-8 but with the "Mini-Computers" I work with for General Electric.

Problem #1: The "T.I." (Texas Instruments) I.C.'s used on the H-8 memory boards (both logic and memory itself) have excellent silver coating/content. The problem is severe oxidation after a period of time (depending on the environment you live in). The problem this creates is very "eratic" and "crazy" H-8 operations.

The solution I've come up with is a silver/precious metal cleaner called "TARN-X". If it's not available in your area then I recommend another high quality "liquid" silver cleaner.

If your H-8 is "acting up" simply look at the "T.I." chips on your H-8 boards (ie. Memory boards; Disk Controller board; and most of all the H-8 C.P.U.! The I.C. "legs" may be severely oxidized/black with oxidation.

Problem 2: The early version H-8's and the teflon/pinned connectors for the "LM" regulators were intermittent and a lot of H-8 owners simply replaced or soldered the "V.C.C." wires directly to the device. Point in mind >>>> Do not forget about the regulators for the "Front Panel" on the H-8!

Conclusion: By cleaning the "oxidized" I.C.'s and making sure the 5 volt regulator connectors are secure or soldered directly to the device will insure super H-8 operation!

Craig Corkery
General Electric Medical Systems
1120 Jorie Blvd
Oak Brook, IL 60521

In Defense of FORTRAN:

I read with great interest the article "Speed on Heath Micros", written by Alfred D. French in Issue 26 of REMark. I found the article to be very helpful to me and provided the last factor in making my mind up to select the Pascal language as my next compiler. I presently use the MBASIC compiler and the Microsoft Fortran 80 compiler. I operate a 64K H89 (2 Megahertz Clock) with three 5 1/4 disk drives and the CP/M operating system.

Because the article was of interest to me I decided to run the prime number algorithm on my system to see how it compared to Mr. French's findings. The program that I wrote and compiled with MBASIC ran close to the time stated in the article (4 minutes and 14 seconds). I came out with 4 minutes and 32 seconds. I received a surprise when I wrote and compiled the program in Fortran. My program ran in 4 minutes and 28 seconds which is appreciably different from the article time of 7 minutes and 5 seconds. Both programs found identical prime numbers, 1228 of them.

The following are the programs I used:

MBASIC:

```
10 DEFINT I:I4=0
20 INPUT "TYPE G TO START
                        THE TEST ";Y$
30 FOR I=3 TO 9997 STEP 2
40 I2=SQR(I)
50 FOR I3=3 TO I2 STEP 2
60 IF (I\I3)*I3 GOTO 100
70 NEXT I3
80 PRINT I
```

```
90 I4=I4+1
100 NEXT I
110 PRINT I4
120 STOP
```

FORTRAN 80:

```
          PROGRAM TEST
          WRITE(1,10)
10        FORMAT(1X,'TYPE G TO START TEST')
          READ(1,15)K
15        FORMAT(A1)
          K=0
                DO 50 I=1,9997,2
                A=I
                A1=SQRT(A)
                I2=A1
                      DO 30 I3=3,I2,2
                      I4=(I/I3)*I3
                      IF(I4.EQ.I) GO TO 50
30                    CONTINUE
                WRITE(1,40)I
40              FORMAT(1X,I5)
                K=K+1
50              CONTINUE
          WRITE(1,40)K
          STOP
          END
```

William Belli
3505 Shelton Road
Portsmouth, VA 23703

This letter is addressed to William W. Moss, MD - 1507 Riverview Lane - Bradenton, FL 33529:

Dear Dr. Moss,

I have just installed HPLINK as submitted to HUG by yourself. A deficiency in this program was immediately evident to me as it will be to anyone who transmits the ESC character with any frequency. Due to the function handling of the program logic the ESC character is transmitted as a semi-random value. The following fix, while not elegant, should handle most situations, including special function keys not used by the program.

Insert the following statement two lines after the identifier ESCTIM:

```
MVI A,ESC
```

Replace the statement following the identifier PCC7 with:

```
JNZ FIX1        IF NOT
```

Insert the following sequence of statements immediately preceding the identifier PCC8:

```
FIX1 STA  INCH      SAVE SPECIAL FUNCTION CODE
FIX2 CALL MOTEST    WAIT MODEM XMITT NOT BUSY
     JZ   FIX2
     MVI  A,ESC     XMITT ESC CHARACTER
     OUT  MODD
     LDA  INCH      GET SPECIAL FUNCTION CODE
```

These modifications have performed satisfactorily for me and should be included with the HUG distribution copy.

Donald E. Horner
1433 Roxburgh
East Lansing, MI 48823

Hi Bob,

With all the talk of the 80 track drives, and having gotten one, I figured people might need a dump program that accessed all the tracks.

The dump program available for HUG on diskette 885-1062A can be easily modified by adding the following to the source file and reassembling the program. The program will then figure out what type of drive the address specified is and allow the proper number of tracks to specified.

At the label GETTRK, replace the call to $TYPTX and the msg with

```
        LHLD    R.SDP+1         FIND UNIT TABLE
        INX     H
        MOV     E,M
        INX     H
        MOV     D,M             DE -> UNIT TABLE
        XCHG
        CALL    $DADA           HL -> UNIT TABLE ENTRY FOR
*                               DEVICE SELECTED
        MOV     A,M             GET UNIT TYPE
        LXI     H,40            BASIC NUMBER OF TRACKS
        RAR
        JNC     SSIDE           ONLY SINGLE SIDED
        DAD                     DOUBLE SIDED
SSIDE   RAR
        JNC     TRK40           ONLY 40 TRACK
        DAD                     80 TRACK DEVICE
TRK40   MOV     A,L
        STA     MAXTRK          SAVE TRACK NUMBER
        DCR     A
        STA     MAXTRK1         SAVE # ORIGINED TO 0
GETTRK  LXI     H,TRKMSG        GET MSG ADR
        SCALL   .PRINT          PRINT 1ST PART
        LDA     MAXTRK1         GET NUMBER OF TRACKS
        MOV     C,A
        MVI     B,0
        CALL    DECOUT          OUTPUT THE NUMBER
        LXI     H,TRKMSG1       GET REST OF MSG ADR
        SCALL   .PRINT          PRINT IT
```

After the label CKTRK, replace the CPI instruction with

```
        LXI     H,MAXTRK        GET MAX TRACK
        CMP     M               GREATER THAN MAX TRACK?
```

After the label NOT9, replace the CPI instruction with

```
        LXI     H,MAXTRK1       GET MAX TRACK ORIGINED TO 0
        CMP     M               SEE IF TRACK IS AT MAX
```

Preceding the label DSKID, add the following

```
TRKMSG  DB      'Track number (0','-'+200Q
TRKMSG1 DB      ') ?',240Q
```

After the label SMDC, add the following

```
MAXTRK  DB      0
MAXTRK1 DB      0
```

Just zapping the old program works, but causes problems trying to step too far if you type a wrong track number. This method keeps the drive happy and the programmer honest.

Vectored to 13 ☞

Introduction To Z-BASIC Part I

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

The **Z-BASIC** Color Language is described in the latest Heathkit Catalog (Christmas 1982) as an extended version of the Microsoft BASIC Language Interpreter. **Z-BASIC** runs under **Z-DOS** (Zenith Disk Operating System) and requires the use of a Z-100 series computer.

A color monitor is a very nice option, however, many of the new features of Z-BASIC work equally well on a black and green monitor. The **Z-120** (All-In-One) computer comes with only one bank of video RAM allowing only one color or shade of green, but all other features of **Z-BASIC** work very well. The additional banks of video RAM may be added at anytime to add additional shades of green on the built-in monitor. (Two sets of video RAM, model Z-219-1, are the required addition.) A color monitor may also be attached to all versions of the **Z-100**, however, full video RAM is recommended (and almost required, since one bank addresses each of the three primary colors, red, green and blue) when using a color monitor.

So much for the introduction of **Z-BASIC**, let's start by explaining some of the new features and how some of the old features have changed. First when requesting **Z-BASIC** from **Z-DOS** the screen is cleared for you and the version numbers and copy-right information is displayed in the upper right corner of the screen.

```
Z-BASIC Rev. 1.0
[Z-DOS/MSDOS Version]
Copyright 1982 (C) by Microsoft
Created: 24-Sep-82
55974 Bytes free
```

Version numbers, copyright dates, creation dates and bytes free will probably be different since this version is a pre-release version.

The first new command that should be used in a program is the **CLS** command for clearing the screen. This command is a little different than the ESCape "E" used on that H/Z19/89/90 display, in that it clears the entire screen including the 25th line. Also the **CLS** does **NOT** use the **PRINT** command in front of it. I chose the **CLS** first because if you don't clear the screen things may start to cause you problems since **Z-BASIC** uses a screen editor. A screen editor is an editor that reads what you have placed on the screen and may read something on a line, at the end of the line, that was not intended to be added. It is best to clear the screen before doing any editing when using a screen editor. More on the use of the screen editor in a future article.

```
10 '            LINE.BAS        Version 09.30.82
20 CLS:'        Clear Entire Screen
30 LINE (215,75)-(425,150),2,BF
40 LINE (1,225)-(640,1)
50 LINE (1,1)-(640,225),7,B
60 LINE -(1,1)
70 END
```

The above program introduces the **LINE** command available for the first time on a Heath/Zenith computer. The **LINE** command is one of the most powerful commands available in any type of BASIC. The **LINE** command has the ability to draw a line (40 and 60), draw a box (line 50), or draw a filled in box (line 30).

When using any command that is drawing lines, circles or addressing the screen for non-character printing, the horizontal position is first, followed by the verticle position. The horizontal positions are from 1 to 640 dots and the vertical positions are from 1 to 225 lines.

Take a close look at the above little program, first the **CLS** command clears the screen and then the first **LINE** (line 30) command draws a box (B) filled (F) in the color green (2) starting at the location 215,75 (215 dots to the right — 75 lines down) and ending at 425,150 (425 dots to the right — 150 lines down). Look at line 30 and note that the first two numbers (inside parentheses) are the starting point and the second two numbers are the ending point with connecting lines drawn to form a rectangle which may also be a square if the correct ratio is used. The number two (2) is the color green and always follows the location numbers. The 'BF' is actually two commands telling the **LINE** command that a box or rectangle is being requested and that it is to be filled in with the specified (2=green).

Color options are:

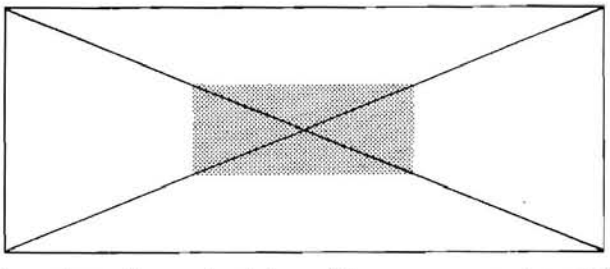
0 Black	4 Red
1 Blue	5 Magenta
2 Green	6 Yellow
3 Cyan	7 White

The next line (40) draws a line from the lower left corner (1,225) to the upper right corner (640,1). Since no color number is used, **Z-BASIC** assumes white (color number for white is 7). The 'BF' command is left off indicating that a line is to be drawn, not a box as in line 30.

So far we have a filled in box in the middle of the screen and a line from the lower left corner of the screen dissecting the filled box and continuing to upper right corner of the screen.

Line 50 is very simialr to line 30 with different coordinates, using white(7) and not filling the box (B). This line command draws a rectangle on the outer edge all around the screen.

Line 60 looks like someone forgot to place the first coordinate after the **LINE** command, however, the starting coordinate may be left off if the line desired is to be continued from the last coordinate, in this case the coordinate was the lower right-hand corner (640,225). Line 60 draws another line from the lower right corner to the upper left corner.



So far we have discussed only two of the new commands available in **Z-BASIC** and yet the possibilities when using the **LINE** command are almost endless. This article is just a beginning of what is hoped to be a regular series of articles on programming in **Z-BASIC.** Each article is planned to be short and explain only one or two related commands.

An Interview With Bill Johnson

President of Heath Company

Bill Johnson, President of Heath Company, is a graduate of the University of Illinois, having majored in advertising and marketing. Bill joined the Army Signal Corps as a 2nd Lieutenant in 1955, and worked out of the old Paramount Studios at Long Island City, New York preparing Army training films. On leaving the Signal Corps in 1957, Bill joined Whirlpool Corporation in Benton Harbor, Michigan, as Assistant Advertising Manager in their Kitchen Division. In 1959, he joined Heath Company as Dealer Sales Manager, but at that time there were no retail operations as Heath was still 100% mail-order.

Bill proceeded to establish a dealer network to handle Heathkit products; Hi-Fi dealers, marine dealers, etc. At this time the idea came forward that maybe Heath should try one of their own retail stores as it was felt that the dealers were not taking care of customer problems after the sale. Heath's first retail store was opened in Denver in 1961. The store was an immediate success doing over four times the annual kit sales that the local independent dealer had been doing. Because of the success of the Denver store and another store that opened in Chicago during 1962, Heath, under the direction of its then parent company Daystrom Inc., began to open new stores around the country at the rate of about two or three a year. In 1968, with approximately thirteen retail operations, Heath was purchased by Schlumberger.

Because of his disenchantment with the speed at which new stores were being opened, Bill accepted an offer to join Columbia Broadcasting Systems in Santa Ana, California as Vice President of Marketing for the Musical Instrument Division. After three years on the Coast, Bill was contacted by then Heath President, Dave Nurse, who indicated that because retail sales had been doing so well, Schlumberger had decided to open many more stores and would Bill like to come back and head up this project.

On returning to Heath, Bill took charge of the retail store program as Vice President of Marketing. The company, over the next few years, opened 35 to 40 more Heathkit stores. His next move was to that of Vice President and General Manager where he controlled retail sales, mail order, international sales and product planning of Heath products.

Then came the Zenith acquisition. Bill was promoted to Executive Vice President with the understanding that President, Dave Nurse, would reach retirement age within the year and Bill would move up to President, the position which he now holds. Bill and his wife Patt have three children, daughter Kristin, a graduate student of the University of Illinois, daughter Sally who has just enrolled at U. of I., and a son Steve, a junior at St. Joseph High School.

The following interview was conducted with Bill Johnson, President of the Heath Company at his office within the Heath Company located on Hilltop Road, St. Joseph, Michigan. The questions asked were selected from those that users have passed along to HUG ... your concerns, problems or curiosity.

REMark: What are Heath's plans for the future as it pertains to micro-computers?

Johnson: In talking for publication it's obvious that I can't divulge detailed long range plans. But, I think it is clear that we have made a major commitment as a company to the micro- computer business. If any company ought to dominate the kit portion of the micro-business that company ought to be the Heath Company. I think that's well evidenced by the fact that the business started out very heavily kit orientated, with manufacturers such as Mits, Altair, Imsai, Processor Technology, Digital Group, Cromemco, North Star, Southwest Technical Products, and others who originally introduced product in kit form ..... essentially they're now all either out of business or only marginally in the kit business. As for assembled units, Zenith Data Systems (which started out as Heath Data Systems), is certainly a different market.

REMark: What is your answer to those who say Heath has abandoned the H-8 Users?

Johnson: We just introduced the soft-sectored double-density disk drive controller card for the H-8. This is a new product for the H-8. I would say there is as much or more existing software out there for the H-8 as any other model we produce. The product has matured, but we didn't just leave it, abandon it, as far as support is concerned. We didn't leave a half-done product line either, it has cassette interface, 5 inch floppies, and now, a double-sided double-density controller card provided for those people who purchased in the past. It has a full set of operating systems, HDOS, CP/M, Pascal plus a full set of languages MBASIC, FORTRAN, Macro80 and COBOL80. It has 64K of RAM. It's expandable. It has a wire-wrap card, parallel card, four-port serial card and we came out with a Z80 CPU for the product. Bread-boarding and extender cards as well as color graphics and music synthesizer board are available. New technology says there are now better ways to do things than the way they were implemented in the H-8. The H-8 is 1976-77 technology. Progress dictates that we continue to move forward. But, the word "abandoned" is the word that really bothers me. We have not abandoned those people. True we are not going to be adding substantially to the H-8 family but, H-8 owners can continue to call us here, talk with our hardware and software consultants and expect continuing support. Our service and parts support will likewise continue as it does on other products for seven years from date of product discontinuation. So nobody's going to get left in the cold. I think most people would want our company looking more forward than backward. We didn't half finish the H-8 family. It's rather a complete grouping of hardware and software that will continue to be supported by Heath Company.

REMark: Now that the H/Z100 is coming out will the H/Z89 be history?

Johnson: I could go back and reiterate my H-8 comments. Same deal. Some day we'll quit selling the 89s, because the cost, features and performance of some newer products are simply going to out date it. But when that day comes, we won't leave an orphan. It's a full line. All the peripherals are there, the major operating systems, the major utilities, languages and all kinds of accessory boards. It's a family of '89, and '90 product that we will continue to support. Yes, there will be a day when we will make our last '89, but we are not abandoning those customers. Incidently, our '100 series product does carry an 8-bit processor to look back at the 8 bit software investment that many of our customers have made.

REMark: In the future, how much production will be in kits as opposed to finished computer products?

Johnson: The sales of finished product is going to grow faster in terms of dollars than kit products simply because the market for assembled products is so much larger than the market for kit products. There are a certain number of people who will build a kit computer. They happen to be very loyal and a very select niche of the market but, it is a small niche of the market. Many, many more people or businesses will want to buy an assembled product. I will say that in both kit and assembled product, we are selling more computers this year than we did last year. The same was true of the year before. So both markets are growing. But the kit market is a rather finite market. We already have the lions share of that market. We don't yet have that kind of market share in the assembled computer business. Lots of opportunities there. Given that the assembled market is so much larger than the kit market, the answer to your question is that we will be producing substantially more assembled product than we do kit product.

REMark: Do you see the Heath (computer) market changing? If so, in what direction?

