$2.50

REMark

Issue 35 • December 1982

A true HERO for Christmas!

See Page 15

Official magazine for users of HEATH ZENITH computer equipment

for Heath/Zenith

KRES™

Happy Holidays from KRES

H/Z89 HARDWARE

Expansion
seven slots - in the machine $395.00

2/4 MHz
double your processor speed - software switchable $79.95

Color Graphics
192 x 256 pixels, 15 colors, 3D sprites, RGB and composite outputs

Music Board
four part harmony in stereo $125.00

H/Z89 ACCESSORIES

Backplate
replacement rear panel with 20 openings $35.00

Fan
supplemental fan helps keep it cool $49.95

Power Supply
5 volts at 10 amps - slips right in

Prototyping Cards
for left, right and ambi-bus $39.95

Joysticks
spring centering or free floating $69.96

SOFTWARE

wide selection - for business and personal use

KRES ENGINEERING
P.O. Box 17328, Irvine, CA 92713
(714) 559-1047 or
(213) 957-6322

REMark Issue 35 • December 1982

on the stack

ON THE COVER: Presenting HERO I, the ET18 Robotic Trainer from Heath Company. The story, by Pat Swayne, on page 31. The cover photo by Jon Falkner.

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

Articles submitted by users and published in REMark, which describe hadware modifications, are not supported by Heathkit Electronic Centers or Heath Technical Consultation.

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

The Super 89

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.

Peripheral Expansion

This important features 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.

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.

NEW SUPPORT PRODUCTS Available for the SUPER 89

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.

Super 89 Electronic Disk

This optional software package for the Heath/Zenith CP/M 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.

MP/M II[TM]

MP/M II is a multi-user, multi-tasking operating system for use with the Super 89. Supports up to four users and the Z37, Z47 and Z67 disk systems. Compatible with CP/M software.

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.

Ideas from the Land of the Bean and the Cod....

where the Cabots speak only to Lodges, and the Lodges only to God!

I have just had the priviledge of returning from my hometown, Boston, after a 23 year absence. The occasion was that of attending the Northeast Computer Fair in the Heath Company display booth.

Gerry Kableman (of ZDS) and I arrived in town on Tuesday, November 9th, from Saint Joseph, Michigan. With nothing pressing to do that afternoon, we set out on a tour of some of my old hangouts. Bright and early the next morning we joined with representatives from many other computer companies in setting up our displays for the show which was to open Thursday morning. Gerry took charge and, with me plus two show laborers in tow, set about erecting the Heath/Zenith booth.

Later that evening, Gerry and myself plus Jim Buszkiewicz of Heath Consultation, and Barry Watzman, Zenith Computer Product Line Manager were taken out to the Wellesley, Massachusettes, Heathkit Electronics Center by Fred Lucia, Senior Technician, for a combined meeting of the HUG North Shore, HUGRI (Rhode Island), HUGEM (Eastern Mass.) Heath User Groups. Bob Weintraub, area Manager for Heath and Manager of the Wellesley store graciously hosted this gathering of over a 100 HUG members. After presenting a slide show on the Heath Company I had the privledge of meeting and talking with a number of these HUGgies.

This great group of Heath User's also had a booth at the NE Computer Show and it presented me with a further opportunity to work and talk with many of them about just what direction computers in general and HUG in particular would be going in the future. Among these fine gentlemen were Perry Miller, HUG North Shore President, Dave Haskel of Wizard Software House, Ron Rocheleau, Jim Teixerira and Charlie Layman are just a few among many who made us welcome.

This proved to be a very interesting and informative trip. The 1982 Northeast Computer Fair broke many attendance records, among them the largest first day crowd at Hynes Auditorium and the largest ever attendance record of any four day event at the Hynes. Also, the 1982 NE Fair was second only to the 1982 National Computer Conference in overall attendance at any computer show including the West Coast Computer Show. It was announced that they lost track of attendies when they past the 55,000 mark because the tickets at this time where un-numbered. It was quite impressive, and I again got to meet and talk with many HUGgies as well as new people interested in Heath/Zenith Computers.

Probably one of the most interesting things that came to light for me during this trip, both in my talks with HUGgies and those interested in Heath/Zenith computers, was the need for software, specifically business and educational software, but most importantly the educational software. The most common statement made was " I have (or want to have) a computer for my business to use at home and I want it to have educational software for my children! What do you have?"

There is some good software already available for business use and more on the way, but not a great deal is yet available for the educational market. This is a wide open field and it is for those of the Heath User's Group who wish to make their mark in computer society and want to further the learning of others to address their attention and talents. I hope to address this subject in more detail starting with the Febuary 1983 issue of REMark.

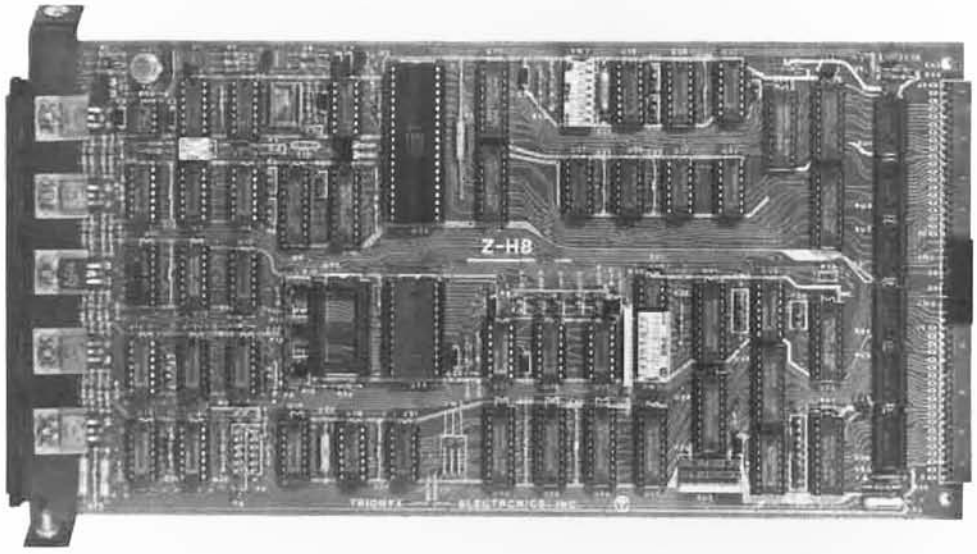
I thourghly enjoyed this trip, at the booth with Gerry, Jim and Barry, plus meeting both HUGgies and those soon to be HUGgies. I also had the opportunity to make many new friendships such as with Bob O'Neill, Warwick, Rhode Island Heathkit Manager and Ken Barton, Assistant Manager of Heathkit in Wellesley. It is from all of these people I get the feeling of a great brotherhood of computer users, specifically Heath/Zenith Users.

From all of us here at the National HUG Headquarters, Bob, Jim, Pat, Terry, Nancy, Margaret and myself, Merry Christmas and a Happy 'HUGable' New Year

Walt Gillespie
REMark Editor

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

BUGGIN' HUG

Dear HUG,

In this letter I will plead the cause of the 'OLD TIMERS' like myself. I have been a Heathkit Computer user for about five years (give or take a few months). I started with the tape operating system and have been able to keep up with most of the updates in hardware and software. Now to the point of this article. I have maintained my H-8 for several years, only to find that now Heath supports mostly the H-89 version. Oh! There is the color graphics board that is only H-8 for the moment, however I am sure we will soon find that to change. In order to keep up with the 89's I have done my homework and come up with a few patches to the old tape system as well as a mod for duplexing the AT:. The first will be the modifications to the tape system.

Because I hate to waste any computing power, I have in the past, I spent several nights burning the midnight oil to find the required memory location to patch E.X. BASIC 10.01.00, E.X. BASIC 10.02.00, HBUG V2.0, TED V2.0, H8 ASM V2.0, as well as all the Games we used to play from the tape operating system, to make it operate at the AT:. This is assuming that, like my system, you are using an H-9 in the AT: slot, and with a H8-5 card. It wasn't too difficult because Heath used a standard console driver in all the above mentioned software.

For Games, DBUG, H8 ASM, TED and BASIC change the following with the front panel.

at 040.107 from 372 to 374
at 040.111 from 372 to 374
at 040.114 from 373 to 375
at 040.117 from 373 to 375
at 040.122 from 373 to 375
at 040.125 from 373 to 375

Plus any patches supplied by Heath to upgrade to their new versions.

Now reset the front panel to 040.100 and push dump. The saved version is now configured to run with the H-9 as the command terminal. CAUTION, you should not have the disks or HDOS running when you run the tape system. Nor will the tapes work with the entire extended configuration option.

The two E.X. BASIC's were a little more difficult. In addition to the modifications supplied above (because of tamper proofing and the PORT command) it was necessary to perform the following change.

at 041.165 from 372 to 374

Now for the duplex mod. How many of you using the H-9 as the AT: are annoyed at the need to switch to half duplex to see just what you are typing and then suffered the further hindrance of having to supply a line feed? This is not only cumbersome, but places unnecessary line feeds in your input text that must be edited out of the file. Well, I have had my fill of this and decided to do something about it. With the advent of the device driver source provided with 2.0 and an understanding of drivers from REMark #20, I went into the code and made these changes.

To ATH84.ASM

I first set the conditional assembly for the H8-5 card. Then located the subroutine called RCHAR (this is the routine of the first mod).

```
RCHAR   LDA     C.CAADR+1
        ANA     A
        STC
        RNZ                     CONSOLE INTERUPT
        CALL    INCHAR
        RZ      RCHAR
        ANI     177Q
```

Delete these next three lines. They will be taken care of in DVDIO.ACM.

```
        CPI     CR
        JNE     RCHAR2          NOT CR
        MVI     A,NL
```

Then in the file DVDIO.ACM perform the following.

In the routine INCHAR at the lines.

```
        ELSE
        MVI     L,USR
        ANI     USR.RXR         'Z' SET IF THERE IS NO DATA
        JZ      INC1
        MVI     L,UDR
        CALL    IN
        CALL    DUPLEX
```

Insert this line to CALL duplex routine

```
        ANA     A
        JMP     INC2
        ENDIF
        INC1    STC

INC2    POP     H
        RET
```

This is where our modification takes place. Insert these lines:

```
DUPLEX  CPI     ENL             NEWLINE + HIGH ORDER BIT
        JZ      DUPLEX2         YES, MAKE A NULL
        CPI     CR+200Q         CR + HIGH ORDER BIT
        JZ      DUPLEX1         YES, DUPLEX CRLF
        CALL    OUT
        RET
DUPLEX1 CALL    OUT
        PUSH    PSW
        MVI     A,10
        CALL    .DLV            5 MS. DELAY FOR 600 BAUD
        POP     PSW
        MVI     A,ENL
        CALL    OUT
        RET
DUPLEX2 MVI     A,NUL           MAKE ANY INPUTED NL A NULL
        RET
```

This is the end of the modification. Assemble and you will no longer have those nasty problems when you replace the old AT: with this one. I did not include any mod for an AT: on the H8-4 card, but following these guidelines you should have no problem on your own doing that. As an extra bonus this mod also allows those users with the H-10 paper tape reader to input through the parallel port on the H-9 with no extra line feeds in the inputted file. A CNTRL-D is still needed to let HDOS know it is the end of the file.

Coming as soon as I complete it is a PT:, parallel driver, for the H8-2 board. It will have the capability to SET bit word lengths, parity, and, most of all, terminal functions like the AT:. It will be talored for the H8-2 card and the H-10, it will also use an asychronous baud rate.

Allen Swayze
Rt. 2 Box 251, Lot #92
Derby, KS 67037

Dear Bob;

I have the H-89 with Benton Harbor BASIC and would like to say that REMark has helped me to operate a computer more than any other publication. The KISS group was probably the best for beginners. Can you present some more under that heading?

Also, I think each issue should have at least one program in Benton Harbor BASIC with detailed remarks.

Good Luck,

Henry K. Bentley
Weymouth Farm
Putney, VT 05346

Dear HUG;

I deceided to purchase and build the H8, 16K memory, 2 cassette tape units and the H9 CRT.

Did I make a mistake in my decision, where are the KISS type programs for this type of H8 system?

Please don't refer me to the HUG list, I have most of the ones that fit.

Where are the great guys of HUG who can convert those super-great 16K programs for 2 tape systems, with an H9 just waiting to display the great results, have they left the planet Earth?

And, if there were such programs available, would REMark take a chance and print them without fear of the""disk-boys" boycotting our magazine?

Please think this letter over, and do your thing for us poor- "tape people".

Your prompt attention to this letter and reply will be greatly appreciated.

Sincerely,
Rudolph Mansi
Rt. 2, Box 45
Alto Loma, TX 77510

ED:) Rudolph, The articles we publish in REMark are, as best we can make them, in proportion to the interests of the majority of our readers. This is not to say, that there are not members out there with minimal systems, or members with very sophisticated systems, but, from the inquiries and article submittals we receive the majority lies somewhere in between. Maybe there is a member with a dual tape system that would be willing to be a "clearing house" for tape software, then "tape" people would have a source to help with the exchange of software. HUG has not produce any new tape software because there aren't a large amount of members that still use tape, to get the reproduction cost down we have to order 500 of one tape at a time and HUG has never sold anything close to 500 of one tape.

Dear HUG;

This is in response to the last question in Q & A for September. I would like some articles for beginners. At this time I have an H8 with 16K and a cassette operating system. An article on how to work with this much of a system would be appreciated.

Sincerely, Jared Freeman
46486 Featherstone Ridge
Utica, MI 48087

ED) Mr. Freeman, To repeat an old refrain, "We publish what we receive". If there is a member of HUG who feels they can put together an article that would address this situation then I would be glad to publish it. The majority of the materials that we receive at HUG are for "disk" systems so this is what generally gets published.

Dear HUG,

I have had a very frustrating two months of trying to interface a Centronics 739-3B with my H8 computer. With the thought that there may be other poor souls trying to use this very fine printer, I would like to record the keys to success in this endeavor.

H8		Centronics
Pin 2<	———	>Pin 2
Pin 3<	———	>Pin 3
Pin 4<	———	>Pin 11
Pin 7<	———	>Pin 7
Pin 20<	———	>Pin 20
Pin 6	Jumpered	
Pin 8	to Pin 20	

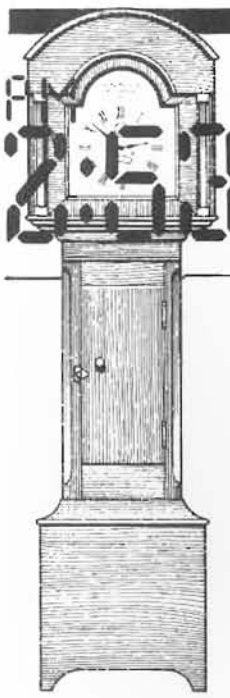
The universal device driver UD.DVD is recommended, with the baud rate set at 4800 and with XON-XOFF (CTRL S/CTRL Q) handshaking. This driver provides for up to eight printer configurations and allows ASCII control characters to be sent for character selection. Thus I have set LPO: for proportional characters (ESC,DC1)<LP1: for standard 10 cpi (ESC,DC3) and LP2: for condensed 16.7 cpi (ESC,DC4).

On Switchpack 1 all DIP switches should be set to OFF except switches 5 and 6 should be ON. Switchpack 2 all DIP switches should be OFF. The port address, of course, should be 340, with no interrupts.

I hope the above info will save others from the long bout of trial and error, long distance phone calls and shipping expense.

Best wishes,

Vel Turner
12484 Seelane Dr.
Florissant, MO 63033



The Hayes Chronograph, an H89 and CP/M

Glen E. Hassebrock, Jr.
c/o Nims Associates, Inc.
363 South Main Street, Suite 420
Decatur, Illinois 62523

After building the Heathkit H89, booting CP/M and using the system for awhile, it became apparent that I would like to put the date and time on displays and printouts. But there was no time or date maintained in the system. When I saw an advertisement for the Hayes Stack Chronograph, I preceived a solution was at hand.

True, the BIOS could be altered to keep track of the time (see REMark November 1981 or CP/M BIOS 2.2.03 from Heath). However, with a fair amount of disk access this "clock" slows down. Also, the need to turn on the display to see the time, and turn it off to use some programs, presented an inconvenience.

The Chronograph costs about the same as a respectable modem, depending on where the purchase is made, from about $189 to $249. If you don't need the time that much, then this might be a bit steep for a "clock". But the Chronograph isn't just a clock, it also keeps track of the date and day of the week.

Physically, the Chronograph is an aluminum box with plastic caps at the front and rear ends. The day of week and time display is a green fluorescent display, similar to the Heathkit Digital Alarm Clock kit GC-1107. On the back is a DCE connector, a write protect switch, a hole for the power pack connector, and calibration access holes. Inside, there is a single circuit board with a place for 3 AA batteries in a holder (all supplied).

Yes that's right, a DCE connector. That means ordinarily connecting it to a DTE connector, of which the H89 with serial interface board has only one, probably already occupied by a modem. Well at first I thought this would present a problem, then I read the owner's manual that comes with the Chronograph.

The owner's manual is complete and has few typographical errors in it. It is complete and fully descriptive of everything a person may wish to know about using the Chronograph, from unpacking to test programs for some popular microcomputers (but not Heath/Zenith). There is also a copy of the two year limited warranty at the back.

Communication to and from the Chronograph is by means of ASCII strings (the characters normally used by Heath/Zenith computers in the USA). There are several commands for setting day, date, time, and dividers for time and date, and for reading the day, date and time. The Chronograph also has an alarm function which can be used to notify a program of a preselected time of day. All responses from the Chronograph end with a carriage return and optionally a line feed.

The first thing I wanted to know was if it survived the delivery service (I ordered it from a mail order company before Heath offered it in their catalog). Following the instructions in the manual, I took the circuit board out and installed the batteries. The display was still blank, as it should be, since only the time, day, and date are retained while the power pack is not supplying power. Plugging in the power pack and connecting it to the Chronograph brought the display to light, and now I pondered how to set it.

The easiest way at hand was to connect the Chronograph to the DTE connector on the H89, just as I would a modem, and use a modem control program to "communicate" with the Chronograph. Since the Chronograph determines baud rate (300 or 1200), parity, etc., from the command sent to it, there was no special modification needed. Following the manual, I set the time, date, and day with no trouble at all. The Chronograph responded with a zero every time, meaning everything was correct. The display had changed to the time and day of week, just as I had set it.

Then I sent the commands to read the date, time and day and got back the date, time and 1 digit for the day. The day is represented by the digit zero meaning Monday, 1 for Tuesday, and so on, to 6 for Sunday. The time can be set in 24 hour mode or 12 hour mode, with A or P for AM and PM indication. Any ASCII character, other than null or carriage return, can be used to separate the date and time digits to provide easier reading. However, these are lost when the Chronograph is running on battery power.

Since my H14 printer is attached at port 340Q and a modem is often attached to port 330Q, this leaves port 320Q available for the Chronograph. This is a DCE connector which would be incompatible with the DCE connector on the Chronograph. But the owner's manual states how to connect the Chronograph to this type of port and referring to a back issue of BYTE, I constructed a null modem cable with male connectors at both ends. Changing the modem control program to communicate with the 320Q port and attempting to get the time showed that this is perfectly acceptable, and now I have the Chronograph attached all the time.

Next, to be useful, there had to be a way to get the Chronograph information without using the modem control program. Attempts to get an MBASIC program to get the time showed that the interpreter was just too slow, if the baud rate for the port is 1200 (300 works fine). Besides I wanted the time and date in other than BASIC programs. My solution: modify the CP/M BIOS so that the Chronograph access code is always available.

