$2.50

REMark®

Issue 45 • October 1983

Official magazine for users of HEATH ZENITH computer equipment.

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

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

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

When you're ready to move up to H-100 Series, remember: these computers are available *only* through the companies that give you solid service and a trustworthy warranty...Heath and Heathkit® Electronic Centers.* The companies that stand by their promise of support to kitbuilders. The companies with the pledge: *"We won't let you fail."*

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

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

Heathkit® ELECTRONIC CENTER*

*Heathkit Electronic Centers are units of Veritechnology Electronics Corporation.

VEC-770

Heath Users' Group

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

REMark Editor .... Walt Gillespie
Assistant Editor .... Donna Melland

Printers .... Imperial Printing
St. Joseph, MI

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

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

*U.S. Funds.

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

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

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

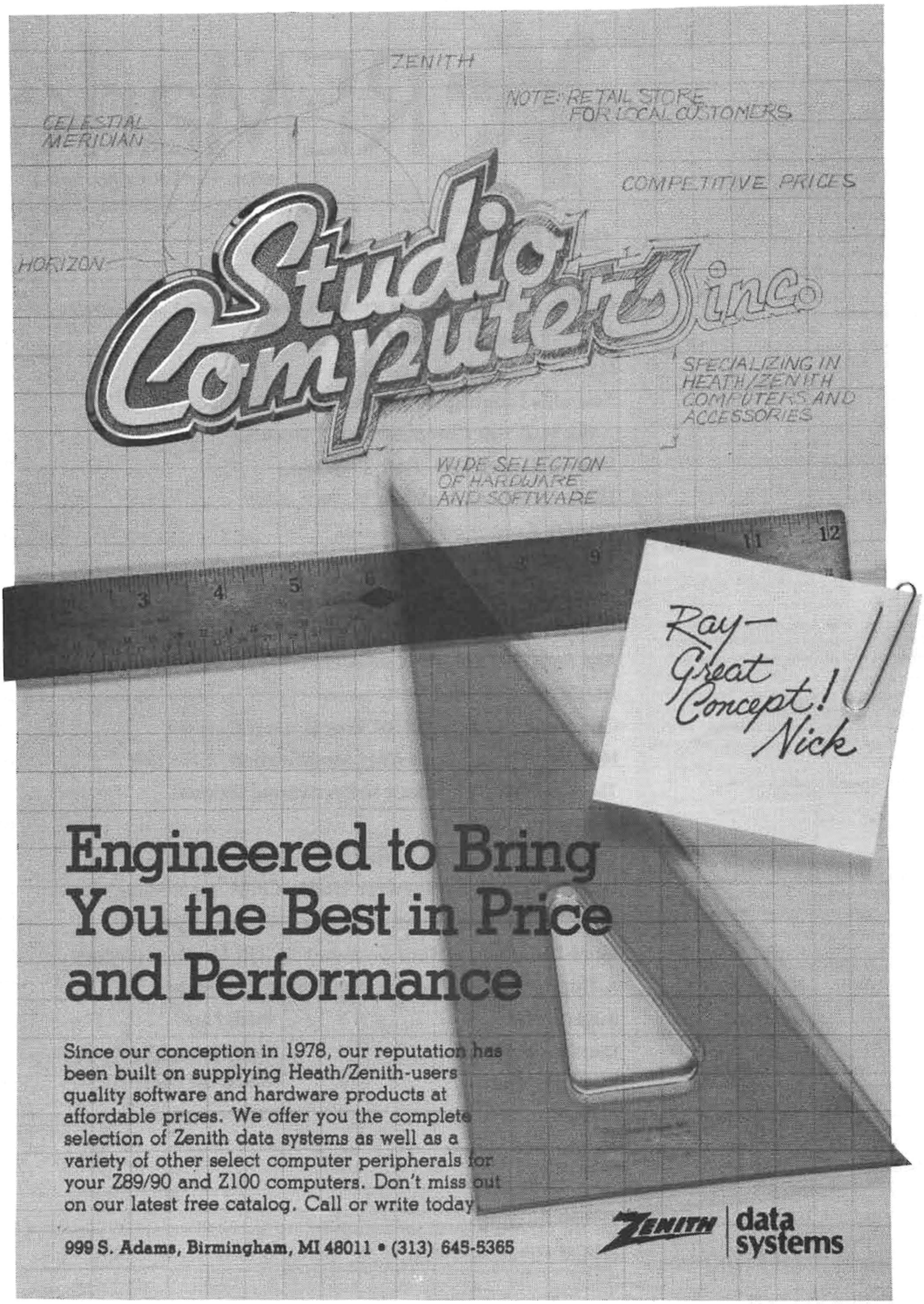
616-982-3463

REMark®

Issue 45 • October 1983

on the stack

ON THE COVER: A photo representation of The Second National HUG Conference. See pages 39-45 for a HUG Staff report.


ZENITH
NOTE: RETAIL STORE FOR LOCAL CUSTOMERS
CELESTIAL MERIDIAN
COMPETITIVE PRICES
HORIZON
Studio Computers inc.
SPECIALIZING IN HEATH/ZENITH COMPUTERS AND ACCESSORIES
WIDE SELECTION OF HARDWARE AND SOFTWARE
Ray— Great Concept! Nick

Engineered to Bring You the Best in Price and Performance

Since our conception in 1978, our reputation has been built on supplying Heath/Zenith-users quality software and hardware products at affordable prices. We offer you the complete selection of Zenith data systems as well as a variety of other select computer peripherals for your Z89/90 and Z100 computers. Don't miss out on our latest free catalog. Call or write today.

999 S. Adams, Birmingham, MI 48011 • (313) 645-5365

ZENITH | data systems

HDOS Goodies

Dear Nancy,

There have been several stories in REMark and SEXTANT about how to save disk space with HDOS or how to run HDOS STANDALONE. Your readers may be interested in some changes I made to HDOS. My version uses only 36 sectors and requires 4K less memory than standard HDOS, while having a few extra features. This helps the disk problem while not using up lots of memory to do it. If anyone wants a copy, they should write to me for details.

Skip Chambers
3822 Westminster Drive
Carrollton, TX 75007

Public Domain "C" Programs

Dear HUG,

I was happy to see Clement Pepper's article on C/80 in Issue #41. I had just ordered C/80 from the Software Toolworks and set a goal of learning C by the end of the summer.

However, in his list of references, Mr. Pepper omitted an invaluable resource, the C Users' Group. The group maintains a library of about 30 volumes of public domain C programs in 6 or 7 different disk formats, including Heath. In addition, it publishes a newsletter at irregular intervals. The address for the group is:

C Users'Group
112 N. Main
Yates Center, KS 66703

The one year membership fee is $10.00.

David Tichane
60 Plaza Street, Apt. 4B
Brooklyn, NY 11238

Evaluation of 2 Parallel Interface Boards

Dear HUG,

I have just finished an evaluation of two parallel interface boards. The reason for the evaluation was to find which boards could drive my company's printers in both serial and parallel so that they could recommend a card to be used for the Heath '89. The cards tested were 2/3rds card from Secured Computer Systems and Wisconsin Intelligent System Engineering's 2S+2P+RTC card.

Vectored to 6 ☞

Questions & Answers

Compiled and edited by Terry Jensen, Software Developer.

(EDITOR'S NOTE: If you need answers to specific questions on software or hardware problems that would be beneficial to other users, please drop us a note to, Questions & Answers, Heath Users' Group, Hilltop Road, St. Joseph, MI 49085. Please keep your questions brief and to the point. We will do our best to answer your questions in a future issue. Some of the Questions & Answers are contributed by Zenith Data Systems Software Consultation.)

Q. When I connected a single-sided 8" floppy disk drive to my Z-100 computer, I was not able to format any 8" disks using CP/M-85. What can I do to correct this?

A. Both CP/M-85 and ZDOS support a maximum of two 5.25" and two 8" disk drives. Although any combination of single-sided and double-sided drives can be used with ZDOS, CP/M-85 does not fully support single-sided 8" drives at this time. However, double-sided 8" disk drives can be used with CP/M-85. If you wish to use single-sided 8" disk drives with CP/M-85, you should call (616) 982-3860, or write to:

Zenith Data Systems, Software Consultation
Hilltop Road, St. Joseph, MI 49085

Q. Is there any way to transfer my H-88 cassette programs to my new disk system under HDOS or Benton Harbor BASIC?

A. The HUG programs BASCON and TXTCON (P/N 885-1077) will allow you to transfer to disk format, cassette Benton Harbor BASIC programs, and text files, respectively. Refer to the HUG Software Catalog, page 26 for a detailed description.

Please be informed that BASCON and TXTCON will transfer files to disk format. The files (programs or text files) may need to be altered or modified to use under the HDOS operating system. For example, the result of PEEKs and POKEs in memory with Cassette Benton Harbor BASIC may not hold under the disk Benton Harbor BASIC, depending on the particular application.

Q. How do I output control characters to the CRT from a FORTRAN program?

A. The following program demonstrates using reverse video from a FORTRAN program:

```
      INTEGER*1 ESC
      ESC = 27
      WRITE(1,100)ESC,ESC
100   FORMAT(1X,A1,'pTEST',A1,'q')
      END
```

Q. The "SAVE 0 GO.COM" command to re-enter CP/M MBASIC (REMark #41, P. 43) seems mystical. Are the string variables lost?

A. A complete explanation of this entry into CP/M can be found in the Buggin' HUG column in Issue 44, page 6 by Craig Pearce.

The "SAVE 0 GO.COM" command saves the null character or simply nothing in the file GO.COM. When CP/M executes GO.COM, it jumps to the beginning address (100H) of the TPA (Transient Program Area). As long as you do not run another program after returning to the CP/M prompt from MBASIC, entering "GO" will return you to the TPA where MBASIC and your program are still loaded.

You will find that the values of the variables will be intact the same as when you exit a program to MBASIC by using the STOP command or a CTRL-C. The CONTinue function will allow the program to continue provided the program has a break point.

Q. I have some H89 software that uses all of the H89 function keys. How can I run this software on my Z-100 when it does not have an ERASE key or the red, blue, and white keys?

A. The F0 key on the Z-100 functions the same as the ERASE key on the H89. The F6, F7, and F8 keys duplicate the red, blue, and white keys.

In the configurations sent, the 2/3rds card with cables and the 2S+2P+RTC with a special parallel cable, the two cost about the same. I was pleased with the "plug and playability" of both cards. All that needed to be done was to make a final cable adaptor for 2/3rds and use a Centronics to Integral Data Systems Converter with the 2S+2P+RTC and away I went. For the serial tests, I used .DVDs and CT.DVD from FBE Research (which if modified, can work miracles with our P80's and P132's).

My conclusions to the evaluation are if you need a real time clock, go with the 2S+2P+RTC card, but if you can live without it, then definitely go with the 2/3rds. The reason I favor the 2/3rds card is that the manual was definitely better written and clearer about how the card functions. I find (as much as I hate extra paperwork) that a good manual can save many hours of head scratching and in some cases, shop repair costs.

Dan Morin
Integral Data Systems
Rt. 13 South
Milford, NH 03055

Screen Control Demo Program

Dear HUG,

I keep thinking the last issue of REMark was the very-best-ever, but each month brings a better one! Issue #39 was an improvement over issue #38, as 38 bettered 37, etc... Just imagine! Four extremely practical articles which I can use on my H8/H19/H17 system without any debugging or conversion as when copying programs from other "foreign-to-Heath" publications! (See pp11, 19, 28, 38, et al.) And all four in one issue of REMark!

The moment our postman delivers REMark, I immediately switch on my system and try everything which pertains to HDOS, Benton Harbor BASIC, ASM, and MBASIC. Sometimes I find solutions to problems which have tormented me for weeks (often months!). But I have yet to find an issue which does not contain new or better ideas which I can adapt for my own use, sometime, sooner or later.

Here is a listing of things I did with Dave Warnick's Screen Control demo program from issue #39. In it I have included remark statements which may be of use or interest to other HUGgites. I found out about not using the semicolon (;) from typing in the huge Christmas graphics program of Jennifer McGraw's in issue #35. The program runs just peachy keen (now that I've corrected all

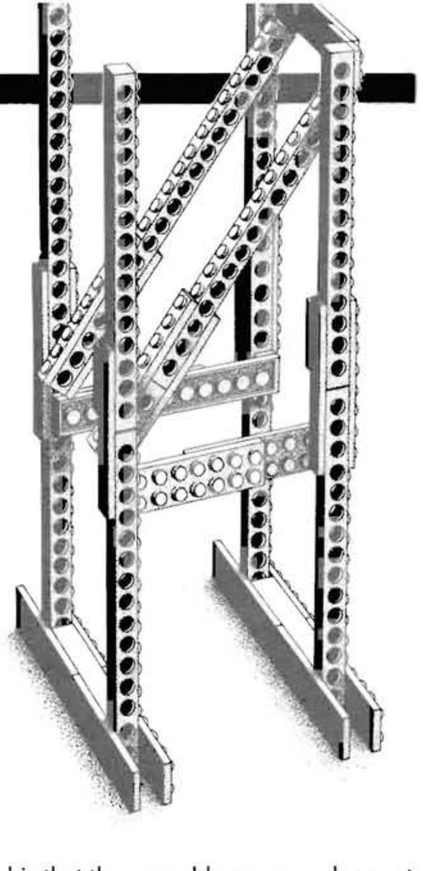
Vectored to 87 ☞

```
2 ' ESCGRAF.BAS     SCREEN-CONTROL GRAPHICS TUTORIAL DEMO PROG.
3 '
4 ' Ref: REMark (R) #39, 1983, pp 19 - 21
5 '
6 ' David E. Warnick's demo program therein.....
7 '
8 ' *Additional remarks by L. E. Geisler, Ann Arbor, MI  48105
9 '
10  E$=CHR$(27)             ' ESCape
20  E1$=E$+"Y"              ' ESC Y - Dirct Cursor Addressing
30  E2$=E$+"E"              ' ESC E - Clear Display
40  E3$=E$+"y5"             ' ESC y5 - Cursor On
50  E4$=E$+"x5"             ' ESC x5 - Cursor Off
60  E5$=E$+"["              ' ESC [ - Enter Hold-Screen Mode
70  E6$=E$+"\"              ' ESC \ - Exit Hold-Screen Mode
80  E7$=E$+"F"              ' ESC F - Enter Graphics Mode
90  E8$=E$+"G"              ' ESC G - Exit Graphics Mode
100 E9$=E$+"p"              ' ESC p - Enter Reverse Video
110 F1$=E$+"q"              ' ESC q - Exit Reverse Video
115 '
120 '     The next line will
130 '          1 - Turn cursor off
140 '          2 - Enter hold-screen mode (no scrolling)
150 '          3 - Erase screen
152 '
153 '  *MBASIC VER 4.82 does not seem to need semicolons (;)
156 '   between ESCape commands on the same line.
158 '
160 PRINT E4$E5$E2$         ' *MBASIC thinks there are semicolons here!
165 '
170 '  Now send cursor to line 5,
180 '     column 10 and print AAA
185 '
190 PRINT E1$"$)";          ' * THIS semicolon is necessary!
195 '
200 PRINT "AAA"
205 '
210 GOSUB 10000             ' * Eliminates redundant typing of delay loop!
215 '
220 '   Now show same operation on line 6,
230 '     column 10 by adding AAA on same line
240 '     as position control characters.
245 '
250 PRINT E1$"%)AAA"        ' * Note: NO semicolon used here.
255 '
260 GOSUB 10000
270 '   Go to line 12 column 4,
280 '      enter Graphics Mode and print
290 '      some symbols
300 PRINT E1$"+#";          ' Line 12 column 4 directive
310 PRINT E7$;              ' Enter Graphics Mode
320 PRINT "aaabbbwwwxxxyyy"    ' Graphic Symbols
330 GOSUB 10000
340 '   Go to line 13, do same with
350 '      single line of instructions
360 PRINT E1$",#"E7$"aaabbbwwwxxxyyy"
370 GOSUB 10000
380 '     REMEMBER:
390 '      Exit from Graphics Mode!
400 PRINT E8$               ' Exit Graphics Mode
410 '  Graphics may also be used
420 '      in Reverse Video, thus:
430 PRINT E1$"-#"E7$E9$"aaabbbwwwxxxyyy"
440 '  Now exit Graphics and
450 '      Reverse Video:
460 PRINT E8$F1$
470 GOSUB 10000
480 '  Before ending,
490 '     1 - Turn cursor back on
500 '     2 - Exit Hold-Screen Mode
510 PRINT E3$E6$
520 PRINT :PRINT"End at 520":PRINT:END  ' I like this better than just OK!
9999 '  *Work and redundancy-eliminating time delay subroutine:
10000 FOR X = 1 TO 5000:NEXT X:RETURN
```

Getting Started With Assembly Language

(or Wow! A Real Program!)