Johnson: When you talk about market .... the Heath Market, I'll say that nothing is as consistent as change. And the micro market is a class A, number one example. In fact, the whole kit business is a prime example of that. As you probably know, roughly 60 percent to 70 percent of our business this year comes from products that didn't exist two to three years ago. We continually need to be out at the leading edge of technology to seek different ideas. The Heathkit customers are a group of people who continually like to be first, they're the avant-garde people who probably had the first Porche on the block, the first digital watches on their wrist, the first video tape recorders in their house, the first satellite dishes in their back yards and the first solar heater for their hot water. They are not interested in yesterday's technology.

There are segments within the kit market. In the last four to five years there has been a huge growth in the scientific/educational aspects of our business. It turned out to be an extension of one of Heath's, distinctive competences, and that's writing manuals that are extremely clear and concise. They lead a customer to learn something relatively complex in step-by-step fashion. We said "why can't we apply that concept to learning digital electronics, or learning about microprocessors or learning AC or DC?" We took our step-by-step kit manual writing talents and applied them to a step-by-step learning process. It worked!

The whole (micro-computer) business started out basically with two, very highly technical, kinds of people. There were hardware freaks and there were software freaks. They bought and built those first computers because they wanted to know about the bits and the bytes, the flips and the flops, and how these things worked. They would put up with interminable problems to get themselves through all of that. The market then moved from those pure hobbyist people to embrace the educational and scientific community. Scientifically oriented people started picking up computers. Started using them for scientific and educational purposes. In the meantime, everybody played games. Then, finally, the software came along that allowed a computer to really do something practical and cost saving. I'm talking about word processing business packages and the spreadsheet products .... VisiCalc, SuperCalc, and boom the market exploded.

For the Heathkit business, we see some big, interesting things coming in the future. Things that have to do with alternate energy, robotics, telecommunications, and there are some terribly interesting things that are going to be happening in the area of voice

synthesis and voice recognition. A lot of products in the future are going to talk to you. If you look back, at Heath we had the very first digital clock. You can now get one if you open a bank account. We had the first digital readout on the screen for a television set. Now it's a standard industry feature. Typically Heath is successfull at the front end of those market places because we have product or product features no one else has. Ultimately when manufacturers start making them by the hundreds or thousands a day, we have to look on to newer things that we can introduce to our customers. Lots of opportunities! Our problem is really selecting what we are going to do. Tough choices, because there are lots of exciting ideas.

REMark: Do you feel the Heathkit concept of the kit computer and the Heathkit customer are being overshadowed by Zenith Data Systems?

Johnson: No! There are two different markets. There will always be a kit market. Those people who really want to understand computers, who are really curious and inquisitive and want to know more. They'll take the time and effort to learn programming and understand it. They like the challenge of understanding computers. It's a special market, a small niche, five percent or less of all the micro users.

On the other hand, Zenith Data Systems is consistently gaining market share in assembled computer systems. Five points of market share in the assembled computer market is probably greater than the entire kit market. I think there's great synergy between ZDS efforts and Heath's kit business because, together, we can bring tremendous resources to bear on our individual markets ... engineering, support, software and all the things that I mentioned earlier that I think are going to be extremely important to the winners .... the survivors .... in the micro market.

REMark: Why hasn't Heath or Zenith brought out a low cost computer such as the VIC-20 or the TRS-Color computers?

Johnson: I can't speak directly for Zenith or ZDS. At Heath we have our hands full addressing the market we have targeted .... basically the kit niche. I don't see Heath in the very low cost home computer business for two reasons. First, it's a mass market and we're not a mass marketer. Second, I think the home computer buyer is not a kit builder. A soldering iron and a box of electronic parts would terrify 99% of the potential buyers for a home computer.

REMark: What is your reaction to the First HUG National Conference?

Johnson: I loved it!I think the HUGgies loved it too. It was a huge success for both of us I think .... what else can I say?

REMark: How do you feel about the competitive advertising in REMark (as a source of income for the National HUG Conference)?

Johnson: Doesn't bother me a bit. If those advertisers don't put their advertisements in REMark, the're going to put them in BYTE or Interface Age, Personal Computing or somewhere else. I think it's great to get all those goodies exposed in one place where they are directly available to the Heathkit and Zenith Data Systems users. The individuals who have a true interest can then see what is available for our family of products. HUG gets the advertising revenue to help the overall organization and to help with the HUG convention. I wouldn't like to see Apple advertised in REMark, but I can't tell HUG what to do .... and I won't.

REMark: If we have a second HUG National Conference would you be willing to attend and speak?

Johnson: I'd be delighted!

REMark: How important is HUG to the rest of the Heath Company?

Johnson: The answer is, **VERY!** HUG is the strongest single source of customer support, objective criticism, suggestions and ideas that we have. It's a valuable sounding board of end users who don't hesitate to tell us about what they like, what they don't like, what they want and what they don't want. HUG members are a tremendous first hand source of the absolute best market research information any company could get. These are people who are using our products, they are knowledgeable and outspoken .... a great source of information on what the real live end user market wants. It's a mix of people all the way from hobbiest to businessman. ZDS customers are members of HUG just as well as Heathkit customers. So it's a great mixture of people, they are very vocal .... Love it! You probably know that Heathkit Electronic Center retail facilities are made available throughout the country for local HUG meetings. We encourage those things. HUG, as a third party operation, can do a lot of things. They (HUG) know that the company encourages their input ... good or bad. Not that we'll do everything they suggest, but I can tell you we listen seriously.

REMark: What are your feelings about alternate publications for Heath/Zenith support? (e.g. H-Scoop or Sextant)

Johnson: I feel a lot about them as I feel about HUG. Although the publications you're talking about are independent, they simply broaden the communications and information base for Heath/ZDS products that are out there. I think that's good. It also helps keep HUG on its toes .... good healthy competition and different points of view. It's a free country and competition has always worked for the benefit of the consumer. I think the more information that's out there that helps spread the word about things that

can be done with Heath/ZDS family of products, the better. Even if some of that is critical, we need to have those sounding boards not unlike HUG.

REMark: What are Heath's plans in software development, expansion, etc.?

Johnson: A big, complex and important subject. Software, I think, is one of the big keys to success for anybody who is going to be one of the ultimate winners in the computer business. Software along with excellent service, support, and certainly distribution. I think the computer itself, the CPU box, is going to get relatively transparent to the ultimate user. He doesn't care. In fact, they'll almost be alike anyway. It's what can I do with it? .... and that's a thing called software, user friendly software.

As you probably know, the Company right now has three different flavors of software. We have, 'Mother and the Flag', fully documented, fully supported software with updates and revisions, and we put the Heath/Zenith name on it. That software is very expensive to develop, and time consuming to develop, but we think the 'Mother and the Flag' products ought to be there for operating systems, key languages, key utilities and certain universal applications, word processing, spreadsheet products, business packages and data bases, etc.

We have SoftStuff, another class below the "Mother and the Flag" software. These are products where we find a very neat piece of software in the open market place that we confirm runs well on our machine. We take a look at the documentation, possibly clean it up a bit, but we don't do the whole 'Mother and the Flag' documentation job on it. It's less expensive, it's less sophisticated and, it's relatively inexpensive by comparison to our fully documented software. And then of course, we have the whole libary of HUG software, which I sort of look at as the flea-market of the software business. It's cheap, the Company (Heath or Zenith) do not stand behind it. If you have a problem, go find the guy who wrote it. But, here it is. It does run on our machines. It **may be** full of bugs .... 'caveat emptor' (buyer beware). But, have some fun. And of course, there is the other side to all this. All the independent third party software that is written and marketed by other people. In order to be first on the marketplace, you can't take six or eight months to fully document and quality check every line of code. That's why we're sometimes not there when we'd like to be. But we just haven't found a way to do it good ... and also fast. Approximately three years ago we had six people in software. The number today is fifteen times that number ... people who are working on all kinds of software, the writing of software, the customizing of software, the quality control of software, the editing of software, and the manufacturing/quality control of software. That gives you an idea how important software is to the Company.

REMark: Do you think the computer kit will ever return to the original basic (bits & piece) kit?

Johnson: It depends on what you mean by, quote "basic kit". CPU cards were provided in assembled form starting with the H8 kit. Likewise the CPU and Terminal Logic boards of the H89 are assembled and tested here. These components are too complex to expect the kit builder to successfully build and test, or require equipment that the normal kit builder wouldn't have. Nobody's going to actually assemble a disk drive. It's a sub-assembly that gets dropped in. But, that isn't anything unusual, we've been doing that to kits for many, many years. Audio products have always had assembled tuners and assembled IF strips that required factory alignment. High voltage power supplies have always been supplied assembled in our television sets, as are the prebuilt sections in our ham gear where there are tuned circuits or other sections of the circuitry that the normal kit builder would not have the equipment or capability to put together.

Yes, I think the nature of kits is changing, but there will always be lots of work for the kit builder. I can remember back when circuit boards came along, they said, "Oh! There goes the kit business", because it was previously all point-to-point wiring. Kit customers found they had a lot of fun loading up the circuit boards with parts and solder. Then along came transistors. They said, "Wow! One transistor in place of 15 components, there goes the kit business." It didn't happen. Then along came IC's. They said "Whew! You can put 150 components on an IC". And then, the microprocessor. What really is happening is that those components, as technology progresses, simply allow us to make more sophisticated products than we ever could have before.

If we tried to make a computer out of discrete components .... it would be huge! What I'm really saying is that these things that have happened over time have not proven to be viable threats to the kit business. What they really have done is expand the horizon of what can be done in a kit.

Vectored from 8

I have enjoyed REMark a lot. I have gotten quite a few hints and helps from your articles.

I also found a source of 80 track TANDON drives:
COMPUTEX, 321 El Dorado Blvd, Webster, TX 77598

The drive is $465 which includes shipping. The device driver from HUG works fine, but if you forget to "TYPE SPACES" when first BOOTing, you will scratch your head for a while.

Hope to have the DUP program modified shortly. Will let you know when I get it done.

Looking forward to many more fine issues of REMark.

Dean B. Peterson Jr.
1001 Harmon Place
Longmont, CO 80501

Dear Walt

1. The members of the OKINAWA Heath Users' Group are elated to see the new change in REMark Magazine. Specifically, the names not only of personnel who submit articles, but also of those who submit new software are now included.

2. Having prinde in submitting a software item to HUG or even for "Jingles in your jeans" may encourage many, but, seeing one's own name in print is a nice (inexpensive?) touch that we are sure will motivate members to support HUG more. This form of reward that has no tangible gain other than recognition or even possible inclusion to a resume of professional development can be an adequate reason for the submission of software items. Some amateur computerites, when they write at all, are better able to write in languages that machines, rather than people understand. We feel that they should be recognized too.

3. As you are the new Editor of REMark the credit for this new policy change probably goes to you. We thank you.

4. NRN

For OKIHUG
Carl H. Eaton

ED:) Thank you Carl, but I can't take the credit for the names being included on software abstracts. This appropriate inclusion comes to you from Terry Jensen, Software Developer. We like to give credit, when we can, to everyone who contributes to HUG.

Vectored to 37

SEXTANT

CHARTER OFFER

Get the Independent Magazine for the Entire Heath/Zenith Computer Community.

More information, innovative ideas, and supplier advertisements are available to you now!—All in an independent magazine for the Heath/Zenith user.

Sextant's first year is already well under way, with lots to offer you. Read articles which have been carefully selected to give you the most interesting and useful information about your system. Learn how to explore your system's capabilities to the fullest. You'll benefit from graphics which expand and clarify the articles. And there are over 50 advertisers in *Sextant*, all of whom have products compatible with Heath/Zenith systems.

You can still get in on our charter offer by subscribing now. Save 15% off the cover price. You pay $19.94 for two years, or $9.97 for one year. As a charter subscriber you'll always get our lowest subscription price offers. Plus, our first two issues are still available, so you can get a complete supply of information.

Order now! Call toll free:

DATATEL™ **800/341-1522**

M-F, 6 a.m.-9 p.m. EST. For orders only.

Charge your subscription to VISA or MasterCard, or we'll bill you later.

To order by mail, simply send us the coupon below and place your order for:

☐ two years/8 issues for $19.94
☐ one year/4 issues for $9.97
Include the following issues in my subscription:
☐ Premiere (Spring 1982) ☐ Summer 1982
☐ Fall 1982 ☐ Winter 1983
☐ Payment enclosed ☐ Bill me
☐ Charge my:
☐ Visa ☐ MasterCard (Code # ____)
Card # _____
Expiration date _____

Name _____

Address _____

_____

Guarantee: Your money will be refunded in full if at any time you are not satisfied with *Sextant*.

Send today to: *Sextant* Fulfillment Dept., 716 E St., S.E., Washington, DC 20003.
202-544-0900

Premiere (Spring 1982)

Heath/Zenith Computers: An Interview and Overview, by John Walker and David Johnstone . . . growth in the Heath/Zenith marketplace.

A Professional Writer Looks At Budget Word Processing, by Hugh Kenner . . . **With an Introduction by William F. Buckley, Jr.**

How to Turn Zenith Screen Graphics into Color Slides, by F. X. "Skip" Millor.

Benton Harbor BASIC Tests Typing Reaction Time, by Raymond Dotson . . . with a program to help you increase your typing speed.

Disk Programming *Without* HDOS, by Richard Smith.

Print Spoolers: A Background and Reviews of Six Software Products to Break Your Printer Bottleneck, by Kenneth A. Patrick.

Rubik's Cube: A Computer Simulation in Microsoft BASIC, by Arthur A. Frost.

A Candid Look at Tiny Pascal, by Arnold R. Madeira . . . a budget version of Pascal which could suit your needs.

A Parallel Interface for the '89, by J.C. Hassall.

What's A Company Like Zenith Doing in the Computer Business?, by Jerry K. Pearlman.

Summer 1982

Heath/Zenith Independent Support Highlights the 7th West Coast Computer Faire, by John Walker.

What to Do with Your H88/89 After the Movers Have Finished Playing Soccer and Dropsy with It, by David Esche.

Crucial Secrets of HDOS Let You Write Device-Independent Programs, by Bill Parrott.

Renumbering Benton Harbor BASIC, by Raymond Dotson . . . to help you clean up the mess BASIC can get you into.

***The Buss Directory* of Independent Suppliers for Heath/Zenith Users: Revision 6.**

The First Ever: National Heath/Zenith Users Conference.

Put Some Structure in Your BASIC Programming, by William Clarkson . . . clearing up the clutter of programming BASIC on your Heath/Zenith system.

Introducing: The Brain of the Z-Machine, by Al Dallas . . . the central processor chip that makes the Z100 line both powerful and economical.

New Zenith Computer Line Offers Something for Everyone, by Charles Floto . . . a look at the Z100 series which includes full color pictures of the interiors *and* exteriors.

Fall 1982

The H8's Fifth Anniversary: Origins and Outlooks, by Frederick Zimmerman.

Putting the "O" in OEM: Magnolia Microsystems, by Hal Glatzer . . . how this family business has expanded.

Experiment with Artificial Intelligence on your Heath/Zenith Computer, by John Mays . . . a review of John Krutch's book on how to use your Heath/Zenith microcomputer to explore artificial intelligence.

The World's Fastest Sort?, by Rick Lutowski . . . mathematics are the answer, not new equipment.

How to Save Kilobucks with a Business Microcomputer, by Don Carter . . . the H/Z89.

Give Your '89 a Boot—Automatically!, by Ray Albrektson.

Budget Word Processing: A Followup, by Hugh Kenner.

Double Precision Trig Functions, by Hugh Kenner . . . a few helpful programs.

Public Domain Software for Your H11, by Edward Judge . . . what *is* available.

The Z100 User's Manual: Clear, Concise and Useful, by Frederick Zimmerman.

Graphics on Your H8, by William Garner, Jr. . . . in color!

Give Your H89 A Bigger Bite

(A Review of the Mako MH89 Plus 3)

Pat Swayne
Software Engineer

The MH89 Plus 3 by Mako is an accessory for H89 computers that allows you to install 6 boards (instead of the usual 3) in the right hand side of your H89 computer. It is a separate pc board that mounts "piggy back" on to the CPU board over the original 3 expansion slots, and replaces them with 6 slots of its own. Photo 1 shows the board installed in an '89 with 3 accessory boards plugged into the right three slots. These slots duplicate the original three, while the other three each have the 4 port lines (SER 0, SER 1, LP and CASS) decoded into separate banks, so that two boards using the same address line can be used without conflict. There is provision to connect the left three slots to an external power supply. Notice the wire bail holding the boards in place. This "Mako Jaw" makes it easier to remove and replace boards than the bracket and screws normally used. It does not hold the boards as securely as the bracket, but unless you regularly throw your computer in the back of a pick-up truck and haul it around, it will hold them

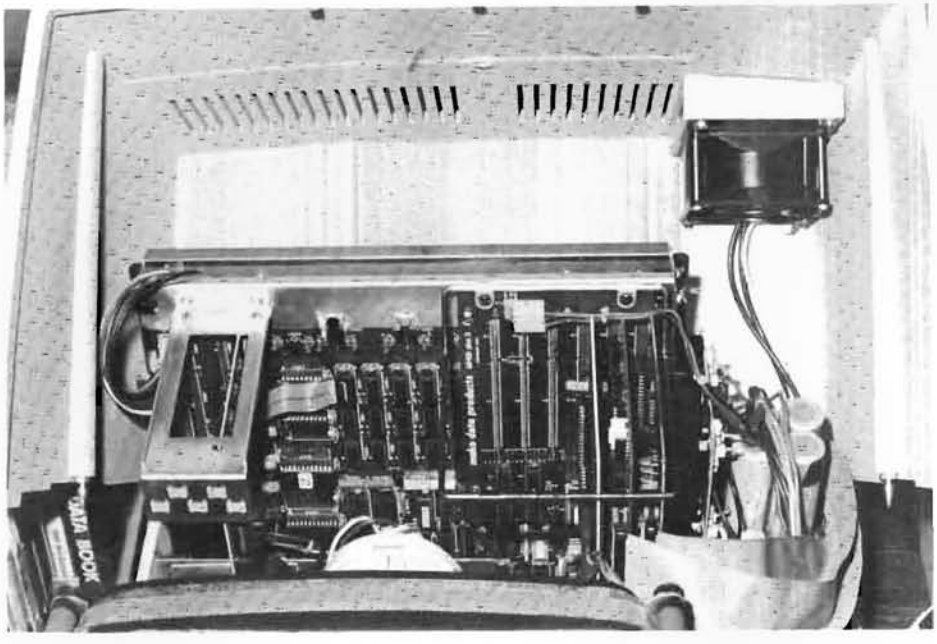


Photo 1. The inside of an '89 with the MH89 in place.

well enough.