Now this really isn't all that difficult. I didn't need to change existing code, I just had to add code. The code needed was (1) set the time and date dividers upon cold boot (2) access the Chronograph for the time, date, and day of week. The difficult part was the need to edit the BIOS with CP/M's ED editor. If you Have the old hard sectored disks, you will need to know how to edit a program on one disk and save it onto another (e.g. A>ED B:BIOS.ASMC:).

Since the new 2.2.03 BIOS can be setup for various disk options, most of the code can be in different locations, depending on the chosen option. Well as any CP/M programmer should know, there is always one thing you can find in the BIOS. That is the jump table, the address of which is at memory locations 1-2. Preceding the warm boot jump (the one actually pointed to by locations 1-2) is the cold boot jump. This jump is not needed after the cold boot has started, so why not use it as a jump pointer to the Chronograph access code!

The first change to the BIOS was to put a TRUE/FALSE indicator in

so that the BIOS could be assembled without the Chronograph code, if need be. This goes near the other such indicators, at the beginning of the BIOS, and should look similar to...

```
        IF      (2-(H17T+H37T+H47T+H67T)) SHR 15
%:      TOO MANY DISK DRIVE TYPES SPECIFIED
        ENDIF
CHRONO  EQU     TRUE   ; <=== ADD THIS LINE FOR CHRONOGRAPH
PARTITN EQU     TRUE AND H67T
```

Next comes the actual access code. A safe and easy place to put this is after the disk parameter tables and just before the warm boot code. The processes involved are

(1) send "read the time command"
(2) get the time
(3) send the "read the date" command
(4) get the date
(5) send the "read the day of week" command
(6) get the day of week code
(7) translate the code to a day literal

The access code is...

```
        DB   1
        DB   8
        DB   0,0,0,0
        ENDIF
        IF   CHRONO          ; <== START ADDING THIS CODE
GETTIM  DB   'ATRT',CR       ; READ TIME COMMAND
GETDAT  DB   'ATRD',CR       ; READ DATE COMMAND
GETDAY  DB   'ATRW',CR       ; READ WEEKDAY COMMAND
        DB   16              ; LOCAL STACK SPACE
DAYS    DB   'MONDAY',0,0,0  ; WEEKDAY LITERALS
        DB   'TUESDAY',0,0   ; AND STACK LABEL
        DB   'WEDNESDAY'     ; EACH DAY MUST BE 9
        DB   'THURSDAY',0    ; CHARACTERS LONG THEREFORE
        DB   'FRIDAY',0,0,0  ; PAD SHORT ONES WITH NULL
        DB   'SATURDAY',0    ; CHARACTER
        DB   'SUNDAY',0,0,0
SAVSP   DS   2           ; SAVE CALLERS STACK POINTER
SDAY    DS   9           ; PLACE TO PUT CURRENT DAY LITERAL
SDATE   DS   9           ; PLACE TO PUT THE DATE
STIME   DS   9           ; PLACE TO PUT THE TIME
TIMER0  LXI  H,0         ; GET THE CALLERS STACK POINTER
        DAD     SP
        SHLD    SAVSP          ; SAVE IT
        LXI     SP,DAYS        ; POINT TO LOCAL STACK
        MVI     B,5            ; COMMAND LENGTH
        LXI     H,GETTIM       ; POINT TO COMMAND
        CALL    CLCKOT         ; SEND IT TO CHRONO.
        MVI     B,9            ; EXPECTED RESPONSE LENGTH
        LXI     H,STIME        ; WHERE TO PUT RESPONSE
        CALL    CLCKIN         ; GET IT
        MVI     B,5            ; REPEAT THE ABOVE FOR DATE
        LXI     H,GETDAT
        CALL    CLCKOT
        LXI     H,SDATE
        MVI     B,9
        CALL    CLCKIN
        MVI     B,5            ; GO GET THE DAY CODE
        LXI     H,GETDAY
        CALL    CLCKOT
        CALL    TTYIN  ; GET THE CODE
        PUSH    PSW    ; SAVE CODE
        CALL    TTYIN  ; GET THE ENDING CR
        POP     PSW    ; RESTORE CODE
        LXI     B,9    ; LENGTH OF DAY LITERALS
        LXI     H,DAYS ; POINT TO LITERALS
        SUI     30H    ; CONVERT CODE FROM ASCII TO BINARY
        JZ      MOVDAY ; IF MONDAY, THEN WE'RE SET TO MOVE
TIMER1  DAD     B      ; IT ELSE INDEX TO PROPER DAY
        DCR     A      ; LITERAL
        JNZ     TIMER1
MOVDAY  LXI     D,SDAY ; POINT WHERE TO STORE CURRENT DAY
TIMER2  MOV     A,M    ; MOVE LITERAL THERE
        STAX    D
        INX     H
        INX     D
        DCR     C
        JNZ     TIMER2
        LHLD    SAVSP  ; RESTORE CALLER'S STACK POINTER
        SPHL
        RET            ; AND RETURN TO HIM
CLCKIN  PUSH    H      ; ROUTINE TO GET CHRONO. RESPONSES
        PUSH    B
        CALL    TTYIN  ; INPUT FROM PORT 320Q
        POP     B
        POP     H
        MOV     M,A
        INX     H
        DCR     B
        JNZ     CLCKIN
        RET
CLCKOT  MOV     C,M    ; ROUTINE TO SEND CHRONO. COMMANDS
        PUSH    H
        PUSH    B
        CALL    TTYOUT ; OUTPUT TO PORT 320Q
        POP     B
        POP     H
        INX     H
        DCR     B
        JNZ     CLCKOT
        RET
        ENDIF                  ; <== LAST LINE TO ADD
WBOOT:  LXI     SP,STACK
        EI
```

I must mention that this routine works if one sets the Chronograph to the 24 hour clock, or military time. In this mode, the time is always the same length; with AM/PM time, the response can be 9 or 10 characters long and one needs to test for the CR instead of a constant length.

To set the time and date dividers and add the jump to the access code, one must next modify the cold boot code. While in ED, enter the command NCB12: ↑ Z and this will take you to the appropriate place. Then add the following code to accomplish the last of the BIOS modification.

```
CB12:
        XRA     A
        STA     LOGDSK
        LXI     H,BIOS
        SHLD    BBIOS
        IF      CHRONO      ; <== START ADDING THIS CODE
TIMER3  MVI     B,6
        LXI     H,SETTDV
        CALL    CLCKOT      ; SEND SET TIME DIVIDER COMMAND
        CALL    TTYIN       ; GET '0' RESPONSE
        CALL    TTYIN
        MVI     B,6
        LXI     H,SETDDV
        CALL    CLCKOT      ; SEND GET DATE DIVIDER COMMAND
        CALL    TTYIN       ; GET '0' RESPONSE
        CALL    TTYIN
        LXI     H,TIMER0    ; GET ACCESS CODE  ADDRESS
        SHLD    WBOOTE-2    ; WRITE OVER COLD BOOT JUMP
        ENDIF               ; <== LAST LINE TO ADD
        MVI     A,BT$CD
        JMP     GOW
        IF      CHRONO      ; <== START ADDING AGAIN
SETTDV  DB      'ATVT:',CR  ;SET TIME DIVIDER
SETDDV  DB      'ATDT/',CR  ;SET DATE DIVIDER
        ENDIF               ; <== THE LAST LINE TO ADD
SMSG0:  DB      CR,LF,LF,0
SMSG1: .....
```

This modifies the BIOS, but "how do you get the date and time from this?" I hear being asked. Well as I said before I wanted to put the time and date on printouts, so I wrote a program to call the access code and print it at the end of the listings on my printer. I gave it the original name... TIME.ASM. Here's its commented listing.

```
; THIS PROGRAM CALLS THE MODIFIED BIOS CHRONOGRAPH ACCESS
; ROUTINE AND THEN PRINTS THE DAY, DATE AND TIME ON THE CRT.
;
; AUTHOR: GLEN E. HASSEBROCK, JR.
;
; COMMAND FOR CRT DISPLAY ONLY:     A>TIME
;
; COMMAND FOR CRT AND LST DISPLAY:  A>TIME P
;
; THIS IS THE 8080 VERSION
;
BASE    EQU     0     ; CP/M EQUATES
TPA     EQU     BASE+100H
;
BDOS    EQU     5
LSTCHR  EQU     5
PSTRNG  EQU     9
CR      EQU     13
LF      EQU     10
DAYOFF  EQU     27    ; OFFSET OF "DAY" FROM EXECUTABLE CODE
DMA     EQU     128
;
        ORG     TPA
TIME:
        LXI     H,0   ; GET CALLER'S STACK AND SAVE IT
        DAD     SP
        SHLD    SAVESP
        LXI     SP,STACK
        ORA     A           ; CLEAR CARRY FLAG
        LHLD    BASE+1      ; GET ADDRESS OF WARM BOOT JMP
        MOV     A,L
        SBI     2           ; POINT BACK TO ADDRESS IN
        MOV     L,A         ; COLD BOOT          \OVERWRITTEN
        MOV     E,M         ; GET ADDRESS OF CHRONO. ACCESS
        INX     H                                \ROUTINE
        MOV     D,M
        XCHG
        PUSH    H           ; SAVE IT
        LXI     D,TIME1     ; GET RETURN ADDRESS
        PUSH    D           ; PUT ON STACK
        PCHL                ; CALL CHRONO. ACCESS ROUTINE
TIME1:
        POP     H           ; GET BACK ADDRESS OF CHRONO.
        ORA     A           ; CLEAR CARRY          \ROUTINE
        MOV     A,L         ; SUBTRACT BACK TO DAY ADDRESS
        SBI     DAYOFF      ; SINCE IT LIES BEFORE START OF
        MOV     L,A         ; ROUTINE EXECUTABLE CODE
        JNC     TIME2
        DCR     H
TIME2:
        LXI     D,MDAY      ; POINT TO DAY IN MSG
        MVI     B,9
TIME3:                      ; MOVE DAY FROM ROUTINE TO MSG
        MOV     A,M
        STAX    D
        INX     H
        INX     D
        DCR     B
        JNZ     TIME3
;
        LXI     D,MDATE     ; POINT TO DATE IN MSG
        MVI     B,8
TIME4:                      ; MOVE DATE FROM ROUTINE TO MSG
        MOV     A,M
        STAX    D
        INX     H
        INX     D
        DCR     B
        JNZ     TIME4
;
        INX     H           ; SKIP OVER CARRIAGE RETURN
;
        LXI     D,MTIME     ; POINT TO TIME IN MSG
        MVI     B,8
TIME5:                      ; MOVE IT TO MSG
        MOV     A,M
        STAX    D
        INX     H
        INX     D
        DCR     B
        JNZ     TIME5
;
        LXI     D,MSG       ; PRINT THE MSG ON CON:
        MVI     C,PSTRNG
```

```
        CALL    BDOS
;
        LDA     DMA         ; CHECK FOR CHARACTER ENTERED WITH
        CPI     2           ; 2 = SPACE + 'P'          \COMMAND
        JNZ     ENDIT       ; IF NOT, WE'RE FINISHED
;
        INX     H           ; POINT TO CHARACTER ENTERED
        INX     H
        MOV     A,M         ; GET IT AND SEE IF IT WAS A 'P'
        CPI     'P'
        JNZ     ENDIT       ; IF NOT, WE'RE FINISHED
;
        LXI     H,MSG       ; ELSE PRINT ON LST: DEVICE
TIME6:
        MVI     C,LSTCHR
        MOV     A,M
        CPI     '$'         ; UNTIL '$' = END OF MSG
        JZ      ENDIT
        MOV     E,A
        INX     H
        PUSH    H
        CALL    BDOS
        POP     H
        JMP     TIME6
ENDIT:
        LHLD    SAVESP      ; RESTORE CALLER'S STACK
        SPHL
        RET                 ; RETURN TO CALLER
MSG:
        DB      CR,LF,'TODAY IS '
MDAY:
        DS      9
        DB      ', '
MDATE:
        DS      8
        DB      '.  THE TIME IS '
MTIME:
        DS      8
        DB      '.',CR,LF,'$'
SAVESP:
        DS      2           ; PLACE TO SAVE CALLER'S STACK
        DS      16          ; LOCAL STACK
STACK:
        END
```

Now for those of you with a desire to stay away from assembly language programming, there are two methods for you to use. The first is to modify the BIOS, as above, and call it from a BASIC program, then peek into memory to get the day, date and time. The second is to write BASIC code to get the information from the Chronograph.

The former is a bit more complex than the latter and would have to be changed if the BASIC program is to be compiled versus interpreted. Therefore, I will present the second option.

In the back of the Chronograph's manual are listings of BASIC routines for systems other than Heath/Zenith, as I stated before. However, it can still be done on Heath/Zenith '89s and H8s (with the H-8-4). The following BASIC program is an adaptation of the manual's programs, and will work on MBASIC 5.21, even compiled if you so desire. To run on earlier versions of MBASIC, the names of all variables must be shortened to two significant characters. As I stated earlier the baud rate for the Chronograph port must be 300 for use by interpreted MBASIC.

```
10 DIM DAY$(6):REM ESTABLISH TABLE OF DAY LITERALS
20 DAY$(0)="MONDAY"
30 DAY$(1)="TUESDAY"
40 DAY$(2)="WEDNESDAY"
50 DAY$(3)="THURSDAY"
60 DAY$(4)="FRIDAY"
70 DAY$(5)="SATURDAY"
80 DAY$(6)="SUNDAY"
90 OST$="ATVT:":GOSUB 300:REM SET TIME DIVIDER
100 OST$="ATVD/":GOSUB 300:REM SET DATE DIVIDER
110 GOSUB 240:REM GET THE TIME
120 GOSUB 260:REM GET THE DATE
130 GOSUB 280:REM GET THE DAY
140 PRINT
150 PRINT "TODAY IS "DAY$", "DATE$".  THE TIME IS "TIME$
160 PRINT
170 PRINT "SHALL I COPY TO LST:? <Y OR N> ":A$=INPUT$(1)
180 IF A$<>"Y" THEN 390
190 LPRINT
200 LPRINT "TODAY IS "DAY$", "DATE$".  THE TIME IS "TIME$
210 LPRINT
220 GOTO 390
230 ' CHRONO TIME ROUTINE
240 OST$="ATRT":GOSUB 300:TIME$=IST$:RETURN
250 ' CHRONO DATE + TIME ROUTINE
260 OST$="ATRD":GOSUB 300:DATE$=IST$:RETURN
270 ' CHRONO DAY ROUTINE
280 OST$="ATRW":GOSUB 300:DAY$=DAY$(VAL(IST$)):RETURN
290 ' CLOCK I/O
300 IF OST$="" THEN 380
310 FOR I%=1 TO LEN(OST$):OUT &O320,ASC(MID$(OST$,I%,1))
320 IF (INP(&O325) AND 32)=0 THEN 320
330 NEXT I%:OST$=""
340 OUT &O320,13:IST$=""
350 IF (INP(&O325) AND 1)=0 THEN 350
360 C%=INP(&O320)
370 IF C%<>13 THEN IST$=IST$+CHR$(C%):GOTO 350
380 RETURN
390 END
```

Now since some of you don't have modem control programs, I suspect that you would like some way to set the Chronograph. Think about it a bit, consider all that Dr. Campbell has taught you in the past issues of REMark, and see if you can write the program on your own. If you need some help, here is a little BASIC program to get you started.

```
5 DIM DAY$(6):REM ESTABLISH TABLE OF DAY LITERALS
10 DAY$(0)="MONDAY"
15 DAY$(1)="TUESDAY"
20 DAY$(2)="WEDNESDAY"
25 DAY$(3)="THURSDAY"
30 DAY$(4)="FRIDAY"
35 DAY$(5)="SATURDAY"
40 DAY$(6)="SUNDAY"
45 GOTO 105
```

```
50 ' CLOCK I/O
55 IF OST$="" THEN 95
60 FOR I%=1 TO LEN(OST$):OUT &O320,ASC(MID$(OST$,I%,1))
65 IF (INP(&O325) AND 32)=0 THEN 65
70 NEXT I%:OST$=""
75 OUT &O320,13:IST$=""
80 IF (INP(&O325) AND 1)=0 THEN 80
85 C%=INP(&O320)
90 IF C%<>13 THEN IST$=IST$+CHR$(C%):GOTO 80
95 RETURN
100 ' MENU SUBROUTINE
105 PRINT
110 PRINT"0.  All finished; end"
115 PRINT"1.  Set to display time"
120 PRINT"2.  Set to display date"
125 PRINT"3.  Set the time"
130 PRINT"4.  Read the time"
135 PRINT"5.  Set the date"
140 PRINT"6.  Read the date"
145 PRINT"7.  Set the day of week"
150 PRINT"8.  Read the day of week"
155 PRINT"9.  Set the date separator"
160 PRINT"10. Remove the date separator"
165 PRINT"11. Set the time separator"
170 PRINT"12. Remove the date separator"
175 PRINT"13. Set the alarm time"
180 PRINT"14. Remove the alarm time"
185 PRINT"15. Turn line feed option on"
190 PRINT"16. Turn line feed option off"
195 PRINT
200 PRINT"Select a function: ";:INPUT F%
205 IF F%<0 OR F%>16 THEN 555
210 IF F%=0 THEN END
215 ON F% GOTO 225,235,245,275,290,315,330,390,405,425,435,
                                    450,460,500,510,520
220 '   FUNCTION 1
225 OST$="ATDT":GOTO 530
230 '   FUNCTION 2
235 OST$="ATDD":GOTO 530
240 '   FUNCTION 3
245 PRINT"Input the current time in ONE of the forms:"
250 PRINT:PRINT"    HHMM00 or HHMM00A or HHMM00P ";
                                                  :INPUT T$
255 IF LEN(T$)<6 OR LEN(T$)>7 THEN 555
260 OST$="ATST"+T$
265 GOTO 530
270 '   FUNCTION 4
275 OST$="ATRT":GOSUB 55
280 PRINT"The time is set to "IST$".":GOTO 105
285 '   FUNCTION 5
290 PRINT"Input the current date in the form YYMMDD: ";
                                                  :INPUT T$
295 IF LEN(T$)<>6 THEN 555
300 OST$="ATSD"+T$
305 GOTO 530
310 '   FUNCTION 6
315 OST$="ATRD":GOSUB 55
320 PRINT"The date is set to "IST$".":GOTO 105
325 '   FUNCTION 7
330 PRINT"0 = MONDAY"
335 PRINT"1 = TUESDAY"
340 PRINT"2 = WEDNESDAY"
345 PRINT"3 = THURSDAY"
350 PRINT"4 = FRIDAY"
355 PRINT"5 = SATURDAY"
360 PRINT"6 = SUNDAY"
365 PRINT"Enter the number corresponding to today: ";
370 T$=INPUT$(1):PRINT T$
375 IF T$<"0" OR T$>"6" THEN 555
380 OST$="ATSW"+T$:GOTO 530
385 '   FUNCTION 8
390 OST$="ATRW":GOSUB 55
395 PRINT"The day is set to "DAY$(VAL(IST$))".":GOTO 105
400 '   FUNCTION 9
405 PRINT"Type the character to use as a separator
                                          in the date: ";
410 T$=INPUT$(1):PRINT T$
415 OST$="ATVD"+T$:GOTO 530
420 '   FUNCTION 10
425 OST$="ATVD":GOTO 530
430 '   FUNCTION 11
435 PRINT"Type the character to use as a separator
                                          in the time: ";
440 T$=INPUT$(1):PRINT T$
445 OST$="ATVT"+T$:GOTO 530
450 OST$="ATVT":GOTO 530
455 '   FUNCTION 13
460 PRINT"Input the alarm time in ONE of the forms:"
465 PRINT:PRINT"    HHMM or HHMMA or HHMMP "
470 PRINT"(note that this form must match the
                                    current time form)"
475 INPUT T$
480 IF LEN(T$)<4 OR LEN(T$)>5 THEN 555
485 OST$="ATAS"+T$
490 GOTO 530
495 '   FUNCTION 14
500 OST$="ATAC":GOTO 530
505 '   FUNCTION 15
510 OST$="ATLS":GOTO 530
515 '   FUNCTION 16
520 OST$="ATLC"
525 '    COMMON COMMAND CHRONO. CHECK RESPONSE
530 GOSUB 55
535 OK=VAL(IST$)
540 IF OK=0 THEN PRINT "DONE":GOTO 105
545 IF OK=8 THEN PRINT "program error for this function.":
    GOTO 105
550 PRINT "The write-protect switch is up."
555 PRINT CHR$(7)"This command was ignored.":GOTO 105
```