Pat Swayne
Software Engineer



Now that the HUG conference is over, and things have settled down a bit around here, I guess it's time to get back to work on learning assembly language. If you are new to HUG and also new to assembly language, you might want to go back to the April issue (#39) and start at the beginning of this series. As for the rest of you, I think that you are far enough along that I can use a real program for this month's example, and not a special tutorial program. So, roll up your sleeves, warm up your computers, and get ready for...

Part VI — Reading Disk Files in HDOS

When you want to get information from a disk file to your computer's memory or somewhere else, there are two basic ways you can do it in either HDOS or CP/M. The first, and easiest way is to deal with the file in the minimum segments of information that the operating system can handle at a time. In HDOS, these segments are called "sectors", and are the same size as the physical sectors on any HDOS disk, 256 bytes. In CP/M, the segments are called "records", and at one time were also the size of the disk sectors, which was originally 128 bytes. These days, CP/M sectors come in many sizes, but the operating system still logically sees 128 byte sectors, and so file records are sometimes called "Logical Sectors".

The other way to read a file is to read the whole thing in one "gulp", or at least as much as will fit in memory. This is called a "block read". In HDOS, it is only a little harder to deal with than reading individual sectors, but in CP/M, a bit more is involved, since the system itself is only capable of reading one record at a time. In this series, I will discuss reading files by sectors or records first in both operating systems, and then discuss block file operations. In this article, I will cover reading a file by sectors in HDOS.

The example program VIEW in the listing accompanying this article is from HUG disk no. 885-1120. It works like the TYPE command in HDOS except that it lists the file a little slower on the screen because it is doing the operation one sector at a time, whereas TYPE (via the PIP program) uses a block read to get the whole file and then start listing it. The following Microsoft BASIC program is an approximation of how the example program works.

```
10 LINE INPUT "ENTER FILE NAME: ";N$
20 OPEN "I",1,N$
30 FOR I=1 TO 2 STEP 0: REM  INFINITE LOOP
40 IF EOF(1) THEN 80: REM  END OF FILE
50 LINE INPUT #1,L$:REM  INPUT A LINE
60 PRINT L$: REM  PRINT THE LINE
70 NEXT I
80 CLOSE #1
```

This program is not exactly like the assembly language program because it reads and displays the file one line at a time, while the assembly program reads and displays it one sector at a time. Each sector may contain several lines or only part of one line. Another difference you may have noticed is that the assembly program does not prompt the user for a file name, because it gets it from the command line. This is a technique that we have not discussed before, so I will explain it before getting into the disk operations.

Both HDOS and CP/M have a way of passing information from the command line to the program that is being run. In either system, if after you type the program name at the prompt, you enter at least one space and some additional characters, those characters are made available to the program. In HDOS, they are placed on the stack, and the stack pointer is positioned before them, so that it points to the string of characters. If a program is expecting information from the command line, all it has to do is see if the stack has been moved from its default location (42200 split octal, or 2280 hex). So, in the example program, the first thing that is done is to get the value of the stack pointer by clearing the HL register pair to zero and adding the stack pointer to it. Then we see if the low byte of the result is 200 octal. If it is, we know that no information was entered after the program name, and a message is printed explaining how to use the program.

If the stack pointer is not at its default location, the program knows that information was typed after the file name. The space or spaces between the program name and the command line information are also placed on the stack, so the program skips over them. Now, the HL register points to the name of the file that is to be processed by our VIEW program. One of the nice things about HDOS is that it can process a file name at the assembly level just as it is typed, while in CP/M the information must be processed into what is called a "File Control Block", as we will see next month. So once we get the HL register pointing to the file name, we are ready to work with it.

As I have pointed out before, HDOS handles all communications with the "outside world" in about the same way, whether it is with a printer, the screen, or a disk. The technique for opening a disk file is the same as for opening the printer driver, as discussed in the June (REMark #41) installment of this series. By way of review, when you open a file, the HL register pair must point to the file name descriptor. The DE register pair must point to a "default block descriptor" which describes the default device (in this case, a disk drive) in case it is not specified in the file name, and the default extension of the name if there is none. For example, if HL points to "README", and DE points to "SY0DOC", HDOS will try to open a file on drive SY0: called README.DOC.

With the HL and DE registers set up, and the number of the channel you want to use for the file in the A register (just about like the channel numbers you use when you open a file in BASIC), the file is ready to be opened for reading with the .OPENR system call. If the carry flag is set after the call, we know the operation was not successful, which usually means that you tried to open a non-existant file.

After the file is opened, the program sets up an "abort" exit to allow the user to quit early before viewing all of a file. This is done using the system call .CTLC, which can set up an interrupt driven vector that will be jumped to if the user types Control-A, -B, or -C. In the VIEW program, we want to use Control-C, so 3 (the value of Control-C) is placed in the A register pair before the call to .CTLC. The HL register pair points to the routine that will be executed if Control-C is typed. Once you set up a vector this way, you can clear it by calling .CTLC again with the control letter in the A register and zero in the HL register.

After the exit vector is set up, the program enters a loop where it reads a sector and then prints the information on the screen. When you read from a file, you must point the DE register to the buffer that is to contain the information read. The BC register pair contains the number of bytes to be read, which in the case of a disk file must be a multiple of 256 bytes (the sector size). In this program, we are reading only one sector at a time, so we put 256 in BC. Note that when you have 256 in BC what you really have is 1 in B and 0 in C. So instead of saying that the number of bytes to be read is in BC, we can say that the number of sectors to be read is in B, with zero in C.

After each read operation, the carry flag is checked. If it is set, the error code (which is always returned in the A register) is checked to see if the error is "End of File" (EC.EOF) or something else. If it is the end, we just quit. Otherwise, an error message is printed. If there is no error, a routine is called that simply prints all 256 characters in the sector buffer. In HDOS, there is no need to check for an end of file character. If the file does not use all of the last sector, the rest of it will be filled with zeros, and there is no harm in sending zeros to the screen.

The program ends with code to process errors, which we have covered before, and the data area containing the default block descriptor and the sector buffer. Next month, I will present a CP/M version of this program. There are some interesting differences, so don't miss it!

```
*       VIEW -- VIEW FILES ON SCREEN
*
*       TO USE THIS PROGRAM, ENTER
*
*       >VIEW dev:FILENAME.EXT
*
*       THE FILE WILL BE LISTED ON THE SCREEN.
*
*       BY  P. SWAYNE, HUG  18-JUN-82

*       HDOS SYSTEM CALLS, ETC.

.EXIT   EQU     0
.SCIN   EQU     1
.SCOUT  EQU     2
.READ   EQU     4
.CTLC   EQU     41Q
.OPENR  EQU     42Q
.CLOSE  EQU     46Q
.ERROR  EQU     57Q
$TYPTX  EQU     31136A          TYPE TEXT FUNCTION (ROM)
EC.EOF  EQU     1               END OF FILE FLAG

*       MAIN PROGRAM

START   LXI     H,0
        DAD     SP              LOCATE STACK
        MOV     A,L
        CPI     200Q            HAS IT MOVED?
        JNZ     GOTFILE         IF SO, CONTINUE
        CALL    $TYPTX
        DB      10,'The correct use of this program is',10,10
        DB      '>dev:VIEW dev:FILENAME.EXT',10,10
        DB      'where dev: is a drive designation '
        DB      '(SY1:, DK0:, etc.),',10
        DB      'and FILENAME.EXT is the file to be VIEWed.'
        DB      10+80H
        XRA     A
        SCALL   .EXIT           EXIT TO HDOS
GOTFILE MOV     A,M             GET A CHARACTER
        CPI     ' '             SPACE?
        INX     H               INCREMENT POINTER
        JZ      GOTFILE         SKIP SPACES
        DCX     H               FIX POINTER
        LXI     D,DEFALT        POINT TO DEFAULTS
        XRA     A               USE CHANNEL 0
        SCALL   .OPENR          OPEN THE FILE
        JC      ERROR           CAN'T DO IT
        LXI     H,END
        MVI     A,3
        SCALL   .CTLC           SET UP CONTROL-C EXIT
RLOOP   LXI     D,BUFFER        PUT FILE HERE
        LXI     B,256           READ 1 SECTOR
        XRA     A               CHANNEL 0
        SCALL   .READ           READ THE SECTOR
        JNC     GOODRD
        CPI     EC.EOF          END OF FILE?
        JNZ     ERROR           NO, MUST BE ERROR
        JMP     END             YES, FOUND END
GOODRD  CALL    TYPBUF          TYPE CONTENTS OF BUFFER
        JMP     RLOOP           READ ANOTHER SECTOR
END     XRA     A
        SCALL   .CLOSE          CLOSE THE FILE
        XRA     A
        SCALL   .EXIT           LEAVE
TYPBUF  LXI     H,BUFFER        POINT TO BUFFER
        MVI     B,0             SET UP COUNTER
TYLOOP  MOV     A,M             GET A CHARACTER
        SCALL   .SCOUT          PRINT IT ON SCREEN
        INX     H               INCREMENT POINTER
        DCR     B               DONE?
        JNZ     TYLOOP
        RET
ERROR   PUSH    PSW             SAVE ERROR CODE
        CALL    $TYPTX
        DB      10,'Error --',' '+80H
        POP     PSW
        MVI     H,7             ASCII BELL
        SCALL   .ERROR
        JMP     END

*       DATA AREA

DEFALT  DB      'SY0DOC'
BUFFER  EQU     *
        END     START
```

Changing your address?

Let us know. We don't want you to miss a single issue of **REMark**, send your change of address to:

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

In a Heartbeat...

- Use all existing Heath/Zenith H89/Z90 software and hardware products.
- Benefit from CP/M®, CP/M Plus®, MP/M II® or HDOS operating systems, providing *incomparable* application software resources.
- Expand your computing power with 128K byte RAM; up to 256K bytes optional (with parity checking).
- Electronic "Disk Emulator" eliminates the need for slow, repetitive floppy disk access operations.
- Multiply mass storage with built-in floppy and large capacity fixed disk drives.
- There is room for growth with five expansion slots and generous power reserves.

The Heartbeat offers advanced features not found on the standard Heath/Zenith computer such as high speed operation, real-time clock/calendar, two RS-232 serial ports, five peripheral expansion slots and provisions for optional AM9511 Arithmetic Processor. The Heartbeat may be used with most popular video terminals including the Zenith Z19, Z29 and ZT-10/11 for full Heath/Zenith software compatibility. The Heartbeat cabinet design provides for inclusion of hard and/or floppy disk drives as well as other desired peripheral interfaces and is color-coordinated for use with the Zenith Z29 and ZT-10/11 video terminals.

Contact your local Heathkit Electronic Center, or DG Electronic Developments for further information.

D·G ELECTRONIC DEVELOPMENTS CO.

700 SOUTH ARMSTRONG
DENISON, TEXAS 75020
(214) 465-7805

HUG NEW PRODUCTS

885-1130 HDOS Star Battle: The Motion Program $20.00

Introduction: STAR BATTLE is a game which you fight against alien ships. The player must solve math problems in order to fire at an enemy or move to another quadrant.

Requirements: This program requires the HDOS operating system on an H8/H17/H19 or H/Z89 with 48K of memory. Only one disk drive is required.

The program is written in Tiny PASCAL and the source code is included. The executable file is included. It is not necessary to have Tiny PASCAL to run this program.

Note: The H19 (H/Z89) terminal is required for the graphics and use of the special function keys.

The following files are included on the HUG P/N 885-1130 HDOS Star Battle games disk:

```
README   .DOC
STARBATT .ABS
MATHTREK.PAS
```

Author: Paul J. Dalton

Program Content: STAR BATTLE uses the special function keys (f1 through f4) to enter commands that aid in fighting and evading the enemy ships. The options are as follows:

f1 — Warp out. This command will move you to another quadrant. Before you can warp out, you must solve a math problem that sets in your course. If you get it wrong, the enemy can take a shot at you.

f2 — Refuel from base. If you happen across a friendly starbase, you can refuel and repair damages.

f3 — Shoot Phasers. This option will shoot at the enemy ship causing damage to the ship.

f4 — Shoot Torpedoes. This command also causes damage, but is most useful for the kill. Before a torpedo can be fired, you must solve a math problem.

The math problems that must be solved in order to complete a command could be addition, subtraction, or multiplication problems.

Comments: This game interacts with the user by requiring a solution to math problems to continue.

Product Update
885-8012[37] CP/M
MAPLE (Modem Application Effector)

This is an announcement for the release of MAPLE 2.1.0 for CP/M on the H8/H19/H17, H/Z89, or Z90 computer systems. The following is a list of some of the new features added to MAPLE 2.0.6:

1) Besides the ASCII and APL keyboard to screen character translations, a TAPE and HEX mode can be set.

2) Both the file SEND and COPY functions have 'text' and 'absolute' settings, in addition to 'wards' mode and the 'line' and 'block' modes of the SEND function.

3) Modem handshaking can now be controlled with the 'Set' mode. Handshaking options are 'none', 'xon/xoff', 'ack/nak', 'DSR' or 'CTS'. Thus, direct computer to computer ties as well as remote connections need not suffer data loss due to high baud rate.

4) The COPYPAD buffer and disk files can be displayed or reviewed page by page or line by line.

5) The modem line signal is constantly monitored. The 25th line and a ring of the bell indicate a change.

6) The screen transfer key (p.page) allows transfers to disk or directly to the remote, up to the present cursor position.

7) The modem can be disengaged, so that local screen activity will not be sent to the remote.

8) Under the Ward Christensen file transfers, you may elect to display transferred characters or not.

9) Remote control of MAPLE is now available, so that a remote system can send commands which will initiate file transfers. A password is required. Remote controlled exit to the CP/M operating system is allowed following a correct password. The remote system will then have remote terminal capabilities for operating the local computer.

For an update to this version of MAPLE, send the original HUG disk P/N 885-8012 or P/N 885-8012-37 and $10.00 to HUG, Hilltop Road, St. Joseph, MI 49085, in care of Nancy Strunk.

885-8022 HDOS SHAPES $16.00

Introduction: SHAPES is an educational program for preschool children, ages two to five. The program is designed to teach basic shape recognition, while at the same time being fun for the child to play. It can also be used to teach the concepts of up, down, left, and right. This program can serve to introduce the very young to the world of computers.

Requirements: This program requires the HDOS operating system version 2.0 on an H8/H17/H19 or H/Z89, Z90 computer with a minimum 16K of memory. Only one disk drive is required.

The program is written in 8080 assembly language and the source code is included.

Note: The H/Z19 terminal graphics are used.

The following files are included on the HUG P/N 885-8022 HDOS SHAPES disk:

SHAPES	.ABS	INSTR	.ACM
SHAPES	.ASM	FORM	.ACM
REFN	.ACM	BLANK	.ACM
INTROD	.ACM	GROUP	.ACM
MENU	.ACM		

Author: Charles Stout, Jr.

Program Content: The SHAPES educational program for preschool children has six different shapes (or figures) arranged together to form a group at the top of the screen. Each of the six shapes is displayed one at a time at the bottom of the screen. The child must move the shape to its correct position by use of the arrow keys of the keypad.

The program has eight selections from the menu drive program. These include versions of Beginner, Intermediate, Advanced, and Random. The child can play a particular level continuously or return to the menu and choose another level.

Comments: This is a very clean program. It would have been nice to combine this KID'S DEVELOPMENT program with similar type programs, but to this date there are none.

885-8023-37 CP/M-85 MAPLE (Modem Appl. Effector) $35.00

Introduction: The popular modem package, MAPLE, is now available under CP/M-85 for the H/Z100 computers. MAPLE is a program designed to allow the H/Z100 computer to communicate effectively with another computer, over the telephone or by direct connection.

Requirements: This program requires the CP/M-85 operating system on an H/Z100 series computer. A printer device is not required but is recommended for producing a hardcopy of anything from the online time.

The user is required to have a modem capable of connecting to the DTE connector on the back of the H/Z100 for telecommunicating with another computer. The modem can be 300 or 1200 baud. (MAPLE is capable of any number of selectable baud rates for direct connect.)

The following program is included on the HUG P/N 885-8023-37 CP/M-85 MAPLE disk:

MAPLE.COM

The instructions are included in a manual.

Author: Dr. William C. Parke

Program Content: MAPLE is a modem communications package, which can be used to connect an H/Z100 computer to another microcomputer or to a time-share host system, e.g. CompuServe, via the telephone lines.

MAPLE, when executed, enters the communications mode, with full file send and receive control. The mode and options are displayed on the 25th line of the terminal and are invoked by depressing the appropriate function key.