Installation

The MH89 Plus 3 installs as easily as the instructions indicate. You have to remove your CPU board and one IC on the CPU board, but there are no trace cuts or jumpers to bother with. The board plugs onto the rightmost accessory pins (P506 and 512), the power pins (P516), and the vacated IC socket (U550). Everything was lined up just about perfectly on the one I received and it slipped right in. Once installed, a screw holds the MH89 securely to the metal bracket at the top of the CPU board. The IC removed from U550 (the IO decoder) plugs into the MH89.

The instructions are well written and easy to follow. In fact, a section on optional power supply modifications looks a lot like the instructions included with the Z-89-37 board, and Pictorial 3 is almost a dead ringer for page 3 of the Z-89-37 Illustration Booklet. Well, I guess that just goes to show you that Heath instructions really are the best, and no one can improve on them.

Evaluation

The MH89 pc board is well laid out and appears to have been thoughtfully designed, at least from the electronics standpoint. The one I received runs with no problems, even at 4 MHz. It was hand soldered, which was well done except for one missed connection (see Photo 2). The accessory pins in the unit are higher than the original ones, which compensates for the fact that boards plugged into it are closer to the front of the computer. The MH89 board itself is higher than the top of the CPU board, and this brings up one problem I had. After I installed the board and was replacing the cover on my computer, I looked in while the top was still open about half an inch (1.3 cm) and noticed that the cover had already contacted the MH89 board. I saw that the cover would push the board down severely if I closed it all the way, so I removed the board, removed the wire bail from it, and snipped off about 3/16th of an inch (.2 cm) of the top edge of the board. Then I bored the wire bail mounting holes so I could mount it lower, replaced it, and replaced the MH89 in my computer. The cover still contacts it slightly. The problem could be solved if the wire bail assembly were mounted on the front of the pc board instead of on the top edge. Apparently, variations in H89 cabinet dimensions are greater than anticipated by Mako.

The accessory slots on the MH89 Plus 3 are closer together than the original slots, and, as the Mako documentation warns, you have to make sure that two adjacent boards do not make electrical contact. They mention that the main problem is open topped programming plugs, but I would also watch vertically mounted electrolytic capacitors, whose metal "cans" may or may not be electrically connected to one of the capacitor's leads. The closeness of the boards also takes away some of the ease of board insertion and removal brought about by the "Mako Jaw".

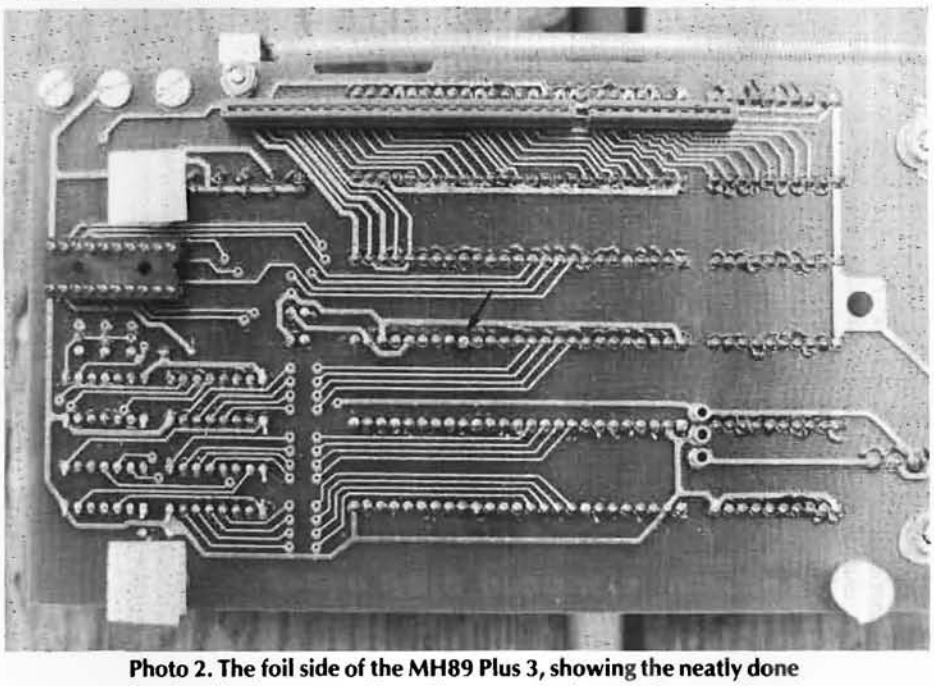


Photo 2. The foil side of the MH89 Plus 3, showing the neatly done hand soldering connections. The arrow points to the unsoldered connection.

As I mentioned before, the ports on the left three slots are decoded into separate banks, which for most purposes would be desirable. But a situation could arise where you might want to have the original ports on one of the extra slots. For example, the HA-89-3 color graphics board normally uses port 320 octal, which is OK in my system because that is the "middle" port on the 3-port serial interface that I do not use. Since all of the software I have for the color board uses port 320, I would want to keep it at that address, even if I put it in one of the left three MH89 slots. The MH89 appears to have been designed with that possibility in mind, because all of the port decoder lines pass through traces on the top of the board (labeled L1, L2, and L3 — see Photo 3), and those traces look like they were just made to be cut. So I could cut the appropriate trace and add a jumper from one of the right slots to the slot I wanted to use, bringing over the port 320 line, and use my color board in one of the extra slots.

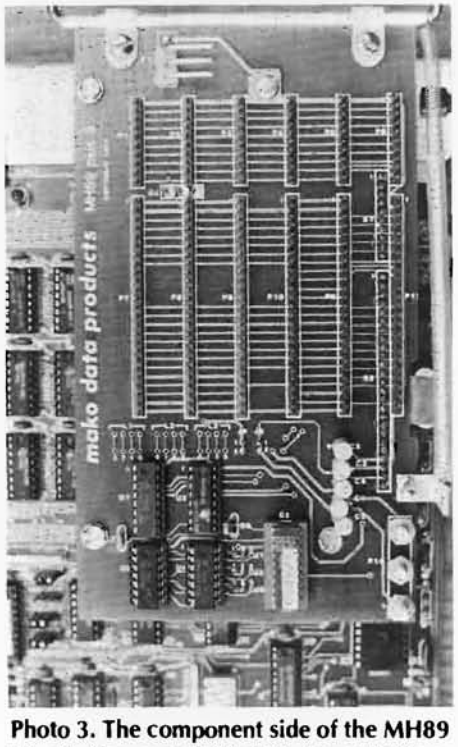


Photo 3. The component side of the MH89 Plus 3. The areas labled L1, L2 and L3 are the traces through which the decoded port signals pass.

Conclusion

The Mako MH89 Plus 3 is a well designed product, with only a minor mechanical improvement needed (the height problem). It installs easily, and works perfectly. If you are looking for a way to put more boards in your H89, it may be the solution for you.

The MH89 Plus 3 is available from Mako Data Products, 1441-B N. Red Gum, Anaheim, CA 92806, Phone (714) 632-8583. It sells for $150 plus $5 shipping ($8 foreign). California residents add 6% sales tax.

The Next Step Is C

William G. Bently
4788 Norfolk Circle
Portage, MI 49002

INTRODUCTION

The first language used by most hobbyists is BASIC, since it is easy to learn and comes with a built-in, easy-to-use programming environment. But after learning some of BASIC's limitations (usually the hard way), many hobbyists begin to look for a more advanced programming language that will provide them greater freedom and greater power over their machines. The most powerful language is assembly language, but assembly language is often difficult to master. Since assembly language is just a symbolic form of the actual machine language, it is very low-level and much effort is expended in performing even simple operations. This effort can be reduced by using higher level languages, which typically reduce the mental effort by about a factor of ten. But which high level language should one choose? Programming languages are complex and difficult to objectively evaluate. Even the experts cannot agree on what criteria should be used to evaluate languages. What is the poor hobbyist to do? He could stick with BASIC and ignore all the progress in language design that has been made since the mid-sixties when Kemeny and Kurtz invented BASIC. A popular answer to this dilemma is PASCAL. But is PASCAL really the best solution? In this article, an alternate viewpoint will be presented. The next step for the serious hobbyist should be the C programming language. This choice can be best defended by carefully examining some of the concepts underlying modern programming languages.

ANCIENT HISTORY

Computers were invented in order to reduce the amount of work involved in performing repetitive and laborious mental calculations. Since that time, scientists have been struggling to invent languages which will reduce the amount of work required to program a computer. Thus, in his great wisdom, man has replaced one tedious and menial task (i.e. calculating) with another tedious and menial task (i.e. programming).

A programming language may be defined as a symbolic means of communication between man and computer. There are two "natural" languages; the language of man (in our case, English) and the language of the machine (binary). The programming language must somehow bridge this gap. Although English is being touted as the ultimate computer language, it is quite deficient. It is much too wordy and not logically precise. Machine language is very precise, but too cryptic (just ask anyone who has programmed in octal). Somehow, the language must be efficient, yet comprehensible to both parties.

In the 1950s it was thought that "automatic programming", brought about through the use of high level languages, would make the programmer an extinct relic of the past. Unfortunately, even though scientists have worked prodigiously and prolifically in creating new languages, there are still many of us "relics" around! Most contempory languages evolved as solutions to problems encountered with earlier languages. Let us examine some of these problems and their contemporary solutions in more detail.

FUNDAMENTAL PROBLEMS

Have you ever written or worked on a BASIC program longer than six pages? Have you ever tried to make a modification in a long BASIC program? If your answer is affirmative, then you may have experienced some of the following difficulties;

1.) Your program does not work properly, and only after the most tumultuous hair-pulling debugging session you discover a collision of variables, i.e. your present use of a variable name conflicts with a previous (and forgotten) use of the same variable name that appeared earlier in the program.

2.) Your program is executing a section of code that should not be executed with the input you have supplied the program. As you begin to follow your goto statements, you feel like you are playing ADVENTURE and sooner or later become totally lost. This type of code is often called "spaghetti" code, since it goes all over the place.

3.) You must go back and make modifications to a program you wrote several months ago. After long hours of looking at handwritten notes made in the margin of the program, you still cannot remember how it works.

4.) You have rewritten your program ten times and it still executes too slowly and takes up too much memory space. You could try assembly language, but then your program may take too long to develop.

5.) One of your best friends visits you with his new brand X computer. Wouldn't it be wonderful to share with him some of the programs that took you months to write? But alas! After downloading your programs, you find that your BASICs are totally incompatible.

Experienced programmers often try to remedy some of these problems by defining their variables in REM statements and making extensive use of subroutines. Such techniques are helpful but not completely satisfactory.

Answer to Problem #1: MODULARIZATION

Engineers faced the problem of building large complex systems long ago. Their usual solution is to reduce complex systems into a hierarchy of smaller, more managable units. Similarly, a large complex program can be broken down into smaller units called modules. Each module performs a single, simple easily- comprehended task. Modules are independent of one another; a small change in one module does not necessitate extensive modifications to other modules. Modules are combined as uniform building blocks in order to construct larger programs which perform more complex processing. A fundamental concept underlying modularity is that the module can be used without knowing how the module handles data internally. The variables used within a module are not directly accessible outside the module, thus preventing the collision of variables. For instance, if the variable "bytecount" appears within a module, it will

not take the value of the similarly-named "bytecount" that appears outside the module. This allows software modules to be used in different contexts. If each module is independent of other modules, they may be easily recombined in new ways to construct entirely different programs. Some call the module a software component and view each module as an extension to the instruction set. This is the approach taken in the ADA language. In C, each module is called a function and is represented in a manner similar to mathematical functions. For example mathematicians often write a function of x as "f(x)". In C, the function is written as the function name followed by the argument (or parameter) list given in parentheses (see LISTING). A C program is just a collection of functions. Standard collections of functions may be stored in special files called libraries. Each module may be compiled spearately before all are linked together to form the final program. Thus when making modifications to a program the only modules that must be recompiled are those which receive the modifications. Since compilation can be a lengthy process, this saves considerable time as compared to recompiling the entire program.

The ability to manage complexity is an important advantage. Most beginners are not aware of this problem since their programs are usually not very complex. Long BASIC programs can be exceedingly difficult to debug, whereas long C programs that are properly modularized are usually no more difficult to work with than small ones.

Answer to Problem #2: STRUCTURES

The spaghetti code problem has been solved through the introduction of programming structures. The main intent of structures is to provide program flow that is easy to follow. Structures organize program flow in much the same way modularization organizes functionality. Complex program flow can be viewed as just a hierarchy of simple, easily-comprehended control statements. C possesses most of the control structures found in PASCAL. This includes statements for decision (if-else), iteration (for and while) and selection (switch).

C also provides a method for organizing data. Several variables of different types may be grouped together and treated as a single unit called a "structure" (not to be confused with the control structures mentioned previously). This construct is called a "record" in PASCAL. These methods of organizing code are used in professionally written programs which may contain thousands of lines of code. Such large programs would be impossible without them.

Answer to Problem #3: READABILITY

As stated above, the earliest and most primitive language, machine language, is extremely difficult to read, even in octal! One of the first high level languages, COBOL, was designed to have English-like statements in order to circumvent this problem. Unfortunately, this makes COBOL a "verbose" language and many programmers prefer abbreviations. PASCAL tends to be wordy. For instance, blocks are delimited by the words "BEGIN" and "END". In C, the brackets '{' and '}' serve the same purpose and are just as readable. C, like PASCAL, provides long variable names, instead of the obscure two-character names used in BASIC. The concise notation of C can be cryptic; thus C programs should always be properly commented.

Answer to Problem #4: EFFICIENCY

Efficiency is especially important on small computers such as the

```
*************************************************************
        LISTING - SIMPLE EXAMPLES OF C FUNCTIONS
*************************************************************
/*-----------------------------------------------------------
|                                                           |
|                       BOX                                 |
|                                                           |
|      Draws a box on H19 screen at line#, column# of       |
| height (hi+2) and width (wide+2).                         |
|                                                           |
|      INPUTS                                               |
|              line    line# of upper left edge             |
|              col     column# of upper left edge           |
|              hi      height of box - 2                    |
|              wide    width of box - 2                     |
|      CALLS                                                |
|              cur()                                        |
|      INITIALIZATION                                       |
|              printf("\033F");   GRAPHICS MODE             |
|              HEATH MODE                                   |
|                                                           |
-----------------------------------------------------------*/

box(line, col, hi, wide)
char line, col, hi, wide;
        {
        int i;

        cur(line, col);
        putchar('f');
        for(i=0;  i<wide;  ++i)          /* TOP */
                putchar('a');
        putchar('c');
        for(i=0;  i<hi;  ++i)            /* RT. SIDE */
                printf("\033B\033D`");
        printf("\033B\033Dd");
        for(i=0;  i<wide;  ++i)          /* BOTTOM */
                printf("\033D\033Da");
        printf("\033D\033De");
        for(i=0;  i<hi;  ++i)            /* LFT. SIDE */
                printf("\033A\033D`");
        }
```

```
/*-----------------------------------------------------------
|                                                           |
|                       CUR                                 |
|                                                           |
|      Positions cursor of H19 screen at line# and          |
| column#.                                                  |
|                                                           |
|      IN                                                   |
|              line    line#                                |
|              col     column#                              |
|      INITIALIZATION                                       |
|              HEATH MODE                                   |
|                                                           |
-----------------------------------------------------------*/

cur(line,col)
char line,col;
        {
        printf("\033Y%c%c",line,col);
        }
```

H89. Memory space is quite limited and execution is slow. Therefore the code produced by the compiler must be efficient in both time and space utilization. Because it is compiled, C code executes much faster than BASIC code which is interpreted. C's excellent performance has been confirmed in standard benchmark tests (reference 1). One of C's advantages over other high level languages is the conciseness of its expressions. C has many operators that are represented by only one or two special symbols. For instance, the value of a variable may be incremented by preceding it with the operator "++." These operators are not difficult to learn, since they are constructed in a very consistent manner. For this reason, C source files tend to be smaller than similar source files written in other languages. A good example of the efficiency of C and its practicality on the H89 is the fact that several of the excellent programs available from Software Toolworks were written in C.

C is so efficient, that it has begun to replace assembly language in many demanding applications. Other languages have been coded in C. For example, Whitesmiths, Ltd. (Parkway Towers, 'B,' 485 US Route 1 So., Iselin, NJ 08830) provides a version of PASCAL written in C. Another proof of C's efficiency is the fact that several operating systems are written primarily in C (see the discussion below). Most neophytes do not appreciate the experienced programmer's obsession with efficiency, since BASIC is adequate for producing programs that are short and simple. It is not until they encounter programs that perform extensive sorting and searching that they begin to perceive the need for languages like C that can substantially reduce the otherwise lengthy execution times.

Answer to Problem #5: PORTABILITY

This is becoming an important issue due to the proliferation of hardware and its rapid obsolescence. More specifically, what will happen to our investment in H89 software after we buy Z100s? Since the fifties, it has been realized that there are too many different machines and that each machine has its own unique machine language. COBOL was designed to "solve" this problem.

Since C is modular, machine dependent functions can be relegated to a small group of interface modules. These provide input and output facilities (i/o) and are incorporated into the "standard i/o library." The source for these modules is usually a mixture of C and assembly language, since C can be combined easily with assembler. Although this library normally follows the de facto standard presented in Kernighan and Ritchie's book (reference 5), it may vary since i/o is not part of the C language. C can often be ported from one machine to another by simply rewriting the machine (or terminal) dependent modules. Although it is easy to port C from one microcomputer to another, it may be difficult to directly transfer a program from a microcomputer to a mainframe computer since some of the primitive C data types are machine dependent. Despite this limitation, C often proves to be more portable in practice than either PASCAL or FORTRAN IV.