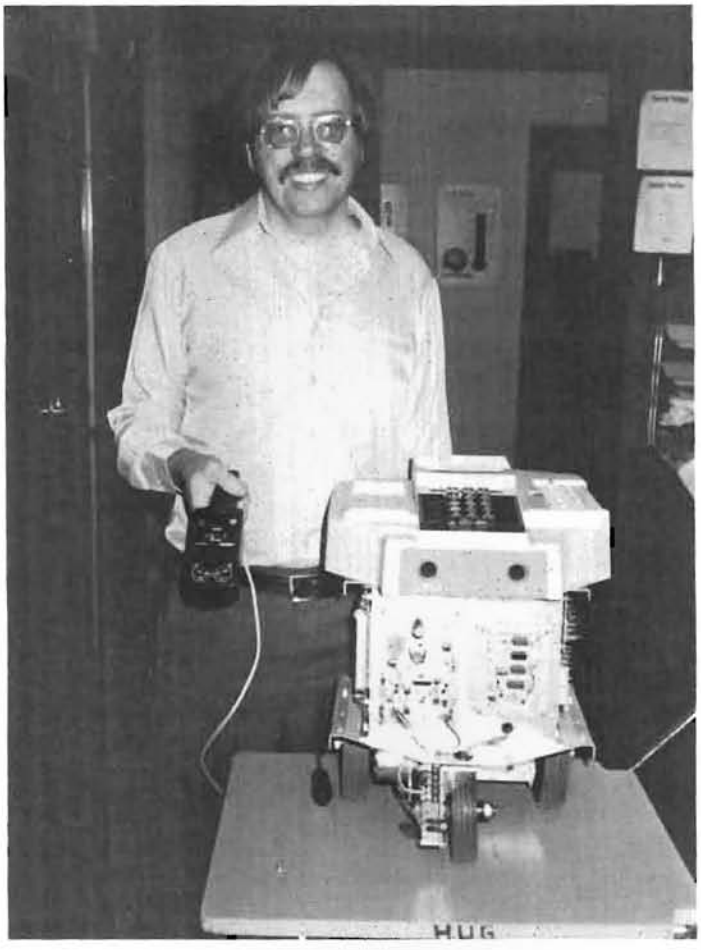
Well my Chronograph is telling me it's time to let you go back to playing with your own systems. I hope this helps those of you who need, or prefer, accurate time, or maybe just the convenience of not having to set it every time the system is booted. My Chronograph has performed flawlessly since I got it, except it has been losing a few seconds per couple of weeks. Now if I could persuade my pock-

etbook to let me get a Heathkit frequency counter...

Some final notes...I set CP/M to run TIME.COM upon cold boot and have the date and time printed as a greeting by the time the CRT warms up. Don't forget, CP/M is a registered trademark of Digital Research, Hayes Stack Chronograph is a registered trademark of Hayes Microcomputer Products, and of course you know Heathkit, H89 and H8 are registered trademarks of Heath Company.

About the Author:

Glen is a data processing consultant for Nims Associates, Inc., located in Decatur, Illinois. His "computer" background includes COBOL, PL/I, FORTRAN, BASIC, TSO, TSO/SPF, IBM's IMS DB/DC & MFS, Z80 assembler, and 8080 assembler. Besides consulting work, he has taught at the Decatur area community college. He graduated from Illinois State University, in December 1977, with a B.S. degree, after majoring in Chemistry and Applied Computer Science. By the time you read this, he will have recently become married, not to his H89, but to his heartthrob of 11.2635 years. His major hobbies are building Heathkit kits (who doesn't?), reading (to keep up with the computers) and trying things out on his H89.



It Walks, It Talks...

(A Preview of the ET-18 Robot)

Pat Swayne
Software Engineer

The lil' fella on the cover of this issue represents one of Heath Company's boldest adventures into new products. It is the ET-18 robot, also known as HERO I (Heath Educational RObot). Designed to teach the principals of robotics to the builder and user, HERO I is microcomputer controlled, mobile, and completely self contained. It can interact with its environment using sensors to detect light, sound, motion, and the passage of time, and can detect built in obstacles in its path with sonar. An optional arm allows it to manipulate small objects, and it can speak using an optional voice board.

HERO I stands 20 inches (50.8 cm) tall and weighs 39 pounds (17.7 kg). It moves on three wheels with the single front wheel serving as both the drive and steering wheel. It has a moveable head (350 degrees), and all sensors and the arm mount on the head. It is powered by 4 rechargable 6-volt gel cell batteries connected in two 12 volt banks. One bank powers the drive motors, and the other powers the electronics. A battery charger is provided which may be left connected and turned on all the time if you wish.

The processor in HERO I is a 6808, a new version of the 6800, which executes the same instruction set. There are several ways to program the robot. It can be moved through a sequence using a special remote control called the "teaching pendant", with each move stored. It can also be programmed using the keypad on its head in either 6800 machine language or a special interpretive "Robot Language". Machine language and Robot Language can also be mixed. Sophisticated programming involving simultaneous operation of several functions and interrupts is possible.

The HERO I Kit

My introduction to HERO I was in the form of a pre-release version of the kit. Being such, I expected it to contain errors. I will not discuss those errors in detail in this article, since, hopefully, they will all be fixed in the release version.

The kit is a kitbuilders dream or a kitbuilders nightmare, depending on your point of view. Like other Heath computer products, the CPU board is already assembled and tested (I had to build mine), but that is where any similarity ends, except possibly with the original H8. Every other board must be assembled by the builder, including the I/O (Input/Output) board, which is larger and more complex than the CPU board. There are 10 boards to assemble in the basic kit, and most of them have several analog parts, which means that you have a lot of different little parts to keep track of, and a lot of work to do.

The mechanical and wiring assembly is where the real work lies. In addition to having to assemble various drive mechanisms and the body and head assemblies, you have to make up a lot of cables and install approximately 200 spring clips on the ends of the wires. There are two ready made cables, with some of the connecters already installed, but they leave some of the work for you on those cables as well.

The "torso" of the completed robot is a box with most of the cables on the inside and all of the boards (except the CPU, which is inside the box, and some boards in the head) mounted outside the box on connecters that allow easy removal and replacement. A door on the back side of HERO I's torso box allows access to the CPU board and the "rat's nest" of wiring (see Photo 2). The wiring will probably not look quite as messy in the release version, since in my kit some wires turned out to be too long, and some too short. Even though the kit is quite complex, it is not really too difficult. It takes time, though. Heath estimates 40 to 60 hours will be needed, and that's about right.

Surprisingly enough, my HERO I worked without any major problems when I finally got him together and fired him up. There were a couple of wires backwards that caused the steering and head motors to turn the wrong way due to errors in the pre-release assembly manual, but I have yet to find an error of my own. I consider myself a pretty good kit builder, but I must also give credit to Heath, who have done their usual excellent job in preparing the assembly manual. At this writing, everything works except the sonar receive circuitry, so I may yet find that I have made an error. By the way, in case they don't put this in the manual, you can isolate a sonar problem to either the transmit or receive section by listening to the transmit tube for the faint clicking of the sonar pulses.

Several tests are provided in the assembly manual to help you make sure your robot is working, once you get it together. The first power on tests and adjustments are on the power supply, then the CPU is tested. The CPU test and the other tests following it

are done by entering simple commands or small programs into the robot's keypad and observing the digital display as you make adjustments.

Programming and Using HERO I

As I mentioned before, you can program the robot by manually controlling it with a remote control, but that is not the best way to do it. You can only perform one operation at a time, except for moving and steering, and you cannot access any of the senses in this mode. The preferred method is to enter carefully worked out programs into the robot's keypad. The value of the remote control is to learn how the robot moves, and to see visually how a certain move translates into the numbers that the robot uses internally. This is made easy because while you are using the remote control, the digital display shows numbers that relate to your actions. For example, if you are moving the robot forward or backward, numbers start counting upward on the display from the time you pull the "trigger" on the control until you release it. You can move the robot a specified distance and see how many "robot units" that movement uses. If you then enter a program into the robot that commands it to move that many units, it will move the same distance that it did with the remote control.

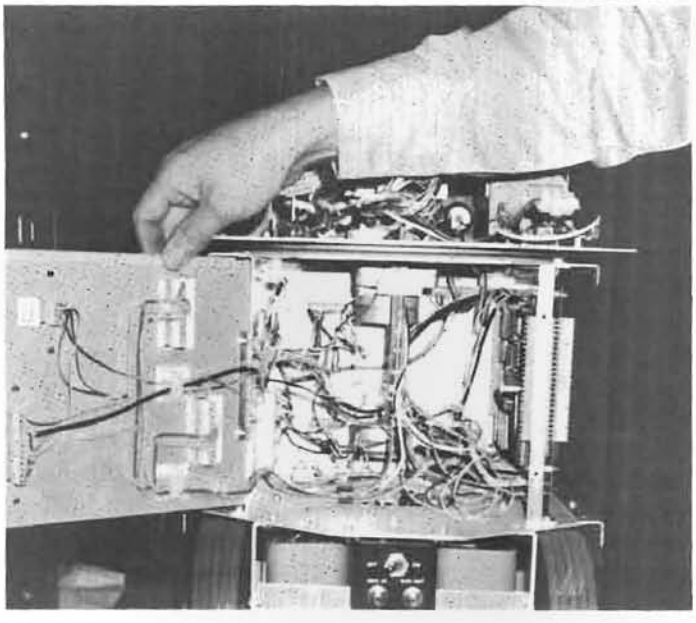
The Robot Language interpreter is one of the most clever things about the robot. It is actually a pseudo machine language, and makes use of opcodes that are not part of the 6800 instruction set. (There are several unused opcodes because the 6800 is a memory oriented processor, not register oriented like the 8080 and Z80. Most of the 8080's opcodes are instructions to move data between registers.) While you are running in the interpreter, you can include 6800 instructions right along with the robot instructions, and they will be executed as normal 6800 instructions. However, they will be executed at a much slower speed because they are being performed by the interpreter, not directly by the 6800. To execute 6800 instruction at full speed, you can make use of a Robot Language instruction to "turn off" the interpreter, and you can turn it back on using the 6800's SWI instruction (Software Interrupt, like the 8080's Restart instruction).

The Robot Language interpreter includes instructions to enable and disable the various senses, and to control the voice and the drive motors. The voice and drive motor instructions have "continuous" and"wait" modes. In the continuous mode, the interpreter will start the action called for and go on to the next instruction immediately. This makes it possible to have several actions taking place simultaneously. In the wait mode, the interpreter will not execute the next instruction until the current action is completed.

There are four Branch if Busy instructions in Robot Language that let you selectively perform operations if parts of the robot are busy doing something. There is also a special Jump if Speaking instruction that lets you string together phrases that may be in different parts of the robot's memory.

The robot's senses can be polled in a running program to make interaction with its environment possible. There are also 3 hardware interrupts that can be used in a program to alter the robots actions in response to stimuli. They are the Motion Detect interrupt, the Trigger interrupt, and the Experimental Board interrupt. The Trigger interrupt is initiated by the trigger on the remote control, and the Experimental Board interrupt comes from a solderless breadboard on the robot's head, where you can build your own experimental circuits.

There are also 3 user programmable keys on the robot's keypad. They can be used to cause a jump to a specific address within your program when one is pressed.



The voltage of the drive and logic batteries is monitored by the CPU and reported on the display and by the voice option if it is installed when the batteries are low. The robot can be put into a sleep mode in which only the most essential functions are kept going for a specified period of time. I am told that in that mode the batteries will last several weeks. It has a clock-calendar built in, so it could be made to sleep until a certain time on a certain day, then perform a task.

Even if you haven't been able to follow all of the previous technical information, you've probably come to the conclusion that the ET-18 is a fairly complex gadget. Heath designed it primarily as a training device, and a course in robotics will be available to accompany it, but many people may buy it as a status toy. The user's manual seems to have been written with that possibility in mind, with humorous suggestions of things to do with it. For example, you could have it wait in a coat closet with the motion sensor turned on and have it greet unsuspecting neighbors or guests when they open the door.

Even though HERO I is a training device and not designed for real work, it could be put to a number of useful tasks. It could serve as a mobile burglar alarm that could monitor pre-determined areas. It could be used to move or manipulate small amounts of toxic or dangerous substances. It can even wake you up in the morning, and remember to let you sleep longer on Saturday and Sunday. The age of robotics is here, and Heath has taken one of the first steps to bring it into being.

from a cartoon appearing in the NOV. 1982 issue of *Creative Computing*.

A User's Review of EDIT19

A. E. McLaughlin Jr.
32 West End Avenue
Brentwood, NY 11717

EDIT19 is, as the name suggests, an editor specifically designed for the Heath/Zenith terminal. It is, in my opinion, the best combination screen and line editor usable on the H8/H19 or H89. At the price of $80.00 for Version 3.0, no other editor usable on these microprocessors even comes close.

EDIT19 Version 2.0 was reviewed in Buss #24 back in July, 1980. At the time I was working with my third editor, PAGED. I was not satisfied with EDIT which came with my HDOS nor the other line editor I purchased. EDIT19 sounded more like what I was looking for than any of the higher priced editors I had heard about and at $50.00 sounded like a good deal. It was, but I still had a problem since its one big shortcoming was that it couldn't handle files larger than the available storage. This forced me to switch back to PAGED whenever I had a large file.

EDIT19 Version 3.0 ended all that! Steve and Kay Robbins have refined and developed this product into one that can handle any size disk file that can be contained on one disk. Using a "virtual memory" concept and three disk drives, EDIT19 can take such a file and, by creating two work files (one for the top and another for the bottom) can still have a large amount of the file in memory PLUS additional workspace for additions. This means that a global search can be done on the WHOLE file with one command!

All the abilities of Version 2.0 have been retained in this latest release with many new features. The latest version, although larger, now allows MORE space in the work area under most conditions since a large number of the commands are now contained in overlays which are only loaded when an overlay command is executed. Unless the AUTLOAD switch is on, the overlay is discarded after execution of the instruction.

A default list allows custom configuration of EDIT19 to load only the desired overlays, start in SCREEN or COMMAND mode, select numerous options or switch settings and, designate device and extension for hardcopy, prolog, macro, catalog, overlays, virtual, help, log and HAND files individually! EDIT19 now has about 150 commands, an increase of almost 50% with user control of so many options that I'm still refining the way I use it. Even after you finally set up all the options and save the new configuration (using DEFSAVE to write them directly into the program on the disk), you can STILL load the program in the opposite mode and/or with NO overlays (to allow you to process a special file) just by following the program name on the command line with a couple of special characters. You could then specify AUTLOAD and whenever you entered a command which was not resident, the overlay which contained it would be permanently loaded creating a custom configuration.

Cursor control is now better than ever with the ability to move the cursor anywhere on the screen through the use of the keypad keys. They allow you to move the cursor horizontally by letter, word, tab of line (using the home key to go to the left margin and the shift home to go to the first position after the last character on the line) and to scroll vertically by line (using the up and down arrows) or page (by using the decimal key to scroll up and the zero key to scroll down). Vertical positioning in a file can also be accomplished using the "P" (for page) command to jump down 24 lines (or up using "-p"), "Para" to jump to the next paragraph ("-para" jumps to the preceding paragraph), "B" jumps to the last line and "T" to the top (Line 0 in command mode).

Many editing commands are available, some of which can be used directly in screen mode through the use of the CTRL and ESC keys. CTRL k will concatinate the current line with the next and ESC followed by RETURN will break the line at the cursor putting the rest on the next line. CTRL-N will change the case of the character at the cursor, CTRL-R and T remove and set tabs and CTRL-E and F scroll up and down a paragraph. The IC, DC, IL and DL keys are active full time also but have broader functions when shifted. Formatting with left or left AND right justification is available for creating beautiful output with automatic left and right margins directly from screen mode.

With the SCREEN overlay resident, you can even execute most commands by just pressing the ENTER key to move the cursor to the command line, type the command and RETURN. EDIT19 automatically redisplays the screen and returns to where you left off after execution. Even faster execution can be achieved by setting one of the special function keys to one command (or a series of commands) since a single keystroke will then cause execution. Since a special function key can be set to any series of ascii characters and be put into IMMEDIATE mode, it can be used to insert a line, phrase or group of non-printing codes anywhere into a file. Want to repeat the last command? Press the "-" and away you go! You don't even have to RETURN!

In a recent issue of Popular Computing, a reader asked if it were possible for a word processor program to convert English pounds to dollars in a text file. The answer was that no word processor type program could do that since it involved not only the ability to differentiate between pounds, the money and pounds, the weight but also it would have to recognize amounts written both numerically and in words. EDIT19, with its MACRO functions and commands has this capability. (A macro is a program which runs within the editor using the editors commands as instructions.) EDIT19's MACRO instructions include all the normal commands plus the following:

APPSPFn	Append to special function key
CALL	Call a MACRO subroutine
COMMAND	Exit the macro, keeping the macro pending.
DISPLAY	Display the next line of the macro
JUMP	Jump to the label given (conditional jumps allowed)
MACLCR	Set the MACRO LOGICAL CARRIAGE RETURN symbol.
RETURN	Return from a macro subroutine.
VARCHR	Set the VARIABLE CHARACTER used for variables, (macro arguments and labels)
XSPFn	Execute a special function key.

Version 3.0 comes with a complete 80 page manual and a separate tutorial disk and manual is available for an extra $20.00. Version 2.0 registered owners can get Version 3.0 for $35.00. For one of the finest programming bargins around, write to Steven Robbins, 4610 Spotted Oak Woods, San Antonio, TX 78249.

Zenith Data Systems

ZF-110-22 (low profile/2 drive) $3240
ZF-120-22 (all-in-one) $3320
Z-100 Software Pack $405
Z-89-81 (48K with HS drive) $2263
ZVM-121 Monochrome Monitor $99
ZVM-134 RGB Color Monitor $570
Z-25AA 150cps Printer $1295
CBASIC for CP/M $99
HDOS Operating System $115
CP/M Operating System $125
UCSD Pascal Operating System $230
Microsoft Basic for CP/M $145
Microsoft Fortran for CP/M $165
Microsoft Cobol for CP/M $340
Microsoft Basic Compiler (CP/M) $195
WordStar $310
DataStar $230
MailMerge $110
SuperSort $160

D.C. Hayes

Smartmodem 300/1200 ..... $525
Smartmodem 300 (only) $219
Chronograph $189

Ashton-Tate

dBASE II $450

Fox & Geller Associates

Quickscreen (for dBASE II) $135
Quickcode (for dBASE II) $239
d-UTIL (for dBASE II) $69

Sorcim

Trans-86 (8080/8088 translator) $105

N.E.C.

12" Hi-resolution RGB Monitor $775

Software Wizardry, Inc.

Z-100 CPU RAM Chips $72/64K
Z-100 Video RAM Chips $64/64K
The HDOS Toolkit (HDOS) $29.95
ZLYNK Modem Utility (HDOS) $24.95
EDBS Database (HDOS or CP/M) $49.95
CRASH disk recovery (HDOS) $24.95
MDS Spooler (HDOS) $24.95

Terms of Sale:

Prices and availability are subject to change without notice. Shipments are FOB our address and purchase orders handled only with prior acceptance. Add 1% shipping and handling plus 4⅛% sales tax in Missouri (continental U.S.A. only). MasterCard and Visa accepted. We handle many items not listed above, so please call for pricing if you don't see it listed here.