The following are the options displayed on the 25th line while in communication mode:

f0 — toggle (on and off) outgoing communication

f1 — type an ASCII file to the screen (the file will NOT be sent to the remote system)

f2 — this is the COPY function, which allows the download of files or data from a remote system; MAPLE has three COPY selections:

text - copy ASCII characters
abs - copy full eight bit words (absolute files)
wards - copy using Ward Christensen protocol

All characters will be saved in a COPYPAD area.

f3 — redisplay any text in the COPYPAD

f4 — send text in COPYPAD to printer

f5 — store to disk the text in the COPYPAD

f6 — clear contents in COPYPAD

f7 — send file to remote

f8 — exit to CP/M-85

f9 — set user options

KEYS - set keyboard setting; ASCII, APL, TAPE, and HEX
CPY - set COPY options; text, abs, wards
SND - set SEND options; e.g. line, block, wards, abs
PROMPT - set 'Line' mode prompt
PRINT - select current output device
DRIVE - select default disk drive
PARITY - select parity check bit
BAUD - set baud rate
WORD - set number of bits per byte and stop bits
ECHO - toggle terminal echo

BREAK — send hard break code to remote.

These are the main options available to the user. There are many other features of MAPLE that are helpful, e.g. the use of the keypad to send predefined messages or instructions to the remote.

MAPLE uses control characters and escape sequences to do a number of functions. A CTRL-P entered while on-line will toggle the printer output. An ESC CTRL-P will send the control-P to the remote.

Comments: MAPLE is one of the most complete modem packages to allow your computer to communicate with another computer.

885-1127 HDOS Soft Sector Support Package $30.00

Introduction: The HUG Soft Sectored Disk Support Package is a collection of programs built around a modified version of the original DKH37 device driver for soft sector 5.25 inch (H/Z37 type) disks provided by Heath Co. on the HDOS update (HOS-5-UP). The Package includes the device driver and programs to test, modify, and duplicate disks while using the driver.

Requirements: This package requires the HDOS operating system version 2.0 on an H/Z89 or Z90 computer with the Z89-37 disk controller, or an H8 computer with the WH8-37 controller.

This package is a two disk set with printed documentation. Disk A contains all of the executable files, and Disk B contains all of the source files except those on Disk A.

Disk A

README	.DOC
H37	.DVD
H37P6	.DVD
H37P9	.DVD
SETUNIT	.ABS
SETUNIT	.ASM
INITAUTO	.ABS
PRINIT	.ABS
SDUP37	.ABS
SDUP37	.ASM
SDUMP	.ABS
TESTCON	.ABS
TESTH37	.ABS
COMBINE	.ABS
LABEL	.ABS

Disk B

README	.DOC
HUG37	.ASM
INIT37	.ASM
H37LIB	.ACM
H37DEF	.ACM
HROM	.ACM
SDUMP	.ASM
COMBINE	.ASM
LABEL	.ASM

Authors:

H37.DVD, SDUP37, SETUNIT, COMBINE, SDUMP, LABEL — Patrick Swayne, HUG.

INITAUTO, PRINIT, TESTCON, TESTH37 — Modifications of Heath/Zenith software. INITAUTO modifications by Dean Gibson, Ultimeth Corp.

Program Content:

H37.DVD — The H37.DVD device driver is a replacement for the standard 5.25 inch soft sector device driver provided by Heath, and offers the following features.

** Faster operation: There is a 77% reduction in time to load large single block files (compared to DKH37.DVD), such as MBASIC.ABS, and when copying such files with PIP. Boot time is also reduced more than 50%. For example, to load MBASIC (from the time you press RETURN until the OK prompt appears) takes about 13 seconds under DKH37, and 3 seconds with H37.DVD (on a fast two sided drive). Boot time on a fast two sided drive goes from 31 seconds to less than 15 seconds.

** Expanded SET options: The step rate can be set for each unit. The head unload delay time can be set to up to 4.5 seconds to prevent "head banging" during rapid disk accesses. The motor on delay time and the number of retry attempts before a hard error can also be set. The SET HELP option shows the current value of each setting.

** New INIT options: The boot step rate can be specified when you run INIT to reduce boot time on fast drives. A fast media check can be performed during INIT so that you do not have to run TEST37 just for that purpose.

** Full HDOS 2.0 compatibility: The patches supplied with HOS-5- UP for HDOS.SYS are not required to use this driver. It will work with standard HDOS 2.0.

** Speed mod compatibility: Versions of the driver are included that are compatible with the programmable 4MHz modification for the H/Z89 presented in REMark issue #34, and the 3MHz H8 modification in issue #38.

SETUNIT — The H37.DVD driver will support a maximum of 4 drives. When you boot, HDOS allocates 256 bytes of space for a table called the GRT for each unit of each disk device driver, whether the drives for those units actually exist or not. The SETUNIT program can be used to reduce the number of drives supported by the driver so that memory space is not wasted. Later, if you add more drives, SETUNIT can be used to adjust the number of units supported again. SETUNIT can also be used with other disk device drivers besides H37.DVD to match the number of units supported to the number of drives.

INITAUTO and PRINIT — These are disk initialization programs that work like the one supplied with HDOS except that INITAUTO produces an auto-boot disk that does not prompt you with the ACTION <BOOT> question when you boot. PRINIT allows you to make your own auto boot patches on one disk and pass them to other disks when you initialize, even if the target disk is of another type. The boot patches must be outside the boot driver area (first two sectors).

SDUP37 — This is a new disk duplication utility that can duplicate any kind of soft sector disk supported by H37.DVD. The source disk can be placed in any drive capable of reading the disk (for example, a 40 track single sided disk in an 80 track double sided drive). The destination must be placed in a write compatible drive.

SDUMP — This is a sector oriented patch utility that lets you alter any HDOS disk by track and sector or any file on the disk.

TESTCON — This program performs a quick check-out of your soft sector controller.

TESTH37 — This program is a modified version of the TEST37 program provided with HOS-5-UP. The Seek test has been modified so that you can enter the seek time you wish the test to start with. This means you do not have to wait for the test to go through the slow seek speeds when you are testing fast drives, such as the H17-4 drive. After the seek test is done, the fastest successful rate is used for all other tests until you use the "U" option to select another drive. If you run a test at the fastest possible rate, it provides a more accurate check of your drive's capability.

LABEL — This is the disk label utility that was originally published in REMark issue #35. It has been modified so that the auxiliary information byte (which contains the number of sectors per track on the disk) is not overwritten by long labels.

COMBINE — This program is required to re-assemble the driver. Instructions are included in the documentation.

Comments: This package is a must for any HDOS user who has a soft sector disk controller and wants to get the best performance from it.

The following four HDOS products are available in soft sectored format beginning this month:

885-1088[-37] MBASIC Graphic Games
885-1066[-37] Utilities Disk X
885-1096[-37] MBASIC Action Games
885-1030[-37] BHBASIC Games Disk

Refer to the HUG Software Catalog for descriptions of the products.

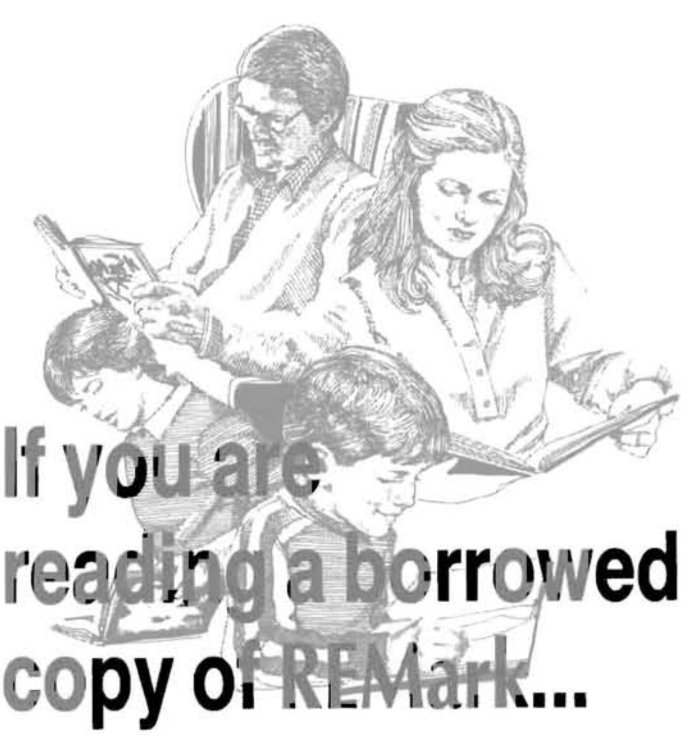
Please remember the "-37" indicates that you want a soft sectored disk. If you do not include the "-37", you will receive hard sectored.

HUG Price List

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

Part Number	Description of Product	Selling Price	REMark Issue
HDOS			
885-1022 [-37]	HUG Editor (ED) Disk H8/H89	$ 20.00	44
885-1029 [-37]	Disk II Games 1 H8/H89	$ 18.00	40
885-1038 [-37]	Wise on Disk H8/H89	$ 18.00	42
885-1042 [-37]	PILOT on Disk H8/H89	$ 19.00	42
885-1044 [-37]	Utilities Disk VI	$ 18.00	43
885-1055 [-37]	MBASIC Inventory Disk H8/H89	$ 30.00	44
885-1060 [-37]	Disk VII H8/H89	$ 18.00	40
885-1062 [-37]	Disk VIII H8/H89 (2 Disks)	$ 25.00	40
885-1064 [-37]	Disk IX H8/H89	$ 18.00	42
885-1067 [-37]	Disk XI H8/H89 Games	$ 18.00	40
885-1071 [-37]	MBASIC SmBusPk H8/H19/H89	$ 75.00	41
885-1078 [-37]	HDOS Z80 Assembler	$ 25.00	42
885-1079 [-37]	Page Editor PAGED	$ 25.00	43
885-1083 [-37]	Disk XVI Misc. Utilities	$ 20.00	43
885-1086 [-37]	Tiny HDOS Pascal H8/H89	$ 20.00	40
885-1089 [-37]	Disk XVIII Misc H8/H89	$ 20.00	41
885-1090 [-37]	Disk XIX Utilities H8/H89	$ 20.00	41
885-1092 [-37[	Relocating Debug Tool H8/H89	$ 30.00	44
885-1093 [-37]	Dungeons and Dragons H8/H89	$ 20.00	43
885-1097 [-37]	MBASIC Quiz Disk H8/H89	$ 20.00	41
885-1107 [-37]	HDOS Data Base System H8/H89	$ 30.00	42
885-1108 [-37]	HDOS MBASIC Data Base System	$ 30.00	41
885-1111 [-37]	HDOS MBASIC Games H8/H89	$ 20.00	44
885-1112 [-37]	Graphics Games Disk	$ 20.00	43
885-1113 [-37]	HDOS Fast Action Games H8/H89	$ 20.00	44
885-1121	Hard Sectored Support Package	$ 30.00	37
885-1122	MicroNET Connection	$ 16.00	37
885-1123	XMET Robot Cross Assembler	$ 20.00	40
885-1124	HUGMAN & Movie Animation Pkg	$ 20.00	41
885-1125	MAZEMADNESS	$ 20.00	41
885-1126	HDOS UTILITIES by PS:	$ 20.00	42
885-8015	TEXTSET Formatter	$ 30.00	42
885-8016	Morse Code Transceiver Ver 2.0	$ 20.00	41
885-8017	HDOS Programmers Helper	$ 16.00	42
885-8021	HDOS Student's Statistics Pkg	$ 20.00	44
CP/M			
885-1211 [-37]	Sea Battle	$ 20.00	36
885-1222 [-37]	Adventure	$ 10.00	36
885-1223 [-37]	HRUN HDOS Emulator	$ 40.00	37
885-1224 [-37]	MicroNET Connection	$ 16.00	37
885-1225 [-37]	Disk Dump and Edit Utility (DDEU)	$ 30.00	38
885-1226 [-37]	CP/M Utilities by PS:	$ 20.00	38
885-1227 [-37]	CP/M Cassino Graphic Games	$ 20.00	38
885-1228 [-37]	CP/M Fast Action Games	$ 20.00	39
885-1229 [-37]	XMET Robot Cross Assembler	$ 20.00	40
885-1230 [-37]	CP/M Function Key Mapper	$ 20.00	42
885-1231 [-37]	Cross Reference Utilities for MBASIC	$20 .00	43
885-3003 [-37]	ZTERM Modem Package	$ 20.00	36
885-8012 [-37]	Modem Appl. Effector (MAPLE)	$ 35.00	36
885-8018 [-37]	FAST EDDY Text Editor and BIG EDDY	$ 20.00	43
885-8019 [-37]	DOCUMAT Formatter and DOCULIST	$ 20.00	43
885-8020 [-37]	CP/M RF Computer Aided Design	$ 30.00	44
ZDOS			
885-3004-37	ZBASIC Graphic Games Disk	$ 20.00	37
885-3005-37	ZDOS ETCHDUMP	$ 20.00	39
MISCELLANEOUS			
885-0004	HUG 3-Ring Binder	$ 5.75	
885-4001	REMark VOLUME 1, Issues 1-13	$ 20.00	
885-4002	REMark VOLUME 2, Issues 14-23	$ 20.00	
885-4003	REMark VOLUME 3, Issues 24-35	$ 20.00	
885-4600	Watzman/HUG ROM	$ 45.00	41

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



If you are reading a borrowed copy of REMark...

maybe now is the time to join the National Heath Users' Group.

● *You will receive a copy of **REMark** filled with new and exciting articles and programs each month.*

● *You will also receive access to the HUG library filled with a large variety of programs.*

And remember, your local HUG is an excellent source of information, support, and comradery. A membership package from the National **H**eath **U**sers' **G**roup contains a list of current local HUG clubs as well as other interesting information

DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES

PRICE BREAKTHROUGH

Super Sale on New Disk Drives

Starting at **$199.95 complete**
with Power Supply and Case!!

Tandon — Siemens — Remex — MPI — Teac — Shugart — Tabor

All of Our Drives are Capable of Single AND Double Density Operation!

RADIO SHACK[1]— HEATH/ZENITH[2]— APPLE[3]/FRANKLIN[4]
IBM/PC-TEXAS INSTRUMENTS[6]& MOST OTHER COMPUTERS

Drive a Hard Bargain!!™
5 M.B. - 12 M.B. with Power Supply Case, Cables & Software
Complete Systems . . . . from $999.95

Dual sided 40 TK or 80 TK drives with Power Supply and Case—Call Toll Free
Diskette Breakthrough — 10 Pack in Library Case — $18.95

TOLL FREE ORDERING GENERAL AND TECHNICAL

1-800-343-8841 1-617-872-9090

Model I/III/IV Drives *(0 1 2 3)*	*starting at* ***$175.00***
Color Computer Drive (1 2 3)	***$Call Toll Free***
Apple/Franklin Compatible Add-On Drives with Case & Cables	$219.95
Apple/Franklin Compatible Drives with Controller	***$259.95***
Model I/III/IV Memory Upgrade	***Call Toll-Free***
Printers — Daisywheel/Dot Matrix	***$Call Toll Free***
Diskettes in Library Cases	$18.95
Cases and Power Supplies — (Single-Dual-**1/2 Height**)	*starting at $49.95*
Printer **Buffers** 8K to 512K	starting at $143.95
Percom Double Density Controller (Model I)	***$Call Toll Free***
Holmes Model I/III Speed-up Mod-VID/80	*starting at* $90.00
Gold Fingered **Edge Card Extenders**	*starting at* $13.00
Cables — Printer/Disk Drive	*starting at $25.00*
DOSPLUS	***$Special Prices***
Repair Services Now Offered — *Fast Turn-a-Round*	***$Call Toll-Free***

Warranty on Disk Drives — 6 Months — *Extended Warranty $ Call Toll Free*

SOFTWARE SUPPORT, INC.

One Edgell Road, Framingham, MA 01701 (617) 872-9090
Hours: Mon. thru Fri. 9:30 am to 5:30 pm (E.S.T.) Sat. 10:00 am to 4:30 pm

Dealer inquiries invited.

TERMS:
M.C./Visa/Amex and personal checks accepted at no extra charge.
C.O.D. please add $3.00.
Shipping: Please call for amount.
Not responsible for typographical errors.

Canada
MICRO R.G.S. INC.
751, CARRE VICTORIA, SUITE 405
MONTREAL, QUEBEC, CANADA, H2Y 2J3
Regular Tel. (514) 845-1534
Canadian Toll Free 800-361-5155

1 ™TANDY CORPORATION
2 ™ZENITH DATA SYSTEMS
3 ™APPLE COMPUTER CORP.
4 ™FRANKLIN COMPUTER, INC.
5 ™IBM CORPORATION
6 ™TEXAS INSTRUMENTS
© Copyright 1983

There Is Always A Catch

(or How To Flip Your LID)

Barry Deer
4202 Carondelet Drive
Dayton, OH 45440

Is there an easier way to open up the lid on your H/Z computer terminal? The answer is yes. Currently, the lid must opened by inserting a screw driver into the slide-latch and pulling the lid up while pulling the screw driver forward. This latch system is tolerable when you seldom access the hardware. But when you need to get in and out often, and when you have printers, disk drives, and (in my case) documentation on both sides of the console, then opening the lid can be a hassle.