C is Small

C is a system programming language. This means that the C language retains much of the power of assembly language, which is traditionally used for system programming. Because it is "close" to assembly language, C is often considered a "not very" high level language. In fact, C retains many of the capabilities inherent in macroassemblers (such as macro expansion, conditional compilation and the ability to include external source files). C was purposely designed to be a "small" language with only a minimal set of built-in functions. For example, C has no built in trig-functions (although they are often provided as part of a standard math library). Contrast this with PL/1, which is a very large language containing so many built-in commands that it is almost impossible for a human to master them all. This makes C easier to learn and more efficient. It also makes it possible to implement the whole language on machines with limited resources (like an H89). Its smallness should not be viewed as a limiting factor, since the language can be extended in a well-controlled manner through the use of libraries.

C and UNIX

C was originally developed as the system implementation language for the famous UNIX operating system from Bell Laboratories. UNIX and UNIX-like operating systems are becoming increasingly popular on the new 16-bit microcomputers. Z100 enthusiasts may be interested in knowing that Microsoft is working on a new version of MS-DOS which will be written in C. UNIX includes powerful software tools that facilitate the development of C programs. The availability of similar tools for use with C on other operating systems will enhance its popularity. Whereas many "fad" languages will have short lifespans, the popularity of UNIX and the extensive use of C within Bell System computers should do much to insure C's longevity.

PASCAL and ADA

As stated earlier, C's major competitor is probably PASCAL. PASCAL was originally written by Niklaus Wirth for the teaching of structured programming techniques. PASCAL is eminently suited for this purpose, but it has several failings when used for programming in the real world. The major problem is that it must be extended considerably to handle practical situations. Even Wirth concedes this weakness, and is presently advocating the use of his new language, Modula II, for real world applications. PASCAL has suffered many incompatible extensions and this severely restricts its portability. For instance, in its original definition (Wirth), PASCAL supports neither random files nor the separate compilation of modules. Features such as these must be added to the language in some nonstandard way. The new DOD language ADA, which is based on PASCAL, may become a strong contender, but most of the experts seem to feel that a full implementation of ADA will not be possible on machines of the H89 generation. Do not despair; one of ADA's principle features, the concept of software components, can be implemented (to an extent) in C. C is presently being used internally by many software houses and should gain popularity as libraries of standard modules appear on the open market. The availability of such pre-packaged, pre-tested building blocks will make it possible to program complex applications in a fraction of the time currently required. Automatic program generators promise similar benefits, but they cannot provide the flexibility possible with software components.

C for the H89

One of the best arguments for C's use by Heathkit hobbyists is the availability of high-quality compilers for the H89 which run under Heath's HDOS operating system. A compiler for a substantial subset of the C language is in the public domain. This great gift to hobbyists was written by Ron Cain and appeared in Dr. Dobb's Journal (references 2-4) as the Small C Compiler. This compiler has been used as the basis for producing more complete compilers such as C/80 from Software Toolworks (14478 Glorietta Drive, Sherman Oaks, CA 91423) which represents one of the best software bargains on the market. C/80 is a nearly complete implementation of the C language and runs under HDOS or CP/M on systems with a minimum of 48K of memory. An even more complete compiler is available from Manx Software Systems (P.O. Box 55, Shrewsbury, N.J. 07701). Although more expensive than C/80, it comes with its own assembler and linker. This makes separate compilation possible on HDOS systems that do not have Macro-80 (from Microsoft). Several other C compilers (mostly spin-offs of Small C) are available for CP/M systems.

Vectored to 23☞

NEW HUG PRODUCTS

885-8012 CP/M
Modem Applications Effector (MAPLE) . $35.00

Introduction: MAPLE is a modem program designed to allow the H8/H19 or H89 computer to communicate effectively with another computer, over the telephone or by direct connection.

Requirements: MAPLE requires the CP/M operating system, version 2.0 or later, on the H19/H8/H17 or H89 with 32K of memory. When transferring files, 48K or more memory is recommended. Only one disk drive is required. A line printer device is not required but is recommended for downloading files from the host computer.

To use MAPLE, a modem and appropriate connector will be required to communicate to another system. MAPLE can be set to run on interrupt 3, 4, or 5 of the DTE connector, port 330Q.

The program is written in assembly language and the source code is not included.

NOTE: *The H19 terminal is required due to the use of the function keys and other escape codes.*

The instructions are contained in a 22 page manual.

Author: Dr. William C. Parke

Program Content: MAPLE is a modem communications package, which can be used to connect the users Heath computer system to another Heath system or to a *time-share host system*, e.g. CompuServe or the SOURCE, via the telephone lines. (Time-share host systems can have several hundred users *on-line* at a single time.)

MAPLE, when executed, enters the *communications* mode, with full file send and receive control. The mode and options are displayed on the 25th line of the terminal and are invoked by depressing the appropriate function key. The main communication "menu" function keys displayed on the 25th line are as follows:

f1 - p.page — Send to the printer any text displayed on the screen. The user may edit the screen display before sending to the printer.

f2 - COPY — Store all screen activity in memory.

f3 - SHOW — Display on the screen the text which has been stored in memory with the *COPY* function.

f4 - P.PAD — Send the text saved in memory to a line printer device.

f5 - STORE — Save the text in memory to a disk file.

ERASE - CLEAR — Clear the text stored in memory.

BLUE - SEND — Send a disk file to the host computer. The file may be sent in *block, line,* or *segment* mode.

RED - Exit — Exit to CP/M.

WHITE - Set — Set or change communication options. These options include this *second level* menu:

f1 — convert between ASCII and APL character sets,
f2 — set mode for sending a disk file,
f3 — set the prompt character in sending a file,
f4 — select line printer device,
f5 — select disk drive,
ERASE — set options selected, and return to conversation,
BLUE — select parity check bit,
RED — select baud rate,
WHITE — set the number of bits per byte, and
BREAK — echo all terminal input.

BREAK - break — Send a *hard break* to the host system. This will stop any transmission by the host and return the user to the host's command prompt. (This feature may not work for all host systems.)

MAPLE uses a number of control characters and escape sequences to do a number of functions. The user can refer to the written documentation for details. *Patch Instructions* are included with the manual to aid the user in customizing MAPLE to his requirements.

Comments: MAPLE, one of the most complete modem packages, includes most any feature required for computer communications. The 25th line display "menu" and documentation make it a very user friendly package.

P/N 885-3003
CP/M-85 ZTERM $20.00

Introduction: This disk contains a modem package for the Z-100 series computers, which enables a user's computer to communicate with another computer. Versions for the H/Z89, Z90, and H8 computers are also included. ZTERM is a modification of the popular CP/M TERM package for the H89/H8 computers.

Requirements: This disk requires the CP/M-85 operating system for use with the Z-100 series. CP/M version 2.2.03 is required for the H8/H19, H89, or Z90 with 32K of memory. The H8 computer must be equipped with either the H8-4 serial interface board or the WH8-47 board. Only one disk drive is required for ZTERM.

The disk contains assembled, ready to use, versions for the Z-100 and H89/H8 computers. The programs are written in assembly language and the source codes are included. The user may modify ZTERM to execute on most any terminal.

ZTERM100.COM — for use on the Z100 version.
ZTERM89.COM — for use on an H89, Z89, or Z90 or H8 using port 340Q.
ZTERM340.COM — for use on the H8 using port 340Q.

NOTE: *The H19 terminal is* **NOT** *required for ZTERM with the H8 computer, but is highly recommended.*

Author: Jim Buszkiewicz

Program Content: This modem communication package allows the user to "talk" or *communicate* to another Heath/Zenith computer or to some other host computer system, e.g. CompuServe and the SOURCE. The package allows the user to change the port used, the baud rate, plus other parameters.

Editor's Note: CompuServe *and the* SOURCE *are two timeshare systems (mainframes) that allow hundreds of individuals to access the* host *computers at a single time. They provide services for these members, e.g. the HUG Bulletin Board.*

ZTERM allows the user to download a file from the host and it will let the user transmit a disk file to the host. The modem package will enable the user to communicate, in the normal fashion with a host, from the main console keyboard. The direct type communication can be either *half* or *full duplex*.

For anyone who uses *TYMNET* and/or *MicroNET*, ZTERM will automatically sign on to both systems by entering one simple escape character sequence. The same is true for those using TYMNET and the *SOURCE*.

General System Requirements: In order to use ZTERM, the computer must have a standard serial RS-232 I/O port. Either an unused or line printer port will suffice. Normally on a Heath/Zenith computer, the user would use the DTE port on the back of the computer.

For systems other than the Z-100 series, an H19 terminal is advantageous but not necessary. The main console must be capable of accepting lower case characters correctly, even though it may convert them to upper case.

Aside from the standard computer system, the only other hardware that is needed is an interconnecting cable, a host computer, and modem.

Configuring ZTERM: ZTERM includes an auto log-on feature, which configures ZTERM to send the User ID number and password. This may be done for MicroNET and/or the SOURCE. Selecting the baud rate is also software selectable. A menu will display the possible baud rates.

If there are other hardware or use requirements that are not met by the pre-assembled versions of ZTERM, the user can alter the assembly source code to meet the requirements. ZTERM was designed to run on just about any computer that can run CP/M, and there are several assembly options that make it easy to configure it to a particular system. The following are some of the possible source code changes:

1) Set for non-Heath system,
2) Set *pause* character,
3) Set *abort* character,
4) Set for a control-H (010Q) in place of delete (07fh), 5) Set for ZTERM to strip the high order parity bit,
6) Set for type and length of data word,
7) Set *port* value, and
8) Set *baud* rate.

Using TERM: ZTERM is not a fancy program that uses graphics or the computer horn, but it does get the necessities done in the most efficient manner. When ZTERM signs on, it displays the size of the storage buffer in decimal bytes. This buffer is used to store characters when down-loading data from the host computer.

While in direct communications mode, ZTERM has nine basic commands available. These commands are executed by entering the appropriate function key across the top row of the Z-100 or H19 keyboard. The command summary is as follows:

Note: *The function key columns refer to the H19 and Z-100 terminals respectively.*

H19	Z100
f1	*f1* — auto-log-on to MicroNET
f2	*f2* — auto-log-on to the SOURCE
f3	*f3* — Duplex toggle - half or full
f4	*f4* — Store all terminal display in buffer
f5	*f5* — Transmit disk file to host
ERASE	*f0* — Display Command Summary
BLUE	*f6* — Save the buffer to a disk file
RED	*f7* — Clear the buffer
WHITE	*f8* — Warm boot CP/M

When downloading files, no type of handshaking is needed. Characters are immediately stuffed into memory eliminating this need. When uploading files to a host, ZTERM will pause when it encounters a pause character.

Comments: ZTERM provides the Z-100 user (or anyone else) with a useful modem communication package.

CHARGE.....IT!

As many of you already are aware, Heath Company agreed long ago to handle the Heath Users' Group software products as we are simply too small to maintain the appropriate HUG inventory levels within our limited office space. Recently, Heath implemented a MasterCard program allowing customers to purchase Heath parts using Visa or their MasterCard. So, why not HUG?

You guessed it! Many of our users have requested the ability to order HUG products by phone using their Visa or MasterCard. With a few minor changes, HUG has now been included in the Heath program to add this increased convenience for our members. You may order any HUG product or products by phoning the Heath Company Parts Department at (616) 982-3571 ($10.00 minimum order, please). Be sure to have the HUG part number (885-) and product description available along with your HUG ID to ensure prompt delivery of the product you desire.

We wish to thank the Heath Parts Department staff for including the Heath Users' Group in their improved program for parts order entry.

HUG PRODUCTS LIST

NOTE: The number in the REM # column refers to the issue of REMark containing a description of the software. Usually, it refers to the "New HUG Sofware" column, but it may refer to an article.

Part numbers shown in **bold print** are available in soft sector 5.25-inch format. Add -37 to the part number to order soft sector. For example, to order 885-1206 in soft sector, use 885-1206-37.

Part Number	DESCRIPTION	Selling Price	REM #
CASSETTE SOFTWARE (H8 and H88)			
885-1008	Volume I Documentation and Program Listings (some for H11)	$ 9.00	
885-1009	Tape I Cassette	$ 7.00	
885-1013	Volume II Documentation and Program Listings	$12.00	
885-1014	Tape II ASM Cassette H8 Only	$ 9.00	
885-1015	Volume III Documentation and Program Listings	$12.00	
885-1026	Tape III Cassette	$ 9.00	
885-1036	Tape IV Cassette	$ 9.00	8
885-1037	Volume IV Documentation and Program Listings	$12.00	8

Part Number	DESCRIPTION	Selling Price	REM #
885-1039	WISE on Cassette H8 Only	$ 9.00	
885-1058	Volume V Documentation and Program Listings	$12.00	

HDOS SOFTWARE H8/H17 or H89 — (5-inch only)

MISCELLANEOUS COLLECTIONS

885-1032	Disk V H8/H89	$18.00	8
885-1044	Disk VI H8/H89	$18.00	
885-1064	Disk IX H8/H89	$18.00	
885-1066	Disk X H8/H89	$18.00	10
885-1069	Disk XIII Misc H8/H89	$18.00	

GAMES

885-1010	Adventure Disk H8/H89	$10.00	4
885-1029	Disk II Games 1 H8/H89	$18.00	8
885-1030	Disk III Games 2 H8/H89	$18.00	8
885-1031	Music 8 & 89 H8/H19 and H89	$20.00	25
885-1067	Disk XI Graphic Games .ABS and B H BASIC (H19/H89)	$18.00	12
885-1068	Graphic Games (H19/H89)	* $18.00	10
885-1088	Graphic Games (H19/H89)	* $20.00	14
885-1093	Dungeons and Dragons Game Requires H89 or H8/H19	* $20.00	16
885-1096	Action Games (H19/H89)	* $20.00	18
885-1103	Sea Battle Game (H19/H89)	$20.00	20
885-1111	HDOS MBASIC Graphic Games	* $20.00	23
885-1112	HDOS Graphic Games	$20.00	23
885-1113	HDOS Fast Action Games	$20.00	23
885-1114	Color Raiders and Goop (HA-8-3)	$20.00	23
885-8009	HDOS & CP/M Galactic Warrior	* $20.00	32

UTILITIES

885-1019	Device Drivers (HDOS 1.6)	$10.00	6
885-1022	HUG Editor (ED) Disk H8/H89	$15.00	20
885-1025	Runoff Disk H8/H89	$35.00	
885-1060	Disk VII H8/H89 SUBMIT, CLIST, FDUMP, ABSDUMP, etc.	$18.00	
885-1061	TMI Cassette to Disk H8 only	$18.00	
885-1062	Disk VIII H8/H89 (2 disks) MEMTEST, DUP, DUMP, DSM	$25.00	
885-1063	Floating Point Disk H8/H89	$18.00	
885-1065	Fixed Point Package H8/H89	$18.00	10
885-1075	HDOS Support Package H8/H89	$60.00	
885-1077	TXTCON/BASCON H8/H89	$18.00	
885-1079	HDOS Page Editor	$25.00	15
885-1080	EDITX H8/H19/H89	$20.00	
885-1082	Programs for Printers H8/H89	$20.00	
885-1083	Disk XVI RECOVER, etc.	$20.00	11
885-1089	MACRO, CTOH, and misc Utilities	$20.00	20
885-1090	Misc. HDOS Utilities CCAT, HPLINK, AH, MBSORT, etc.	$20.00	22
885-1092	RDT Debugging Tool H8/H89	$30.00	14
885-1095	HUG SY: Device Driver HDOS 2.0	$30.00	18
885-1098	H8/HA-8-3 Color .ABS/.ASM	$20.00	19
885-1099	H8/HA-8-3 Color in Tiny Pascal	$20.00	19
885-1105	HDOS 2.0 Device Drivers MX-80, Paper Tiger, Clock, etc.	$20.00	24
885-1116	HDOS Z80 Debugging Tool	$20.00	27
885-1119	B H BASIC Support H8/H19 or H89	$20.00	29
885-1120	HDOS "WHEW" Utilities	$20.00	33
885-8001	SE UCSD-Style Screen Editor	$25.00	28
885-8003	B H BASIC to MBASIC Converter	$25.00	28
885-8004	UDUMP and FAKEMNT Disk Manipulation Utilities	$35.00	28
885-8005	MAPLE Modem Program	$35.00	29
885-8006	HDOS SUBMIT	$20.00	31
885-8007	EZI-TRANSfer	$30.00	30

PROGRAMMING LANGUAGES

885-1038	WISE on Disk H8/H89	$18.00	
885-1042	PILOT H8/H89	$19.00	
885-1059	FOCAL-8 H8/H89	$25.00	13
885-1078	HDOS Z80 Assembler	$25.00	21
885-1085	PILOT Documentation	$ 9.00	
885-1086	Tiny Pascal H8/H89	$20.00	13
885-1094	HUG Fig-Forth H8/H89 2 Disks	$40.00	18

BUSINESS, FINANCE AND EDUCATION

885-1047	Stocks H8/H89	$18.00	
885-1048	Personal Account H8/H89	$18.00	
885-1049	Income Tax Records H8/H89	$18.00	
885-1055	Inventory H8/H89	* $30.00	
885-1056	Mail List H8/H89	* $30.00	
885-1070	Disk XIV Home Finance H8/H89	$18.00	
885-1071	Small Business Package III 3 Disks H8/H19 or H89	* $75.00	17
885-1091	Grade and Score Keeping	* $30.00	14
885-1097	Educational Quiz Disk H89 or H8/H19	* $20.00	18
885-1118	Payroll	* $60.00	30
885-8010	HDOS Checkoff	$25.00	32

DATA BASE MANAGEMENT SYSTEMS (DBMS)

885-1107	Amateur Radio Logbook and TMS	$30.00	23
885-1108	Telephone/Mail Info. System	* $30.00	23
885-1109	Retriever 3 Disks	$40.00	23
885-1110	Autofile 2 disks	$30.00	23
885-1115	Aircraft Navigation H8/H89	* $20.00	25
885-8008	Farm Accounting System	* $45.00	30

AMATEUR RADIO

885-1106	Morse-89 H8/H19 or H89	$20.00	22

* Means MBASIC is required

H11 SOFTWARE

885-1008	Volume I Documentation and Program Listings (some for H11)	$ 9.00	
885-1033	HT-11 Disk I	$19.00	
885-1053	H11/H19 Support Package EXEC Modem Software, etc.	$20.00	27
885-1117	Pirate's Adventure for H11/H19	$20.00	28

CP/M SOFTWARE (5-inch only)

885-1201	CP/M (TM) Volumes H1 and H2	% $21.00	
885-1202	CP/M Volumes 4 and 21-C	%% $21.00	
885-1203	CP/M Volumes 21-A and B	%% $21.00	
885-1204	CP/M Volumes 26/27-A and B	%% $21.00	
885-1205	CP/M Volumes 26/27-C and D	%% $21.00	
The above CP/M products are 2 disks each.			
885-1206	CP/M Games Disk	* $20.00	11
885-1207	TERM and H to C	$20.00	26
885-1208	HUG Fig-Forth H8/H89 2 Disks	$40.00	18
885-1209	Dungeons and Dragons Game MBASIC and H89 or H8/H19	* $20.00	19
885-1210	HUG Editor	$20.00	20
885-1211	Sea Battle Game for CP/M	$20.00	20
885-1212	CP/M Utilities I	$20.00	21
885-1213	CP/M Disk Utilities	$20.00	22
885-1214	Amateur Radio Logbook	* $30.00	23
885-1215	BASIC-E	$20.00	26
885-1217	HUG Disk Duplication Utilities	$20.00	26
885-1218	CP/M MBASIC Payroll	* $60.00	31
885-1219	CP/M Aircraft Navigation H8/H89	* $20.00	31
885-1220	CP/M Action Games	* $20.00	32
885-1221	Watzman ROM Source Code (2 disks)	$30.00	33
885-3003	CP/M ZTERM Modem Communications	$20.00	34
885-8011	CP/M Checkoff	* $25.00	32
885-8012	CP/M MAPLE Modem Program	$35.00	34

% Means CP/M 1.43 only (ORG-4200).
%% Means CP/M 1.43 or 2.2 (Heath).
MBASIC programs on these disks are for version 4.8 or earlier.
Other CP/M disks are for 2.2.
* means MBASIC is required.