Software Wizardry, Inc.

122 Yankee Drive, St. Charles, Mo. 63301

(314) 946-1968

VISA MasterCard

Introduction To Z-BASIC Part II

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

This is the second article in a series of articles dealing with the new commands of the **Z-100**'s **Z-BASIC** over **BASIC-80**. Last month's article dealt with the new **CLS** and **LINE** commands of **Z-BASIC**, this month we will discuss the use of the **LOCATE** command and the 25th line. Some of you out there will say that the 25th line is not a new feature of the Heath/Zenith computer line, however the **Z-100**'s **Z-BASIC** uses the 25th line in a slightly different way.

The **LOCATE** command is similiar to direct cursor addressing except it does not require the use of a **PRINT** command and it is a little easier to understand than direct addressing.

NOTE: The direct cursor addressing may no longer be used to print characters on the screen. It may still be used with the erase commands to erase to the end of the line, end of the page or beginning of the page. All character displaying must be done with the **LOCATE** command.

The **LOCATE** as you may suspect is used to locate something on the screen. The syntax for using the **LOCATE** command is the word **LOCATE** followed by the veritcal row (1 to 25), a comma (,) and then the horizontal position (1 to 80).

Example: Print the word "Test" in the middle of the screen.

```
10 CLS:'                Clear Screen
20 LOCATE 13,38:'       Select Location
30 PRINT "Test":'       Print Message
```

The reason line thirteen was chosen and not twelve is that there are actually twenty-five lines on the screen. The number thirty- eight was chosen to place the middle of the word "Test" in the very middle of the horizontal line.

Any position may be addressed using the locate command, however if a value for the vertical position is larger than twenty-five (25) or the horizontal position is larger that eighty (80) an error of an 'Illegal function call' will occur. The same error message will appear if the value of either number is a negative number or the value is less than one half (.5). All numbers are rounded to the nearest whole number, rounding .5 up to the next whole number.

The twenty-fifth line may also be adddressed using the **LOCATE** command and may be erased using the same code as used on the **H/Z-19/88/89/90** the ESCape (CHR$(27)) "y1". When using the ESCape "y1" to turn off the 25th line you must turn it back on using the ESCape "x1". **Z-BASIC** defaults to the 25th line being turned on.

Example: Let's number the lines from 1 to 25, print a message in the 25th line, pause, erase the 25th line and display another message in the 25th line.

```
10 CLS:'                Clear the screen
20 E$=CHR$(27):'        E$=ESCape
30 FOR I=1 to 25:'      Start Loop
40 LOCATE I,10:'        Select Location
50 PRINT I;:'           Print Line Number
60 NEXT I:'             Continue Loop
70 PRINT "This is the 25th line turned on."
80 FOR I=1 to 1000:NEXT I:'Pause
90 PRINT E$"y1":'       Turn Off 25th Line
100 PRINT E$"x1":'      Turn On 25th Line
110 LOCATE 25,1:'       Address 25th Line
120 PRINT "This is the 25th line back ON again."
```

The 25th line may be used for displaying special messages to assist the program user or the programmer. These messages may be information about the program such as titles or they may be information on how to use the special function keys. More on the special function keys later.

The 25th line **WILL** be erased by the **CLS** command, however the ESCape "E" may be used to erase only the first twenty-four lines. When using the ESCape "E" the cursor must not be on the 25th line or only the 25th line will be erased. As a precaution when using the ESCape "E" you may choose to goto the upper left hand corner of the screen first before erasing the screen.

Example: First clear the screen, then set the variable E$ equal to ESCape, locate the word 'Test' in the middle of the screen, print a message on the 25th line, goto the upper left corner erasing everything except the 25th line, and ask for the input of the variable A$.

```
10 CLS:'                Clear Screen
20 E$=CHR$(27):'        E$=ESCape
30 LOCATE 13,38:'       Selected Middle Of Screen
40 PRINT "Test":'       Print The Word 'Test'
50 LOCATE 25,10:'       Select 25th Line
60 PRINT "We are on the 25th line.";
70 PRINT E$"H";:'       Go Home (Upper Left Corner)
80 PRINT E$"E";:'       Clear Screen Except 25th Line.
90 LINE INPUT A$:'      Wait For Input
```

The program displays the word 'Test' in the middle of the screen, writes the message (line 60) on the 25th line, then clears the screen and then the cursor will return to the 25th line and wait there for the variable A$ to be inputted. When **Z-BASIC** returns to the command mode the cursor will exit the 25th line and return to the 24th line. No more getting stuck in the 25th line. Have you ever listed a program in the 25th on the **H/Z-19/88/89/90**? It is pretty hard to find an error if everything comes out on one line. It looks kind of like one of those hand held computers with everything jumping around in one line.

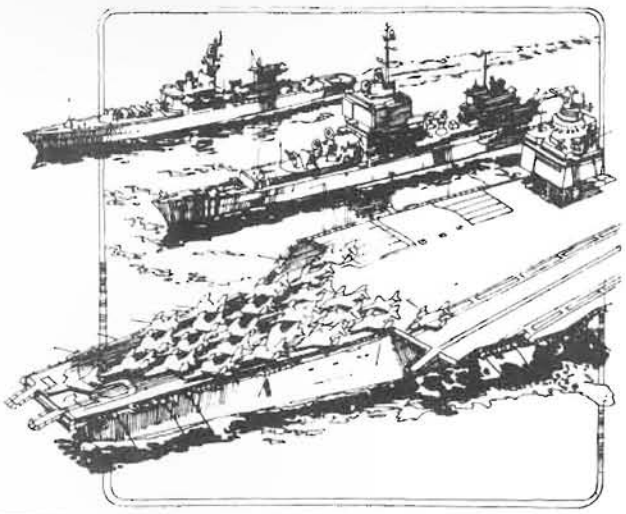
Let's take a look at one more feature of the **Z-100**'s **Z-BASIC** that will be very useful when using the 25th line. The commands that are to be used are the **KEY** command which has four options. They are **OFF**, **ON**, **LIST** and 'assigning of the special function keys'.

Z-BASIC using the special function keys to assist the programmer by providing some very common commands available by pressing only one key. The special functions key are initially, programmed or assigned the following values:

F1 LIST
F2 RUN
F3 LOAD"
F4 SAVE"
F5 CONT
F6 PRINT
F7 AUTO
F8 FOR
F9 NEXT
F10 GOSUB
F11 TRON
F12 TROFF

Vectored to 23 ☞

NEW HUG PRODUCTS



885-1211

CP/M Sea Battle $20.00

Introduction: This version of SEA BATTLE will execute on the H8, H89 (Z89), and the Z-100 computers. (This is simply a modified version to run on the Z-100 series machines.)

SEA BATTLE is a video action game that uses the H19 graphic metures. The role of the SEA BATTLE player is to command a Destroyer force in opposition to an aircraft carrier and its escort. The escort includes fighters, bombers, and submarines, all of which attack the player ship.

Requirements: SEA BATTLE requires the CP/M operating system, version 2.2.02 or later, on an H19/H8/H17 or H89 with 32K of memory. Only one disk drive is required.

For the Z-100 computer, SEA BATTLE requires the CP/M-85 operating system. The Z-100 requires the soft-sectored disk format. (Order with a "-37" suffix.)

The program is written in assembly language and the source code is included.

NOTE: *The H19 (H89) terminal is required for the graphics codes of the game.*

The following files are included on the HUG P/N 885-1211 CP/M Sea Battle graphics game disk:

README	.DOC
SEABATTL	.DOC
SEABATTL	.ASM
SEABATTL	.COM

The SEABATTL.DOC file gives detailed instructions of the game. Most players will have fun learning by trial and error.

Author: Victor Abell

Program Content: SEA BATTLE is a very interactive game which requires the entire screen of the H19 terminal. The outline of the game structure is as follows:

A) The Aircraft Carrier and its Escort

The escort consists of an endless number of fighter planes, five heavy bombers, and a number of submarines. The fighter planes are replenished from the aircraft carrier as each is shot down. Each time the aircraft carrier is hit a submarine appears on the horizon.

B) The Player Ship

The player ship has two guns which may only be fired one at a time. The ship moves back and forth across the bottom of the screen dodging the shells and bombs from the air force. The player ship can take a few hits from the fighters, but is destroyed when hit by a bomb. The player ship must flee from the submarines.

C) Scoring

The score (and other data) is displayed on the 25th line throughout the game. The following is the scoring table:

Down a Fighter Plane	5 points
Down a Bomber	50 points
Hit the Carrier	50 points
Sink the Carrier	250 points plus Variable Bonus

The Variable Bonus is determined by the success of staying within the center of the screen, where the most action takes place. The program sets a timer and the player must stay within the marks or the countdown begins.

D) Sinking the Carrier

The aircraft carrier will sink when it has been hit by fifteen shots from the player ship guns. Once the carrier has been sunk, the player can watch it sink into the sea. Then another carrier is there to replace the previous carrier. The next carrier escort, more aggressive than the previous carrier's escort, appears on the screen.

E) Sinking the Player Ship

The player ship may be sunk by 12-15 hits from the shells of the fighters, by a single hit from a bomber, or by being caught by a submarine.

F) Summary

The player must find a battle strategy which will lift him in victory over as many aircraft carriers as possible. Each successive carrier is more dangerous than it's predecessor.

Comments: SEA BATTLE is one of the finest games which has been released from HUG.

885-1222
CP/M Adventure Disk $10.00

Introduction: This version of ADVENTURE has been adapted for the H89 (Z89) CP/M and CP/M-85 for the Z-100.

ADVENTURE is one of the most well known and best liked computer games. It is an adventure through a giant cave to search out and find treasures. Many dangers are in store for the user who ventures into its midst.

Requirements: This game requires the CP/M operating system, version 2.2 or later, on H8/H17 or H-89 with a minimum of 24K of memory. Only one disk drive is required.

For the Z-100 computer, CP/M-85 is required. The soft-sectored disk format is required. (Order with a "-37" suffix.)

NOTE: *The H19 terminal is not required to play this game.*

The following is a list of the files on the HUG P/N 885-1222 CP/M Adventure Game disk:

README	.DOC
ADVENT	.COM
ADVENTUR	.DTB
NEWGAME	.CAV

Authors: This program was originally developed by Willie Crowther. Most of the features of the current program were added by Don Woods (Don @ SU-AI). This microprocessor version was done by G. Letwin of Heath Company.

Program Content: ADVENTURE

Somewhere nearby is a colossal cave, where others have found fortunes in treasure and gold, though it is rumored that some who enter are never seen again. Magic is said to work in the cave. I will be your eyes and hands. Direct me with commands of one or two words. I should warn you that I look at only the first four letters of each word, so you'll have to enter "northeast" as "NE" to distinguish it from "north", "dnstream" for "downstream", etc. Should you get stuck, type "help" and "info" for some general hints.

HELP

I know of places, actions, and things. Most of my vocabulary describes places and is used to move you there. To move, try words like forest, building, dnstream, enter, east, west, north, south, up, or down. I know about a few special objects, like a black rod hidden in the cave. These objects can be manipulated using some of the action words that I know. Usually you will need to give both the object and action words that I know. Usually you will need to give both the object and action words (in either order), but sometimes I can infer the object from the verb alone. Some objects also imply verbs; in particular, "inventory" implies "take inventory", which causes me to give you a list of what you're carrying. The objects have side effects; for instance, the rod scares the bird. Usually people trying unsuccessfully to manipulate an object are attempting something beyond their (or my!) capabilities and should try a completely different tack. To speed the game, you can sometimes move long distances with a single word. For example, "building" usually gets you to the building from anywhere above ground except when lost in the forest. Also, note that cave passages turn a lot, and that leaving a room to the north does not guarantee entering the next from the south.

SUGGESTIONS

Try "ENTER BUILDING".

When you see an object, pick it up.

Go "DNST" (downstream) if you want to find the cave.

HELPFUL WORDS

INVEntory	List items you are carrying
SCORe	Show your current score
LOOK	Type a long description of your current loaction
BACK	Go back the way you came
QUIT	Stop the game and give final score

COMMAND STYLE

Remember, ADVENTURE takes one or two word commands, only. Make them straight-forward, like:

ATTACK DRAGON
EAT BIRD
THROW AXE
GET GOLD
WEST
UNLOCK GRATE
DOWN

INFO

If you want to end your adventure early, say "QUIT". To see how well you're doing, say "SCORE". To get full credit for treasure, you must have left it safely in the building, though you get partial credit just for locating it. You lose points for getting killed, or for quiting, though the former costs you more. There are also points based on how much (if any) of the cave you've managed to explore; in particular, there is a large bonus just for getting in (to distinguish the beginners from the rest of the pack). I may occasionally offer hints if you seem to be having trouble.

****** G O O D L U C K ******

Comments: ADVENTURE will provide many months and even years of fun trying to reach the 350 points.

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	$18.00	
	SUBMIT, CLIST, FDUMP, ABSDUMP, etc.			
885-1061	TMI Cassette to Disk	H8 only	$18.00	
885-1062	Disk VIII	H8/H89 (2 disks)	$25.00	
	MEMTEST, DUP, DUMP, DSM			
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
885-1222	CP/M Adventure Disk		$10.00	35
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 23
885-4002	REMark VOLUME II	$20.00

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

Vectored from 19

The special function key may be used for programming, during the operation of a program or in the command mode. To use any of these keys simply press the desired key. For example, to load a program into memory simply press the **F3** key, type the program name, and the **RETURN** key. To run the program simply press the **F2** key only. The **RETURN** is not necessary when using the **F2**, **F5**, **F11**, or **F12** keys.

NOTE: F0 puts out a **CTRL-U** to clear the current line and may not be assigned another value.

Before assigning new values to any of the special function keys you may want to look at the present values by using the **KEY LIST** command, which prints the above list. Also try turning the 25th line into a status line for the special function keys by using the **KEY ON** command.

When the **KEY ON** command is used the 25th line becomes a status line displaying the first ten keys (F11 and F12 are not displayed). The assigned value of the keys may be up to fifteen (15) characters, however only six (6) characters for each key will be displayed on the 25th line.

The **KEY OFF** erases the 25th line only if the **KEY ON** has been previously used. The entire line is erased including all parts that have been overwritten using the **LOCATE** command.

The last and most important feature of the **KEY** command is the ability to assign new values to the special function keys. With this command we can change the value of the special keys to almost any combination of fifteen ASCII characters.

Example #1: Create a program which uses the function keys to go to different sections of a program.

```
10 CLS:RT$=CHR$(13):' Clear And RT$ Equals The Return Key
20 KEY 1,"Demo 1"+RT$
30 KEY 2,"Demo 2"+RT$
40 KEY 3,"Demo 3"+RT$
50 KEY 4,"END"+RT$
60 KEY LIST
70 LOCATE 4,1:PRINT CHR$(26)"What Is Your Option";
80 LINE INPUT A$
90 IF A$="END" THEN END
100 IF A$="Demo 1" then 1000
110 IF A$="Demo 2" THEN 2000
120 IF A$="Demo 3" THEN 3000
130 GOTO 70
    ....
1000 PRINT "Demo 1":GOTO 70
2000 PRINT "Demo 2":GOTO 70
3000 PRINT "Demo 3":GOTO 70
```


In this example the **KEY LIST** is actually used as part of the menu and since we are not using all of the special function keys the cursor was placed at line five and the rest of the screen erased (CHR$(26) or CTRL-Z erase to the end of the screen). Using the special function keys this way allows the user to input mulitple keys by using only one key.

Example #2: Let's assign the values of the keys that were used on the Z/H19/88/89/90 to the special function keys of the Z-100. This allows a program that was using the these keys to be used with **Z-BASIC** without having to rewrite each input selection.

```
10 CLS:E$=CHR$(27)
20 KEY 1,E$+"S":'      F1   f1
30 KEY 2,E$+"T":'      F2   f2
40 KEY 3,E$+"U":'      F3   f3
50 KEY 4,E$+"V":'      F4   f4
60 KEY 5,E$+"W":'      F5   f5
70 KEY 6,E$+"P":'      F6   BLUE
80 KEY 7,E$+"Q":'      F7   RED
90 KEY 8,E$+"R":'      F8   GRAY (White)
100 KEY 9,E$+"J":'     F9   ERASE
    ....
210 A$=INPUT$(1)
220 IF A$=E$ THEN B$=INPUT$(1):PRINT B$
230 GOTO 210
```

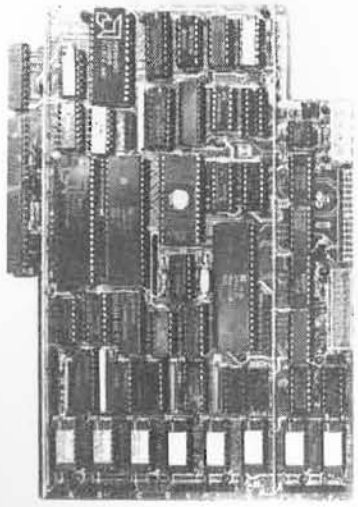
The above routine is intended to allow the function keys of the **Z-100** to act the same as those of the **H/Z-19/88/89/90**. The last three lines (210 to 230) are a simple example of how to input using the special function keys.

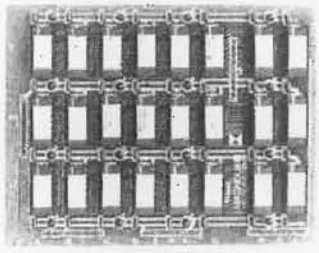
When using the special function key with ESCape codes as shown above the **KEY ON** and **Key LIST** will not show the codes being generated by the special function keys.

NOTE: The hardware of the function keys is always sending the same codes, only **Z-BASIC** interprets them differently to give the programmer some real flexibility.

Try the examples and it will become apparent that the **Z-100**'s **Z-BASIC** has some really powerful commands.

INTRODUCING -- MAGNOLIA's MOST POWERFUL BOARD YET!





The Z89/90 is a good solid computer. But it does have certain limits.

We've had many requests to increase its CPU speed to 4MHz, but nothing met our standards for both field installation and reliability. And who needs anything less.

We increased its RAM to 176K with our 128K board, allowing use of MP/M™. However, without DMA, multi-user operations under MP/M are slow.

Networking '89s together has been a long-term goal of ours. Through the Corvus Constellation multiplexer we provided the first stage -- allowing computers to share Winchester disk storage. But that technique leaves much to be desired -- since each computer independently manages its disk allocation there are significant operational limitations on sharing disk files.

We first announced our impending 'RS422 Network Interface' at the 1982 West Coast Computer Faire. Our design goals included:

- RS422 Network Communication at 800K baud
- On-board Network Control and Data Buffering

As it progressed, we realized that we had designed an 'I/O Board' which was FAR MORE POWERFUL THAN THE '89 ITSELF!

After agonizing delays, the board is now in production as the '77422 Network Controller'. Compare these specifications with the Z89 itself [or the new 'upgrade' CPU boards]:

- 4MHz Z80™ CPU
- 64K RAM, 256K [bank selectable] optional
- 8 K EPROM
- 4 CHANNEL DMA controller
- 2 Serial Ports: RS422 at 500KBaud
  RS232 at up to 19.2 KBaud

We also changed its name from '77321 Network Interface' to '77422 Network Controller'. Our 773xx products must be used in a Z89/90 computer. Although this product CAN be an I/O card in an '89 or 90, it can also be used independently as a stand-alone station on the network:

- permitting a terminal [Z19 or other] as a network station [with local disk storage], or
- permitting a printer to spool data directly from the network.