In order to make opening of the lid easy, I made a latch release system. When installed, the lid can be opened by pulling two knobs, one on each side of the screen (see Figure 1). The concept is simple. A plastic knob, mounted through the terminal, is suitably connected to the latch release. When the knob is pulled, the latch is released.

Once you understand the concept, you probably can invent your own linkage between the knob and latch. The one described below is the one currently in use on my H-89 and the H8 of several members of our HUG.

First, let me describe the release mechanism. It is shown in Figure 2. An internally threaded spacer (about 1-1/4 inches long) is stuffed at one end with a stiff wire (4 inches long). Using a vise, the spacer is crimped in order to hold the stiff wire in place and also to hold a close fitting washer from sliding off the end (see Figure 1). A medium strength spring is slid over the spacer and a final close fitting washer is placed over the spacer. Two holes are drilled into the terminal front over the spacer, at an approximate location suggested by the template in Figure 3. One hole is drilled to the left and one to the right of the CRT. The template can be cut out and held to the face of the terminal, with the template arrows pointing at the intersection of the keyboard plate and the plate surrounding the CRT and disk. (On the right side, this is just below and to the right of the Heathkit logo.) Before drilling a hole, make sure that the knob you have selected will clear the sides of the terminal lid, and that the mechanism you will use will clear in the back.

The size of the drilled hole should be roughly the outside diameter of the spacer. Also, as the drill is going through the chassis, it is pointed down slightly, as if the tip of the drill will point directly at the lid latch. After locating the holes, make sure there is adequate clearance for the spring and the spacer to clear the CRT and cabinet sides. When drilling, you may wish to hold a small block of wood behind the console to protect the CRT. Observe all safety rules for working around the CRT. After the holes are drilled, the installation is next. Compress the spring on the internally threaded spacer. Insert the non-crimped end through the drilled holes from the back side of the console. Thread the plastic knob into the threaded spacer until the plastic knob cannot turn anymore. At this point, you should be able to pull the knob out and have it travel at least 3/4 of an inch. When the knob is released, the spring should pull the knob against the front panel. Next, bend the wire so that it points to the lid latch (Figure 3). Make a second sharp bend in the end of the wire at the point where it could be stuck through a hole in the sliding lid latch. The location of the bend should be at a point which does not cause tension to be placed on the sliding catch, but will allow the catch to be moved forward enough to release the lid when the knob is pulled. Stick the end of the wire through the hole and crimp it over so that it will stay attached to the sliding lid latch.

The particular link you use can be dictated by the available parts around your work bench. But, we found that the right knobs (well insulated and threaded) which allowed us to mount them quickly, were not readily available. When we did find them, we had to buy in large quantity. Therefore, we went ahead and made a batch of them for our HUG, and put them in a kit with instructions and template for drilling. These kits are available for $7.00 by writing to the above address.

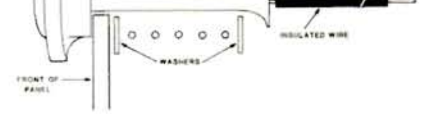


Figure 1. Side view of front panel (cut-away view) showing location of assembled parts.

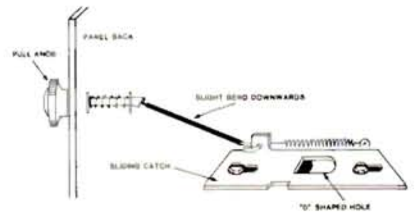


Figure 2. View from right side.

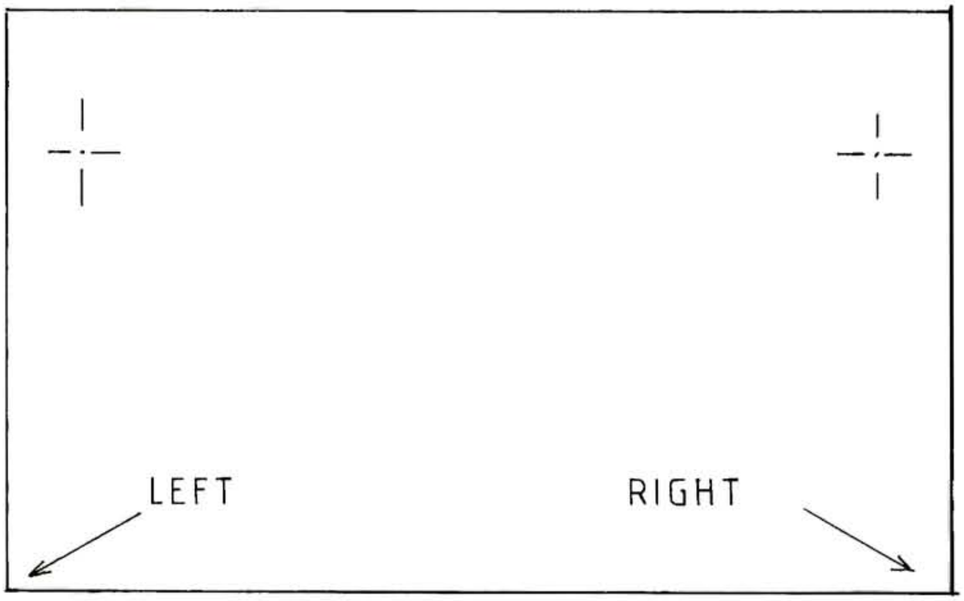
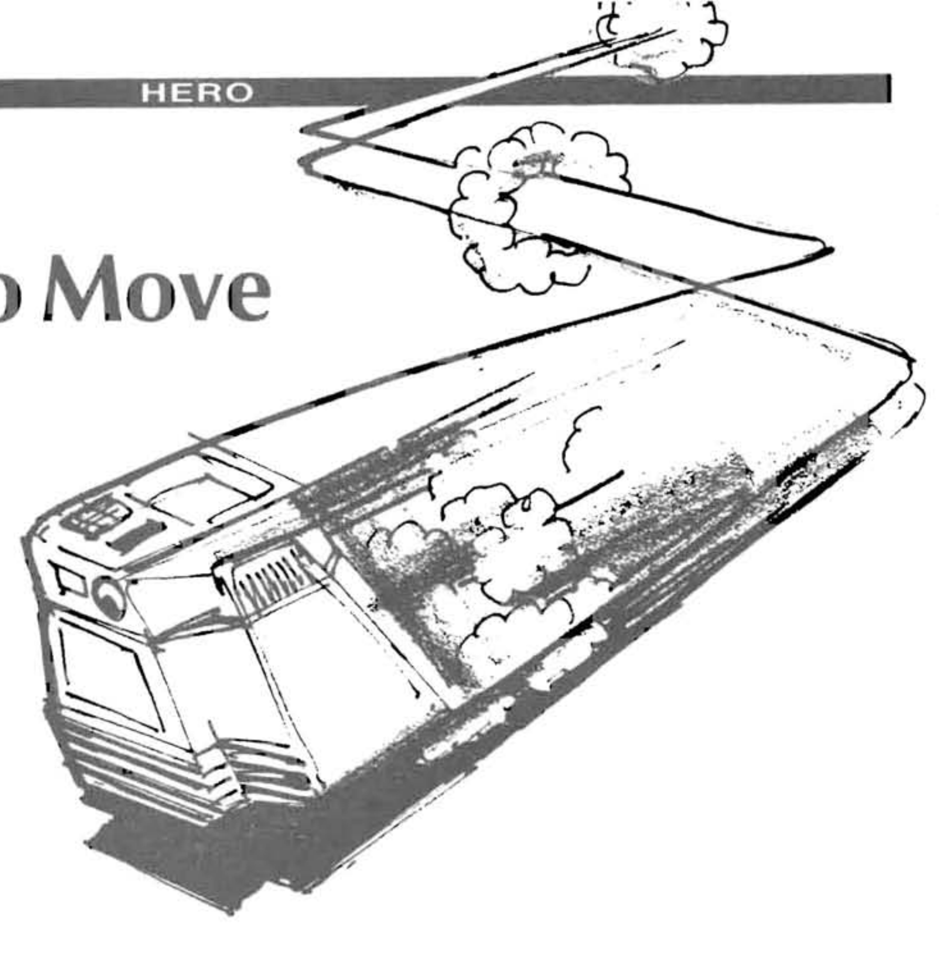


Figure 3. A template which can be used to locate the approximate location of the holes.



Making Hero Move

Mike Curiale
23 Sullivan St.
Charlestown, MA 02129

To me, a robot is a machine capable of determining, plotting, and executing motion. It need not perform any truly useful function as a robot arm must do in a factory environment. Instead, it need only act as a house pet to warm my heart. I have always been fascinated with the idea of having a robot. So the morning I saw the ad for the Heathkit Hero 1 in Computers and Electronics magazine, I ordered it. Weeks later I had the kit, months later it was working. Since Hero 1 comes without software except that dandy robot interpreter ROM, I knew that my efforts would focus on making Hero mobile so that it would conform to my expectation of what a robot should do. As time progressed, it became apparent that much experimentation would be required. The program included with this article is one in a series of programs that I have written and am writing in an attempt to transform Hero into a rambling, walking machine. Admittedly, this is not my latest nor best program, but it is fun to play with and will hopefully inspire you to explore robot motion, too.

Coma Program 3 (CP3), Coma being the name of my Hero robot, was originally hand assembled and entered via the robot's keypad. For purposes of this article, I have reassembled the program with a 6800 cross assembler on the Apple II computer. In the process, I have done very little to modify it. I mention this because I had written most of the program to be executed under control of the robot interpreter. While this would have been fine for a short program, it becomes very clumsy in longer programs. It is much better to write for execution in machine code with switching to interpreter mode only when necessary. This approach, I am told, is recommended in Heath's robot course.

After you observe the robot running CP3 for awhile, the programming logic should become apparent. However, before you enter and run the program, a short description might be of interest. After typing AD0100 to start the program, Hero asks whether you re-

```
                    *
                    *          CP3
                    *
                    *   by Mike Curiale - 1983
                    *
                    *--------------------------------
                    *
                         .in  robot macros 7/31

                         .or  $100
                         .tf  cp3
                    *--------------------------------
                    *
0005-               headpos     .eq  $05
0006-               steerpos    .eq  $06
000A-               drivepos2   .eq  $0a
0010-               rangehit    .eq  $10
0011-               range       .eq  $11
00F6-               counter     .eq  $f6
00F7-               lastdrive   .eq  $f7
00F9-               mindist     .eq  $f9
00FA-               bestsonar   .eq  $fa
00FB-               no.ent      .eq  $fb
00FC-               tblpoint    .eq  $fc   ;& fd
00FE-               motorx      .eq  $fe   ;& ff
                    *
                    *--------------------------------
                    *
0EFC-               random      .eq  $0efc
F65B-               clrdis      .eq  $f65b
F777-               inch        .eq  $f777
F7AD-               outbyt      .eq  $f7ad
F7C8-               outch       .eq  $f7c8
                    *
                    *--------------------------------
                    *
                    *   MAIN PROGRAM LOOP
                    *
                    mpl
0100-                      >swi               ;switch to interpreter
0100- 3F       0000>          .HS   3F
0101- BD 02 49             jsr   runquest
0104- BD 01 4A             jsr   inithead
0107- BD 01 56             jsr   initsteer
010A- 86 50                ldaa  #$50         ;min sonar reading before stopping
010C- 97 F9                staa  mindist
010E- CE 02 2B      .2     ldx   #tblfront    ;head & steer data
```

ally wish to run CP3. You answer yes or no by pressing the appropriate key beneath the display prompt. If you answer yes, the robot turns its head and then steering wheel to the robot's left to initialize the appropriate memory addresses. After this process is completed, the main program loop is entered. The robot turns its head 45 degrees to the right and takes a sonar reading. If the new sonar reading is greater or equal to the last one, the display is updated. The head moves to the forward position, takes a reading, and updates the display if necessary. The front wheel is then turned in the direction indicated in the display and the drive is engaged to make the turn before the drive wheel is centered. If the sonar value was below a specified minimum-stored at address $f9 (mindist)-neither the steering nor driving motions will be done. This is to prevent further movement toward something blocking its path. Instead the robot looks 90 degrees to either side before initiating a tight right-angle turn. Once the robot has decided that it is safe to proceed in a direction, it will continue going in that direction until the sonar indicates an impending collision and causes the drive to abort; again by comparing the sonar reading to the value stored at $f9. The scan routine is then repeated and so on.

Hero voices several comments during the execution of the program. To reduce the frequency at which each comment is spoken, the program uses the pseudo-random number generated by the clock to decide whether a comment will be made. There is a 50-50 chance that the comment will be made.

Practical experience proved that sonar ranging without moving the robot head would not prevent all collisions, especially with items close to the ground, corners of furniture, and doorjams. Each of these collisions caused Hero to become immobile with the drive continually trying to move the robot forward because the sonar ranging said the path was clear. To circumvent this problem, I developed the bump subroutine. It monitors the value at drive address $0a. If the value does not change, a collision is flagged and the robot takes evasive action. The bump subroutine has not been incorporated into all the drive routines, for example, while making a turn. This task is left to you; I recommend doing it. Why haven't I included it? Well, I'm working on an advanced version of this program at this time…

```
0111- B6 03     1460          ldaa  #$3
0113- BD 01 62  1470          jsr   headscan
0116- 96 FA     1480          ldaa  bestsonar
0118- 91 F9     1490          cmpa  mindist
011A- 24 18     1500          bcc   .1           ;best sonar greater so branch
011C- BD 02 BF  1510 .3       jsr   comment2     ;"something in the way"
011F- CE 02 3D  1520          ldx   #tblside     ;90 d left or right
0122- B6 02     1530          ldaa  #$2
0124- BD 01 62  1540          jsr   headscan
0127- BD 01 D1  1550          jsr   steer.
012A- CE 02 37  1560          ldx   #tblstrght ;straighten head and steer
012D- B6 01     1570          ldaa  #$1
012F- BD 01 62  1580          jsr   headscan
0132- 20 DA     1590          bra   .2
0134- BD 01 D1  1600 .1       jsr   steer.
0137- BD 02 F7  1610          jsr   comment3     ;"clear the way"
013A- BD 01 DF  1620          jsr   drive.
013D- 81 FA     1630          cmpa  #$fa         ;bump flag
013F- 27 DB     1640          beq   .3
0141- BD 01 9F  1650          jsr   sonar
0144- 91 F9     1660          cmpa  mindist
0146- 24 D4     1670          bcc   .3           ;false stop
0148- 20 C4     1680          bra   .2
                1690 *
                1700 *--------------------------------
                1710 *
                1720 *   INIT HEAD AND STEER
                1730 *
                1740 inithead
014A- 86 E0     1750          ldaa  #$e0
014C- 97 05     1760          staa  headpos
014E-           1770          >mwai head+slow  ;turn head left completely
014E- C3 C8 00  0000>            .DA    #$C3,HEAD+SLOW
0151- 4F        1780          clra  headpos    ;reset
0152-           1790          >mwai head+slow+$0062
0152- C3 C8 62  0000>            .DA    #$C3,HEAD+SLOW+$0062
0155- 39        1800          rts
                1810 *
                1820 initsteer
0156- 86 B0     1830          ldaa  #$b0
0158- 97 06     1840          staa  steerpos
015A-           1850          >mwai steer+slow ;steer to left completely
015A- C3 E8 00  0000>            .DA    #$C3,STEER+SLOW
015D- 4F        1860          clra  steerpos   ;reset
015E-           1870          >mwai steer+slow+$004a
015E- C3 E8 4A  0000>            .DA    #$C3,STEER+SLOW+$004A
0161- 39        1880          rts
                1890 *
                1900 *--------------------------------
                1910 *
                1920 *   SCAN ROUTINE
                1930 *
                1940 headscan
0162- DF FC     1950          stx   tblpoint
0164- 97 FB     1960          staa  no.ent       ;number of 6 byte entries pointed at
0166- 7F 00 FA  1970          clr   bestsonar
0169- A6 01     1980 .4       ldaa  1,x          ;destination of head
016B- 91 05     1990          cmpa  headpos      ;current position
016D- 25 0A     2000          bcs   .1           ;branch to move head left
016F-           2010          >mcri head+slow+$0003
016F- DC C8 03  0000>            .DA    #$DC,HEAD+SLOW+$0003
0172-           2020          >mcri head+med+$0003
0172- DC D0 03  0000>            .DA    #$DC,HEAD+MED+$0003
0175-           2030          >mwax #$0          ;indexed
0175- E3 00     0000>            .DA    #$E3,#$0
0177- 20 08     2040          bra   .2
0179-           2050 .1       >mcri head+slow+rev+$0003
0179- DC CC 03  0000>            .DA    #$DC,HEAD+SLOW+REV+$0003
017C-           2060          >mcri head+med+rev+$0003
017C- DC D4 03  0000>            .DA    #$DC,HEAD+MED+REV+$0003
017F-           2070          >mwax #$0          ;indexed
017F- E3 00     0000>            .DA    #$E3,#$0
0181- BD 01 9F  2080 .2       jsr   sonar
0184- 91 FA     2090          cmpa  bestsonar
0186- 25 09     2100          bcs   .3           ;branch if not better
0188- 97 FA     2110          staa  bestsonar    ;better reading
018A- BD 01 B1  2120          jsr   enddis       ;showit
018D- DE FC     2130          ldx   tblpoint
018F- DF FE     2140          stx   motorx
0191- 08        2150 .3       inx
0192- 08        2160          inx
0193- 08        2170          inx
```

The robot arm is at the source of an unsolved problem. Being attached to the robot's head, it is likely to strike something when the head is turning causing the robot to loose track of where its head is. Several simple solutions are possible: 1) initialize the head periodically (a time waster), 2) move the arm to the down position so that it is out of the way, or 3) remove the arm. If anyone comes up with a better solution, I would very much like to hear from you.