MISCELLANEOUS

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

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

23

Vectored from 19

Getting Started in C

C is not a good language for computer neophytes! If your main background is BASIC and you decide to make the switch to C, be forewarned. C is a structured language; therefore C programs must be more carefully planned out ahead of time than BASIC programs. C must be edited, compiled, assembled and linked. This development process is neither as fast nor as easy as the interpreted environment provided in BASIC. C is more dangerous than PASCAL, since it does not perform rigorous type checking. This makes C quite powerful but beginners should be careful, especially in using pointers.

C operates on a fairly low-level. The serious student should first become acquainted with assembly language programming, since many of the basic data types and operations are similar. A C-like interpreter called Tiny-C (Tiny-C associates, P.O. Box 269, Holmdel, NJ 07733) may be of interest to those wanting to learn structured programming. The interpreter facilitates learning by providing quick and easy experimentation with the language. The interpreter supports a highly modified subset of C, making Tiny-C programs difficult to transport to standard C environments. Execution is slow. Tiny-C's authors should consider producing an interpreter that supports a standard subset, as this product could be an excellent program development tool. Although there is no royal road to learning C, there are two books which may be of great help along the way (references 5-6). For those of you who wish a more detailed presentation of the C language, there is an excellent article written by its principle designer. (reference 7). Hopefully, more C programs will be published, thus providing much needed examples of C programming techniques.

*

References

1. Jim Gilbreath, "A High-Level Language Benchmark", Byte, Vol. 6, No. 9, pp. 180-198, Sept. 1981.

2. Ron Cain, "A Small C Compiler for the 8080's", Dr. Dobb's Journal of Computer Calisthenics & Orthodontia, Vol. 5, Issue 5, pp. 5-19, May 1980.

3. Ron Cain, "A Runtime Library for the Small c Compiler", Dr. Dobb's Journal of Computer Calisthenics & Orthodontia, Vol. 5, Issue 8, pp. 4-15, Sept. 1980.

4. J. E. Hendrix, "Small-C Expression Analyzer", Dr. Dobb's Journal of Computer Calisthenics & Orthodontia, Vol. 6, Issue 12, pp. 40-43, December 1981.

5. B. W. Kernighan and D. M. Ritchie, The C Programming Language, Englewood Cliffs, N. J., Prentice-Hall, 1978.

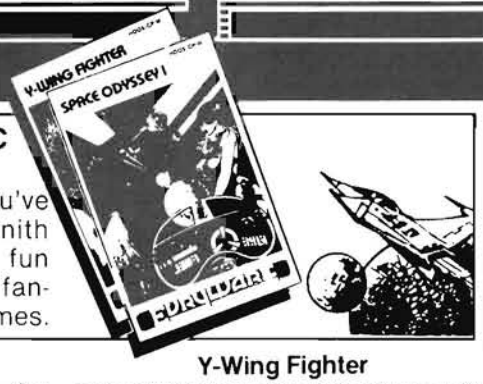
6. Alan R. Feuer, The C Puzzle Book, Englewood Cliffs, N. J., Prentice-Hall, 1982.

7. D. M. Ritchie, S. C. Johnson, M. E. Lesk, and B. W. Kernighan, "The C Programming Language", Dr. Dobb's Journal of Computer Calisthenics & Orthodontia, Vol. 5, Issue 5, pp. 20-29, May 1980.

About The Author:

Bill is employed as a Computer Systems Engineer with the Upjohn Company in Kalamazoo, Michigan. He has a B.A. in Mathematics from Oberlin College and an M.S. in Biology from Ball State University. An insatiable desire to build intelligent machines has driven him to become involved with computers both as a hobby and as a profession. An avid homebrewer, his first computer was an original design based on the Intersil IM6100, a 12-bit CMOS microprocessor. Years of wirewrapping the boards for this project convinced him to buy a Heathkit. His primary interest is in the subfield of Artificial Intelligence dealing with programs that learn. Bill is married and has two children; a six year old son who has an interest in computer games and a one year old daughter who has already demonstrated a great interest in the H89 keyboard. He credits Tony Brewer of the Data General Corporation for introducing him to the wonders of C.

Software From EVRYWARE



FAST ACTION GRAPHIC GAMES

Featuring graphics like you've never seen on your Heath/Zenith computer. Experience the fun and excitement of these fantastic machine language games.

Exterminator

You are an EXTERMINATOR, the highest rank in your planet's space force. Your elite position gets you selected for the dangerous job of protecting the earth colony on Yargon from the savage attack of the hammerhead anthropod. Arriving on Yargon, you are armed to the teeth with your trusty raygun by your side, plenty of remote controlled bullets, and a small fleet of star-copters. Your mission is to exterminate these invaders before they evolve into the carnivorous anthropod, the terror of the galaxy. (48K) **$19.50**

Space Odyssey I

What dangers lie ahead deep in the mysterious Orion cluster? Looking out the front window of your spaceship, you see asteroids, Xylon attack ships, and a variety of other mysterious objects coming at you in 3-D space. Maneuver around the objects, vaporize them with your gamma ray, or retrieve them with your tractor beam before they hit your shield. Keep the heading gauge lined up to stay on course while watching your fuel level, reading navigation messages, and studying the Deep Space Scanner to see what's ahead. You have five warp levels for forward speed and a hyperwarp button if the action gets out of control. Now the excitement of space travel can be yours! (48K) **$21.50**

Galactic Warrior

The battle has just begun! The deadly Evils have launched a massive attack against your home galaxy. Blast thru waves of Evil attackers and then hit the central reactor of their Space Station. Your ship is armed with deep space missiles, a short range laser, and a powerful force field to shield you against enemy fire.
(48K) **$19.50**

Y-Wing Fighter

Flying just above the ground, you will meet some of the hostile inhabitants of a strange planet. They will do their best to prevent you from reaching the enemy's home base. Control the speed and altitude of your Y-WING FIGHTER to out maneuver them and use powerful front and rear phasers when they become too aggressive. But watch out, the volcano is just ahead. (48K) **$19.50**

Missile Control

You are under attack by a powerful enemy. He has many long range guided missiles and bombers all aimed at your cities. You can launch missiles from either (or both) of your launch pads to protect your cities from the onslaught of this menace. If you ever wanted to know what it feels like to be responsible for millions of (electronic) lives, now is your chance. (32K) **$17.50**

Invasion

Three missile launchers have been put under your command in a last ditch attempt to save earth. The INVASION will do their best to blow you to smithereens by dropping bombs and radioactive fallout. (32K) **$17.50**

INTERACTIVE FICTION

Interactive Fiction puts you in a literary setting—a narrated story in which you are a character. What you say and do determines the outcome of the story—and your fate. They require Microsoft Basic, 48K and two disk drives.

Six Micro-Stories

Take part in six short stories, each with many outcomes. Be a spy in Hitler's Third Reich, the pilot of a doomed 747, and more **$17.50**

Dragons of Hong Kong

A terrible secret takes you to the Far East where you have a chance to free the world of an age-old blight, clear your name of a despicable crime, and finish the story wealthy beyond your wildest dreams **$21.50**

His Majesty's Ship Impetuous

Go back in time to the days when England's sailing ships ruled the seas. As the captain of the "IMPETUOUS," you must contend with a mutinous crew, treasure ships, and Napoleon's French fleet **$21.50**

AVAILABLE FOR: H/Z89, Z90, Z100, and H8/H19 systems at most Heath and Zenith retailers or order direct from us. Specify HDOS or CP/M, Hard or Soft Sector. Please add $1.00 per order for shipping and handling. Ca. residents add sales tax.

VISA

CALL TOLL FREE TODAY
800-547-5995 ext. 142

master charge

AMERICAN EXPRESS Card

for VISA, MC or AmExp orders or mail your order to:

EVRYWARE

Dept. S1, P.O. Box 60802, Sunnyvale, Ca. 94088

A Programmable 4 MHz Modification For The H89

Pat Swayne
Software Engineer

The H89 computer runs at a 2 MHz (MegaHertz), which is fine for most folks, but if some of the jobs you give your computer seem to take forever, you may want to try modifying your '89 to run at 4 MHz. Many users have already modified their computers to run at 4 MHz by simply tapping a 4 MHz output on a divider IC but I have two objections to that approach. First, the 4 MHz outputs of the divider are not symetrical square waves, and second, you must modify the disk boot code in the system ROM. So I developed a method for 4 MHz operation with the following features.

1. A symetrical divide-by-three counter is used to divide the 12 MHz oscillator to 4 MHz. This ensures stable operation and allows you to use a Z80A processor (available from HEATH) instead of the more expensive Z80B that may be required if you use the other method.

2. The system comes up at 2 MHz, so no modifications to the boot ROM are required. This also ensures compatibility with obsolete operating systems that you may be using that you cannot modify for 4 MHz operation.

3. The system can be programmed to run at 4 MHz at any level. It can be programmed at the operating system level, or, if you do not wish to modify your operating system, it can be programmed at the user program level — even from BASIC.

NOTE: The modification presented here is for H88 and H89 computers. Although it should work on an H88A or H89A, the physical descriptions of the modification will not apply to those systems. If you are not sure what model you have, see if your CPU board has an unused IC position between U512 and U506. If it does, you can build this modification as presented here. I should also point out that this modification is not compatible with the 56k modification presented in REMark #33, because they both use unused outputs of the General Purpose port on the H89.

The Modification Circuitry

Photo 1 shows the modification mounted on an H89 CPU board. It requires 3 TTL IC's (available from HEATH), a small piece (approximately 1 by 2 inches) of perfboard,

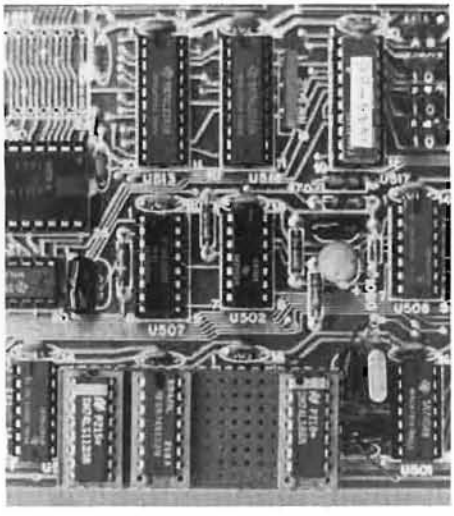


Photo 1. The 4MHz adapter mounted on an H89 CPU board.

some wire wrap sockets, and wire. Photo 2 shows how the circuit is installed on the CPU board. Notice that the center socket is soldered into the unused IC position with the outboard sockets positioned so that there is clearance for their pins, which have been cut as short as possible after the wire wrapping was done.

Figure 1 is a schematic of the circuitry. U1 is the center IC in the photographs, with U2 to the left and U3 to the right. U2 and U3 combine to make a symetrical divide-by-three counter. U2 is a conventional divide-by-three circuit using the 2 JK flip-flops. Its output is applied to an exclusive-OR gate through which the clock (from the 12 MHz

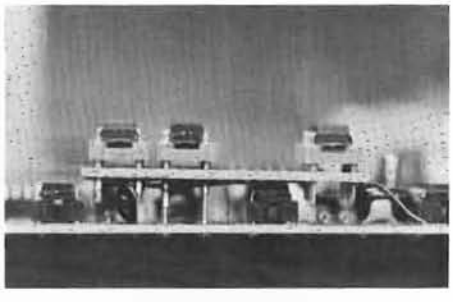


Photo 2. Side view of the adapter showing the pins on the two outboard sockets cut short, and the pins on the center socket mounted in the unused IC position.

oscillator) to the flip-flops must pass. This inverts the polarity of the clock halfway through its cycle, and the result is a 4 MHz square wave from U2. Note: This circuit will not work using an older JK flip-flop IC, such as the 74LS73. Use a 74LS112, as specified in the schematic. The 443 numbers on the schematic are the Heath part numbers for the IC's.

U1 is a quad 2-input NAND Schmitt trigger IC that is used to select either the 2 MHz clock (from U502 pin 8) or the 4 MHz clock from U2. An unused output from the General Purpose port (U552 pin 6) is used to make the selection, and the result is sent to U512 pin 3, where it is distributed to the system. I used a Schmitt device instead of a regular NAND gate so that any noise in the circuit would not affect its output.

Construction

To build the circuit, unplug your computer and remove the CPU board. If there is any solder in the pad holes at the unused IC location between U506 and U512, remove it with a solder suction tool or solder removal braid. Also remove any solder in the pad hole just below pin 7 of U501.

Mount two 14-pin and one 16-pin wire-wrap sockets on a 2 and 1/8th by 1 inch (2.5 by 5.5 cm) piece of perfboard with .1 inch (.25 cm) spaced holes, as shown in photos 1 and 2. You can get the perfboard and wire-wrap sockets at local electronic stores, such as Radio Shack. You will probably notice that I used 16 pin sockets in all three locations, which is acceptable if that is all you have. Cut two pins off at one end of two of the sockets if all of them have 16 pins. Wire wrap the connections shown in the schematic, and in addition, make the following connections: U1 pin 7 to U1 pin 8 and U3 pin 7. U1 pin 14 to U2 pins 16,15,14,10, and 4, and to U3 pin 14. Connect a 4-inch (10 cm) piece of wire-wrap wire to U3 pin 1, but do not connect the other end at this time. Leave all of the insulation on this wire except what you must remove to make the connection. Cut the pins on U2 and U3 as short as possible without damaging the wire-wrap connections and check your wiring to make sure it is correct. Once you install the board, it will be impossible to make changes with-

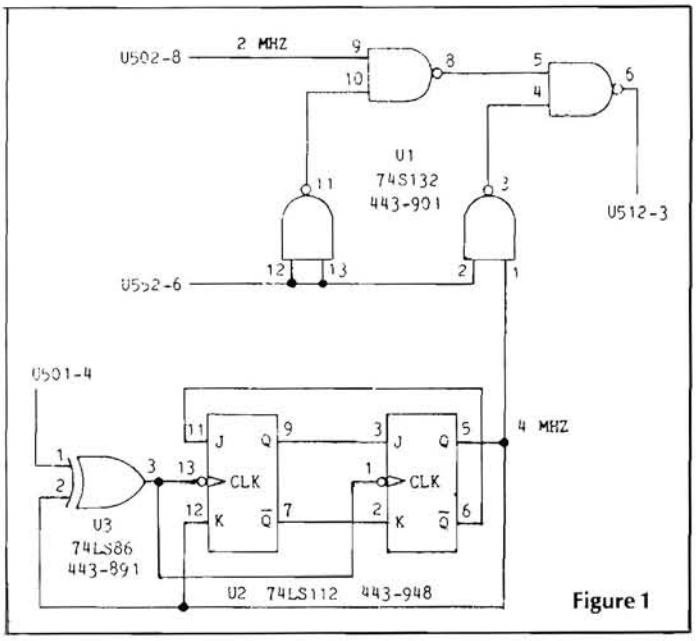


Figure 1

out removing it.

Install the assembled modification on the CPU board by inserting the uncut wire-wrap pins into the unused IC pads. Mount the mod board as close as you can to the CPU board and solder the pins. Bring the unconnected 4-inch wire through the pad hold below IC 501 and connect it to pin 4 of that IC on the solder side of the board. Shorten it if necessary before making the connection so that the wire takes a direct path with no loops. Cut the trace coming from U512 pin 3 as shown in Figure 2. Now make the connections from the U1 to the CPU board that are shown in the schematic. Use wire-wrap wire to make the connections on the foil side of the board from the pads of the unused IC position (which is now U1) to the pads of the other IC's as shown. Insert the appropriate IC's into U1, U2, and U2, and remove the IC at U504 (Z80) and replace it with a Z80A or MK3880-4 (Heath part no. 443-953). Examine U562, and if it is marked 74LS132 or 443-792, replace it with a 74S132 (443-901).