Network applications software is currently being developed and tested.

Software for an exciting non-network application is included. The workload is split between the Z89/90 CPU and the CPU on the '422 board:

- The Z89's 2MHz CPU continues to run Magnolia CP/M™, handling all physical I/O [console, printer, floppy and Winchester disks]
- The '422 board's 4MHz CPU runs an interface program which 'looks like' CP/M to an application program -- but passes CP/M system calls to the Z89's CPU for execution, allowing a 63K TPA on the '422 board!

With this division, the following Z89 limitations are overcome:

- The application runs on a 4MHz CPU
- The application has 63K of RAM available, independent of BIOS size
- The DMA controller [making a slave I/O processor of the '89 board] frees the application CPU from detailed I/O functions.

This last advantage is especially significant when executing MP/M on the '422 board, using the optional bank-selectable memory.

Hardware-dependent programs [including utilities like 'format'] which perform actual physical I/O, must run on the 'slow' Z89 CPU. A special utility is provided to 'mark' those application programs which can execute more efficiently on the '422 board CPU.

The board is presently available in 64K and 256K versions which mount in an I/O slot [P504/P510] on the '89 CPU board. To limit power dissipation within the '89, no disk drive may be used internally.

64K Network Controller Board
order 77422-064-xx **$695**

256K Network Controller Set
order 77422-256-xx **$1295**

CP/M Operating System by MMS if needed
order CPM-xx add **$100**

MP/M II Operating System
requires 77422 order MPM-xx **$595**

CP/NET™ Network Software
requires 2 or more 77422 boards, one with MPM
order CP/NET-xx **$450**

Use 'xx' to specify preferred media format

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

The MPI 99G Printer

Terry Jensen
Software Developer

Photos by Ed Quinones

Introduction:

Over the past several months, a relatively unknown company to Heath users, has been receiving quite a bit of attention on their new printer. Many HUG members, including staff, got their first look at the MPI 99G dot matrix printer at the National HUG Conference back in August.

Micro Peripherals Inc. (MPI), an American company, is located in Salt Lake City, Utah, and has been producing dot matrix printers for almost seven years.

The MPI printer line consists of three models. The 88G (updated from the 88T model) has a seven wire print head mechanism. The second model, the 99G, has a nine wire print head mechanism for true descenders. The third model, the MP-150, is a high speed dot matrix printer, which will feed standard or narrow width computer paper. All three models were created to print any number of predesigned character *fonts*. (A font is a particular style and size of character type.) Each of the printers have been designed to operate on other computer systems in addition to the Heath/Zenith line of computers.

MPI boasts many features, in addition and superior to, the well known Epson printer line. This review is not intended to present a comparison, however, features of the MPI 99G may require a comparison to other printers for clarification.

General Overview:

The printer is packed in a case that includes the ""QT" sound cover, the single sheet feeder, the 88G and 99G reference manuals. The schematic and parts list is available free of charge for the asking.

The *AP-PAK* package may be shipped separately or together with the printer. It contains the connector cable, software diskette, the Style Writer General Reference Manual, directions for installing Style Writer, and the warranty information.

At first glance, there was too much reading material, however, after scanning through the manuals, the information that was included was necessary, generally easy to follow, and written in a well organized manner. The following sequence seemed the most logical and the easiest to follow:

1) The Style Writer General Reference Manual was the first manual that caught my eye. I wanted to find out more about this "Style Writer". The manual was very well done for the individual not familiar with MPI printers. The text was written in a question/answer format to lead the reader step by step through the operation and commands.

2) Not fully understanding all the commands, I found the "Installing Style Writer" was the next manual that continued a logical sequence of learning the system. This manual was written for the very beginner. It leads the reader step by step through the CP/M *Boot, FORMAT, MOVCPM17, SYSGEN, PIP, and CONFIGUR* routines, to aid the reader in creating a system disk prepared for using Style Writer.

NOTE: *It should be noted at this time that the program "Style Writer" is written for the CP/M operating system. The MPI 99G printer may be used under the HDOS operating system with a normal LP: (line printer) device driver, however, none of the special features of Style Writer can be implemented.*

3) The 88G and 99G Reference Manuals, were the typical reference manual type. The previously mentioned manuals contained most of the necessary information to get the MPI printer ready for use, but not quite all. This became apparent when it was time to remove the top to connect the connector cable inside the case. I was expecting to find the information in the installation manual, however, I soon found the diagram indicating the I/O cable routing in a full page pictorial in section 3, "Interfacing", of the 88G Reference Manual.

While searching through the reference manuals, the paper feeding installation turned up some interesting facts. The MPI 99G will accept paper feeding from the front, back or underneath, including rolled, continuous or separate sheets of paper. The paper can be pressure or tractor fed. The pressure roller can be engaged or disengaged by setting two levers. (I found the levers hard to move at first but after working them a few times found they moved more easily.)

The single sheet feeder which connects very easily to the printer, I found is great for loading continuous pin hole paper as well as single sheets. The feeder acts as a guide, and with it, I may never have to improvise again to load paper. I did find, however, that the front of the printer scratches easily by the feeder whenever the feeder is moved. When removing or mounting the feeder, care must be taken to lift the hooks over the lip of the front panel for minimal face damage. ☞

The long life, ribbon cartridge is located on the back of the printer. The ribbon can be replaced easily without removing the top of the printer. The ribbon is a half inch wide and when installed is inclined at approximately one degree for utilizing the entire ribbon surface for printing. This may look incorrect when installing a ribbon for the first time.

The remaining major sections of the Reference Manual contained information on specific features of programming and graphics, which were briefly discussed in the Style Writer and Installation manuals. Also the inspection and testing information was contained in an appendix of the 88G manual.

4) The Schematic and Parts List revealed an interesting fact, which was apparent after removing the top of the 99G: the printer design is clean, uncluttered, contains a minimal number of parts and should be easy to maintain.

Operation of the Printer

After unpacking the 99G, I tried the printer just to see how it would work as a normal printer. After configuring the LST: device on a CP/M system disk for PORT 340Q and 1200 baud, the printer worked just great.

The 2K buffer of the 99G, contributes to the modest 100 characters per second output. This is 20 characters per second faster than the EPSON MX-80.

The printer pauses for brief periods of time when printing special fonts or character formats through the Style Writer. This is due to Style Writer sending approximately 12 times as many characters to the printer. Each character of a special font is made of many smaller "characters".

I found the noise level to be about the same as the Heath H-25 printer without the squeal of the H-25 high speed head and the annoying "buzzing" of the H14.

The paper advance switch on the front of the printer will scroll the paper one line if pressed once. If held longer, the paper will be advanced 10 lines per feed until the top of the form is reached. There is no means for backing up if the paper is advanced too far.

The Setup Procedure for Style Writer

After scanning through most of the manuals and following through the instructions, I created a system disk and prepared for using Style Writer. The system disk must be configured for four (4) K less than the computer memory size. My H8 has 64K of memory, so I configured CP/M for 60K bytes. (For 48K of memory, CP/M is to be configured for 44K.)

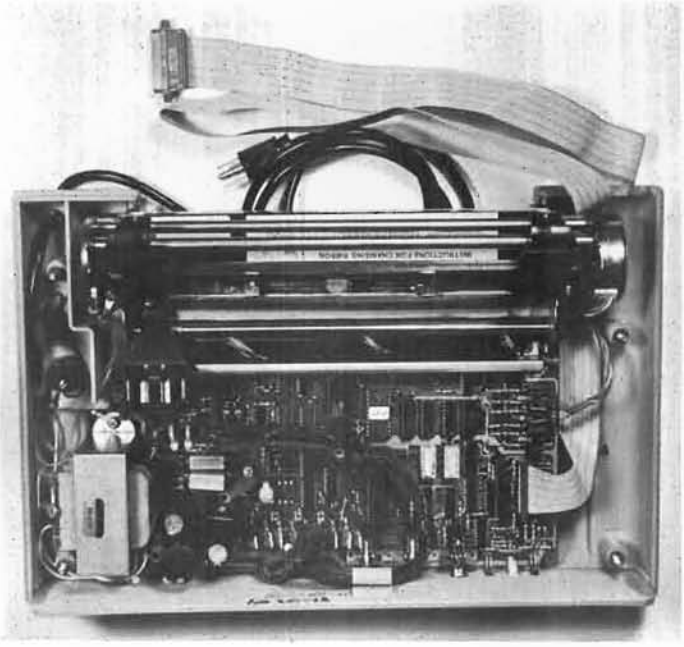
What a disappointment to find that Style Writer would not run on the 8080 processor of the H8! The printer worked fine as a normal printer, but when an attempt was made to load STYLE64, the whole system hung up.

NOTE: *Style Writer requires the Z80 processor board on the H8.*

After installing the Z80 processor board in the H8, I was pleased to see the output of some of the 99G features, which up to that time, I had only read about in the manuals.

Style Writer

Style Writer is a program that is loaded from CP/M by the program STYLE64, (STYLE48 for 48K of memory). The four (4) K of memory "reserved" above CP/M is where Style Writer is loaded and stored. Style Writer actually intercepts all output from the BIOS to the printer, and translates any codes it understands, while the remaining text is sent unchanged. That is the reason why the fonts are *software* selectable; the *coding* is done in memory before it is sent to the printer.



The fonts, which are released on the disk with the AP-PAK, are shown in Figure 1. The example fonts are the result of sending the characters "ABCDEFabcdef" through Style Writer to the printer. These fonts are ready to be used by Style Writer.

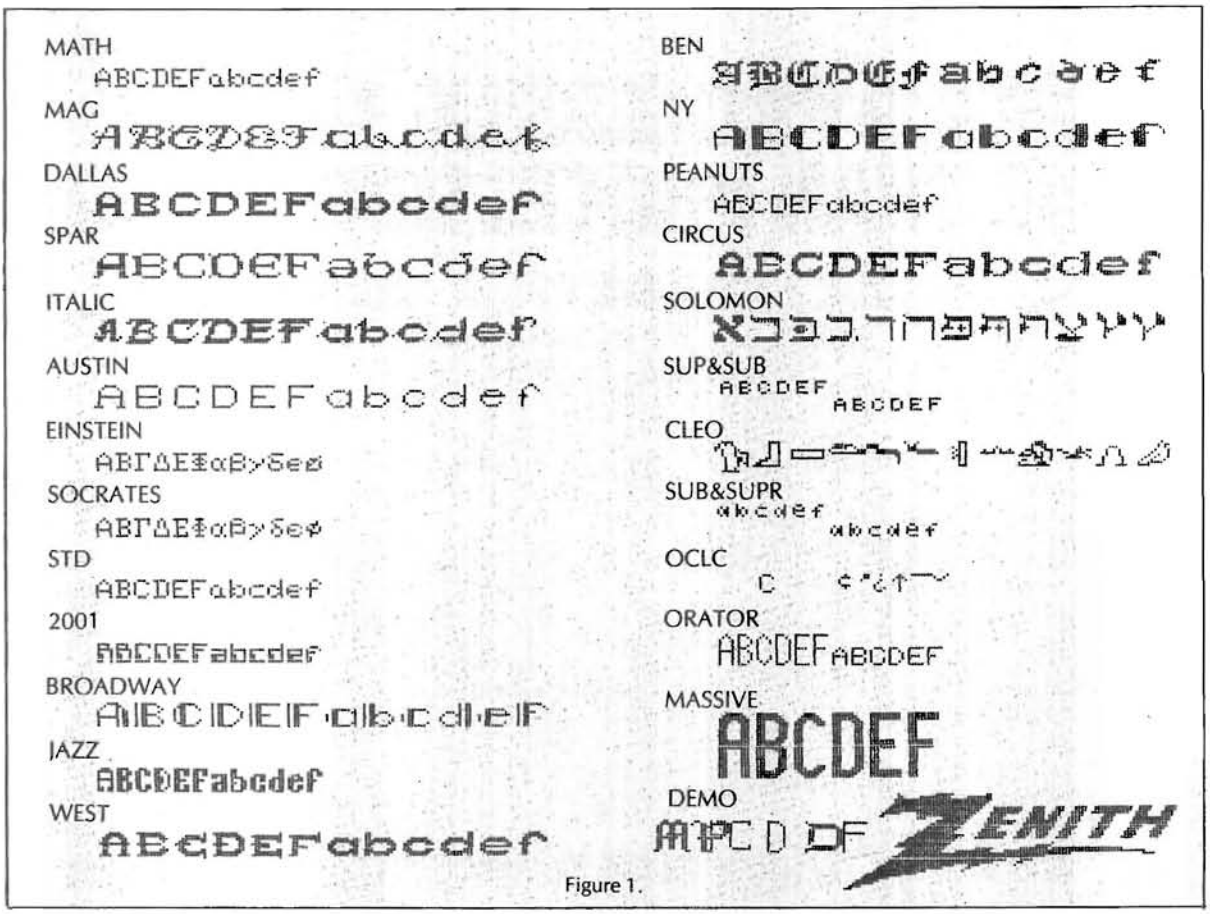


Figure 1.

In addition to the fonts, the MPI 99G can be commanded to print in different formats. The H19 graphics and reverse characters are two of the features that should raise interest among Heath users. In addition, characters can be printed double high and double wide. All of the features are detailed in the manuals.

Application of Style Writer

Style Writer can be used with BASIC or with a word processor. All that is required to enter a particular format, is to send the escape character and an appropriate letter. The remaining text is sent as output to the printer.

I had no difficulty using Style Writer with MBASIC (BASIC-80). The most difficult part was remembering to load Style Writer into memory before running MBASIC.

I did run into a real hassle trying to use Style Writer with WordStar. As it turns out, the problem was not the fault of Style Writer. The Style Writer Manual made mention of using WordStar, but did not give any hints or details for setting up WordStar. I am making special note of this for anyone else who may encounter the same frustration.

WordStar must be configured for a "Teletype Like Printer" using the Install program of WordStar. This ensures that the only characters sent to the printer are those required for the MPI 99G.

Once this was done, I had no difficulty in using WordStar to output the special fonts and characters to the 99G.

Two additional comments relating to the use of WordStar with Style Writer.

1) When copying a file created by WordStar with PIP, from CP/M to the printer, the last character of each word in the text is replaced with a space. WordStar sets the eighth bit on the last character of each word in the text, which causes Style Writer to insert the space. A solution is to print the file (in WordStar) to a disk file and then PIP the new disk file to the printer.

2) The WordStar system disk does not have enough space to hold a font and a text file. Any fonts to be used in a text file must be copied to a documentation disk. WordStar must then be invoked from the disk drive containing the documentation disk.

One Last Note

On the Style Writer disk is a program called CALLIGRAPHER, which allows the user to create any pattern or font. Any pattern chosen should be laid out on a grid and then transferred into a new font with CALLIGRAPHER. The newly created font or pattern is called by the same means as the normal fonts. (CALLIGRAPHER is menu driven and very user friendly.)

Conclusion

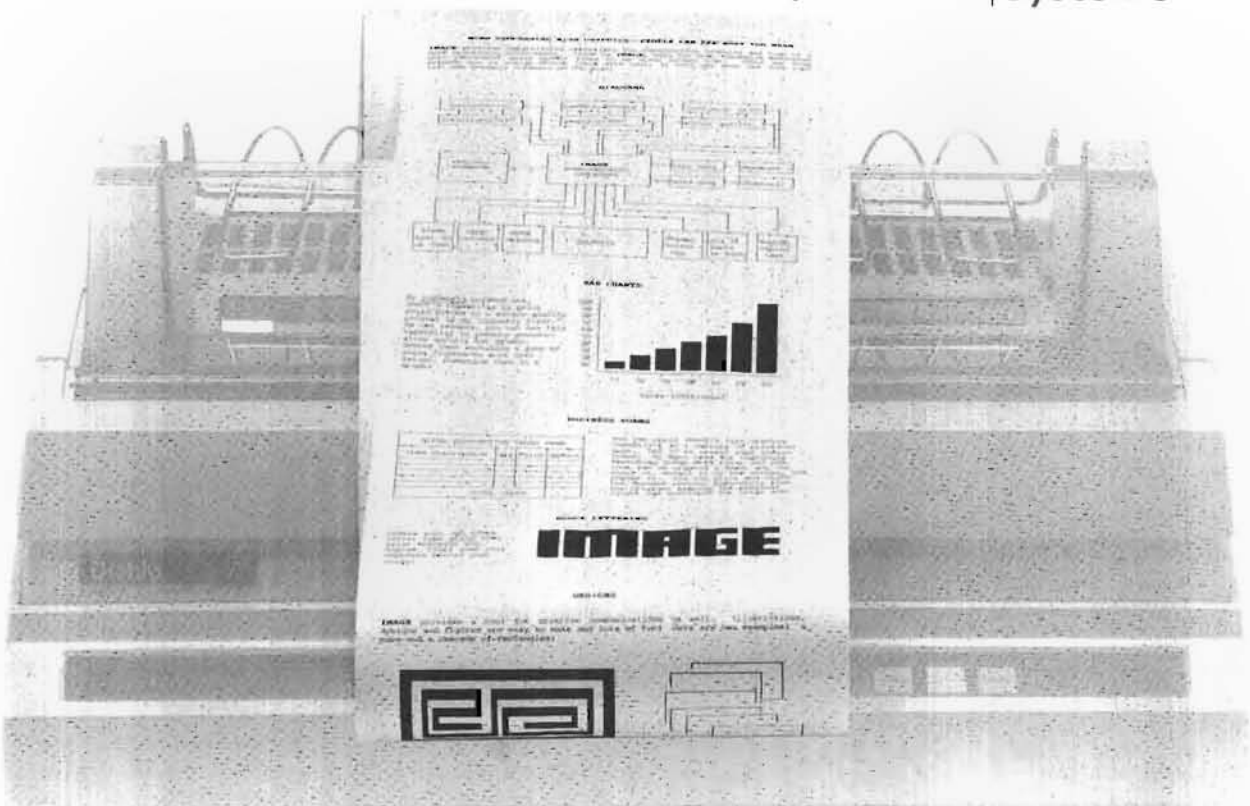
One of the points of the MPI 99G printer which I was most impressed, has nothing to do with the functional operation of the printer or Style Writer. It is the first thing that most everyone says when they walk into my office. "Wow, neat printer!"

The MPI 99G dot matrix printer has a suggested retail price of $695.00. A Style Writer demonstration disk, part #A-501 Z-89, is available at no additional charge from:

Micro Peripherals Inc.
4426 South Century Drive
Salt Lake City, Utah 84107

WORD PROCESSING IS NOW MORE THAN PROCESSING WORDS.

ANNOUNCING **IMAGE**™ FOR ZENITH data systems



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

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

TOP MARKS FROM INFOWORLD

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

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

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

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

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

IMAGE is a trademark of MicroArt Corporation.

InfoWorld
Software Report Card

Image

	Poor	*Fair*	*Good*	*Excellent*
Performance	☐	☐	☐	■
Documentation	☐	☐	☐	■
Ease of Use	☐	☐	☐	■
Error Handling	☐	☐	☐	■

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

EXTRAORDINARY VALUE...

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

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

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

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

Mastercard, Visa and COD orders accepted.

Keyboard Polling from BASIC

Roy Reichert
29 Blazier Road
Warren, NJ 07060

In recent discussions with other programmers, I have been asked about problems encountered when attempting to use the INP(232) instruction in Microsoft BASIC to set up a keyboard polling routine. The problems encounterd are easily overcome, once the programmer has gained some insight into what is happening and what facilities are available to correct the difficulty.

First, it might be helpful to define just what it is we are attempting to do, and then demonstrate the approach commonly used (where the problems are encountered). We will then show the means of solving the problems and achieving an efficient, useful polling scheme.