Hero running CP3 is still far from meeting my definition of a robot. While it does allow the robot to wander about, Hero does so without an apparent goal. It would appear that there is still much that home robots must learn to do.

```
0194- 08        2180        inx
0195- 08        2190        inx
0196- 08        2200        inx
0197- DF FC     2210        stx    tblpoint
0199- 7A 00 FB  2220        dec    no.ent
019C- 26 CB     2230        bne    .4           loop till all positions examined
019E- 39        2240        rts
                2250 *
                2260 *--------------------------------
                2270 *
                2280 *  SONAR READING
                2290 *
                2300 sonar
019F- 7F 00 10  2310        clr    rangehit
01A2- 7F 00 11  2320        clr    range
01A5-           2330        >eur              ;enable sonar
01A5- 45        0000>          .HS    45
01A6-           2340        >pse  $0008       ;stabilize .5 secongs
01A6- BF 00 08  0000>          .DA    #$BF,$0008
01A9- 96 11     2350        ldaa   range
01AB- 26 02     2360        bne    .1         ;reading not zero
01AD- 86 FF     2370        ldaa   #$ff       ;set to maximum
01AF-           2380 .1     >dur              ;sonar off
01AF- 5B        0000>          .HS    5B
01B0- 39        2390        rts
                2400 *
                2410 *--------------------------------
                2420 *
                2430 *  DISPLAY A BYTE
                2440 *
                2450 enddis
01B1-           2460        >norm
01B1- 83        0000>          .HS    83
01B2- 36        2470        psha
01B3- BD F6 5B  2480        jsr    clrdis
01B6- DE FC     2490        ldx    tblpoint
01B8- A6 01     2500        ldaa   1,x        ;head dest.
01BA- BD F7 AD  2510        jsr    outbyt
01BD- 20 05     2520        bra    ctrdis1
                2530 *
                2540 ctrdis
01BF-           2550        >norm             ;leave robot interpreter
01BF- 83        0000>          .HS    83
01C0- 36        2560        psha
01C1- BD F6 5B  2570        jsr    clrdis
                2580 ctrdis1
01C4- 4F        2590        clra
01C5- BD F7 C8  2600        jsr    outch      ;output blank
01C8- BD F7 C8  2610        jsr    outch      ;output blank
01CB- 32        2620        pula
01CC- BD F7 AD  2630        jsr    outbyt     ;output byte
01CF-           2640        >swi              ;robot int.
01CF- 3F        0000>          .HS    3F
01D0- 39        2650        rts
                2660 *
                2670 *--------------------------------
                2680 *
                2690 *  STEERING
                2700 *
                2710 steer.
01D1- DE FE     2720        ldx    motorx     ;set by headscan
01D3- 08        2730        inx
01D4- 08        2740        inx               ;fetches steer instructions
01D5- DF FC     2750        stx    tblpoint
01D7-           2760        >mwax #$0         ;indexed steering
01D7- E3 00     0000>          .DA    #$E3,#$0
01D9-           2770        >mwax #$2         ;indexed 2,x drive instructions
01D9- E3 02     0000>          .DA    #$E3,#$2
01DB-           2780        >mwai steer+slow+$0049 ;steer forward
01DB- C3 EB 49  0000>          .DA    #$C3,STEER+SLOW+$0049
```

```
                3420 tblstrght
0237- D8 62     3430        .hs    d862       ;head staight forward
0239- F0 49     3440        .hs    f049       ;steer same
023B- 08 00     3450        .hs    0800       ;drive turn
                3460 *
                3470 tblside
023D- D8 9A     3480        .hs    d89a       ;head right 90 d
023F- F0 92     3490        .hs    f092       ;steer same
0241- 08 16     3500        .hs    0816       ;drive turn
                3510 *
0243- D8 2A     3520        .hs    d82a       ;head left 90 d
0245- F0 00     3530        .hs    f000       ;steer same
0247- 08 16     3540        .hs    0816       ;drive turn
                3550 *
                3560 *--------------------------------
                3570 *
                3580 * RUN THIS PROGRAM?
                3590 *
                3600 runquest
0249-           3610        >spw  runquest1
0249- 72 02 7B  0000>          .DA    #$72,RUNQUEST1
024C-           3620        >norm
024C- 83        0000>          .HS    83
024D- BD F6 5B  3630        jsr    clrdis
0250- 4F        3640        clra
0251- BD F7 C8  3650        jsr    outch      ;blank
0254- 86 3B     3660        ldaa   #$3b       ;y character
0256- BD F7 C8  3670        jsr    outch
0259- 4F        3680        clra
025A- BD F7 C8  3690        jsr    outch
025D- BD F7 C8  3700        jsr    outch
0260- 86 15     3710        ldaa   #$15       ;n character
0262- BD F7 C8  3720        jsr    outch
0265- BD F7 77  3730 .2     jsr    inch       ;await keypres
0268- 81 0F     3740        cmpa   #$0f       ;f pressed?
026A- 27 09     3750        beq    .1
026C- 81 0D     3760        cmpa   #$0d       ;d pressed?
026E- 26 F5     3770        bne    .2
0270-           3780        >swi
0270- 3F        0000>          .HS    3F
0271-           3790        >spw  ok          ;speak "OK"
0271- 72 02 A8  0000>          .DA    #$72,OK
0274- 39        3800        rts
0275-           3810 .1     >swi
0275- 3F        0000>          .HS    3F
0276-           3820        >spw  later       ;speak "maybe later"
0276- 72 02 B0  0000>          .DA    #$72,LATER
0279-           3830        >rex              ;return to "ready"
0279- 3A        0000>          .HS    3A
027A- 39        3840        rts
                3850 *
                3860 *--------------------------------
                3870 *
                3880 * "shall i run program cp3?"
                3890 *
                3900 runquest1
027B- 03 03     3910        .hs    0303
027D- 1B 01 18
0280- 26 2D 3E  3920        .hs    1b0118262d3e
0283- 11 05 18
0286- 03        3930        .hs    11051803
0287- 15 00 09
028A- 29 03     3940        .hs    1500092903
028C- 2B 32 31
028F- 0D 03     3950        .hs    2b32310d03
0291- 25 2B 35
0294- 1C 2B 2F
0297- 00 0C 03  3960        .hs    252b351c2b2f000c03
029A- 1F 3C 29
029D- 03        3970        .hs    1f3c2903
```

```
01DE- 39          2790         rts
                  2800 *
                  2810 *-------------------------------
                  2820 *
                  2830 *    MAIN DRIVE
                  2840 *
                  2850 drive.
01DF- 86 08       2860         ldaa   /drive+slow
01E1- 97 FC       2870         staa   tblpoint
                  2880 cont
01E3- BD 01 9F    2890         jsr    sonar
01E6- 91 F9       2900         cmpa   mindist
01E8- 25 0F       2910         bcs    .1          ;below min. allowed
01EA- 97 FD       2920 .2      staa   tblpoint+1  ;dist to travel
01EC- BD 01 BF    2930         jsr    ctrdis      ;show distance
01EF- BD 02 06    2940         jsr    bump
01F2-             2950         >bbb   #cont
01F2- 1C EF       0000>          .DA    #$1C,#CONT-:1
                  0000> :1
01F4-             2960         >mcae tblpoint   ;move according to data
01F4- FC 00 FC    0000>          .DA    #$FC,TBLPOINT                  at tblpoint
01F7- 20 EA       2970         bra    cont
01F9- BD 01 9F    2980 .1      jsr    sonar       ;recheck sonar
01FC- 91 F9       2990         cmpa   mindist
01FE- 24 EA       3000         bcc    .2          ;false stop
0200-             3010         >adm               ;abort drive
0200- 02          0000>          .HS    02
0201- BD 01 BF    3020         jsr    ctrdis      ;show last dist.
0204- 4F          3030         clra               ;leave without error
0205- 39          3040         rts
                  3050 *-------------------------------
                  3060 *
                  3070 *   COLLISION DETECTION
                  3080 *
                  3090 bump
0206- 96 F6       3100         ldaa   counter
0208- 84 02       3110         anda   #$02
020A- 27 04       3120         beq    .1
020C- 7C 00 F6    3130         inc    counter
020F- 39          3140         rts
0210- 7F 00 F6    3150 .1      clr    counter
0213- 96 0A       3160         ldaa   drivepos2
0215- 91 F7       3170         cmpa   lastdrive
0217- 27 03       3180         beq    .2
0219- 97 F7       3190         staa   lastdrive
021B- 39          3200         rts
021C-             3210 .2      >adm
021C- 02          0000>          .HS    02
021D- 7F 00 F7    3220         clr    lastdrive
0220-             3230         >pse   $0010
0220- 8F 00 10    0000>          .DA    #$8F,$0010
0223-             3240         >mwai drive+slow+rev+$0010
0223- C3 0C 10    0000>          .DA    #$C3,DRIVE+SLOW+REV+$0010
0226- 32          3250         pula
0227- 32          3260         pula               ;upon rts return to mpl
0228- 86 FA       3270         ldaa   #$fa        ;collision error flag
022A- 39          3280         rts
                  3290 *-------------------------------
                  3300 *
                  3310 *   DATA
                  3320 *
                  3330 tblfront
022B- D8 7F       3340         .hs    d87f        ;head right 45 d
022D- F0 6E       3350         .hs    f06e        ;steer same
022F- 08 10       3360         .hs    0810        ;drive turn
                  3370 *
0231- D8 46       3380         .hs    d846        ;head left 45 d
0233- F0 24       3390         .hs    f024        ;steer same
0235- 08 10       3400         .hs    0810        ;drive turn
                  3410 *
```

```
029E- 25 3C 29
02A1- 03          3980         .hs    253c2903
02A2- 79 6B 7C
02A5- 69 3F       3990         .hs    796b7c693f
02A7- FF          4000         .hs    ff
                  4010 *
                  4020 *-------------------------------
                  4030 *
                  4040 * "ok"
                  4050 *
                  4060 ok
02A8- 35 77 03    4070         .hs    357703
02AB- 19 20 29
02AE- 3F          4080         .hs    1920293f
02AF- FF          4090         .hs    ff
                  4100 *
                  4110 *-------------------------------
                  4120 *
                  4130 * "maybe later"
                  4140 *
```

QUALITY FIRST

HEATH/ZENITH

HEATH/ZENITH OWNERS We now offer you 27 well documented easy to use software products developed over 3 years.

GAMES Choose arcade style graphics, adventure type, or intellectual (like new WORD WIGGLE). Many play on any system.

SPREADSHEET Full featured - now only $49.95

WORD PROCESSING 50,000 word proofreader, file difference detector and more.

LANGUAGES & UTILITIES Top-rated "C" Compiler (fastest in BYTE benchmark), Z80 macro assembler. Plus five utilities and more.

AVAILABLE through Heathkit stores and other Heath/Zenith dealers near you.

FREE CATALOG Call or write today for detailed 16 page catalog.

The Software Toolworks

(213) 986-4885
15233 Ventura Blvd.
Suite 1118, Sherman Oaks, CA 91403

Dealer inquiries invited.

FREE CP/M-Plus™ from MAGNOLIA MICROSYSTEMS

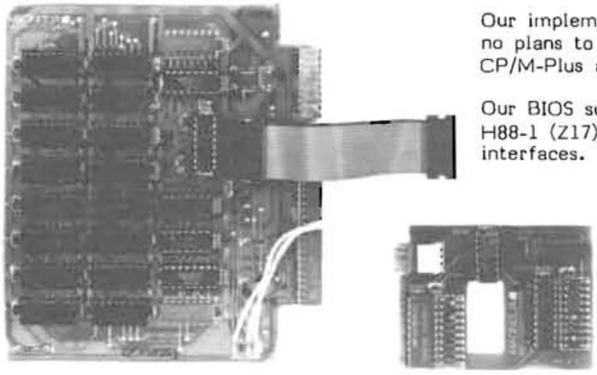
With purchase of 128K RAM Board at $595

Digital Research's list price for CP/M-Plus is $350, but it will be included AT NO EXTRA CHARGE with purchase of our 77318 128K RAM board at its regular price of $595 by mentioning this ad when ordering.

CP/M-Plus^tm support is available for Z89 and Z90 computers with our 128K RAM board and either the Z89-37 (Z90) or our own 77316 Double Density disk controller. Over 700K of files are included, so it will not be distributed on Hard Sectored media.

Using banked RAM, disk performance is enhanced through Hashed Directory tables, Directory Buffers, and LRU Data Buffers.

New utilities and features include a 'Help' command; optional Password protection and Time and Date Stamping of files; Console Redirection to or from disk files; and many others.



Our implementation of CP/M-Plus REQUIRES the use of our 128K RAM board. We have no plans to implement an un-banked memory version, most of the advantages of CP/M-Plus are neither available (nor practical) in an un-banked version.

Our BIOS supports both our 77316 and Zenith's Z89-37 Double Density controllers, Heath's H88-1 (Z17) Single Density controller, and our 77314 CORVUS and 77320 SASI-bus interfaces.

We are including the SOURCE code for our BIOS, together with Digital Research's MAC, RMAC, LINK, and SID software development tools, you may customize it to your specific application.

Orders must specify the Double Density controller and boot drive: the Z89-37 (as used in the Z90) or our 77316 (5-inch or 8-inch).

Earlier purchasers of the 128K RAM board, who are registered owners of Magnolia Microsystems CP/M 2 (S/N 2-175-xxxx) can update to CP/M-Plus for $100 (plus shipping and handling).

DISK INPUT/OUTPUT (I/O) BOARDS

These boards are made available apart from our subsystems for use by 'Systems Integrators'. End-users should carefully consider their skills and desired level of involvement before undertaking this responsibility; purchasing a complete subsystem from a qualified integrator may be a wise choice.

Although similar in name and apparent function, these boards fall into two distinct classes:

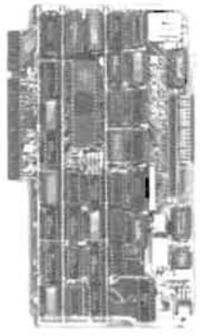
- INTERFACE boards connect a disk subsystem (containing a controller) to the computer. These boards include the Z89-47 and Z89-67 and our 77314, 77317, and 77320 interfaces.
- CONTROLLER boards actually control a plain disk drive. These boards include the H88-1 (Z17) and Z89-37, and our 77316 Double Density Controller.

All Magnolia I/O boards use proprietary techniques to expand the 89's I/O addresses, allowing use of more than Zenith's 5 (3 serial ports, 2 disk controllers). Since our INTERFACE boards also contain 3 RS232 ports (replacing the machine's H88-3) allowing expansion of the I/O capability without using an additional card slot.

Each Magnolia I/O board package includes:
- The disk I/O board itself
- A new I/O decoder PROM
- A new Monitor EPROM
- A copy of MMS CP/M 2.2

Magnolia's CP/M software supports over 20 soft-sectored media formats, including all existing Zenith formats.

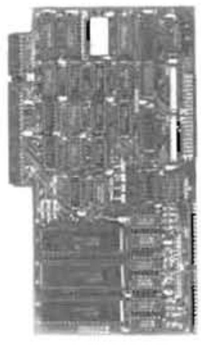
Double Density Disk Controller — reduced to **$495**
order number 77316



This board adds complete hardware and software support for four 8" single- or double-sided AND four 5" SS or DS, 48- or 96-track-per-inch (40- or 80- track) drives, at both single and double recording densities.

This controller is compatible with 8" drives by Shugart, Qume, and Tandon; 5" drives by Siemens, Tandon, and MPI; and most other 'industry standard' drives (possibly depending on the skill level of the experienced 'Systems Integrator').

SASI-bus Winchester Interface — **$350**
order number 77320



This board (functionally replacing the Z89-67 interface) contains 3 RS232 ports and can support up to 8 SASI-bus disk controllers, with up to 4 drives on each!