When the modification is complete, you can re-install your CPU board. Make sure that the mod board and IC's clear the metal bracket at the bottom of the computer. Re-connect all cables and plug in and turn on the computer. It should beep twice normally. If it does not, check your wiring first, then check or replace the new IC's. If everything works normally, you can start checking operation at 4 MHz.

Testing at 4 MHz

The areas where the H89 is most likely to fail at 4 MHz operation are its memory systems, both ROM and RAM. The first test you should run is the built in RAM test. This test clears the General Purpose port when it first starts, so you will need to enter a small machine language program that will set up 4 MHz operation and then jump to the test past the part where it clears the port. Using the Substitute command, enter the following code

```
H: Substitute 40100
040100 000 72
040101 000 71
040102 000 40
040103 000 331
040104 000 157
040105 000 331
040106 000 76
040107 000 4
040110 000 323
040111 000 362
040110 000 303
040111 000 122
040112 000 7          (hit RETURN)
H:
```

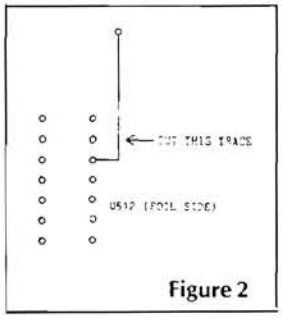


Figure 2

Note: The old data at these memory locations, shown here as 000, may be anything. After you enter this program, enter

```
H: Go 40100
```

to start the test. This code, in addition to setting up 4 MHz operation, sets up the radix if you have MTR90. If you have MTR88 or MTR89, that part of the code is ignored.

If the test starts up normally and runs for several minutes without any problems, then your system is ready for use at 4 MHz. If the test starts normally but reports a memory error, then the problem is probably with RAM, and you will probably have to replace your RAM chips with faster ones. Before you replace all of your chips, you may want to try to isolate the bad ones by exchanging them. The address of the failure, printed by the RAM test will locate the 16k bank where the failure occurred. Remember that the banks start at 40000 (split octal) while you are in the monitor's control. Even if you only isolate the problem to a bank, that will mean that that you only have to buy 8 IC's. There are several sources for faster RAM chips. One is Jameco Electronics, 1355 Shoreway Road, Belmont CA 94002 (phone 415-952-8079). They sell 8 150-ns chips for 16.95 plus 5% postage and $1.50 insurance. Order part no. 4116N-2, and include sales tax if you live in California.

If the RAM test does not start up at all, then your Monitor ROM prob-

ably cannot operate at 4 MHz. There are two options. If you have access to an EPROM programmer, you can copy your ROM into a faster chip. If you do not have access to a programmer, you will have to buy another ROM from Heath and hope it works at 4 MHz. If you decide to buy another ROM and you now have MTR88 or MTR89, I would strongly suggest that you buy MTR90 (Heath part no. 444-142). Note that this is a new part number for MTR90 (the old one was 444-84). If you are getting MTR90 for the first time, you will also need to get 444-83 (replaces U550) and make a small modification to the CPU board. First, locate 4 programming plugs on the left side of the board. Move the middle two from position 0 to 1 and remove the upper one entirely. Now connect the middle connector where the upper programming plug was to pin 14 on one of the left 25-pin connectors, such as P508. You can wire-wrap the connection if you wish.

There are two reasons for upgrading to MTR90. The first is that it is programmed into a 2732 IC, which is a newer part than the three supply 2716 used for MTR88 and MTR89, and the chances are greater that it will work at 4 MHz. The second is that it is a much better monitor, with more commands and hex as well as octal operation.

If the indications are that your ROM will not work at 4 MHz, and you have Heath/Zenith CP/M, there is another test you can perform. First, boot up on CP/M and run MBASIC. Then load a game or other BASIC program (Robot NIM is a good one for this). Then enter

POKE 13,38:OUT 242,38

to turn on 4 MHz operation and RUN the program. If it runs normally, your RAM is probably OK, but your ROM will have to be replaced. That was the condition of my H89 when I first modified it. After you have run the program, enter

POKE 13,34:OUT 242,34

to switch back to 2 MHz and run the program again to see the difference between 2 and 4 MHz operation. If you don't see any difference, you probably made a mistake in the modification. Make sure you are in 2 MHz operation before you return to CP/M.

The above illustrates how to switch speeds from BASIC in CP/M. In HDOS, you would use POKE 8246,6:OUT 242,6 to go to 4 MHz, and POKE 8246,2:OUT 242,2 to return to 2 MHz.

You have probably noticed that the above examples do not just write to a port, but modify a memory address as well. This is because during each clock interrupt in an H89, a program may write the value stored at the address we POKEd (called the "control byte") to the general purpose port. In fact, in CP/M all you really need to do is change the control byte, because its value is written to the port during each clock interrupt.

You can also switch speeds in an assembly language program in a similar way. For example, in HDOS, this code would do the trick.

```
MVI     A,6             6 = SPEED BIT + CLOCK BIT
STA     40066A          SET CONTROL BYTE
OUT     362Q            SWITCH TO 4 MHZ

MVI     A,2             2 = CLOCK BIT ALONE
STA     40066A          SET CONTROL BYTE
OUT     362Q            SWITCH TO 2 MHZ
```

And in CP/M, you could do it this way.

```
LDA     0DH             GET CONTROL BYTE VALUE
ORI     4               SET 4 MHZ BIT
STA     0DH             REPLACE CONTROL BYTE

LDA     0DH             GET CONTROL BYTE VALUE
ANI     -1-4            RESET 4 MHZ BIT (SET 2 MHZ)
STA     0DH             REPLACE CONTROL BYTE
```

Here, we did it a little differently. Only the 4 MHz bit is affected (set or reset), and the port is not written to. This is the preferred method to use when you know that the control byte is in use by other programs, and the port is being written to on every clock interrupt.

NOTE: Your system may not operate at 4 MHz even with faster RAMs and a faster monitor ROM. There is a slight chance that one of the other ROM's (I/O decoder, RAM decoder), or the floppy static RAM, or another IC may not be capable of 4 MHz operation. It is up to you to troubleshoot your system and find the "slow" component, since (obviously) Heath cannot support such modifications.

Modifying Your Operating System

Once you experience the thrill of running at 4 MHz, you probably will want to do it all of the time. This will require modifying your operating system. Specifically, you will have to modify your HDOS disk device driver(s) so that 4 MHz is set when the driver is loaded, and all processor speed dependent delays are fixed. In CP/M, you will also have to fix the delays in the BIOS and modify the cold boot code to make the speed switch.

Modifying HDOS

In HDOS, the ideal place to make the switch to 4 MHz is in the disk device driver, when it is loaded, since critical delay constants have to be changed anyway. The source code for the 5.25 inch hard sector and 8 inch (H47) disk drivers is supplied with HDOS 2.0, so I will present modifications to them.

To fix the 5.25 inch hard sector driver, you will have to modify the files SYDVD.ASM and SYINIT.ASM and re-assemble the driver.

First, modify the file SYDVD.ASM as follows. Locate the label SYLOAD, and a few lines below it, add the lines shown here in bold print.

```
SYLOAD  EQU     *

        XRA     A
        OUT     UP.FC           Set Fill character = 0

*       Set up the original vectors

        LHLD    SYDD+1
        PUSH    H               Save current system device
        LXI     B,BOOTAL
        LXI     D,BOOTA
        LXI     H,D.CON
        CALL    $MOVE           Move in constants and vectors
        LXI     B,12
        LXI     D,BOOT4
        LXI     H,D.CON+2
        CALL    $MOVE           Move in 4 MHz constants
        MVI     A,6
        STA     40066A          Update control byte
        OUT     362Q            Switch to 4 MHz
        POP     H
        SHLD    SYDD+1          Restore system device
```

Now, locate the label TDT and add one line.

```
TDT     DB      DEF.TDT           Track delay time
BOOT4   DB      40,10,32,9,20,15,250,5,7,46,46,240
        DW      0                 Dummy Relocation address
```

Modifications to the HUG SY: device driver are similar, but must be done in the file MFDVD.ACM, and the added lines must be preceded by IF MFBOOT and end with ENDIF.

The file SYINIT.ASM must be modified if you want to initialize disks while operating at 4 MHz. Locate the line

```
MVI     A,H17SDL
```

and replace it with

```
MVI     A,H17SDL*2
```

If you have the HUG SY:, modify MFINIT.ACM by replacing the line

```
LXI     DE,15*2048/35
```

with

```
LXI     DE,15*2048/17
```

After you modify SYDVD.ASM and SYINIT.ASM, you will have to assemble them. These files require several .ACM files to assemble correctly, and there are two approaches you may take to gathering the files. One is to go through the ASM file looking for XTEXT statements (not all of them are at the beginning of the file), and copy down the names after the word XTEXT. Then locate the appropriate files on the Software Tools or Driver Source disks supplied with HDOS 2.0 and copy them to a working disk, along with the file to be assembled. For example, if you see the line XTEXT TYPTX, then you would copy TYPTX.ACM to your working disk. If the file you are assembling is SYDVD.ASM, and your working disk is in SY1:, you can begin the assembly by entering

```
>ASM SY1:SYDVD=SY1:SYDVD
```

assuming that ASM.ABS is on SY0:. Another approach to gathering the needed .ACM files is to make a separate .ACM disk by copying all of the files that end in .ACM from the Software Tools and Driver Source disks to another disk. Then, with this disk in SY2: and your ASM file in SY1:, you could enter

```
>ASM SY1:SYDVD=SY1:SYDVD,SY2:
```

and the assembler would know to look on SY2: for the .ACM files. Once the assembly of the two parts of the device driver, SYDVD and SYINIT is completed, they must be combined into one program. The source code for a program to do this (MAKMSD.ASM) is supplied on the Driver Source disk. This program does not work properly when assembled as supplied (at least on the disks I have), but you can modify it by replacing the lines

```
ORG     USERFWA-ABS.COD
DB      377,FT.ABS
DW      LOAD
DW      MEML-LOAD
DW      ENTRY
```

with

```
ORG     USERFWA
```

Then assemble MAKMSD.ASM to get MAKMSD.ABS. Then copy the two assembled device driver files, SYDVD.ABS and SYINIT.ABS to SY0:. Rename SYDVD.ABS to XX.DVD and rename SYINIT.ABS to XXINIT.SYS. Now, run MAKMSD by entering

```
>MAKMSD XX:
```

and it will combine XX.DVD and XXINIT.SYS into one file called XX.DVD. If you do a directory (CAT) after running MAKMSD, you will see that XX.DVD is now larger than it was before you ran MAKMSD. It has become your completed device driver, ready to use. Just delete the old SY.DVD (or DK.DVD) from a system disk and copy XX.DVD to it, renaming it to SY.DVD (or DK.DVD). For example, if you have a system disk in SY1:, you could enter

```
>DELETE SY1:SY.DVD
>COPY SY1:SY.DVD=XX.DVD
```

Then run FLAGS and set the S flag on the new SY.DVD. You can delete XXINIT.SYS after running MAKMSD if you wish.

Boot up on a system disk with the new driver, and if you did everything correctly, you will be operating at 4 MHz.

NOTE: When you BYE from HDOS while operating at 4 MHz and wish to re-boot, you will have to reset the computer and boot from the monitor (from the H: prompt). You cannot just press RETURN to re-boot because the boot code on the disk or in the ROM has not been modified for 4 MHz operation, and the computer still operates at that speed until you reset (SHIFT-RESET).

To operate H47 disks at 4 MHz, you will need to modify the files DDDVD.ASM and H47LIB.ACM. At the beginning of DDDVD.ASM, add the following line before the first STL statement.

```
*
PRG4MHZ SET     0               ASSEMBLE FOR 4 MHZ
        STL     'Assembly Constants'
```

Locate the label DDLOAD and add the lines shown in bold print.

```
DDLOAD  CALL    RST
        IF      PRG4MHZ
        MVI     A,6
        STA     40066A          SET CONTROL BYTE
        OUT     362Q            ENABLE 4 MHZ
```

```
        ENDIF
        ANA     A               Ignore any errors
```

That completes the modifications to DDDVD.ASM. Near the beginning of H47LIB.ACM, add a line after IF .SMALL.

```
        SPACE   4,10
        IF      .SMALL
PRG4MHZ SET     1               NO 4 MHZ DURING BOOT
        ELSE
```

Locate the label DLY and add 6 lines.

```
DLY     MVI     A,040Q
        ENDIF
        IF      PRG4MHZ
        ADD     A               DOUBLE DELAY FOR 4 MHZ
        ENDIF
        IF      .SMALL
        ELSE
        ANA     A               F = 'NC'
```

Now find the label WDN1 and add these lines.

```
WDN1    DCX     B
        IF      PRG4MHZ
        MVI     A,0             WASTE SOME TIME IF 4 MHZ
        MVI     A,0
        MVI     A,0
        MVI     A,0
        ENDIF
        MOV     A,B
```

Finally, find WND1 and add these lines.

```
WND1    DCX     B
        IF      PRG4MHZ
        MVI     A,0             WASTE SOME TIME IF 4 MHZ
        MVI     A,0
        MVI     A,0
        MVI     A,0
        ENDIF
        MOV     A,B
```

After you have made the changes, assemble DDDVD.ASM and DDINIT.ASM and combine the two .ABS files using the procedure described for the hard sector device driver. Unlike the hard sector driver, this driver will still operate at 2 MHz, so you can freely switch speeds while testing software, etc.

Modifying CP/M

All modifications for going to 4 MHz with CP/M are made to the file BIOS.ASM (supplied with CP/M), which must then be re-assembled. This discussion will cover Heath/Zenith CP/M version 2.2.03, but modifications to 2.2.02 should be similar. Copy BIOS.ASM to a working disk before you modify it, and if it is write protected, remove the protection with STAT.

The first modifications are for 5.25 inch hard sector disks. Locate the line WHDA EQU 20 and modify six of the constants as follows.

```
WHDA    EQU     46
WHNA    EQU     46
WSCA    EQU     160
WRITA   EQU     44
WRITB   EQU     10
WRITC   EQU     44
READA   EQU     100
```

The next modifications are for H47 disk operation. These modifications may be skipped if you are not using an H47. Locate the label W4D1:, and add these lines.

```
W4D1:   CALL    H47INS
        ANI     DSDONE
        JNZ     W4D2
        DCX     B
        MVI     A,0
        MVI     A,0
        MVI     A,0
        MVI     A,0
        MOV     A,B
```

These modifications are for 5.25 inch soft sector disk operation (H/Z37). You may skip them if you are not using that kind of disk. NOTE: These modifications have not been tested, since I do not have a Z89-37 on my modified H89.

Find the label RESH371:, and change the line BEFORE it.

```
        MVI     A,20
RESH371:
        DCR     A
```

Find RDYH37B:, and change the line before it.

```
        MVI     A,20
RDYH37B:
        DCR     A
```

Locate the label WBS37:, and add 4 lines below it.

```
WBS37:  MVI     A,150
WBS371: DCR     A
        NOP
        NOP
        NOP
        NOP
        JNZ     WBS371
```

The next change must be done regardless of what kind of disks you are using. It causes the computer to switch to 4 MHz operation at cold boot. Locate the label CBOOT:, find the line LXI H,CTLPRT a few lines below it, and add two lines.

```
        LXI     H,CTLPRT
        MOV     A,M
        ORI     4
        MOV     M,A
        OUT     H88CTL
```

This completes the modifications to the BIOS. It must be re-assembled using the MAKEBIOS program supplied with CP/M. If the procedure given in the Heath/Zenith manuals is too difficult to understand, you may want to try the method I outlined in the article "Making Sense of MAKEBIOS" in REMark #26 (March 1982). There are two typographical errors in that article in the listing of MAKE.SUB in the left column on page 14. Where it says $R/W, change it to $$R/W, and where it says $DIR, change it to $$DIR.

After you assemble the new BIOS, you will need to install it on a sys-

tem disk. You should first prepare a test system disk (with FORMAT and SYSGEN) and copy STAT.COM, MOVCPMnn.COM (that is, MOVCPM17 for hard sector disks, MOVCPM47 for H47, etc.), PIP.COM, CONFIGUR.COM, and STAT.COM to it. Then delete the old BIOS from it with these commands.

A>STAT BIOS.SYS $R/W
A>ERA BIOS.SYS

Now, copy the new BIOS.SYS to this disk and enter

A>MOVCPMnn * A:

replacing nn with 17, 47, or 37 as appropriate. When MOVCPM is finished, run SYSGEN and hit RETURN when it asks for a source disk, and type A when it asks for a destination disk. Then reset your computer and re-boot on the test disk. CONFIGUR will run if everything is working, and you can set up things as you wish and make other system disks from the new one.

If you want to format disks under CP/M while running at 4 MHz, you will have to modify the FORMAT program. To do it, use the S command in DDT to make the following changes to FORMAT.COM.

Address	Old data	New data
103	43	CD
104	6F	CF
105	70	06
106	79	C3
107	72	1B
108	69	04
418	CD	C3
419	CF	03
41A	06	01
4DE	01	02
4FB	01	02
5AA	01	02
607	30	60
613	50	F0
75E	14	28
76B	14	28
876	0A	14

These modifications affect delays for all three disk formats that I have presented BIOS modifications for. The soft sector 5.25 inch part has not been tested.

A Final Note

Once you have modified your operating systems, you should experiment with test disks, running several different programs, until you are satisfied that everything works. You will notice different effects on different programs. For example, an action game that uses the computer's 2 ms clock to time its actions (such as Sea Battle — 885-1103) will run at the same speed, while a game that is processor speed dependent (such as Galactic Warrior — 885-8009) will run faster. Most programs will run much faster, and it's almost like getting a new computer for the price of a few IC's, so enjoy it.

The Heath/Zenith 88-89 CPU with a Future

SUPER 89

The Super 89 replaces the central processor board in the Heath/Zenith 88-89 series computers to bring your 88-89 to current state-of-the-art technology. The Super 89 gives you features that are useful today and allow expansion of your capabilities. The Super 89 is fully compatible with all Heath/Zenith products and also supports many peripheral devices from other manufacturers. New software and hardware are enhanced with the Super 89 by using all the features of the Z80 technology.