"Keyboard Polling" means that the program is written so that it continually loops and senses the output of the keyboard port. Unlike the usual INPUT statement which causes the program to halt and wait for input from the keyboard, polling simply "looks" at the keyboard port and accepts whatever character happens to be there at the time. The program continues to loop, performing other functions as well. If no characters are entered at the keyboard, the polling routine will "see" the same character each time around. This will be whatever character was entered at the last time the keyboard was touched.

It is important to understand that we are looking at the keyboard "port"; not the input buffer in HDOS! The port is never "flushed" when it is read by the program, as is the input buffer. Whatever character is entered into the keyboard port, remains there until it is replaced by striking a DIFFERENT key.

Polling is very useful in many applications. One common use is in games where keys are used to control some action, with multiple strikes of a given key interpreted as intensifying the particular action of that key. Many versions of BASIC have commands such as INKEY, which are polling input commands. Such commands are not available in the Benton Harbor or Microsoft versions of BASIC as used on Heath Computers.

Microsoft BASIC for the Heath system has the INP(#) command which reads any input port, where # is the number of the port. Using this command, we can construct our own "INKEY" equivalent. The keyboard of your Heath computer is port #232, so the command INP(232) will read the keyboard port. Using this command, it is relatively easy to code a routine to poll the keyboard. The commonly used method is as follows:

```
100 ' POLLING DEMONSTRATION
110 CLEAR
120 GOSUB 500
130 IF X<>0 THEN PRINT CHR$(X);
140 PRINT "*";
150 GOTO 120
  -
  -
  -
500 ' KEYBOARD POLLING
510 IF P=0 THEN P=INP(232)
520 X=0
530 C=INP(232)
540 IF P=C GOTO 580
550 P=C
560 X=C
570 A$=INPUT$(1)
580 RETURN
```

Let's examine the code and see what happens here. The polling is done in the subroutine at line 500. Variable "P" is the "previous" character found at the port. Variable "C" is the "current" character at the port. The first time through the routine P is undefined, so it is zero. Thus, it is initialized in line 510 to whatever is in the keyboard port at the time. Variable "X" is the output variable of the subroutine.

Line 530 reads the port. If the character found is different from the previous character (P), it means another key has been pressed. Line 550 sets P to this new character, and line 560 stores the new character into the output variable (X). Each new character is treated by HDOS in the usual manner, meaning that the character is also placed into the keyboard input-buffer. If we were to leave it there, the buffer would accumulate new input characters up to the limit of 100 characters. At that point, no further keyboard input would be accepted and the program function would cease. Line 570 is used to flush this new character out of the buffer, thus preventing this problem. It is important to remember that if there in no NEW character in the buffer, line 570 must not be executed since it will halt the program to wait for an input character. Thus, line 540 jumps around this if there is no new character.

The main program test if there is an output character in "X" and if so, prints the character. The print statement at line 140 simply prints a continuous line of "*" to show that the program is really looping and polling the keyboard without stopping for input. Run the above code on your system and be sure you understand what is happening. You will be able to test it and discover the first obvious problem with the routine.

The first problem is simply that the routine works as long as each key pressed is different from the previous one. If multiple strikes are made on the same key, the routine will ignore all but the first.....ALMOST! AHA!!! Now for the second problem!.....

Multiple strikes on the same key will continue to dump characters into the input buffer which WILL NOT BE FLUSHED OUT! Eventually the program will come to a point of failure due to a full buffer. Many applications may be able to tolerate these problems. In such cases, the above routine is usable as it is, with no need to refine it further. However, where these problems must be solved, we can do so easily if we know a little more about HDOS.

HDOS makes use of a highly useful feature called the "type-ahead buffer". This is simply a buffer for storing keyboard input characters as they are entered. Thus, you can type in commands to your system prior to any need for them by the program. They will be stored, and when an input statement is encountered the input will be taken from the buffer in the same order in which you entered them (First-In, First-Out [FIFO]). Note that this feature is NOT similarly available under CP/M.

The type-ahead buffer is a circular buffer; i.e. as characters are

entered and the buffer fills up, subsequent characters are placed at the start of the buffer again as in a continuous loop (assuming that the program has been pulling characters out of this part of the buffer). In order to know where the next input character is coming from, HDOS uses pointers to track the beginning and end of the current contents of the buffer. It is important to understand this since it is the key to the solution of the polling problem.

As characters are flushed (or read) out of the buffer, the pointer to the "start" of the buffer contents is incremented, while the pointer to the "end" of the contents is unaffected. As characters are typed, and thus placed into the buffer, the pointer to the "start" of the buffer contents is unaffected, while the pointer to the "end" of the contents is incremented to include the new character(s). We will use this feature to solve the polling problem. Refer to the following code.....

```
100 ' POLLING DEMONSTRATION FOR HDOS 2.0
110 CLEAR
120 GOSUB 500
130 IF A$<>THEN PRINT A$;
140 PRINT "*";
150 GOTO 120
-
-
-
500 ' KEYBOARD POLLING
510 M=64880!
520 IF PK=0 THEN PK=PEEK(M)
530 A$=
540 K=PEEK(M)
550 IF K=PK GOTO 580
560 PK=K
570 A$=INPUT$(1)
580 RETURN
```

The location of the end-of-queue pointer is at 64880 (decimal) in HDOS 2.0. Line 510 defines M as this memory location. (This should be location 64883 in HDOS 1.6.) On the first pass through the subroutine, the value of PK is undefined, or zero, so it is defined as the current value of the pointer by using the PEEK command in line 520. The variable PK is used as the "previous" value of the pointer. The variable K is the "current" value of the pointer and is defined in line 540.

We now compare the previous and current values of the pointer in line 550. If they are different, we know that a key has been struck and that a character has been placed in the buffer. Thus, at line 570 we read the character from the buffer using a normal INPUT$(1) statement. Note that the INP(232) statement is not used in this routine. The INPUT$(1) statement gets the character for use within the program and it also flushes the buffer. (This has no effect on the pointer value, so our new value of PK is still valid!)

If, in line 550, we find that P and PK are the same, we know that no key has been struck, so we jump to line 580. We can see that the routine uses the pointer to detect only the striking of a key and does not care what the actual character may be. Also, we only flush the buffer when we know that there is a character present.

We are not finished with the development yet, however. We must consider the effect of multiple keystrokes still further. If the main

Vectored to 35 ☞

WHAT'S IN A NAME?

Pat Swayne
Software Engineer

Way back in REMark issue #9, we presented the source code for a program that allowed you to change the label on an HDOS disk. In a recent letter, a HUG member told me that he could not get the program to work, because it needed some XTEXT files that were not present in the magazine. I looked in every nook and cranny here at HUG headquarters for those files and couldn't find them, so I decided to write a new label changer program for that HUG member. Besides, the old one would only work on hard sector disks.

I had learned previously that there is no reason why a disk's label must be limited to 60 characters, and that by using a patch utility such as DUMP I could make labels much longer (up to the end of the label sector). I also had learned that disk labels can contain multiple lines and graphics, and made such a label on the SEA BATTLE disk (885-1103). So I decided to give my new label program these capabilities.

The result is the assembly listing LABEL.ASM accompanying this article. Type in the source code with your editor and assemble it to get LABEL.ABS. An example of this program's use would be

```
>LABEL SY1:
```

In this case LABEL.ABS is on SY0: and we are going to work with the label on SY1:. The disk that you are examining the label on does not have to be mounted. LABEL will first print the existing label on the disk and prompt you for a new one. If you do not want to change the label, just hit RETURN before typing anything else. If you type anything else as the first character, it becomes part of the new label, and RETURNs can also be put in the label. When you are finished typing the new label, type CTRL-D to write it to the disk and return to HDOS. Do not type a RETURN at the end of the label unless you want two of them printed by HDOS when you mount the disk. If you decide you do not want to change the label after you have typed a new one in, you can exit to HDOS by typing two CTRL-Z's.

Here is a sample label that you can make with this program.

```
H D O S  SYSTEM DISK
```

To make this label, you would type in the following when you run LABEL.

```
<ESC>F<cr>
c f sac fac fac<cr>
vat ' ' ' ' eac<cr>
d e uad ead ead<ESC>G SYSTEM DISK<CTRL-D>
```

In this example, <ESC> means to type the escape key, <cr> means to type RETURN, and <CTRL-D> means to type Control-D. You will not see what I have shown here as you type, but you will see the graphic characters instead.

Your new label will be read and displayed correctly by HDOS when you mount, dismount, or boot the disk, and also by CCAT from HUG disk 885-1090, but there are other programs that will only display the first 60 characters of a long label. Among them are INIT.ABS and the LAB command in Jim Teixeira's SYSMOD (otherwise a fine product).

For those of you who are assembly hackers, this program is an example of reading and writing on a disk directly through the disk device driver rather than through HDOS (with SCALL). At the label READSEC, the program tests which version of HDOS we are using, because versions before 2.0 did not really have disk device drivers, and all direct access to the disk can be done through D.SYDD, the "System Device Driver". If the version is 2.0, you can't use D.SYDD, because you will not necessarily be using the system device. But you can find the address of the device you are using by loading it (with SCALL .LOADD, even though it may already be loaded). That gets you the address of the device table in the HL register pair, and when you add DEV.JMP (3) to it, you get the address of a JMP into the device. There are other ways of finding the address of the device table, but an added benefit of this method is that you would have to load the driver anyway if it was not already loaded, and this takes care of it.

After the LABEL program gets the address of the device driver, it tells it which unit to use (the 0 in "SY0" is the unit number) by placing the binary (not ASCII) value of the number in AIO.UNI. Commands to the disk device driver, such as READ or WRITE, are made via a code in the A register. These are explained in the article "HDOS Device Driver Programmer's Guide" in REMark #20, pages 10 and 11, and their values are found in DDDVD.ACM, included with HDOS 2.0. When you read or write, you must supply the sector number in the HL register pair. You do not need to supply the track because in HDOS the disk is assumed to be one continuous track of sectors (a 5.25-inch single density single sided disk would have sectors 0-399), and the device driver will figure out the physical track and sector from the sector number given.

There are two commands for reading from a disk, DC.REA (Read) and DC.RER (Read Regardless). With 5.25 inch hard sector disks, DC.REA will not work until the device driver is supplied the volume number of the disk, but DC.RER disregards this "volume protection". So the LABEL program uses DC.RER to read the label sector, which contains the volume number of the disk as well as the label. There is no Write Regardless command, so the device driver must give the volume number before write operations can be done on hard sector disks (other disks do not have volume protection). This is done with the DC.MOU (Mount) command, with the volume number in the L register. The first track of a disk is always volume 0, so LABEL sets up the driver with volume no. 0 and writes the new label to the disk. Then it issues the Mount command again to restore the original volume number.

Using this program as a starting point, it would be possible to

write a disk patch utility that would work on any HDOS supported disk.

```
*       LABEL -- READ OR CHANGE DISK LABEL
*
*       WITH THIS PROGRAM YOU CAN READ OR CHANGE THE
*       LABEL ON ANY HDOS DISK
*
*       TO USE THIS PROGRAM, ENTER
*
*       >LABEL DEV:
*
*       THE EXISTING LABEL ON THE DISK WILL BE
*       PRINTED, AND YOU WILL BE PROMPTED FOR A NEW
*       LABEL.  ENTER RETURN IF YOU DO NOT WISH TO
*       MAKE A NEW LABEL.
*
*       DEV: IS ANY HDOS DISK DEVICE (SY0:, DK1:, ETC.)
*
*       BY P. SWAYNE, HUG  25-AUG-82
*       BASED ON M. DUTTWEILER'S PROGRAM IN REMARK #9

*       DEFINITIONS

.EXIT   EQU     0
.SCIN   EQU     1
.SCOUT  EQU     2
.VERS   EQU     11Q
.LOADD  EQU     62Q
$TYPTX  EQU     31136A
DC.WRI  EQU     1           WRITE TO DISK
DC.RER  EQU     2           READ FROM DISK
DC.ABT  EQU     7           ABORT DEVICE
DC.MOU  EQU     8           MOUNT DEVICE
DEV.JMP EQU     3           JUMP TO DEVICE DRIVER
DFF.LAB EQU     9           LABEL SECTOR
D.SYDD  EQU     40130A      SYSTEM DEVICE DRIVER (OLD HDOS)
AIO.UNI EQU     41061A      UNIT NO.

        ORG     42200A

*       GET DEVICE TO USE

START   LXI     H,0
        DAD     SP          FIND STACK
        MOV     A,L
        CPI     200Q        HAS IT MOVED?
        JZ      READSEC     NO ARGUMENT, USE SY0:
        LXI     D,DEV       PUT DEVICE HERE
        MVI     B,3         3 CHARACTERS
GETARG  MOV     A,M         GET A CHARACTER
        ORA     A           END OF ENTRY
        JZ      READSEC     IF SO, GO ON
        CPI     ' '         SPACE
        INX     H           INCREMENT POINTER
        JZ      GETARG      SKIP SPACES
        STAX    D           STORE CHARACTER
        INX     D           INCREMENT POINTER
        DCR     B           DECREMENT CHARACTER COUNTER
        JNZ     GETARG      GET MORE CHARACTERS

*       READ LABEL SECTOR

READSEC SCALL   .VERS       CHECK VERSION
        JC      OLDVER      OLD ONE
        CPI     20H         2.0?
        JC      OLDVER
        LXI     H,DEV       POINT TO DEVICE
        SCALL   .LOADD      LOAD THE DEVICE (TO GET ADDR)
        JC      CANTRD      CAN'T READ LABEL
        LXI     D,DEV.JMP
        DAD     D           FIND JUMP INTO DRIVER
        SHLD    DEVJMP      SET IT UP
OLDVER  LDA     DEV+2       GET DRIVE UNIT NUMBER
        SUI     30H         REMOVE ASCII BIAS
        STA     AIO.UNI     SET IT
        LXI     H,0
        MVI     A,DC.ABT
        CALL    DISK        ABORT DEVICE
        JC      CANTRD
        LXI     H,DFF.LAB   GET LABEL SECTOR
        LXI     D,BUFFER    PUT IT HERE
        LXI     B,256       READ ONE SECTOR
        MVI     A,DC.RER
        CALL    DISK        READ LABEL SECTOR
        JC      CANTRD
```

```
*       PRINT CURRENT LABEL

        CALL    $TYPTX
        DB      'Current label is:',212Q
        LXI     H,BUFFER+17             LABEL IS HERE
LABLP   MOV     A,M             GET A CHARACTER
        ORA     A               END OF LABEL?
        JZ      GETLAB          YES
        SCALL   .SCOUT          PRINT CHARACTER
        INX     H               INCREMENT POINTER
        JMP     LABLP

*       GET NEW LABEL

GETLAB  CALL    $TYPTX
        DB      12Q,'Enter new label (or RETURN to exit).'
        DB      12Q,'Type Control-D when done:',212Q
        LXI     H,BUFFER+17     PUT NEW LABEL HERE
        MVI     B,230           ALLOW 230 CHARACTER LABEL
        SCALL   .SCIN           GET A CHARACTER
        JC      *-2
        CPI     12Q             RETURN?
        JNZ     NOTRET          NO
        XRA     A
        SCALL   .EXIT           ELSE EXIT
GETLAB1 SCALL   .SCIN           GET A CHARACTER
        JC      *-2
NOTRET  MOV     M,A             STORE CHARACTER
        INX     H               INCREMENT POINTER
        DCR     B               DECREMENT COUNTER
        JZ      LABOVL          LABEL TOO LONG
        CPI     4               CONTROL-D?
        JNZ     GETLAB1         IF NOT, CONTINUE
        DCX     H               BACK UP TO LAST CHARACTER
        MVI     M,0             TERMINATE LABEL WITH 0
        MVI     A,12Q
        SCALL   .SCOUT          ISSUE NEW LINE

*       WRITE NEW LABEL SECTOR

        MVI     L,0
        MVI     A,DC.MOU
        CALL    DISK            MAKE VOLUME NO. 0
        LXI     H,DFF.LAB       GET LABEL SECTOR NO.
        LXI     D,BUFFER        SECTOR IS HERE
        LXI     B,256           WRITE ONE SECTOR
        MVI     A,DC.WRI
        CALL    DISK            WRITE THE SECTOR
        PUSH    PSW             SAVE ERROR CODE
        LDA     BUFFER          GET ORIGINAL VOLUME NO.
        MOV     L,A
        MVI     A,DC.MOU
        CALL    DISK            RESTORE IT
        POP     PSW
        JNC     EXIT            GOOD WRITE
        CALL    $TYPTX
        DB      12Q,'Can''t write new label.',212Q
EXIT    XRA     A
        SCALL   .EXIT

*       CALL DISK DEVICE DRIVER

DISK    JMP     D.SYDD          GO TO DEVICE DRIVER
DEVJMP  EQU     *-2             ALTER FOR HDOS 2.0

*       CAN'T READ LABEL

CANTRD  CALL    $TYPTX
        DB      12Q,'Can''t read disk label.',212Q
        XRA     A
        SCALL   .EXIT

*       TOO MANY CHARACTERS

LABOVL  CALL    $TYPTX
        DB      12Q,'Too many characters in
        XRA     A                       label (230 max).',212Q
        SCALL   .EXIT

*       STORAGE

DEV     DB      'SY0:',0        DEVICE NAME
BUFFER  DS      256             LABEL BUFFER

        END     START
```

Single Page Writer for the MX80

Charles 'Karl' Romer
P.O. Box 8796 CRB
Tuscon, AZ 8538

Do you like to use your computer to write letters? And do you have a friction/traction type printer like the Epson MX-80 FT? If so, you might be interested in a little MBASIC CP/M program that will make it easy to use single sheets and a pre-printed letterhead first sheet.

The program presented is strictly 'bare bones', no comments no frills. However, it is packed with all kinds of nice goodies and features! Since it is an MBASIC program many of the parameters can be readily changed to suit one's own ideas or desires. Some of the features are as follows:

1. Following the letterhead page, each page is numbered at the top center of the page.

2. You can review your letter on the CRT display page by page in its entirety prior to printout.

3. During the review display you can detect if there is an error, hanging line, etc. This can then be corrected by re- editing.

4. Most significant, any single page of a multi-page letter can be printed without printing the entire letter.

5. During printing the display remains clear until specific instructions appear on the CRT for inserting sheets into the printer. Note: These instructions appear on the screen BEFORE the printer has finished printing the previous page. Just let the printer finish and stop before following the instructions!

Since there are no comments within the program a line by line description follows:

Line 10. Assigns string variable CL$ to CLEAR SCREEN. Also, identifies program file as 'SPW.BAS'.

Line 20. Clears screen and displays program title.

Line 30. Choice for either display or printout.

Line 40. The POKE 3,105 sets all LPRINT statements to appear on the CRT screen. This is changed only when a printout is selected. This line also insures that a 'D' or 'P' has been selected.

Line 50. Opens the desired file for input.

Line 60. Permits printout from page 1 (letterhead) and on or from any other page and on. Note: The printer STOPS at the END of each page so that you can print just the one page or any number of pages following.

Line 70. Selects display or printout as per line 30. Also, if you had selected to start printout at page one this is when the previous POKE statement is changed to set all LPRINT statements to go to the printer.

Line 80. The stand-by notice appears on the screen ONLY when a print-out is requested for page 2, 3 etc.

Line 90. The FIRST instruction if you had selected printout at page one.

Line 100. This statement stops the printer until a carriage RETURN (CR) is pressed. This permits you to insert a new sheet and line it up in the printer.

Lines 110 thru 130. These will display or printout ONLY the first page of your letter. Note: Page length is set for 52 lines. If your letterhead has more than 3 lines or takes up more head space the 52 lines may be too many. Change accordingly! Remember, this is ONLY FOR THE FIRST PAGE! Also, the left margin is set by the value in TAB(n). To change the left margin change this value. If you make a change here be sure and do the same at line 190. Sorry, there is no provision for automatic right margin. If necessary, you can set the right margin by adding spaces between words in each line when in the editor.