It is compatible with Winchester controlers by Xebec, DTC, Shugart, and others, as well as Zenith's Z67. CP/M software support for popular controller/drive combinations is included.

MAGNOLIA MICROSYSTEMS, INC. • 2264 - 15th Ave West
(206) 285-7266 (800) 426-2841 • Seattle, Washington 98119

CP/M is a registered trademark, and CP/M-Plus, MAC, RMAC, and SID are trademarks of Digital Research

Assembly Language Programming in CP/M-85 with the H/Z-100 Computers

D. C. Shoemaker
HQ USEUCOM Box 897
APO, NY 09128

I'd like to share a simple piece of software with those of you who may have recently acquired one of the Heath/Zenith 100 series computers. In the process of presenting this program, we'll also take a look at assembly language programming for the very new beginner. No frills, no esoteric explanations, just good old empirical program writing. The goal is twofold: first, to get a useful program into your software library, and second, to "break the ice" on assembly language.

First, remember when you're writing software that it's impossible to harm the computer (unless you pick it up and throw it against the wall). The worst thing that can happen is that you may crash a disk, causing something to overwrite a vital part of CP/M or some other program already on the disk. But that's why we make back-up copies of our disks (a hint).

I recently finished building an H-120 kit, with which I'm particularly pleased. It's my third Heath computer kit (following an H8 and an H89), and without a doubt the finest effort yet from Heath. As with all things, however, there is always at least one thing about any computer that misses the ideal, and in my case it's the key click feature that's built into the system. I have it on authority from Bob Harris, Technical Consultant with Heath, that there's no way to defeat the click from the keyboard itself, since it lacks the "OFF- LINE" key present on the H/Z-19 and the H/Z-89. This means that the key click can only be turned off by telling the computer to do it for you. According to the H-100 operator's manual, the key click feature can be turned off by transmitting an ESCape x2 comand to the computer.

The first way I thought of was to write a short program such as the one in Listing 1 that would send out the required escape code.

```
10 PRINT CHR$(27);"x2"   ' Send escape command to computer
20 SYSTEM                ' Return to CP/M via warm boot
```

Listing 1. Defeating the key click in MBASIC.

While this works, it isn't very efficient since you have to load MBASIC, which takes five or six seconds, load the program, run it, and return to CP/M. So we'll bypass this approach and press on with our assembly language program.

In reviewing the manuals that came with CP/M-85, I found that in the case of Heath's version, there's a feature that allows you to identify an executable (or .COM) file that you want CP/M to run every time you either boot the system from power-on, or every time you press CTRL-C for a warm boot, or both. The .COM file can be anything you want, and it occurred to me that if I had a small assembly language program that sent the proper control code to the computer, my problem would be solved.

The approach I finally settled on was to write such a program and have CP/M run it each time I started the computer and every time either a program or I caused CP/M to do a warm boot. The program in Listing 2 is my solution to the problem.

```
        ;RESET.ASM cancels the H/Z100 key click
        ;by sending an ESCx2 to the computer.
        ;
        ORG     0100H           ;Assemble at 100H
CONOUT  EQU     2               ;CP/M system call
ESC     EQU     001BH           ;Define ESC code
X       EQU     0078H           ;Define 'x'
TWO     EQU     0032H           ;define '2'
ENTRY   EQU     0005H           ;Warm boot address
        ;
START   MVI     E,ESC           ;Load register E with ESC
        MVI     C,CONOUT        ;Load register C with syscall
        CALL    ENTRY           ;Send out to computer
        MVI     E,X             ;Load register E with 'x'
        MVI     C,CONOUT        ;Load register C with syscall
        CALL    ENTRY           ;Send out to computer
        MVI     E,TWO           ;Load register E with '2'
        MVI     C,CONOUT        ;Load register C with syscall
        CALL    ENTRY           ;Send out to computer
        JMP     ENTRY           ;Jump to warm boot address
        ;
        END
```

Listing 2. Source code of the RESET program.

This program is short and simple, and should not pose a problem for even the newest owner of an H/Z-100 computer. In fact, whether or not you have any desire to defeat the key click feature, it's a good introduction to the use of the assembler that comes with CP/M-85. The program has comments for clarity, but unless you've dug into the Intel 8085 microprocessor manuals for what each instruction means, they all probably look like Greek to you. While that's not important for the time being, you really do owe it to yourself to spend a little time becoming familiar with assembly language if you want to get the most from your computer.

There are only six instructions involved in our program, and this is a good point to look at them. The first is the ORG instruction, it tells the assembler that the final product is supposed to start (ORiGinate) at memory location 100H. This location is the point where almost all CP/M programs begin.

The next instruction is EQU, short for EQUate, which means that we're setting something equal to something else. This is usually the way program variables are identified to the assembler for later use in the program. EQUates are supposed to make the program easier to read by using words that remind you of something (mnemonics) rather than forcing you to deal with a string of numbers that might or might not remind you of the instruction you wanted to use.

Next comes MVI, which means MoVe Immediately to some memory location, in this case a register in the CPU chip itself. The register is just a special part of the microprocessor's memory in which small pieces (or bytes) of data can be stored for short times for fast access and use. The letter (e.g., "E") that follows the instruction identifies the register to be used.

The CALL instruction is one of the more important ones to note. It's really a system call (sometimes referred to as a "syscall") to CP/M to do something that we would otherwise have to spell out in great detail. These system calls save a lot of programming effort, and handle things like disk access, printer output and, as in our program, output to the console or video monitor.

Next is the JMP instruction. This tells the program to JuMP to the memory location specified. Note that here, as elsewhere, we used a name or a label instead of just the memory location 0005H. This makes for easier readability since we can easily understand what the programmer meant by "ENTRY", but 0005H might not tell us much at all.

Last comes the END instruction. As you might suspect, it tells the assembler that this is the end of the program. In our example here, the program itself never actually gets to the END, as the JMP instruction sends it off to the CP/M warm boot entry point. Nevertheless, it's essential to tell the assembler that we're through. You'll get an error message if you forget.

Note the lines of text to the right of the semicolons. These are the comments that will be used by others trying to figure out just what it was that we were doing when we wrote the program. As such, they're very important, and not to be omitted.

Here are the steps we will take to get our program into the computer and working. First, if you haven't done so already, I recommend that you make up a working disk including the CP/M operating system, the assembler (ASM.COM), LOAD.COM, and whatever you use for an editor. I happen to prefer TXTPRO from Newline Software, but you can use ED.COM that comes with CP/M-85. You'll also want a copy of CONFIGUR.COM and STAT.COM on the disk, for reasons that will become clear later.

With the working disk made and booted up in your computer, call up your editor and type in the text of the program exactly as it appears in Listing 2.

After the program is typed into the computer's memory with the editor, save it to the disk using the file name RESET.ASM. You could call it anything you like, but this name made sense to me. The usual convention is to call assembler source code files something ending with .ASM, just so we'll all understand what the others are doing.

With the program saved to a disk file, we're ready to call up the assembler. Do this by typing ASM RESET.ASM. This will cause CP/M to load the assembler, and cause the assembler to look for a file named RESET.ASM. It will then create its output in the form of a file of numbers. This file, while not yet ready to run, is rather interesting in itself, and instructing to look at. When the assembler has finished with the file, call it up on the terminal by the command TYPE RESET.HEX and take a look. You should see what appears in Listing 3. It's not yet what's referred to as machine language. That's composed of binary numbers (0's and 1's) that the computer can read directly, but in a bit we'll look at a CP/M utility that does just that.

```
:100100001E1B0E02CD05001E780E02CD0500
    1E320C
:0B0110000E02CD0500C3050003D
:0000000000
```

Listing 3. The hex output from the assembler.

For now, these are the numerical instructions that represent the mnemonic instructions you typed into the computer's memory and made into a source code file. Some instructions require one byte, some require two, and some take three, making the .HEX file difficult to read. The numbers are in a notation called "hexidecimal" or "hex" for short. They're based on 16 instead of 10 for reasons that aren't important to us now. We'll see more of them when we look at the other file the assembler produces, the .PRN file. We can view that one by typing TYPE RESET.PRN. When you do, you'll see a display like Listing 4.

In looking at Listing 4, you'll notice that the entire source code is present, including the comments. This makes a useful record of the program, and it's not a bad idea to print it out and save it if you have a printer.

The next things you may spot are the column of hex numbers on the left-hand side of the listing. These are the same numbers contained in the .HEX file we looked at before. The left-most hex number is a line number generated by the assembler. This is just to help you keep your place while you read the listing, and to aid in locating errors that the assembler spots. The other numbers are the actual hex instructions that will be used in the next step. Each instruction consists of two characters (remember that a hex number can run from 0 to 9 and A to F). The numbers that we used in the program, like 0005H for the warm boot entry point for CP/M, are printed with the least significant byte first. This is the technical way of saying that they're written with the bytes in reverse order, and it's done to make it easier for the assembler to do its work. Thus, 0005H appears as 0500 in the .PRN file listing. You may have noticed that it's that way in the .HEX listing also.

Something you didn't see in the listing above is an error listing. If you make a mistake in typing the source code file, the assembler will usually spot it and identify it in the .PRN file. Then you can refer to it, determine where you went wrong, call up the source code file and make the correction, and run the assembler again. You may have to do this several times before you get an error-free run.

Now that you have the .HEX file on the disk, you're ready for an operation that CP/M calls

```
                              ;RESET.ASM cancels the H/Z100 key click
                              ;by sending an ESCx2 to the computer.
                              ;
0100                          ORG     0100H         ;Assemble at 100H
0002 =              CONOUT    EQU     2             ;CP/M system call
001B =              ESC       EQU     001BH         ;Define ESC code
0078 =              X         EQU     0078H         ;Define 'x'
0032 =              TWO       EQU     0032H         ;define '2'
0005 =              ENTRY     EQU     0005H         ;Warm boot address
                              ;
0100 1E1B           START     MVI     E,ESC         ;Load register E with ESC
0102 0E02                     MVI     C,CONOUT      ;Load register C with syscall
0104 CD0500                   CALL    ENTRY         ;Send out to computer
0107 1E78                     MVI     E,X           ;Load register E with 'x'
0109 0E02                     MVI     C,CONOUT      ;Load register C with syscall
010B CD0500                   CALL    ENTRY         ;Send out to computer
010E 1E32                     MVI     E,TWO         ;Load register E with '2'
0110 0E02                     MVI     C,CONOUT      ;Load register C with syscall
0112 CD0500                   CALL    ENTRY         ;Send out to computer
0115 C30500                   JMP     ENTRY         ;Jump to warm boot address
                              ;
0118                          END
```

Listing 4. The RESET.PRN file generated by the assembler.

"LOADing a file". I'm not certain why that particular term was chosen; you're not going to "load" anything. What you'll be doing here is converting the hex numbers in the .HEX file into binary numbers that the computer can work directly with. To do this, type LOAD RESET.HEX, and the LOAD program will locate the RESET.HEX file and do the conversion. This results in an executable file called RESET.COM that the computer can run exactly as it would run BASIC or ED. You can't TYPE this one to the screen; binary files can only be viewed through an intermediary program like DUMP or DDT, but that's not a part of this story.

Finally comes the test phase. Up until now, every time you pressed a key, you heard a short "beep" from the keyboard speaker that makes it sound as if the keyboard produces a sharp "click". Type RESET and our program will send "ESCx2" to the computer, telling it to stop making that noise. From now on until CP/M is reset, there will be no key click. Success!

But wait. When we reboot the system, either as the result of running some kinds of programs (TXTPRO resets CP/M when it exits) or when we turn on the system, we have to type RESET each time we want to use it. This is certainly one way to do it, but there's another.

One of the most useful utility programs that come with CP/M is CONFIGUR.COM. Heath's version of CP/M includes the best configuration program I've ever seen. It allows you to set all sorts of options in CP/M that many other computer owners only dream about. And one of the more useful things you can configure is the ability of CP/M to run a program of your choice any time you power-up the computer, or press CTRL-C, or both. If the idea appeals to you, here's how we do that.

Call the configuration program by typing CONFIGUR. (Remember that CP/M files can have only eight letters; that's why you'll see some strange file names.) When CONFIGUR comes up, it will offer you the opportunity to configure the Command Lines. Select this option by typing C and a carriage return. You'll get another menu, this one offering the choice of Cold Boot or Warm Boot. Select each in turn, typing RESET for each, then exit the program as directed. You'll find now that when you boot CP/M at power-up, the key click is turned off. The same thing will happen after each warm boot (done by pressing CTRL-C, for instance). Listing 5 shows the steps we took, and what the display looks like for each step.

Now for STAT.COM. STAT allows you, among other useful things, to "flag" or identify a file as a system file. This means that it won't show up in the directory listing when you type DIR. It will only reveal itself when you type STAT *.* to display the status of all files on a disk. This helps keep the directory down to a manageable size by "hiding" those files which, like CP/M itself, are there, and you know they're there, and you don't have to be reminded each time you type DIR. To do this, just type STAT RESET.COM $SYS. To undo the process, type STAT RESET.COM $DIR. Since RESET.COM is supposed to reside on your working disks and be invoked with each cold and warm boot, there's really no reason why it needs to appear in the disk directory all the time, so I suggest you "hide" it.

```
          CP/M-85 System Configuration Utility version 2.2.101
               Copyright (C) 1982 by Zenith Data Systems

                           *** MAIN MENU ***


                    P - Printer Configuration
                    M - Modem Configuration
                    C - Command Configuration
                    I - I/O map Configuration

                    X - Exit

                    Selection [P,M,C,I,X or ?] :C

                *** Command Line Configuration ***



                C - Cold Boot Command Line = RESET
                W - Warm Boot Command Line = RESET
                ? - Brief Help message

                X - Exit

                Selection [C,W,? or X]:X
```

Listing 5. Displays for installing RESET.COM for execution by CP/M-85 on both cold boot (power-up) and warm boot (CTRL-C or on exit to CP/M by some programs).

It's probably occurred to you by now that you could do the same thing with any other escape code you might want to send to the console. What you may not know is that by changing CONOUT and its associated value of 2H (2 in hexadecimal), you could send any escape code to any other part of your system. The printer, for example, uses the system call 5 for list device output. By creating a series of files like RESET, each one containing a particular escape code you need for printer control, you can configure your printer to do all sorts of things that you might not have thought about before. For instance, by sending the proper code, your printer might be made to provide listings in Swedish or Italian (assuming your printer has those capabilities built in).

We've reached the end of the story. Hopefully, you won't feel intimidated by the thought of doing assembly language programming after having seen how easily this simple program can be created, entered, assembled, loaded, and run. Even if you like the key click, you may have occasion to turn it off. I leave it to you to figure out how to turn it back on again. (Hint: you might start by looking up the required escape code in the operator's manual.)

Special Notice:

Heath Company will discontinue manufacture and sale of its H-8 computer, as well as expansion cards and accessories for the H-8, effective immediately. Heath will continue to supply spare parts for the H-8, upon request.

Heath understands that Trionyx Electronics, Inc. of Santa Ana, CA (no connection with Heath Company), a long time servicer of H-8 computers and manufacturer of accessories for the H-8, plans to continue to service the H-8 and to expand its line of H-8 compatible accessories.