The Highlights of the Super 89:

- Twice the operating speed (4MHz+)
- Memory Capacity to 256 K in Software Bank Selectable 64K blocks
- CP/M and HDOS Compatible without modification
- Twice the number of expansion slots (Six)
- Real time clock on-board
- Two serial I/O Ports
- Designed for multi-user capability
- Parity checking for RAM assures integrity of memory transfer operations
- Arithmetic processor provision facilitates mathematic operations

These features, along with an enhanced monitor to access to all the Z80 CPU, give you power from your 88-89 that only large computers can claim.

High Speed Processing

The Super 89 runs twice as fast as the standard H/Z CPU board. Time savings on running programs are significant.

Expanded Memory Capacity

This feature allows you to use the advantages of the more sophisticated programming languages; enables you to use enhanced memory software such as print spoolers and electronic disks to increase speed; allows the use of "scratch pad" memory to increase efficiency; the bank select features give you high speed data handling and manipulation; and provide for multi user capabilities.

Super 89 Electronic Disk

This optional software package for the Heath/Zenith CP/M 2.2.02 and 2.2.03 allows the Super 89 user to access auxiliary RAM as a very fast mass storage device. Provides up to 180K bytes of storage area that is accessable without the slowness of disk drives. The Electronic Disk also includes display capability for the Real Time Clock.

Peripheral Expansion

This important feature lets you use your Super 89 in more ways with peripherals from DG, Heath and many other manufacturers. Some of these important enhancements are: Additional floppy disk controllers; Modem or Printer serial interfaces; Color video controllers; IEEE 488 BUS for test equipment and measuring interface; Analog/Digital interface; Parallel interface for high speed printers; Hard disk system controllers; Bread-board development cards; Computer game controllers; and Production process controllers.

Enhanced Super 89 Monitor

Gives you all the features of Heath's MTR-89 monitor plus the ability to display all the Z80 register contents; Single-step through a program and set up break-points; Supports H/Z and other manufacturers of disk systems; Improved system diagnostic routines; and Supports the Super 89 Real Time Clock.

Real Time Clock

The Real Time Clock allows you to program activities and control functions according to time; allows the use of interactive time functions with an electronic disk; and is very useful in accounting functions.

Parity Checking

This features ensures the integrity of memory transfer operations. The Super 89 alerts you if a parity error occurs.

Full CP/M and HDOS Compatibility

The Super 89 has full compatibility with either the HDOS or CP/M disk operating systems that does not require hardware modifications. This feature gives you the best of both worlds in the amount of existing software you may use.

Arithmetic Processor Provision

The Super 89 has on-board provisions for the optional AM9511A. This is a separate processor that features basic arithmetic as well as exponential, logarithmic, trigonometric and binary functions. Calculations are high speed and can be accomplished as a "hardware subroutine". This device is a must for anyone using any amount of mathematical computation whether complex functions or arithmetic calculations.

Ease of Installation

The Super 89 is simple to install and takes only minutes. No soldering required. Simply remove the old CPU board, configure and install the Super 89 to multiply the capabilities of your H/Z 88-89.

D·G ELECTRONIC DEVELOPMENTS CO.

Ordering Information: Products listed available from DG Electronic Developments Co., 700 South Armstrong, Denison, Tx. 75020. Check, Money Order, VISA or MasterCard accepted. Phone orders (charge only) call (214) 465-7805. Freight prepaid. Allow 3 weeks for personal checks to clear. Texas residents add 5%. Foreign orders add 30%. Prices subject to change without notice.

Period Cruncher!

A Diablo Lettering Program

Frank Clark
402 West Ferry
Berrien Springs, MI 49103

ED: The period on the Diablo printer is the only recommended character to be used for the characters you generate using the following article. The period is a bronze insert that will take the punishment over any other character if you are not using a metal print-wheel. Further, the following program "builds" a new character set for all the ASCII characters of the key board. However, you may wish to allow one of the ASCII characters to equal your own creation (a picture, etc.). Experimentation with this program can produce some interesting results.

The READER.BAS program is an MBASIC program designed to read a file of letters or characters, compress them and output them to a Diablo printer creating attractive lettering or pictures. The letters or characters are in a file called "READFILE.DAT". After the program reads in the letters or characters it accepts lines of input. The characters in each of the input lines are sent to the Diablo in the format specified by the letter file which you can create using a standard EDitor of your choice. To end the program type the TAB key and RETURN.

The "READFILE.DAT" shown in Figure 1 is supplied for your reference indicating the construction of the Old English letters M and Z. Figure 2 is the actual output of the program printed on the Diablo. The user can easily modify this file with an Editor. The entire file used for the program consists of 94 sets of 24 lines for each of the ASCII graphic characters except the space. The first line is ignored and the next 23 are printed exactly as they appear on the screen except that they are printed closer together on the Diablo.

The original letter file is about 22k and requires 64k when read into memory with the MBASIC program. There is no protection against a line that is too long for the printer and/or paper. A Diablo with tractor feed does not work too well since the best character appearance requires precise positioning and the tractor allows a little slack. Be sure to set the print force switch inside the cover on the lowest setting or a little more wear than normal will be observed on the period character after extended usage.

Simple modifications to the program are easily made to provide a few different character sizes if desired. The standard size provides a solid black character of any width line when using a period as the printing character. Carbon ribbons provide the sharpest lettering image though cloth ribbons can provide an acceptable image.

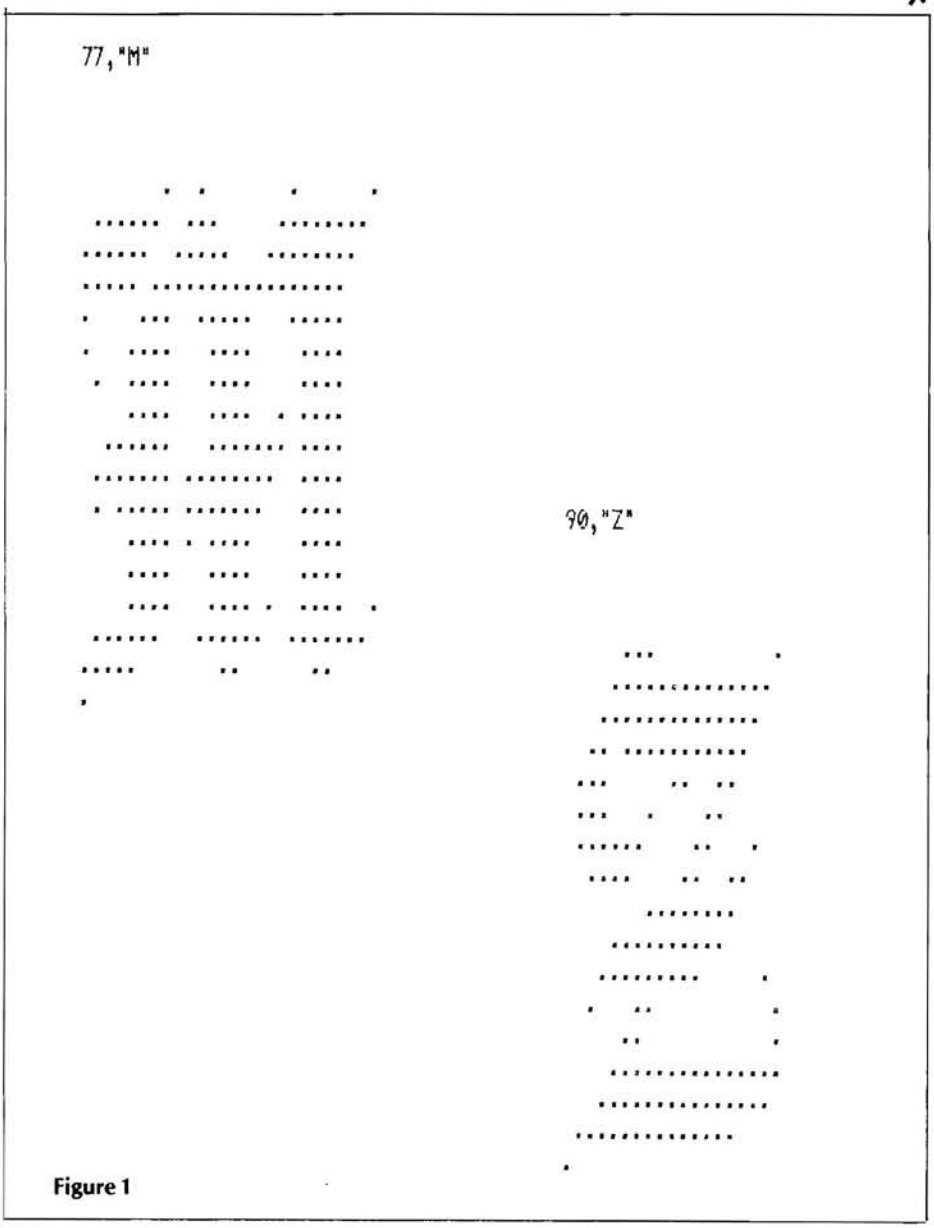


Figure 1

Figure 2

```
10 PRINT:PRINT"DIABLO lettering program"
20 PRINT:PRINT"This program requiries 64k of memory."
30 PRINT:PRINT"This program is brought to you courtesy of:
40 PRINT"Frank T. Clark - Copyright 9/3/82 - Version 2
50 PRINT:PRINT"One minute while I read the letter file.":PRINT
60 OPEN"i",1,"READFILE.DAT"
70 DIM S$(93,23)
80 '
90 'fetch all the lettering records to memory
100 '
110 FOR I=0 TO 93
120 LINE INPUT#1,S$
130 VALUE=VAL(S$)
140 PRINT 126-VALUE;
150 IF VALUE<>I+33 THEN PRINT"sequence error in the file":STOP
160 FOR J=0 TO 22
170 LINE INPUT#1,S$(I,J)
180 NEXT:NEXT
190 PRINT:PRINT:PRINT"Ready
200 PRINT:PRINT"Input a line to letter on the diablo.
210 '
220 ' set hmi vmi
230 '
240 E$=CHR$(27)
250 HMI=2:VMI=1
260 LPRINT E$CHR$(30)CHR$(VMI+1)E$CHR$(31)CHR$(HMI+1);
270 '
280 'get characters
290 '
300 LINE INPUT "? ";L$
310 FOR I=1 TO LEN(L$)
320 S$=MID$(L$,I,1)
330 V=ASC(S$)
340 IF V=32 THEN L=20:LPRINT E$CHR$(10);:GOTO 500
350 IF V=9 THEN 570
360 '
370 'output characters
380 '
390 V=V-33
400 IF V<0 THEN 510
410 L=0
420 FOR J=0 TO 23
430 LPRINT E$CHR$(10);
440 NEXT
450 FOR J=0 TO 22
460 LPRINT S$(V,J)
470 L1=LEN(S$(V,J))
480 IF L<L1 THEN L=L1
490 NEXT
500 LPRINT STRING$(L+2," ")E$"9"
510 NEXT
520 LPRINT E$CHR$(128+9)CHR$(1)E$"9"STRING$(23,CHR$(10))
530 GOTO 300
540 '
550 'reset and stop
560 '
570 LPRINT CHR$(12)
580 PRINT"Goodbye
590 FOR I=0 TO 299:NEXT
600 LPRINT E$CHR$(13)"P"
610 END
620 GOTO 190
```

Local HUG Clubs

Floyde Adams 2001 S. Valley Drive Las Cruces, NM 88005 is interested in starting a HUG club for Las Cruces, NM or Las Cruces/El Paso, TX. Please contact him at (505) 523-7511 or write.

Hal Davis, who lives in the Ann Arbor area would like to contact other HUGgies in his area to possibly start a group to communicate by asking/answering questions, and passing ideas on to one another. Hal lives at 6501 Nollar Road Whitmore Lake, MI 48189 (313) 665-7919.

SPOHUG (Spokane HUG) has a new address: c/o Chuck Ballinger S. 3810 Havana Spokane, WA 99203 Telephone number is the same.

PRHUG (Puerto Rico HUG) has a new address: c/o Joseph Gonzalez 22 Yardley Place Ocean Park, PR 00911 Phone (809) 725-1612.

VHUG (Vancouver HUG) is now the Portland/Vancouver HUG. They meet at the Vancouver Heathkit store the first Thursday of each month at 7:30 p.m.. Dan Heims is the contact person.

LAHUG (Los Angeles HUG) has a new president Ray Livingston. New address and phone: PO Box 5334 Pasadena, CA 91107 (213) 792-4763.

Stephen Weinrich, 1429-C Nike Plaza Redstone Arsenal, AL 35808 (205) 837-3357 invites Heath users in the Huntsville, Alabama area to contact him regarding forming a local users group.

T.W. Longfellow at 11414 Avondale Road #12 Redmond, WA 98052 (206) 881-7190 would like to start a local HUG that serves both the Bellevue and Redmond Washington area. His intention is to start a users' group this is dedicated to expanding its knowledge in such areas as programming in assembly language and BASIC, software development techniques, hardware concepts and in general, just getting the most out of the computer. Perhaps, most importantly, dedicated to the beginner. Please contact Mr. Longfellow by mail or phone if interested.

Yet another HUG has formed in Florida. The NWFHUG (Northwest Florida HUG) meets monthly the second Wednesday at 7 p.m. at the offices of DATATEC INC. They currently have 30 members. Anyone interested may contact George A Repasy at 812 Cherokee Rd Eglin AFB, FL 32542 (904) 651-2108.

HEY!
I Just Discovered PILOT!

Jennifer T. McGraw
12741 SW 68th Terrace
Miami, FL 33183

The other day, while looking for programs that would be easy to demonstrate at an educational computer display, I turned to the HUG disks. Of course I was aware of the MBASIC Quiz disk, and Score and Grade Keeping. But Hangman is good for only so many hours. Then I bumped into this phrase, Programmed Inquiry, Learning Or Teaching, (hope that's right), and decided to pull out disk #885-1042 at HEC #21, and take a look. There was something on that disk called TREKQUIZ.PIL. "Why is a program called a pill?", I ask. But, in our area, if you CAN'T STAND the evening disaster reports, you turn to STAR TREK. I am therefore a minor Trekkie. And no one can resist a trivia quiz anyway. (Judge and jury both, right?). So I did the quiz. Not satisfied with being a Denebian Slime Worm, I took it again, and again, etc.

If I can get hooked, who else can I hook? So, I looked at a listing, as advised by our good Doctor Readme. Easy. It took an hour at the most. And I wrote a quiz.

I remember learning BASIC. All that editing. All those Syntax Errors. We will not mention ASM.

There are about 21 basic instructions in PILOT. They are very simple mnemonics and the syntax makes Tarzan a grammarian. Put the instruction, frequently one letter, at the beginning of a line, follow it with a COLON and do a fair amount of typing, and RUN.

But, as usual, I tried doing more and more. It's amazing what you can do with an interpreter that can't count past 99. Especially if you use an editor or word processor to write the program, so you can easily correct typos, although it is not difficult to program with PILOT itself, just tedious in the editing phase.

Basically, programming in PILOT is simply asking questions, getting answers, seeing if the answer matches the correct one, and informing the user whether he is right or wrong. Consider the following program:

```
R: SKY.PIL
T:WHAT COLOR IS THE SKY?
A:
M:BLUE
TY:YES THAT IS CORRECT.
TN:YOU LIVE IN LOS ANGELES, MAYBE?
E:
```

R: = Remark, for notes by the programmer.
T: = Type this line on the screen.
A: = Accept keyboard entry or wait for an Answer.
M: = Match with this word or list.
TY:= Type if Yes (Yes there is a match)
TN:= Type if No (No there is not a match)
E: = End, either of program or subroutine.

I really think that PILOT should be the first language for someone to learn. It teaches several basics used in most languages. For instance it can jump (J:LABEL) to another spot in the program just like GOTO. It can use subroutines to save tedious repetition, (U:SUBR), which is the same as GOSUB in BASIC, or CALL in ASM. It introduces string and numeric variables and a LET statement, (C: X=X+2 , C: being short for Compute). While the syntax is simple it must be correct.

There are simple disk I/O commands, SAVE: and OLD:, shades of BHBASIC. In long programs, it's best to put frequently used subroutines at the beginning, because PILOT begins its search for a label at the beginning.

It will also encourage one to find a language that does more, without so much of the frustration found in first learning any computer language.

No, it doesn't have the capability of COSINing which is a bad habit anyway, and you do have to supply the answers. But you can use it to teach the kids or spouse something about what it is that has you speaking in foreign tongues. Heck, write a program to teach basic HDOS usage. Or how about one for CP/M and MAKEBIOS? When you get it written, please forward to me.

And having been a teacher, I should think it a godsend for writing simple quizzes for all those bright ones, to keep them out of trouble. Or let them write the quiz. (Actually, I'm not too sure about that. I am extremely jealous of the up and coming generation and all the opportunities they have for learning about computers, the new playmates of the month.)

All in all, I feel serendipitous over PILOT. I wasn't looking for it, but, jimineez, what a neat little language.

NOTE: For those who like Turnkey operations, it can be done. PILOT has this weird habit, when first up, of casting aside every other letter, or something like that. So fool it with spaces. Using DOCOM in HDOS which has been SYSMODed, courtesy of Jim Teixeira, enter it like this:

```
DOCOM GO=PILOT;     O L D : T R E K Q U I Z ;     ;     R U N ;
```

Those long blanks are 9 spaces each. You now have a program named GO.ABS which will automatically run PILOT, load TREKQUIZ.PIL and run it, all on SY0:, of course. This could probably be modified for SUBMIT.