Line 140. Resets the 'N' counter for the next page and tests for display or printout starting at page 2.

Lines 150 and 160. Instructions for insertion of all pages in the printer from page 2 and on - when a printout is selected.

Line 170. Prints the appropriate page number at the top of each page starting at page 2 and provides for 2 spaces before beginning the text. This can be changed to provide more space by adding ':LPRINT' for each additional space desired. However, if you do this be sure and REDUCE the 'N' value in line 200 accordingly.

Lines 180 thru 200. These will continue the display or printout of all the following pages of the letter from page 2 and on.

Line 210. This is the GOSUB line which decides when to make the LPRINT statements go to the printer.

Line 220. Returns program to follow the GOSUB statement.

Line 230. This closes the file and ends the program. Note: The POKE 3,169 statement may seem superfluous and for the most part it is. However, I found I could not do an LLIST printout right after I ran this program on display 'D'.

In conclusion I have a couple of tips that may be helpful. I found that I couldn't use single sheets on the EPSON due to the 'Out of Paper' feature. To overcome this feature I first used a small piece of stiff paper and taped it under the paper guide plate on the left side of the printer. This is where you insert paper into the printer. Later on, when I had the cover off the printer for some other purpose I tied a thin string to the paper- out switch. WARNING: With the paper-out switch disabled be very careful when printing too close to the bottom of a page - it doesn't do the platen much good if the printhead inks it! For various reasons I prefer to disable the 'Out of Paper' feature permanently!

The other tip has to do with printing close to the top of a sheet and particularly when using a better grade of bond paper. To set the paper in the printer so the page numbers are printed close to the top of the sheet I found that I had to recess the paper, in the Epson, JUST below the top of the printhead ribbon guide. This meant that I had to disengage the bail to allow for paper clearance. Unfortunately there isn't enough clearance as the heavier bond paper tends to come out from under the platen and hit the bail. What a mess! I tried holding the right edge of the paper into the platen with a narrow strip of stiff paper but it usually

was a touch and go situation. Every so often the printhead would hit my strip of paper before I could snatch it away in time. I found a much simpler way to solve this problem. This applies to any weight paper. Here is what to do. Before you insert the paper in the printer take hold of it at the bottom edge and lay it FACE UP about half way over the straight edge of a desk or table. Then with your other hand lightly press down on the upper half of the sheet, pull the paper down and over the edge. This should put a curve to the upper portion of the paper allowing it to clear the bail when in the printer. If at your first attempt the paper does not show a slight curve try it a time or two until it does. Remember, when printing starts, to re-engage the bail just as soon as you can. I find that this works just great!

Hope you have as much fun with this program as I have had. Good luck and happy letter writing!

```
10 CL$=CHR$(27)+"E" 'SPW.BAS FOR CP/M 2.203 (20 Oct 82)
20 PRINT CL$ TAB(12)"*** S I N G L E   P A G E   W R I T E R ***":PRINT
30 INPUT"PRESS 'D' FOR DISPLAY OR 'P' FOR PRINTOUT - THEN <CR>....";M$
40 PRINT:POKE 3,105:PG=1:IF M$="D"THEN 50 ELSE IF M$<>"P"THEN 30
50 INPUT"TYPE 'FNAME' OF LETTER (EX: BILL5 or C:BILL5)....";L$:OPEN"I",1,L$
60 PRINT:IF M$="P"THEN INPUT"SELECT PRINTOUT AT PAGE # - 1,2,3 etc.";X
70 PRINT:IF M$="D"THEN 110 ELSE GOSUB 210
80 IF X>1 THEN PRINT CL$"PLEASE STAND-BY....":GOTO 110
90 PRINT"INSERT LETTERHEAD SHEET - TURN ON PRINTER - THEN HIT <CR>...."
100 LINE INPUT Z9$
110 IF EOF(1)THEN 230 ELSE LINE INPUT #1,X$:N=N+1:IF X>1 THEN 130
120 LPRINT TAB(6)X$
130 IF N=52 THEN 140 ELSE 110
140 N=0:PG=PG+1:IF M$="D"THEN 170 ELSE GOSUB 210:IF X<=PG THEN 150 ELSE 180
150 PRINT CL$:FOR I=1 TO 9:PRINT:NEXT:PRINT"INSERT FOR PAGE #"PG"- ";
160 PRINT"TURN ON PRINTER - THEN HIT <CR>....":LINE INPUT Z9$:PRINT CL$
170 LPRINT TAB(38)"-"PG"-":LPRINT:LPRINT
180 IF EOF(1)THEN 230 ELSE LINE INPUT #1,X$:N=N+1:IF X>PG THEN 200
190 LPRINT TAB(6)X$
200 IF N=56 THEN 140 ELSE 180
210 IF X=PG THEN POKE 3,169
220 RETURN
230 CLOSE:POKE 3,169:END
```

Vectored from 30

program is long and the looping rate is slow, it is possible to enter many keystrokes between each successive pass through the subroutine. These will all go into the buffer and the pointer will increment many times. We must set up to handle this situation.

In such a case, the difference between K and PK will be a count of the number of keystrokes made. By placing this number into the INPUT$(X) statement, we will flush all of the characters out, into the variable A$ as before. If desired, the main program can examine this string to see if there are different characters present, or whatever you wish to do with the string.

One final task...The input buffer is circular. That is, when the buffer pointer gets to the end of the buffer it resets to the beginning again. The range of values which this pointer can have are from 126 to 226. We must allow for this reset in calculating the keystroke count. The listing which follows is the final polling routine. All of the above problems have been accounted for in this routine.

```
100 ' POLLING DEMONSTRATION FOR HDOS 2.0
110 CLEAR
120 GOSUB 500
130 IF A$<>THEN PRINT A$;
-
200 ' DELAY TO SIMULATE A LONG-LOOP PROGRAM
210 FOR J=1 TO 300:NEXT J
-
300 PRINT "*";
310 GOTO 120
-
-
500 ' KEYBOARD POLLING
510 M=64880!
520 IF PK=0 THEN PK=PEEK(M)
530 A$=
540 K=PEEK(M)
550 IF K=PK GOTO 600
560 N=K-PK
570 IF N<0 THEN N=(226-PK)+(K-126)+1
580 PK=K
590 A$=INPUT$(N)
600 RETURN
```

This routine is a simple, yet efficient polling scheme which will be quite useful in many applications. It should be noted that the use of HDOS in the Stand-Alone mode may change the address of the buffer pointer, so the value of M will have to be changed accordingly.

OOOooopppppssss!!!!

In "A 4 MHz Mod" REMark No. 34, at the bottom of page 25, last column, U1 pin 7 to U1 pin 8 *should read* U1 pin 7 to U2 pin 8.

Sorry for any inconvenience this may have caused.

Christmas Graphics

An On-going Christmas Display Sure To "WOW" Family And Friends!!!