Newline Software

P.O. Box 402 • Littleton, MA 01460
617-486-8535

Software For Heath/Zenith Computers

Software for Your H8/H89 and HERO-1 Robot

MFG.*	OS	PRODUCT NAME	PRICE
NL	HERO-1	Keypad Utilities Cassette	$ 24.95
NL	cp/m	ASM68 HERO-1 Cross-assembler	$ 59.95
NL	hdos	ASM68 HERO-1 Cross-assembler	$ 59.95
NL	hdos	TEXT PROCESSOR 4.1	$ 59.95
NL	cp/m	TEXT PROCESSOR 4.1	$ 59.95
NL	cp/m	TOUCH TYPIST 2.0	$ 29.95
NL	hdos	TOUCH TYPIST 2.0	$ 29.95
NL	hdos	VIDEO ARTIST PLUS	$ 39.95
NL	hdos	H25/Z25 GRAPHICS GENERATOR	$ 44.95
AT	cp/m	DBASE II	$465.00
AT	cp/m	FINANCIAL PLANNING LANGUAGE	$549.00
AT	cp/m	THE BOTTOM LINE STRATEGIST	$299.00
NL	hdos	TOUR-700	$ 19.95
NL	cp/m	TOUR-700	$ 19.95
NL	hdos	UNIVERSAL SPOOLER	$ 29.95
NL	hdos	RESISTIVE NETWORK DESIGN	$ 24.95
NL	hdos	MX-80 DEVICE DRIVER	$ 29.95
NL	hdos	MODEM COMMUNICATIONS DEVICE DRIVER	$ 34.95
NL	cp/m	DOMESTIC MGT. SYSTEM	$ 39.95
NL	cp/m	CHECK PROCESSOR	$ 29.95
NL	cp/m	SNAP	$ 29.95
NL	cp/m	MBASIC LANGUAGE CAI	$ 59.95
NL	hdos	MBASIC LANGUAGE CAI	$ 59.95
LS	hdos	DIAMOND ACCOUNTING SYSTEM	$ 74.95
SS	hdos	SUPER SYSMOD 2	$ 29.95
SS	hdos	UD.DVD Printer Driver	$ 29.95
SS	hdos	UD.DVD Printer Driver with SPOOLER	$ 39.95
SS	hdos	MX.DVD Printer Driver	$ 29.95
SS	hdos	MX.DVD Printer Driver with SPOOLER	$ 39.95
SS	hdos	H25.DVD Printer Driver	$ 29.95
SS	hdos	H25.DVD Printer Driver with SPOOLER	$ 39.95
SS	hdos	TINY PASCAL & DISASSEMBLER	$ 25.00
SS	cp/m	TINY PASCAL & DISASSEMBLER	$ 25.00
SS	hdos	GERMAN HANGMAN	$ 29.95
SS	hdos	ITALIAN HANGMAN	$ 29.95
SS	hdos	SPANISH HANGMAN	$ 29.95
SS	hdos	FRENCH HANGMAN	$ 29.95
SS	cp/m	GERMAN HANGMAN	$ 29.95
SS	cp/m	ITALIAN HANGMAN	$ 29.95
SS	cp/m	SPANISH HANGMAN	$ 29.95
SS	cp/m	FRENCH HANGMAN	$ 29.95
SS	hdos	SUPERMATH 1	$ 39.95
SS	cp/m	SUPERMATH 1	$ 39.95
SS	hdos	HFORTH-79	$ 34.95
SS	cp/m	UNIVERSAL PRINTER DRIVER & SPOOLER	$ 39.95
PB	hdos	LUCIDATA PASCAL	$ 55.00
PB	hdos	P-CODE to MACRO-80 TRANSLAT for LUCIDATA PASCAL	$ 25.00
PB	hdos	GRAPHICS and MATH CHIP SUPPORT for LUCIDATA PASCAL	$ 25.00
PB	cp/m	LUCIDATA PASCAL	$ 55.00
PB	cp/m	P-CODE to MACRO-80 TRANSLAT for LUCIDATA PASCAL	$ 25.00
PB	cp/m	GRAPHICS and MATH CHIP SUPPORT for LUCIDATA PASCAL	$ 25.00
NL	hdos	DENSE-PAC for Data COMPRESSION/ENCRYPTION	$ 29.95
NL	cp/m	DENSE-PAC for Data COMPRESSION/ENCRYPTION	$ 29.95

RUN HDOS 2.0 ON YOUR Z-100

Run your old HDOS 2.0 software on your new Z-100! Twice the speed on most programs. Available now! Optional Device Drivers, Txt Pro, Super Sysmod 2, and more. This is actual HDOS 2.0, not an emulator. Requires standard Heath HDOS 2.0 diskettes.

Software for Your Z-100

MFG.*	OS	PRODUCT NAME	PRICE
NL	zdos	Z-100 HDOS 2.0 UPGRADE	$ 79.95
NL/SS	z-100 hdos	SUPER SYSMOD 2	$ 29.95
NL/SS	z-100 hdos	UD.DVD Printer Driver	$ 29.95
NL/SS	z-100 hdos	UD.DVD Printer Driver with SPOOLER	$ 39.95
NL/SS	z-100 hdos	MX.DVD Printer Driver	$ 29.95
NL/SS	z-100 hdos	MX.DVD Printer Driver with SPOOLER	$ 39.95
NL/SS	z-100 hdos	H25.DVD Printer Driver	$ 29.95
NL/SS	z-100 hdos	H25.DVD Printer Driver with SPOOLER	$ 39.95
NL	z-100 hdos	TEXT PROCESSOR 4.1	$ 59.95
NL	zdos	TEXT PROCESSOR 4.1	$ 59.95
NL	cp/m-85	TEXT PROCESSOR 4.1	$ 59.95
NL	zdos & cp/m-85	TEXT PROCESSOR 4.1	$ 99.95
NL	zdos	TOUCH TYPIST 2.0	$ 29.95
NL	cp/m-85	TOUCH TYPIST 2.0	$ 29.95
NL	zdos	CHECK PROCESSOR	$ 29.95
LS	zdos	DIAMOND ACCOUNTING SYSTEM	$149.95
AT	zdos	DBASE II	$465.00
AT	zdos	THE BOTTOM LINE STRATEGIST	$299.00

DISKETTES

NASHUA DISKETTES (LIFETIME GUARANTY)	1st Box	2nd-9th Box Each	10 Boxes
Hard 10 Sector, SS, 48 TPI	$ 25.00	$ 20.00	$199.00
Hard 10 Sector, DS, 48 TPI	$ 33.50	$ 28.50	$280.00
Soft, SS, 48 TPI	$ 25.00	$ 20.00	$199.00
Soft, DS, 48 TPI	$ 33.50	$ 28.50	$280.00
Soft, DS, 96 TPI	$ 42.00	$ 37.00	$365.00

* Manufacturers - NL – Newline Software
AT – Ashton-Tate
SS – SoftShop
LS – Lindley Systems
PB – Polybytes

Attention Program Authors

NEWLINE SOFTWARE is searching for original, high quality programs for the H8, H89, and Z100 computers. We are interested in GAMES, GRAPHICS, BUSINESS SOFTWARE, UTILITIES, LANGUAGES, COLOR GRAPHICS or any other software for the businessman or hobby computerist. Programs must be written for ease of use, take advantage of the hardware features, and be fully documented (source code not required). You can earn high royalties for your programs without interferring with your hobby. We will evaluate your programs in strictest confidence. Materials will be returned only if return postage is provided.

SEND FOR FREE CATALOG

Part 3

Modem Tutorial 83

Walt Jung

Pickaprogram!

With the first two installments already behind us, we are now ready to get into modem program packages themselves. This installment will cover what a general purpose modem program does on a Heath/Zenith computer, under the various operating systems. If you missed the first two installments of the series (May and July), beg, borrow or steal those copies. You'll need them as background for what follows below.

Whatsaprogram?

First of all, let's define just what we are talking about with regard to a modem program, since 'modem program' can mean different things to different people. For our purposes, the discussion will be general, in the sense that we will be discussing programs which allow a variety of modem functions to be performed. This can include everything from the most simple terminal programs (where your computer emulates a terminal), up to full-featured versions with that plus autodial, block mode error checked file transfers, auto logon, and other goodies. The programs discussed here will allow communications with a variety of other systems, ranging from other personal computers and BBS systems up through large time sharing systems.

Thus, these programs can be used for just about any modem communication use you might desire. But, like the modems themselves, modem programs come in all varieties, and with a truly staggering array of features. In fact, the range of features is such that it can literally overwhelm you if this game is still relatively new (and maybe even if it isn't!). Like the previous advice regarding modem hardware, you should take it slow at first, that is, you need not jump into the most complex program right away.

A fortunate factor with modem programs is that some of them are in the public domain, and are therefore nominally free (a phone call or copy costs). So, you can actually try one before you buy and gather some experience inexpensively. Many of the public domain programs available are truly excellent, and since they are free, they are obviously very popular. Some may not feature all of the hardware related bells and whistles of the Heath specific commercial packages, but you can customize them if you wish with the available source code.

The big difference which should be understood between any public domain program and a similar commercial counterpart is that public domain software is not generally supported. So, unless you are already experienced with programming, this should be a factor in your selection, if a program requires any customization (not all do). If you have a local HUG with members active with modems, you may have all the support you'll need, and there may be a program ready to go on disc (check for disc library availability on this). On the other hand, you know you can get support with the commercial programs, and some of those discussed here offer exceptional value as well as that crucial support.

Program Functions and Features

Obviously, for a modem program to be useful, it must provide you with the operational features you require. These features range from simple to sophisticated, and like a fancy modem, all of them will not be necessary for all users. The various features are described below as to their function, and again later on, with regard to specific programs. Once the understanding of the various features is clear, one can then appreciate the basic differences in the programs. These features are numbered sequentially and referenced in the program comparison in Table I. It is assumed that all programs listed are ready to run on the Heath/Zenith operating system, as listed, and that 300 baud full duplex communications is the standard default.

To begin, there are just a few very basic features which are all you will require for conversing with message systems, either a local BBS or another personal computer user. You may need to manually dial your phone, or, if you use a Smartmodem (trademark Hayes Microcomputer Products), you can dial from the keyboard (note that this is true, whether or not the program itself also supports autodial). Here then are these minimum features. Note the number prefix [N], associated with the particular feature (function), which is keyed to the functions listed in Table I. The program-by-program discussion which follows this part fills in with details on specific packages.

[1] ASCII send

With this function, you can send an ASCII file from your disc to the remote system. This is useful for prepared messages, either for the purpose of saving your actual on line time, or to better control formatting by preediting off line. The better programs will also allow the control of delay time, between lines, or the sending of lines after a prompt character is received from the remote. It is also desirable to have the prompt character be variable to suit different remote systems.

[2] ASCII save

This one allows you to capture incoming ASCII data into a memory buffer for a subsequent save to disc under a specified filename. This function is useful for the capture of messages or other text where an occasional glitch is not harmful. It is generally not optimum for saving program files, since block mode error checked transfers are readily available (below). This save function should be easily toggled on and off, to allow "on-the-fly" editing of the incoming data.

Better programs will display on the screen

Table I
Modem programs and their functions

FUNCTION #	1	2	3	4	5	6	7	8	9	10	11	12	13
PROGRAM (cost) (op system)													
CPS ($40) (H,C Z?)	● [See text]	●		●	●	●	●	●	●				●
ZLYNK ($25) (H)	● [ZLYNK 2 $40 (CZ,C), see text]	●	a	●	●	●	●	●	●	●			●
REACH 2 ($20) (H,C)	●	●	a			c		b	●				●
TERMINAL ($25) (H)	● [See text]	●	●	●		●		●	●	●			●
AMCALL ($150) (C,CZ,Z)	● [See text]	●	●	●			h	b	i	j			
ACCESS ($40) (C,CZ)	● [See text]	●	a	●					●	●			●
COMMX ($99) (C,CZ)	●	●	●	●		●	e	b	c	j	f	g	
MAPLE ($35) (H,C,CZ)	● [See text]	●	a	●		●	●	●	●	●	●	●	●
HTERM ($20) (H)	●	●	●	●									
ZTERM ($20) (C,CZ)	●	●		●		●							●
H/M/ZPLINK (0) (Note 2)	●	●	●			●							●
MDM7nn (0) (C,CZ; note 3)	●	●	a	●		●	d	b	●	●			
CTRM89 (0) (C,CZ)	●	●		●			●						●
YAM (0) (C,CZ)	●	●	a	●			d	●	●	j		●	●

Notes:

1) Operating systems: H=HDOS, C=CP/M-80, CZ=CP/M-85, Z=ZDOS.

2) HPLINK (H), MPLINK (C), ZPLINK (CZ).

3) MDM7nn implies latest version, as MDM711, MDM712, etc.

4) The '*' indicates function available, while 'n' indicates it is available with a special note, under 'n' below.

a) Printer output is buffered.
b) Automatic re-dial feature.
c) Macro capability.
d) Wildcard (batch) transfers, both checksum and CRC.
e) Uni-directional MODEM, plus two-way COMMX, both with wildcards.
f) User modifiable translate table.
g) Additional program CONSOLX required.
h) MODEM; MCALL-C, MCALL-T (wildcards), Western Union, RCA Globcom, ITT Timtron.
i) Flexible modification, ZIP file or command line.
j) Very flexible directory, other housekeeping.

the remaining buffer space to prevent buffer overflow. Or, better yet, when full, automatically do an XOFF to the remote to freeze incoming data, write out the buffer contents to disc, and then do an XON to restart the remote, and continue with the save. This way, the buffer effectively becomes as large as your disc capacity allows.

[3] Printer logging

Incoming data can also be sent to the printer for a direct hard copy log of a session. This function should be easily toggled on and off again (usually by ↑ P) to control the data logged. Because both incoming data rates as well as printer speeds can vary, characters can sometimes be lost with the use of this function. The better programs buffer the data sent to the printers to avoid this problem.

[Some or all of the above functions can be found in several basic terminal programs, such as CTRM89, HPLINK, MPLINK, and ZPLINK, as well as HTERM and ZTERM (see Table I). If you have had no prior experience with modems, one of these programs will be a good starting point since they are easy to use and won't overwhelm you with options and commands which you may not fully understand as yet. CTRM89 is an excellent choice for starters since it not only includes the terminal functions, but also allows block transfers, and importantly, is complete and ready to go just as it stands.

See Table II for the sources of these programs, and note their applicable operating systems listed in Table I. An effort has been made to have versions of the public domain programs available for download on the Compuserve Information Service (CIS) HUG SIG, the BHEC RCPM, and on disc from the CHUG library.]

[4] Change baud rate

Although 300 baud is still the mainstay of personal computer telecommunication, a number of systems also respond at 450 and 1200 baud. It is therefore useful to conveniently change the baud rate of the modem port on the fly, or from within the program.

Usually, even when this option is available, the program will set the baud rate upon entry to the standard 300 baud. Some more complete programs will also display the current baud rate for you. If you are not sure of the baud rate of the system you call, always try 300 baud at first. The system should respond after you hit a RETURN or two. Just remember to make sure it has answered first, and the modems are locked up, before hitting RETURN.

[5] Connect time log

A very useful feature for maintaining a time log of sessions is a connect time clock, and/or a logging feature. The most simple implementation is to display a running time clock, for example on the 25th line. A more complete implementation is to store the log time information in a connect time log file. Not very many programs do this, but it is a useful feature for talking to time sharing systems, where your connect time relates to dollars, as well as it is for long distance calls. Note that this feature is only likely to be found in versions which take advantage of the Heath specific features, such as ZLYNK and CPS. Note also that this function may be subject to timing variance due to disc accesses.

[6] Auto logon

It is highly convenient to be able to send a unique logon string to the remote computer, to simplify and standardize your access and signon procedure. This can be just your name and/or ID number, but often a password as well. Simple strings can be stored within a program and called by a special key, often one of the function keys or a control key. The better programs support prompted logon sequences which are completely automatic; this will be invaluable for accessing time sharing systems such as CIS.

[7] Block transfers

The best way of transferring program files between computers is via a block transfer method, which sends or receives data in a series of blocks with integral checksum or CRC data. Using this type of transfer, the receiving computer evaluates a received block for integrity, and if an error is detected, requests a re- transmission of the bad block. In this fashion, files of any length can be transmitted with very high confidence. This method also (usually) allows 8 bit transfers, permitting object code program files to be exchanged as well as conventional ASCII (7 bit) files. This of course eliminates the necessity for BINARY- HEX (and back) conversions.

The defacto standard block transfer protocol for personal computers has become Ward Christensen's MODEM, also often referred to as XMODEM. While originally implemented in CP/M, the use of this protocol has extended to other operating systems, now including both HDOS and ZDOS. This means that files can be exchanged over the normal disc barriers of these operating systems.

The more complete modem programs support this protocol, and some of them may also include the 16 bit CRC option of use as well as the ability to batch mode send (or receive) a series of files, that is with wildcard filenames. For serious modem use, you should not be without some program which supports the Christensen MODEM protocol. It is the key to standardized, reliable data transfers on personal computers.

Another very important protocol is the CIS executive protocol, which is used for block transfer to or from the HUG SIG and others on CIS. Few programs support both this and the Christensen protocol, but those that do offer exceptional utility. Examples would be YAM, as well as the new ZLYNK and the new TERMINAL. However, several customized modem packages are also available on the CIS system, and handle error checked file transfers exceptionally well. They will be discussed with the CIS part of this series.

Many commercial packages also feature proprietary protocols of their own design. For example, AMCALL supports a host of protocols, not only MODEM, but the MicroCALL 'MCALL-C' and 'MCALL-T'. COMMX has its own 'COMMX' protocol in addition to MODEM. The important thing to remember about **any** block mode protocols is that **both** ends must be operating under the same protocol system for a transfer to take place.

[8] Autodial, re-dial (Smartmodem)

With the advent of the intelligent modem and the defacto standard dialing commands of the Smartmodem, quite a few programs offer the feature of automatic dialing of a number, either from program memory or a disc file. This is supported by many packages and is a great operational convenience. Typically, a single keystroke selects a number from a library and dials the number. Some programs provide a re-dial of a busy number in addition, which can also be automatic (AMCALL, COMMX, MDM711).