More on I/O BYTE Assignment

F. J. James
3709 Elkanah Place
Randallstown, MD 21133

In the September issue (32) of REMark Bob Sutherland describes a method of changing the I/O (Input/Output) device assignment while running under the CP/M operating system. The BASIC statments, Poke 3,171 and Poke 3,169, are used to switch the Console to UC1: and back to the CRT:. This procedure sets the entire I/O byte to predetermined values. For example, when using this procedure if the list device, as defined in the I/O assignment for your system, is UL1:, it will be reset to LPT:. The objective of changing the I/O device assignment should be to reset only BITS 0 and 1, the two which change the console assignment, without changing any others. The I/O assignment code is stored in memory location 3 and determining your system assignment can be accomplished by: (1). reading and saving the number stored in memory location 3. Example:

A%=Peek(3).

A% equals the value for your system I/O assignment as it currently exists. Referring to Figure 1, looking at the CON: row, you can see that the values would change for other configurations. In the CON: row you will note that the difference between the CRT and UC1 codes is 2. If we wish to change the I/O assignment between the two devices we need only to modify the system I/O assignment number by adding two. (2). The statement

Poke 3,A%+2

will direct console output to the printer and (3) the statement

Poke 3,A%

will switch it back. No other bits (assignments) will be changed. This logic assumes that the original console assignment was to the CRT. If this is not the case a 1 or a 3 would be added to A% instead of 2. Refer back to Figure 1 in the CON: row to determine the proper number for your configuration.

Switching the console output to the listing device permits selective CRT or hardcopy output from BASIC programs using PRINT and LIST instead of LPRINT and LLIST. This will permit outputting selectively to either device under program control. It can also be useful for such things as "recording" keystrokes, computer responses and other terminal directed text for illustration purposes.

For many applications switching console assignments is exactly what is needed. However, in other cases the fact that all output is to the listing device may be a serious disadvantage. I for one find it very disconcerting typing to a blank screen! A useful alternate is to assign CRT: as the listing device. When debugging BASIC programs all LPRINTS, for example, will go to the terminal. Printing to the terminal is faster than printing to the printer, easier to edit and saves reams of paper. The LPRINTS in a program can be directed either way with one POKE statement which can be entered either in immediate mode or at the proper place in the program.

The next logical question is why not make any desirable temporary device assignment changes from BASIC rather than going back to SYSTEM, using CONFIGUR and then back to BASIC, a time consuming procedure. The only problem is how to easily find the proper number to POKE for the desired assignments. The elegant and time consuming way is to study I/O BYTE assignment values in the CP/M alteration guide, determine the BIT pattern for the desired assignments and convert to one of the number systems that your system will accept. I find the assignment chart, Figure 1., a much easier way of selecting the proper decimal number. You use this chart just like you carve an elephant. First you get a block of wood. Then you carve off everything that doesn't look like an elephant! Choose the four assignments you want then forget the remaining blocks. The sum of the numbers in the blocks you have left is the elephant that you wanted.

Example: to assign: CONSOLE=CRT: (01)
READER =UR1 (08)
PUNCH =UP1: (32)
LIST =LPT:(128)
TOTAL = 169

POKE 3,169 OR POKE 3, 1+'+=×+128 will make the proper assignment.

I/O BYTE ASSIGNMENT CHART

	ASSIGN NUMBER	ASSIGN NUMBER	ASSIGN NUMBER	ASSIGN NUMBER
CON: BIT 0-1	TTY: 00	CRT: 01	BAT: 02	UC1: 03
RDR: BIT 2-3	TTY: 00	PTR: 04	UR1: 08	UR2: 12
PUN: BIT 4-5	TTY: 00	PTP: 16	UP1: 32	UP2: 48
LST: BIT 6-7	TTY: 00	CRT: 64	LPT: 128	UL1: 192

Figure 1.

The table can also be used to find which elephant you have. I have two printers available, one a DIABLO 1640 (which requires UL1:) for letter quality printing and the other a Z25 (which uses LPT:) for higher speed. If at print time I am not sure which device is assigned I can find out by entering "PRINT PEEK(3)" in immediate mode. If the number returned is 192 or greater UL1: is assigned. If it is 128 to 191 the assignment is LPT:.

Figure 2 shows how this method of switching could be used within an MBASIC program. This program has no function except to alternate the LPRINT output between the printer and the terminal.

Lines 10 through 90 define smart terminal commands. All of these have been covered in previous issues of REMark and will not be

```
10 '**************************PROGRAM DEMO**********************************
30 Z$=CHR$(27):CL$=Z$+"E"'-----------------------DEFINE ESC & CLEAR PAGE.
40 ZE$=Z$+"l"'-------------------------------------------------ERASE LINE.
50 ZJ$=Z$+"j":     ZK$=Z$+"k"'------------SAVE CURSER-CURSER TO PREV. POS.
60 ZL$=Z$+"x1":    ZO$=Z$+"y1"'-----------------ENABLE/DISABLE 25th LINE.
70 ZP$=Z$+"p":     ZQ$=Z$+"q"'----------------ENTER & EXIT REVERSE VIDIO.
80 ZY$=Z$+"Y";'-----------------------------------DIRECT CURSER ADDRESS.
90 DEF FNZC$(R,C)=ZY$+CHR$(R+31)+CHR$(C+31)'POSIT. CURSER TO ROW R COL. C.
100 T$="CURRENT ASSIGNMENT IS TO THE "
110 '************************PROGRAMS START HERE****************************
130 PRINT CL$;"THIS PROGRAM DEMONSTRATES ONE METHOD OF SWITCHING LPRINT"
140 PRINT "FROM PRINTER TO THE CRT WITHIN THE PROGRAM."
150 PRINT:PRINT "CURRENT MODE IS ALWAYS SHOWN ON THE 25th LINE."
160 FOR K=1 TO 2000:NEXT K
170 A%=PEEK(3):GOSUB 350
180 B%=A%'---------------------------------------SO WE CAN RESET LATER
190 PRINT CL$:INPUT"DO YOU WANT TO CHANGE I/O ASSIGNMENT? Y/N";X$:PRINT
200 IF X$="Y" OR X$="y" THEN GOSUB 320
210 FOR J%=1 TO 10
220     LPRINT J%;
230 NEXT J%
240 LPRINT
250 PRINT:INPUT"ANY MORE RUNS? Y/N ";X$
260 IF X$="Y" OR X$="y" THEN 190
270 POKE 3,B%'-----------------------------PUT IT BACK LIKE YOU FOUND IT.
280 MS$="":GOSUB 370'--------------------------------AND CLEAR 25th LINE.
290 END
320 '================SUBROUTINE TO SWITCH MODES==========================
330 IF A%=105 THEN A%=B% ELSE A%=105
340 POKE 3,A%
350 IF A%=105 THEN A$="CRT" ELSE A$="PRINTER"
360 MS$=T$+A$
370 '***********************SUB ROUTINE 25TH LINE************************
380 PRINT ZJ$;ZL$;FN ZC$(25,1);ZP$;ZE$;MS$;ZQ$;ZK$;:RETURN
```

Figure 2.

explained here except for my use of Z's in the names. I have a file of these and other commonly used functions. Most of these have names starting with "Z" and all are compatible with each other, that is their names do not conflict. By avoiding the use of variable names starting with "Z" within the main program, conflicts there are avoided as well.

PROGRAM SUMMARY:

100 T$ is text for use in the message on line 25.

170 Sets A% = to the value of the I/O BYTE before running the program and goes to the subroutine at line 350 to prompt the current assignment.

180 Stores the original value found in the I/O BYTE as B%.

190 260 Values of 1 to 10 are printed using LPRINT. If a "Y" option is chosen in line 190 then the GOSUB 320 in line 200 is executed which results in the new list device being selected.

270-280 Restores the system to the original configuration before ending.

330 A "toggle". A% will be switched between the original LST: assignment and the CRT: Note that if CRT: is not your terminal assignment, the 105 must be changed.

350-360 Either "CRT" or "PRINTER" is added to the string T$ to define the message MS$ to be printed on the 25th line.

380 Prints 25th line. The sequence is:

(1) Save curser position.
(2) Enable 25th line.
(3) Position curser to row 25 column 1.
(4) Enter reverse video.
(5) Erase line.
(6) Print message (MS$).
(7) Exit reverse video.
(8) Return curser to previous position.
(9) Return to calling routine.

To summarize, the method described can be used to change the I/O BYTE assignment either from within an MBASIC program or as an immediate mode statement. The number to POKE into memory location 3 can be found by summing the numbers found in the blocks in Figure One that correspond to the desired assignments. The method is valid for CP/M 2.2.02 and 2.2.03. A cold boot restores the system to normal configuration.

Vectored from 13

Dear Walt

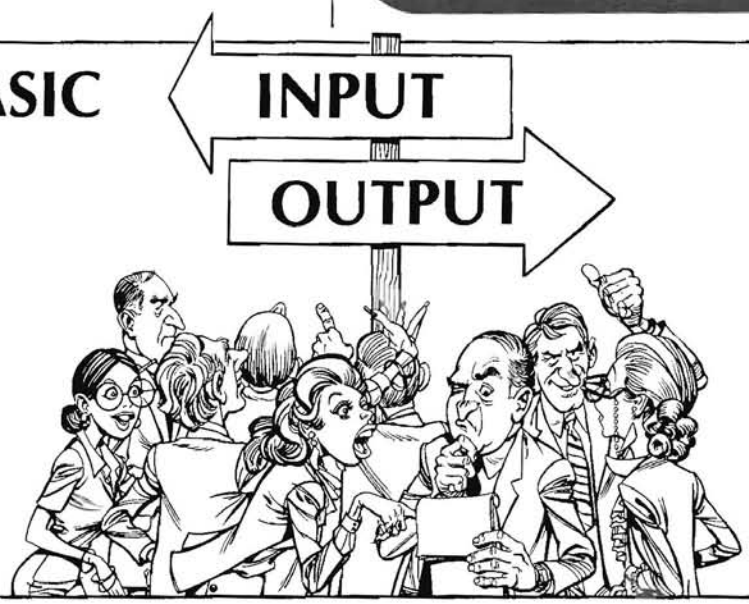
The following is an example of using DIRECT.SYS to compile information in sequential files. I used it to write a history of a family with 14 people involved. Over a long period of research I filed, with the HDOS editor, paragraphs pertaining to each person. I used their initials plus a number as the file name and started each entry with the name of the file, ex. JED1.TXT. When I was ready to unscramble two disks of information I ran the following program. The output was all of the data collected on each individual. Using this output and the TEXT EDITOR again, the information was re-arranged in the proper order. I can think of many applications of this idea and I haven't seen it in REMark before.

John Dailey
808 South A Street
Richmond, IN 47374

```
15 REM ADAPTED FROM LUIS E SUAREAZ ARTICLE REM #18 PAGE 25
25 REM PROGRAM USED TO REARRANGE RESEARCH FILES
55 REM VARIABLES IN THIS PROGRAM ARE DECIMAL
75 REM DELETE ODD LINES TO SAVE SPACE
85 REM DIMENSIONING VAR A$
90 DIM A$(11)
105 INPUT "WHICH DRIVE (SY0:) ";D$
107 INPUT "KEYWORD (JED.TXT) ";K$
110 PRINT CHR$(27)+"E"
120 N=0
125 D1$=D$+"DIRECT.SYS"
130 OPEN D1$ FOR READ AS FILE #1
135 REM IF NO NAME, JUST READ ALL 23 BYTES UPDATE COUNTER AND BACK AGAIN
136 Z=0
140 B=CIN(1):IF B=255 THEN FOR A=2 TO 23:B=CIN(1):NEXT:N=N+23:GOTO 140
145 REM IF END OF FILE PRINT BLANK LINE AND FINISH
150 IF B=254 THEN PRINT:END
151 IF B=0 THEN PRINT:END
155 REM IF VALID, STORE CHARACTER IN B$
160 B$=B$+CHR$(B)
165 REM READ BYTE 2 TO 8, STORE THEN IN B4 AND ADD A PERIOD
170 FOR A=2 TO 8:B$=B$+CHR$(CIN(1)):NEXT:B$=B$+"."
175 REM READ BYTES 9 TO 11. STORE THEM IN B$. THIS IS THE NAME EXTENSION
180 FOR A=9 TO 11:B$=B$+CHR$(CIN(1)):NEXT
185 REM JUST READ BYTES 12 TO 14
190 FOR A=12 TO 15:B=CIN(1):NEXT
205 REM JUST READ BYTES 16 TO 23
210 FOR A=16 TO 23:B=CIN(1):NEXT
```

```
220 PRINT B$,;
221 IF LEFT$(B$,3)=LEFT$(K$,3) THEN GOTO 4000
222 Z=Z+1:IF Z=4 THEN PRINT CHR$(13)
223 IF Z>4 THEN Z=0
225 REM CLEAR VARIABLE B$
230 C=0:B$=""
235 REM UPDATE COUNTER. IF 506 BYTES WERE
          READ, READ NEXT 5 AND RESET COUNTER
240 N=N+23:IF N=506 THEN FOR A=1 TO 6:B=CIN(1);NEXT:N=0
245 REM BACK AND START AGAIN
250 GOTO 136
260 REM LINES ADDED TO INCLUDE ALL DRIVES
4000 REM PRINT ROUTINE FOR TEXT FILES SELECTED
4001 OPEN "LP:" FOR WRITE AS FILE #2
4010 D5$=D$+B$
4012 OPEN D5$ FOR READ AS FILE #3
4020 J=CIN(3)
4030 IF J=0 THEN GOTO 4070
4040 LINE INPUT #3,;L$
4045 L$=CHR$(J)+L$
4050 PRINT #2,L$
4060 GOTO 4020
4070 CLOSE #2:CLOSE #3
4080 GOTO 222
```


On MBASIC



Hugh Appleby
14906 E. 39 Terrace
Independence, MI 64055

In my application of MBASIC programs, I use many sequential data files. The OUTPUTing and INPUTing of these files is very straight forward, but when it comes time to change, add to, insert new data, or correct the always present typing mistakes, it becomes very tedious. The following is the way I now handle this problem and may be of interest to other HUG members.

Normally, if an attempt is made to LOAD a sequential data file from disk into memory, a DIRECT STATEMENT IN FILE error results. However, if each data record in the sequential file is preceded by a different numerical character, the file can be LOADed into memory and LISTed on the CRT exactly as a normal MBASIC program.

By making the data file in the format of a MBASIC program and SAVEing it in ASCII by - **SAVE "SY1:TEST.DAT",A** - a sequential file can now read it, it can be LOADed into memory, it can be LISTed and the CRT, and, most important of all, the main purpose for this procedure, it can be edited with the EDIT feature of MBASIC.

One method to accomplish this is to just simply write the file as if a MBASIC program:

```
10 'This is data one\This is data two\This is data three\
20 'Data one here\And, data two here\Here three\Here four\
30 etc.
```

Note that each line is a remark statement. This must be done to prevent MBASIC from changing the entries. With this method, leading and trailing blanks, commas, and items enclosed in quotations can be used. Note also, the use of reverse slash as a delimiter between data fields and at the end of each record.

In using this method, do not use "@" to break a line that is to long for the screen, but continue typing and use the WRAP AROUND AT END OF LINE feature of HDOS.

Each data record can have as many different data fields as needed. I the above example, line 10 has three, line 20 has four.

A second method is to write the data file in normal MBASIC:

```
10 CLEAR 1500
20 OPEN "O",1,"SY1:TEST.DAT"
30 X=10:X$=STR$(X)
40 C$=X$+"'"
50 LINE INPUT "ENTER DATA - TYPE -1 TO END - ";B$
60 IF B$="-1" THEN 80
70 C$=C$+B$+"\":GOTO 50
80 PRINT #1,C$
90 PRINT "ANYMORE ENTRIES - Y/N - ":Y$=INPUT$(1)
100 IF Y$="Y" THEN X=X+10:X$=STR$(X):GOTO 40
110 IF Y$<>"N" THEN 90 ELSE END
```

A variation on this could be a program for something like a mailing list.

```
10 CLEAR 1500
20 OPEN "O",1,"SY1:TEST.DAT"
30 X=10:X$=STR$(X)
40 C$=X$+"'"
50 LINE INPUT "ENTER NAME - ";B$
60 IF B$"-1" THEN END
70 GOSUB 200
80 LINE INPUT "ENTER ADDRESS - ";B$
90 GOSUB 200
100 LINE INPUT "ENTER CITY AND STATE - ";B$
110 GOSUB 200
120 PRINT #1,C$:X=X+10:X$=STR$(X):GOTO 40
200 C$=C$+B$+"\":RETURN
```

To retrieve the data from the file, use normal MBASIC:

```
10 CLEAR 1500
20 OPEN "I",1,"SY1:TEST.DAT"
30 IF EOF(1) THEN END
40 LINE INPUT #1,B$:B$=MID$(B$,INSTR(B$,"'")+1)
50 A$=LEFT$(B$,INSTR(B$,"\")-1)
60 PRINT A$
70 B$=MID$(B$,INSTR(B$,"\")+1)
80 IF INSTR(B$,"\")=0 THEN 30 ELSE 50
```

With this procedure all data is read from each data record regardless of how many data fields may be in each record.

If it is required that a string variable be assigned to each data field, the program could be something like this:

```
10 CLEAR 1500
20 OPEN "I",1,"SY1:TEST.DAT"
30 IF EOF(1) THEN END
40 X=1
50 LINE INPUT #1,B$:B$=MID$(B$INSTR(B$,"'")+1)
60 A$=LEFT$(B$,INSTR(B$,"\")-1)
70 B$=MID$(B$,INSTR(B$,"\")+1)
80 IF INSTR(B$,"\")=0 THEN 90 ELSE X=X+1:GOTO 60
90 FOR Y=1 TO X
100 PRINT Z$(Y):NEXT Y
110 GOTO 30
```

As can be seen, the requirements for this procedure are very simple and should be adaptable to most any program.

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

CUT ALONG THIS LINE

HUG MEMBERSHIP RENEWAL FORM

When was the last time you renewed?

Check your ID card for your expiration date.

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

Name _____

Address _____

City-State _____

Zip _____

REMEMBER — ENCLOSE CHECK OR MONEY ORDER

CHECK THE APPROPRIATE BOX AND RETURN TO HUG

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

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

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

Heath
Users'
Group

Hilltop Road
St. Joseph, MI 49085

POSTMASTER: If undeliverable, please do not return.

885-2034