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

The XMASPROG.BAS is a graphics display program for the H19, H89 Terminal. It requires almost 48K of RAM under MBASIC. The program will turn off the terminal keyboard, but turns it back on at the end when the two stars are flashing. This is written to be a non-stop display type program. There aren't any remark statements because of the length of the program, however, it is fairly simple, consisting mostly of GOSUBS to print the desired picture. The graphics themselves were first constructed using ED-A-SKETCH from the Software Toolworks.

```
470 PRINTP$Y$"$"CHR$(64)" "Q$"}"P$Y$"$E "Q$"}"P$;
475 PRINTY$"$J "Q$"rx"P$Y$"$O "Q$"}"P$Y$"$S "Q$"rx";
480 PRINTP$Y$"$X "Q$"}"P$Y$"$\ "Q$"}"P$Y$"$` ";
485 PRINTQ$"}"G$Y$"%( "F$Y$"%/ "P$" "Q$C$"|";
490 PRINTC$" "P$" "Q$C$"|"C$" "P$" "Q$C$"|";
495 PRINTP$Y$"%"CHR$(64)" pppp "Q$"}"P$Y$"%J "Q$"}"P$;
500 PRINTY$"%S "Q$"}"P$Y$"%\ "Q$"}"P$Y$"%` "Q$"}";
505 PRINTP$Y$"&0 "Q$C$"|"C$" "P$" "Q$C$"|";
510 PRINTC$" "P$" "Q$C$"|"P$Y$"&"CHR$(64)" ppppp"Q$"]";
515 PRINTP$Y$"&J "Q$"}"P$Y$"&S "Q$"}"Y$"&\"D$;
520 PRINTP$"ppp "Q$"}"P$Y$"'` "Q$"}"Y$"(]"D$P$"pp";
525 PRINTQ$"rx"P$Y$", r"Q$"ppppppp"P$D$Q$"y"P$;
530 PRINTY$"-  "Q$"}"Y$"-(p"P$Y$"-, "Q$"}"P$Y$"->p";
535 PRINTQ$"]"P$Y$"-M "Q$"}"P$Y$".  "Q$"}"P$;
540 PRINTY$"., nppp"Q$"]"P$Y$".6p"C$"pppp"Q$"]"P$C$"p";
545 PRINTQ$"]"P$Y$".Cppppp"Q$"]"P$Y$".Kpp np"Q$"]"P$;
550 PRINTY$".Sp"Q$"]"P$"ppp"Q$"]"P$"ppp"Q$"]"Y$".`{"P$"ppp";
555 PRINTQ$"{"P$"p"Q$"]"P$Y$".jppppp"Q$"]"P$Y$"/ ";
560 PRINT" "Q$"}"P$Y$"/, "Q$"rx"C$""D$P$D$Q$"y";
565 PRINTP$Y$"/6 "Q$"rx"P$Y$"/; "Q$"}"P$C$" ";
570 PRINTQ$"}"P$Y$"/C "Q$"}"P$Y$"/M "Q$"}"P$;
575 PRINTY$"/S "Q$"}"P$Y$"/W "Q$"}"P$Y$"/[ "Q$"}";
580 PRINTP$Y$"/` "Q$"}   "P$" "Q$"}"P$Y$"/j "Q$"}";
```

```
10 REM***CHRISTMAS GRAPHICS BY JENNIFER McGRAW*****
15 REM !!! IF YOU REMOVE MY NAME MAY GREMLINS INFEST YOUR ICs !!!
20 E$=CHR$(27):Y$=E$+"Y":G$=E$+"G":F$=E$+"F":P$=E$+"p":Q$=E$+"q"
25 D$=CHR$(95):J$=E$+"j":K$=E$+"k":B$=E$+"B":C$=E$+"C"
30 L1$=CHR$(45):L2$=CHR$(44)
35 WIDTH 255
40 PRINTE$"E"E$"x5";:A1=0:A=0
45 PRINTE$"}"
50 DL$=E$+"M":IL$=E$+"L":DC$=E$+"N"
55 H$=E$+"H":QU$=CHR$(34):EX$=E$+"y5"+G$+Q$+Y$+"6 "
60 GOSUB430:A1=0:GOSUB 640
65 PRINTY$"4!";:FORI=1TO40:PRINT"* ";:NEXT:PRINT
70 FOR P=1TO2000:NEXT
75 GOSUB160
80 FORP=1TO1000:NEXT
85 FORJ=1TO15:PRINTH$;:FORI=1TO23:PRINTDC$:NEXT:NEXT
90 GOSUB 1510:FORP=0TO5000:NEXT
95 FORJ=1TO24:PRINTH$;IL$;:NEXT:PRINT
100 GOSUB 715:FOR I=1TO5
105 FORP=1TO300:NEXT:GOSUB 965:FORP=1TO700:NEXT:GOSUB 970:NEXT
110 FORI=0TO23:PRINTH$;DL$;:NEXT:PRINT
115 A1=1
120 GOSUB160:FORP=1TO500:NEXT:FORI=0TO10:PRINTH$;DL$;:NEXT:PRINT
125 GOSUB975
130 GOSUB160:FORP=1TO400:NEXT:GOSUB1595
135 FORP=1TO5000:NEXT
140 PRINTE$"(":GOSUB430:FORP=1TO20:GOSUB 640:GOSUB 685:NEXT
145 GOTO40
150 END
155 FORJ=1TO10:PRINTH$;:PRINTDL$;:NEXT:PRINT
160 GOSUB315
165 IFA1=1THENGOSUB240:GOSUB 275:RETURN
170 FORP=0TO1000:NEXT
175 PRINTY$;CHR$(37);CHR$(103);".";
180 FORP=0TO2000:NEXT
185 PRINTY$;CHR$(37);CHR$(102);F$"^ ";
190 FORP=0TO2000:NEXT
195 FORI=1TO4
200 ONIGOSUB240,255,265,275
205 FOR P=0 TO 500:NEXTP
210 NEXTI
215 GOSUB290
220 FOR P=0TO200:NEXT
225 GOSUB240:FOR P=0TO500:NEXT
230 A=A+1:IF A<7 THEN 215
235 RETURN
```

```
240 PRINTE$"w"H$G$Q$F$Y$"#e  "Y$"$d ";
245 PRINTP$"r"D$Q$" "Y$"%d "D$"r "Y$"&e  "G$Q$
250 RETURN
255 PRINTE$"w"H$F$Y$"#f¦"Y$"$c{{";
260 PRINTY$"$g{{"Y$"&f¦"G$Q$:RETURN
265 PRINTH$F$Y$QU$"f¦"Y$"$b("Y$"$i("Y$"'f¦"G$;
270 PRINTQ$:RETURN
275 PRINTH$F$Y$"!f¦"Y$"#cy";
280 PRINTY$"#hx"Y$"$^{{{{"Y$"$j{{{{"Y$"&cx"Y$"&hy"Y$"(f¦"Y$")";
285 PRINT"f¦"G$Q$:RETURN
290 PRINTH$F$Y$" f¦"Y$;
295 PRINTQU$"by"Y$QU$"ix"P$Y$"#er"D$Q$Y$"$\{{";
300 PRINTP$Y$"$dr"Q$"r"D$P$D$Q$Y$"$n{{";
305 PRINTY$"%d"D$P$D$"r"Q$"r"Y$"&e"D$"r"Y$"'bx"Y$"'";
310 PRINT"iy"Y$"*f¦"G$Q$:RETURN
315 PRINTE$"w"H$E$"J"Y$" Q*"Y$" h*"Y$"!#.";
320 PRINTY$"!/*"Y$"!?*"Y$"!N*"Y$"!S*"Y$QU$" ."Y$QU$"$*";
325 PRINTY$QU$"(."Y$"$5*"Y$"$9*"Y$"$N*"Y$"$Q*"Y$"$b."Y$"%";
330 PRINT"2*"Y$"&.*"Y$"&B*"Y$"&J*"Y$"&f."Y$"''*"Y$"'Q*"Y$"'";
335 PRINT"U."Y$"'j*"Y$"(/*"Y$")**"F$Y$"*#*"G$Y$"*9*";
340 PRINTY$"*G*"Y$"*a*"F$Y$"+!."G$Y$"+,."Y$"+G."Y$",5.";
345 PRINTF$Y$"-8fcfc"G$Y$"-?*"Y$"-W*"F$Y$"-h*"Y$".&*";
350 PRINTY$".8`on`"Y$".K*"Y$".S."Y$".Z("C$"("Y$".j."Y$"/).";
355 PRINTY$"/2xzy";
360 PRINTY$"/9¦)"G$" "F$Y$"/G."Y$"/Z¦p)"Y$"0 y"Y$"00xz m zy";
365 PRINTY$"09¦)"Y$"0N(((((""Y$"0Ze"P$"p"Q$"d"Y$"1!y{{{"Y$"1.";
370 PRINT"xz    m  zy¦) "Y$"1B{{{"P$"i"Q$"zzzzz¦az"C$;
375 PRINT" n1 ¦zam{{{) ¦{{{{"Y$"2%zzzy{";
380 PRINT"("Y$"2.i p       "P$"i"Q$"¦)aa("P$"i"Q$"zzzm  "P$"i"Q$" p";
385 PRINT"   a"P$"i"Q$"zzzzzzm  aaaaz z ! "P$"i"Q$G$Y$"2o ";
390 PRINTF$Y$"3*yy("C$"i{{{{{ (  "P$"ii"Q$"  az y{{{{{"P$"i";
395 PRINTQ$" o    "P$"i"Q$"p   "E$"pii"Q$"   "P$"iiiii"Q$" n   ";
400 PRINTP$"i"Q$"zzzzz¦"Y$"3o "Y$"4%{{{{{{{x"Y$"44y{{{{"P$"i"Q$"a(";
405 PRINT"aazz"Y$"4Fy{{{aa"P$"i"Q$"aaaaai{{x zzzaaaaaaz";
410 PRINTY$"4dzzzaaazzy{{{"Y$"5 zzzzz";
415 PRINT"    y      {{{{aaazzz"SPC(27)"y          yaaa";
420 PRINTG$"V"F$"{{{x"Y$"6 xzzVx"SPC(50);
425 PRINT"\{{{{{           xzzVzzzzaa"G$Q$:RETURN
430 PRINTH$P$F$Y$"!0r"Q$"rppp"P$"r"Q$"rppp"P$;
435 PRINTD$Q$"y"Y$QU$"/ "P$" "Q$C$"¦"C$" "P$" ";
440 PRINTQ$C$"¦"P$Y$QU$": "Q$C$"¦"Y$"#/ ";
445 PRINTP$" "Q$C$"¦"C$" "P$" "Q$C$"¦"C$" ";
450 PRINTP$" "Q$C$"¦"P$Y$"#"CHR$(64)"pppppp"Q$"]"P$;
455 PRINTY$"#Jp"C$"pppp"Q$"]  "P$"p"C$"pppp"Q$"]"P$Y$"#";
460 PRINT"\p"Q$"]"P$Y$"#`p"Q$"]"Y$"$/ "P$" "Q$C$"¦";
465 PRINTC$" "P$" "Q$C$"¦"C$" "P$" "Q$C$"¦";
585 PRINTP$Y$"0  "Q$")"P$Y$"0(p"Q$"]"P$Y$"0, ";
590 PRINTQ$")"G$" "P$F$Y$"01 "Q$")"P$Y$"06 ";
595 PRINTQ$")"P$Y$"0> "Q$")"Y$"0Cpppp"P$" "Q$")"P$;
600 PRINTY$"0M "Q$")"P$Y$"0S "Q$")"P$Y$"0W "Q$")";
605 PRINTP$Y$"0[ "Q$")"P$Y$"0` "Q$")   "P$" "Q$")";
610 PRINTG$C$" "F$C$"pppp"P$" "Q$")"Y$"1 "D$;
615 PRINTP$D$"pppppp"Q$"rx"P$Y$"1, "Q$")"P$Y$"11 ";
620 PRINTQ$")"P$Y$"16 "Q$")"P$Y$"1> "Q$")"P$;
625 PRINTY$"1Cpppp "Q$")"P$Y$"1M "Q$")"P$Y$"1S "Q$")";
630 PRINTP$Y$"1W "Q$")"P$Y$"1[ "Q$")"P$Y$"1` npp";
635 PRINTQ$"m"P$" p"Q$"]"P$Y$"1jpppp "Q$")"G$:RETURN
640 PRINTE$"w"H$F$Y$"!%!";:IFA1=1THENPRINTY$"2j!";
645 PRINTY$QU$QU$"y"Y$QU$"'x";
650 IFA1=1THENPRINTY$"3gy"Y$"31x";
655 PRINTY$"# (("P$Y$"#$r"D$Q$Y$"#((";
660 IF A1=1THENPRINTY$"4e(("P$Y$"4ir"D$Q$Y$"4m((";
665 PRINTY$"$$"D$"r ";
670 IFA1=1THENPRINTY$"5i"D$"r ";
675 PRINTY$"%"QU$"x"Y$"%'y";:IFA1=1THENPRINTY$"6gx"Y$;6"]y"Y$"7j!";
680 PRINTY$"&%¦"G$Q$;H$:RETURN
685 PRINTH$Y$"!% "Y$"2j ";
690 PRINTY$QU$QU$"      "Y$"3g      ";
695 PRINTY$"#           "Y$"4e          ";
700 PRINTY$"$$   "Y$"5i   ";
705 PRINTY$"%"QU$"      "Y$"6g      ";
710 PRINTY$"&% "Y$"7j "H$:RETURN
715 PRINTH$J$Y$QU$"a,sss,"Y$"#`o";
720 PRINTF$"SSSS$"G$"o"Y$"$F,sss,"P$F$Y$"$[r"Q$"zz";
725 PRINTP$D$Q$G$C$"S"F$"S"P$"^.^"Q$"S"G$"S";
730 PRINTP$F$C$"r"Q$"zz"P$D$Q$G$Y$"%";
735 PRINT"EoS"F$"SSSS"G$"o"P$E$"F"Y$"%Zr"Q$G$C$"  ";
740 PRINTP$F$C$D$Q$C$D$P$" "G$"O"E$;
745 PRINT"F "Q$"r"P$C$"r"Y$"%]"D$Q$G$Y$"&+,sss,";
750 PRINTP$F$Y$"&"CHR$(64)"r"Q$"zz"P$D$Q$E$;
755 PRINT"G"C$"SS"P$F$"^.^"Q$"S"G$"S"P$F$C$"r";
760 PRINTQ$"zz"P$D$Y$"&Yr"Q$G$C$"     "F$"r";
765 PRINTG$"     "F$D$"¦"P$Y$"&m"D$E$;
770 PRINT"q"G$Y$"'*oSSSS"F$"S"E$"Go"P$F$Y$"'?r"Q$;
775 PRINTG$C$"  "P$F$C$D$Q$C$D$;
780 PRINTP$" "G$"o"F$" "Q$"r"P$C$"r'Y$"'Q"D$;
785 PRINTY$"'Yr"Q$"  "G$"   "F$"x"G$"       "F$"y"P$Y$"'m";
790 PRINTD$Y$"(%r"Q$"zz"P$D$Q$G$C$"SS"E$;
795 PRINT"p"F$"^.^"Q$"S"G$"S"P$F$C$"r"Q$"zz"P$;
800 PRINTD$Y$"()r"Q$G$C$"    "F$")"P$"r"G$"MMmM";
805 PRINT"M"F$D$Q$"¦"P$Y$"(R"D$Y$"(Yr"Q$;
810 PRINTY$"(^x"G$" "P$F$"r"Q$"sssss"P$D$Q$;
```

```
815 PRINTC$"y"P$Y$"(m"D$Y$")$r"Y$"))"D$Q$;
820 PRINTC$D$P$" O "Q$"r"P$C$"r"Y$")6"D$;
825 PRINTY$"))>r"Q$"  "G$"   "P$F$"x  x`y "G$" "F$"y"Y$")";
830 PRINT"R"D$Y$")Yr"Q$Y$")]x "G$" "F$"rzzzzz"D$Y$")iy";
835 PRINTP$Y$")m"D$Y$"*#r"Q$Y$"**r"G$"     "F$;
840 PRINTD$";"P$Y$"*7"D$E$"Y*>r"Y$"*Cr"G$" "F$"x y`x y";
845 PRINTG$" "F$D$Y$"*R"D$Y$"*Yr"Q$Y$"*\x  ";
850 PRINTF$"         "Y$"*jy"P$Y$"*m"D$Y$"+#r";
855 PRINTQ$Y$"+)x"G$"   "C$"   "F$"y"P$Y$"+7"D$;
860 PRINTY$"+>r"Q$Y$"+C"D$P$"    w    "Q$"r "P$E$;
865 PRINT"Y+R"D$Y$"+Z"D$Q$C$"zz"D$";"G$"       "F$;
870 PRINTC$"rzz"P$C$"r"Y$",#r"Q$Y$",(x"G$" "P$;
875 PRINTF$"r"Q$"sssss"P$D$Q$C$"y"F$Y$",7";
880 PRINTD$Y$",>r"Q$Y$",A   "P$"y  x y  x"Q$"   "P$;
885 PRINTY$",R"D$Y$",["D$Q$Y$",^}"G$C$"       "F$;
890 PRINTC$";"P$Y$",kr"Y$"-#r"Q$Y$"-`x "G$" "F$"rzzzz";
895 PRINT"z"D$Y$"-3y"P$Y$"-7"D$Y$"-?"D$Q$C$"   ";
900 PRINTP$" zz"G$"   "F$"zz "Q$"   "P$C$"r"Y$"-\r"Q$"r";
905 PRINT")"G$"        "F$C$";"D$P$D$Y$".#r"Q$;
910 PRINTY$".&x  "G$"        "F$" "Y$".4y"P$Y$".7"D$;
915 PRINTY$"."CHR$(64)D$G$Y$".D         "F$Y$".Pr"Q$;
920 PRINTY$".^){{{{{{{{{;"P$Y$"/$"D$Q$C$"zz"D$";"E$;
925 PRINT"G       "F$C$"rzz"P$C$"r"Y$"/Ar"Q$"r"P$;
930 PRINTG$C$"         "Q$F$C$D$P$D$Q$;
935 PRINTY$"/^}uuuuuuuuu;"P$Y$"0%"D$Q$Y$"0()"G$;
940 PRINTC$"       "F$C$";"P$Y$"05r"Y$"0D(((((((((Q$;
945 PRINTY$"0"D$"zzzzzzzzz"P$Y$"1&r"Q$"r)"G$"        "F$;
950 PRINTC$";"D$P$D$Y$"1Duuuuuuuuu"Q$Y$"2()(((((((((;"Y$;
955 PRINT"2Dzzzzzzzzz"Y$"3()uuuuuuuuu;"Y$"4)zzzzzzz";
960 PRINT"zz"G$Q$:RETURN
965 PRINTP$F$Y$"%c^"Y$"'HQ"Y$")-^"Q$G$:RETURN
970 PRINTP$F$Y$"%cO"Y$"'H^"Y$")-O"Q$G$:RETURN
975 X=30:Y=30:Z=30:K1$=K$+B$+J$
980 FOR CY=1TO10
985 FORI=1TO8:IFCY>1THENX=X+2:IFCY>3THENY=Y+2:IFCY>5THENZ=Z+2
990 IFCY=1THENPRINTY$L1$" ";:GOSUB1045:GOTO 1025 '8FOR#1
995 IFCY=3 THENPRINTY$L2$" ";:GOSUB1045 '8FOR#2
1000 IFCY=5THENPRINTY$L1$" ";:GOSUB 1045 '8FOR#3
1005 IF CY>5 AND Z<110 THEN PRINT Y$L1$CHR$(Z);:GOSUB 1050 '#3
1010 IFCY>3ANDY<110THEN1015 ELSE 1020
1015 PRINTY$L2$CHR$(Y);:ONIGOSUB1450,1285,1335,1400,1450,1285,1335,1400
1020 IF X<110 THEN PRINT Y$L1$CHR$(X);:GOSUB 1050 '#1
1025 IFCY=1THENFORP=0TO250:NEXT
1030 NEXT
1035 NEXT
1040 RETURN
```

```
1275 PRINT F$" "D$Q$"  "P$" "D$Q$"  "P$" "D$Q$" ";
1280 PRINT P$" "D$G$Q$:RETURN
1285 PRINT J$F$"                ";
1290 PRINT P$"r;"G$") "F$"y"D$Q$" "F$K1$"       ";
1295 PRINT P$"r"Q$"r.:."D$P$D$Q$"  "P$"V "Q$"rzz;";
1300 PRINT " "F$K1$"     "P$"r "Q$": : :"P$" "D$"r  "Q$"((";
1305 PRINT "x "K1$"    x"P$"  wwwww  y  "Q$" "F$"* ";
1310 PRINT K1$"   x "P$"  x     (  "Q$"r "K1$"  o  "P$" x";
1315 PRINT Q$"r     "P$") "Q$" "K1$"     "P$" ;"Q$"      "P$") ";
1320 PRINT Q$" "K1$"     "P$" ;"Q$"      "P$") "Q$;
1325 PRINT " "K1$"     "P$" y"D$Q$"     ";
1330 PRINT P$") "D$G$Q$" ":RETURN
1335 PRINT J$F$"                "P$"r;"G$") "F$"y"D$;
1340 PRINT Q$" "K1$"      "P$"r"Q$"r.:."D$P$;
1345 PRINT D$Q$"  "P$"V "Q$"rzz; "K1$" "F$"    "P$"r ";
1350 PRINT Q$": : :"P$" "D$"r  "Q$"((x "K1$"    x";
1355 PRINT P$"  wwwww  y  "Q$" * "K1$"   ";
1360 PRINT " ;"P$"y  x    ;  "Q$"r "K1$"   o";
1365 PRINT P$"r) "Q$"r     "P$" x"Q$" "K1$" ";
1370 PRINT "   "P$" "Q$"r"D$P$D$Q$"    "P$"r";
1375 PRINT Q$"r"P$" "Q$" "K1$"    "P$" "Q$;
1380 PRINT "  "D$P$D$Q$"  "P$"r";
1385 PRINT Q$"r "P$F$" "Q$" "K1$"  ";
1390 PRINT "  "P$" "D$Q$"  "P$" ";
1395 PRINT D$"r "Q$"  "P$" "D$Q$" "G$:RETURN
1400 PRINT J$F$"                "P$"r;"G$") "F$"y"D$;
1405 PRINT Q$" "F$K1$"      "P$"r"Q$"r.:."D$P$;
1410 PRINT D$Q$"  "P$"V "Q$"rzz; "F$K1$"     "P$"r ";
1415 PRINT Q$": : :"P$" "D$"r  "Q$"(("")x "K1$"    x"P$"  wwwww     "Q$;
1420 PRINT "  * "K1$"    ;"P$"y  x      (";
1425 PRINT Q$"r "K1$"    n"P$") "Q$"r     "P$;
1430 PRINT " x"Q$" "K1$"     "P$") "Q$"      "P$" ;"Q$" ";
1435 PRINT K1$"     "P$") "Q$"      "P$F$" ;"Q$;
1440 PRINT " "K1$"     "P$") "D$Q$"     "P$" y";
1445 PRINT D$G$Q$" ":RETURN
1450 PRINT J$F$"                "P$"r;"G$") "F$"y"D$;
1455 PRINT Q$" "K1$"      "P$"r"Q$"r.:."D$P$;
1460 PRINT D$Q$"  "P$"V "Q$"rzz; "K1$"     "P$"r ";
1465 PRINT Q$": : :"P$" "D$"r  "Q$"(("")x "K1$"    x"P$"  wwwww  y  "Q$" * ";
1470 PRINT K1$"   x "P$"  x(    y   "Q$"r "K1$"   n "P$" ";
1475 PRINT Q$"r"P$" "Q$"r    "P$"y "Q$" "K1$"     "P$" "Q$;
1480 PRINT " "F$D$P$D$Q$"    "P$" ";
1485 PRINT Q$D$P$D$Q$" "K1$"     "P$" "Q$"  ";
1490 PRINT D$P$D$Q$"   "P$" "Q$" "D$P$D$;
1495 PRINT Q$" "K1$"     "P$" "D$;
1500 PRINT Q$"  "P$" "D$Q$"  "P$" "D$Q$" "P$" ";
```

```
1045 ONIGOSUB 1060,1070,1080,1095,1125,1155,1195,1230:RETURN
1050 ONIGOSUB 1285,1335,1400,1450,1285,1335,1400,1450:RETURN
1055 ONIGOSUB 1450,1285,1335,1400,1450,1285,1335,1400:RETURN
1060 PRINT E$"w"J$P$F$"y";
1065 PRINTD$Q$K1$"z¦"K1$"x"G$:RETURN
1070 PRINT J$P$") "F$"y"D$Q$K1$"rzz¦"K1$"((x";
1075 PRINT K1$" *"G$Q$:RETURN
1080 PRINT J$P$F$"r¦"G$") "F$"y"D$K1$"V ";
1085 PRINT Q$"rzz¦"P$K1$"  "Q$"((x"P$K1$"  "Q$"  ";
1090 PRINT "*"P$K1$" "Q$"r "G$Q$:RETURN
1095 PRINT J$F$"  "P$"r¦"G$") "F$"y"D$Q$;
1100 PRINT K1$"  "P$"V "Q$"rzz¦"P$ K1$""D$"r  "Q$"((x "P$;
1105 PRINT K1$" y  "Q$" "F$"* "P$F$K1$"   ";
1110 PRINT Q$"r "P$K1$"y "K1$" "Q$D$P$D$K1$" ";
1115 PRINT Q$" "F$D$P$D$K1$" "D$Q$" "P$" ";
1120 PRINT D$G$Q$:RETURN
1125 PRINT J$F$"    "P$"r¦"G$") "F$"y"D$Q$;
1130 PRINT K1$D$P$D$Q$"  "P$"V "Q$"rzz¦"K1$"."P$" ";
1135 PRINT D$"r  "Q$"((x"P$K1$"w  y  "Q$" *"P$;
1140 PRINT K1$"  (  "Q$"r"K1$"  "P$") "Q$K1$"  "P$") ";
1145 PRINT Q$K1$"  "P$") "Q$K1$"  "P$") ";
1150 PRINT D$G$Q$:RETURN
1155 PRINT J$F$"       "P$"r¦"G$") "F$"y"D$Q$;
1160 PRINT K1$".:."D$P$D$Q$"  "P$"V "Q$"rzz¦"K1$" : :";
1165 PRINT P$" "D$"r  "Q$"((x "P$;
1170 PRINT K1$"wwww     "Q$"  * "P$K1$"x   y  ("Q$;
1175 PRINT "r"K1$"     "P$" x"Q$K1$"    "P$"r";
1180 PRINT Q$"r"P$" "K1$D$Q$"  "P$"r";
1185 PRINT Q$"r "P$" "K1$" "D$"r "Q$"  ";
1190 PRINT P$" "D$G$Q$:RETURN
1195 PRINT J$F$"         "P$"r¦"G$") "F$"y"D$K1$"r";
1200 PRINT Q$"r.:."D$P$D$Q$"  "P$"V "Q$"rzz¦"P$;
1205 PRINTK1$" "Q$": : :"P$" "D$"r  "Q$"((x"P$K1$" wwwww  y  ";
1210 PRINT Q$" "F$" *"P$K1$"  x    ¦  "Q$"r "P$;
1215 PRINT K1$" "Q$"r     "P$F$" x"K1$" "Q$"      "P$;
1220 PRINT " ¦"K1$" "Q$"      "P$" ¦"K1$" "D$;
1225 PRINT Q$"     "P$" y"D$G$Q$:RETURN
1230 PRINT J$F$"           "P$"r¦"G$") "F$"y"D$;
1235 PRINT Q$K1$"  "P$"r"Q$"r.:."D$P$D$Q$"  "P$"V ";
1240 PRINT Q$"rzz¦"K1$" "P$"r "Q$": : :"P$" "D$"r  "Q$"((x";
1245 PRINT K1$"x"P$"  wwwww  y  "Q$" "F$"*"K1$" "P$"  x( ";
1250 PRINT "  y   "Q$"r"K1$" "P$" "Q$"r"P$" "Q$"r    "P$"y ";
1255 PRINT Q$K1$" "P$" "Q$" "D$P$D$;
1260 PRINT Q$"    "P$" "Q$D$P$D$Q$;
1265 PRINT K1$" "P$" "Q$"  "D$P$D$Q$"   "P$" ";
1270 PRINT Q$" "D$P$D$Q$G$K1$" "P$;
```

```
1505 PRINT D$Q$" "G$Q$:RETURN
1510 PRINTH$Y$"!m."Y$"#h*"Y$"%e.";
1515 PRINTY$"&l*"F$Y$"*bxzy"Y$"*i(("Y$"+b¦"C$")"Y$"+hxxyy";
1520 PRINTY$",dx"Y$",gv((((t"Y$"-d¦"Y$"-h!a))"G$Y$".I       ";
1525 PRINTF$Y$".d¦"Y$".hy())"G$Y$"/I       "F$Y$"/d¦";
1530 PRINTY$"/hfu yy"G$Y$"0I       "F$Y$"0d¦"Y$"0gx"G$" ";
1535 PRINTF$"   ¦"G$Y$"1I       "F$Y$"1U(((((Y$"1c)"P$;
1540 PRINT"b"Q$"zz x   ¦"G$Y$"2I    "F$"(((((((xx ^ e¦"Y$"2d!zz)ssss¦";
1545 PRINTG$Y$"3I   "F$"x "G$"ccccc"F$" z"G$"c"F$"faaz";
1550 PRINTY$"3d¦"Y$"3g)uuuu¦"G$Y$"4I   "F$"zy "G$"cccccc"F$"x";
1555 PRINTG$Y$"4Y  "P$F$"rx ^ e"Q$G$"  "F$C$"¦";
1560 PRINTY$"4g)    ¦"G$Y$"5I    "F$"))zzzz"P$"r "G$"ccccc";
1565 PRINTF$" z"G$"c"F$" "Q$"ppz"G$"  "F$C$"¦"Y$"5g) ";
1570 PRINTG$"   "F$"¦"Y$"6Nz"G$Y$"6R  "F$D$P$" ";
1575 PRINTG$"cccccc"Q$F$"r"G$"   "F$Y$"6d¦"Y$"6g)((((¦"E$;
1580 PRINT"G"Y$"7T "F$"q"G$" "F$Y$"7Y q"G$"   "F$Y$"7d¦";
1585 PRINTY$"7gxuuuuy"G$Q$H$
1590 RETURN
1595 PRINTH$;
1600 PRINTY$",+              "Y$",0   ";
1605 PRINT"      "F$"(("G$"         "F$;
1610 PRINTY$"-)    ((              "Y$"-K          ";
1615 PRINT"xz((yxzzzzy     "Y$".'    xz((yxzzzzy       (((((";
1620 PRINT"((((((((((((    xzzzzzxxwyzzy\\y     "Y$"/%  xzzzzz";
1625 PRINT"xxwyzzy\\y"G$"    "F$"xzz(              ";
1630 PRINT"zzyxx////"G$"  "F$"ww"G$"  ";
1635 PRINTF$"\\yzy     "Y$"0% xx////  ww  \\yzy   x  ";
1640 PRINT"xOy            (   y       ww            "G$;
1645 PRINTY$"1%         "F$"ww"G$"          ";
1650 PRINTF$")"G$" "F$" x y"G$"    ";
1655 PRINT"       "F$"xOy"G$"   "F$"¦";
1660 PRINTG$"       "F$"ww"G$"           ";
1665 PRINTF$Y$"2%         ww          ) )yyx!   ***";
1670 PRINT"   )xy y  ¦"G$"       "F$"ww"G$;
1675 PRINT"            "Y$"3%         "F$"ww";
1680 PRINTG$"          "F$")"G$" ";
1685 PRINTF$") yx¦"G$" "F$D$;
1690 PRINTP$"    "Q$"r"G$"  "F$"` y";
1695 PRINT"()"G$"  "F$"¦"G$"       ";
1700 PRINTF$"ww"G$"            "F$;
1705 PRINTY$"4&        ww  (((((((() )   ¦  w  w   `    y ¦(((";
1710 PRINT"(("G$" "F$"ww   (((((((  "Y$"5'";
1715 PRINT"   zzzzzzzz       zzzzzz   zzzzzzzzzzzzzzz      zzzzzz";
1720 PRINTG$"     "F$"  z"G$Q$
1725 RETURN
```

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.

885-2035