A disc based number library has the advantage of easy update, while a program integral library is more compact but can be more difficult to update. The better programs sup-

Table II
Modem programs and their sources

Discussed here

Source	**Program**
Hilgraeve Incorporated PO Box 941 Monroe, MI 48161	"ACCESS"
Heath Company Hilltop Rd. St. Joseph, MI 49085	"CPS"
Hawkeye Grafix 23914 Mobile Street Canoga Park, CA 91307	"COMMX"
Software Wizardry Inc. 122 Yankee Drive St. Charles, MO 63301	"ZLYNK"
Software Toolworks 15233 Ventura Blvd. Suite 1188 Sherman Oaks, CA 91403	"REACH 2"
Studio Computers 999 S. Adams Birmingham, MI 48001	"TERMINAL"
Heath Users' Group Hilltop Rd. St. Joseph, MI 49085	"MAPLE" "HTERM" "ZTERM"
MicroCALL Services 9655-M Homestead Ct. Laurel, MD 20707	"AMCALL"
Public Domain (RCPM & BBS systems, CIS HUG SIG (XA), CP-MIG (XA) etc.) (CHUG, Box 2653, Fairfax, VA 22031)	"MDM7nn" (MDM710 & up) "CTRM89" "MPLINK" "HPLINK" "PLINK"

Data incomplete

Source	**Program**
Software Subscription PO Box 5379 Richmond, CA 94805	"FTCOM"
MicroProcessor Assoc. PO Vox 3438 Nashua, NH 03061	"EMAIL-89"
Microstuff, Inc. 1845 The Exchange, Suite 12 Atlanta, GA 30399	"CROSSTALK"
Mycroft Labs PO Box 6045 Tallahassee, FL 32301	"MITE"
Newline Software PO Box 402 Littleton, MA 01460	"MD:"
Westico Software 25 Van Zant Street Norwalk, CT 06855	"ASCOM"
Husker Sys. of Neb. 4517 North 61st St. Omaha, NB 68104	"ZED-COMM"

Note: Some packages are also sold in your local HEC, so check for availability there as well.

port auto-dial even to the extent of including the long distance dialing service prefixes, and some can even re-configure modem parameters as part of the dialing routine.

[9] Modifiable defaults

Given the range of variation in communication parameters possible with various systems, many instances will require changes to one or more program defaults within a session. Examples might be to vary a protocol selection from the default of XON/XOFF, etc. It is important that such parameters be easily and quickly modified, either from within the program, or as it is called up.

There is no single approach to how such parameters are modified. Methods using install utilities and in-program menus are used, as well as run time configuration files. The more complete programs offer extensive custom variations, such as flexible control character processing, and some even have the ability to save session parameters to disc for recall (MAPLE). The method used by AMCALL seems very flexible and uses a special disc file of the type 'ZIP', which can be called up to change defaults. ZIP files can be stored for different uses. AMCALL also supports command line default mods.

[10] Housekeeping

This general term can mean a number of things specifically, but the general concept is to perform operating system functions from **within** the program, rather than making an exit necessary. In practice, different programs support various degrees of housekeeping, such as logging a new disc (or disc/user for CP/M), a directory function (preferably sized and sorted), and the renaming or deletion of files.

The built in DIR features of both COMMX and AMCALL are exceptional and allow a paged, sorted, and sized directory display. YAM has housekeeping tricks which are nothing short of amazing; the DIR even does the transmission time for you, and files can be CRC checked, as well as TYPEed, even USQ'd if you need! These goodies are in addition to the logging of drive/user, and maintaining a transmission log.

[11] Character translation

If you communicate with large mainframe computers using non-ASCII character sets, you may need a program which provides character translation, such as APL to ASCII (MAPLE), or simply encoding or decoding, for security reasons (COMMX).

[12] Remote operation

A couple of programs go so far as to offer the ability of being remotely controllable via a modem line, MAPLE being one of them. Another one is COMMX, with the addition of the CONSOLX software package. YAM has an optional compilation called XYAM, which allows remote control of transfers in an 'XMODEM' fashion.

[13] Terminal dependent features

While not absolutely essential to the basics of modem telecommunications, the use of such terminal features as the 25th line, the function keys, and keypad go a long way towards a friendly environment. A number of programs provide these features, which will obviously appeal to all Heath users.

The Programs Themselves...

With the data of Table I at hand, we are now ready to discuss the overall features of the each of the programs.

CPS

CPS is a well established and very flexible modem program written by Bob Mathias and available from Heath's 'Softstuff'. It runs under both HDOS and CP/M systems on the H/Z-89 or H8, and a new ZDOS version is currently in the works.

This program does most of all the things you are likely to need in a modem program, yet it is still relatively easy to use. It uses a series of menus for quite thorough on line configuration, even including incremental baud rate change and port assignments, and it can log you onto CIS, as well as auto-dial from keypad stored strings.

Two block mode transfers are available, one of these MODEM compatible (H89-to-H89 mode, CP/M only) and the other a CIS protocol. Also, there is a mode for standard ASCII transfers.

CPS is weak in the area of housekeeping functions, but in an overall sense, it is still a useful and powerful package, in spite of its age. An outstanding virtue of the program is the ease of use. It can be readily used by a newcomer without intimidation. If you are running under ZDOS, watch for the new version from Bob.

ZLYNK

Dale Lamm's ZLYNK quickly established a mark as a very comprehensive HDOS program, the first of such to include the Ward Christensen MODEM block protocol. The original program includes all of the features noted in Table I, plus such innovative features as edited screen dumps and a connect time log file.

ZLYNK comes with a CONFIG program for infrequently changed parameters, while more frequently altered ones such as baud rate can be changed within the program. The program is menu driven, but uses defaults for many items. For example, you can go from the initial main menu directly to terminal mode simply by hitting RETURN (as well as 'T'). With all of the features available, ZLYNK may take some getting acquainted for new modem users. Fortunately, the 36 page documentation should explain things very well. The program also comes with two other utilities, XLATE and FILTER, useful for processing files off line.

A new version of ZLYNK, ZLYNK 2 for CP/M-85 on the H/Z-100 has been in the works for some time. When available, this version should greatly enhance the communications power of that machine. Look for such things as multiple block mode protocols, and an even greater range of more general features. Hopefully, the CP/M-80, HDOS, and ZDOS versions will not follow too far behind the CP/M-85 release of ZLYNK 2.

REACH 2

REACH 2 is a recent update to the popular program from Software Toolworks, available as before for both the HDOS and CP/M operating systems.

This program is basically a terminal program, but what it does it does well. The printer output is buffered, and it has a sophisticated auto dial and redial capability, as well as support for highly customized logon sequences. A configuration program is provided for baud rate change at installation.

TERMINAL

TERMINAL is an HDOS program by L. Boufford, available from Studio Computers. It offers a flexible combination of features including auto dial for the Smartmodem and auto logon.

The program is menu driven with a variety of screens, and allows not only the array of communications functions noted, but also a host of HDOS housekeeping functions (reset disk, delete file, etc). ASCII file transfers are supported, but not block mode.

The EDITERM utility also comes with the program for the creation of custom logon sequences, including the modification of defaults for unique system communication. The documentation consists of 44 pages plus appendices. New versions for HDOS, CP/M, and ZDOS are planned for a September release, featuring support for the Christensen as well as CIS block mode protocol transfers.

AMCALL

AMCALL is a full featured communication package written in 'C' by Tim J. Pugh, Jr. It is a highly developed program, having been evolved from the original MCALL. The C language is highly portable, and both H/Z-89 and H/Z-100 CP/M AMCALL versions are available, with a ZDOS version likely to be available when this article appears. A 56K (min) system is required to run AMCALL.

The program is highly flexible, and is available for many computers and UART configurations beyond the Heath version discussed here, which supports the Hayes Smartmodem 1200 (300 and 1200 baud). AMCALL is probably one of the most flexible programs around in terms of configuration, as it can be customized for baud rate, protocols, etc. either by command line switches or via a 'ZIP' file, called when it is invoked. Custom ZIP files are created with the supplied utility, CONFIG-A, and saved to disc for use with specific systems. AMCALL uses keyboard sequences of ESC plus a character for its commands, and has a logical command and internal configuration menu system.

The Smartmodem is dialed from a disc PHONE.NUM (ASCII) file created with an editor. This file has three fields, a letter, a descriptor, and number for each system. Dialing is done by entering the appropriate letter, and both re-dial and long distance dialing service features are available for 26 entries.

AMCALL comes with other utilities; XD which is a very flexible directory program, and UNLOAD (COM to HEX), as well as a number of help files. A recent update has added several additional protocols, making it one of the most flexible programs in this area. The variety of protocols, the flexibility of configuration, and the intelligent dialing are major features of this program. Documentation comes in the form of a 41 page indexed manual and the mentioned disc files.

ACCESS

ACCESS is a CP/M terminal program by John Hile, available from Hilgraeve Inc. It is an easy to use program with a variety of nice features. While it is currently available for the H8, H/Z-89, and H/Z-100 under CP/M, a ZDOS version is planned also.

The program is menu driven and uses the H/Z-19 terminal features to good advantage in a well integrated package with easy modification of many parameters. Changes can be made from menus, but ACCESS also stores customization parameters in an 'ACCESS.OPT' file for later recall. Up to seven different option arrangements can be so defined and stored.

Currently, the program has an ASCII file transfer capability, but a block mode Christensen type protocol feature is being added, as is auto-logon, and a timer function. Documentation consists of 10 pages (for the version reviewed), which while short, is clear and straight-forward. The new features were reportedly being finalized as this article was being wrapped up, so they will likely be available as you read this.

COMMX

Will Pierce's COMMX program is part of a comprehensive communication package from Hawkeye Grafix, called 'COMMX-PAC'. COMMX is the modem utility portion of this package reviewed here, with features as described in Table I, and is available for the H/Z-89 and H/Z-100 computers under CP/M. The complete COMMX-PAC sells for $150, and includes the additional programs MBORD, a bulletin board system; MBORDM, a data base manager; and CONSOLX, a host remote access utility. This package allows one to set up the computer for completely unattended operation using a Hayes Smartmodem.

COMMX is a flexible and capable program, yet it is also relatively easy to use through a series of menus. Smartmodem dialing is done via a disc file, COMMXDIR.DAT, which includes not only the number to be dialed, but also configuration parameters (baud rate, data bits, macro file, etc.). Dialing can be either conventional or via a long distance dialing service with re- dial automatic. Dialing (as well as other functions) can also be specified from the command line, simply by appending a dial command (D) and the number.

Auto logon is supported via a KEYMAP file, which is created by the user, and can be configured for a variety of characteristics and time delays. This file is referenced in the dial command to auto log onto a particular system. COMMX supports the MODEM protocol, but only for receive, and the COMMX protocol bi- directionally. Both protocols handle wildcard transfers.

While COMMX is highly useful as a stand alone modem utility, its value is enhanced greatly when operated in conjunction with the CONSOLX program, which is intended to be used for electronic mail and order receipt functions. With the macro processing capability, batch processing operations can be set up to automatically transfer files at pre-determined times to multiple points.

COMMX is high on housekeeping, allowing discs to be re-logged, and it has a good directory program. In addition to the utilities mentioned, COMMX also comes with a set of the Richard Greenlaw public domain SQUEEZE and UNSQUEEZE file compression utilities. The manual is nicely finished in a 108 page binder and is indexed. Overall, the operating features of COMMX are well balanced between communication utility and ease of use.

MAPLE

MAPLE is a program originally written for the H/Z-89 under HDOS by Dr. William Parke, but since expanded to CP/M, and soon also to CP/M on the H/Z-100. Dr. Parke has been adding features all along to this program which is available through HUG.

MAPLE really exercises all of the terminal features of the H/Z-89, using the 25th line and the function keys to control and monitor virtually all communication parameters. The program can send and receive in a multiplicity of modes (even a Christensen compatible batch mode, in the most recent version), it can edit and send or save screens, save number strings on the keypad, and it can even be remotely operated by password access. With it, you are constantly aware of your current status, via the thoughtful use of the 25th line. When finished a given session, MAPLE saves the settings at exit to disc for you.

MAPLE is a program with a broad array of features, more in fact than are obvious at first. Because of this multi-level array of features, a user new to this game might be advised to take it a step at a time. But, the more experience you gain with it, the more you are likely to appreciate it. The program comes with 52 pages of thorough documentation, including patching instructions. They may not be needed in many instances however, since virtually everything is controllable from within the program.

HTERM

HTERM is an HDOS terminal program from HUG, written by Pat Swayne. It has a wide range of baud rates (110-9600), and can send and receive ASCII files.

The program comes with both the source code, as well as executable files, assembled to operate with both interrupt 4 as well as the standard 5.

ZTERM

ZTERM is a CP/M terminal program from HUG, written by Jim Buszkiewicz. Like HTERM, it also features a wide range of baud rates (from 75-9600), and in addition uses the terminal features of the H/Z-19 and the H/Z-100. It can send and receive ASCII files.

The program comes in three executable versions, one for the H/Z-100, another for the standard H/Z-89 (at port 330Q), and a third for the the H8 (at port 340Q). The source is also supplied for customization, but you can configure it for your own unique CIS (or Source) ID and password simply by using the built in patch routine.

H/M/ZPLINK

These programs are adaptations by Dr. William Moss from an original CP/M user group program, PLINK. They are simple and easy to use terminal programs, well adapted to the Heath hardware. Since they are all in the public domain, you might want to try one of them out before one of the more complex programs.

MDM7nn (MDM710, MDM711 and up)

Irv Hoff's MDM711 (as of late July) is the latest chapter in a continuing series of modem programs based on the Christensen MODEM protocol which run on CP/M computers. Although many versions of this program exist for different machines, the ones pertinent to Heath are M711H8 for the H/Z-89 and H8, and M711Z for the H/Z-100. The program is highly evolved and has many sophisticated features, yet it is in the public domain for all of us to use and enjoy.

This program, like the predecessors, is optimized around the block transfer mode feature, and the capability is quite impressive in that mode. Single and batch transfers are possible for any number or size of files, ASCII or binary, in either a checksum or the (preferred) 16 bit CRC mode.

While error checked file transfer is the major feature of the program, it has menu selectable auto dial and re-dial for the PMMI (original) and now with versions 7.10 and later, including the Hayes Smartmodem. The program is good in housekeeping with a built in ERA and DIR function, and the ability to re-log both drive and user. The buffered printer output allows easy error- free hard copy logging as well.

If you use CP/M, particularly with a Smartmodem, you should investigate this program. Be sure to get all relevant files, as the package includes the source code, a DOC file, and a phone number directory overlay, as well as the H/Z-89 and H/Z-100 overlay files. Check for the latest version, as there are frequent updates.

CTRM89

CTRM89 is Bill Pearson's neat 'C' language program for the H/Z-89 and H/Z-100, under CP/M. While it is basically a very easy to use terminal program, it also has a block mode MODEM protocol feature, and uses the 25th line for prompts. Like the H/M/ZPLINK series, it is a good start, but more so because it is painless to get going, and it has the bonus of the error checking.

YAM

YAM (for Yet Another Modem) is a very unique modem program, as it is one with a broad range of useful features, yet it was specifically written around the H/Z-19 (H/Z-89) and it's terminal features. Written by Chuck Forsberg in BDS 'C', YAM can be customized readily, using the available source code.

The standard features include both ASCII and block mode file transfers, with both the Christensen as well as the CIS 'A' protocol. Thus, this program is equally adept for use with both BBS and RCPM systems, as well as it is with CIS. A slick circular buffer technique is used for ASCII saves, allowing re-examination of the contents as well as save to disc.

Housekeeping is a big plus of YAM, as it supports intelligent DIR functions as well as a transmission time calculation, at your baud rate. And, it can TYPE disc files, even in a squeezed format, as well as do CRC checks of disc files.

It can be optionally compiled for Smartmodem dialing, if this is desired, which is done from a disc number library. An RCPM style host version can also be compiled, as 'XYAM' (Xmodem YAM), and offers the YAM features for remote system use. YAM may not be a program learned overnight, but it can be a source of fascination, particularly with the compilation options.

Other programs

The above comments summarize the programs of Table I for their features of operation. It is recognized that this may not be all the programs which run on Heath machines, just those which could be researched for this article. While preparing this article, a number of other suppliers of programs for Heath computers were contacted, unfortunately with no reply. The interested reader can write for information on these programs listed in Table II.

Summary

Writing a broad and easily understandable summary of modem packages is a difficult job, but the information above should lead you to a choice, hopefully one which will be right for you.

Some qualifiers must be placed on the information given, particularly with the use of Table I. Everyone's needs are different, to at least some degree, so take the time to evaluate the programs for your intended type of use. Don't rush to any decision without serious thought, and by all means do compare experiences with fellow users.

You are now ready to go on line with a program and modem, but some final words of advice. If you are new to all of this, please, do be sure to read thoroughly all of your program documentation to understand how it works, before calling your BBS. Very likely, the environment of the BBS you call will be new, as well. Be prepared to **save** everything you see on its documentation as you logon. Read that carefully off line, as well, and you'll then be able to get more from it the next time.

Don't be afraid to ask questions of the system operator. But, be sure you've read everything first beforehand, as often the answers are there somewhere. Usually, first time users see a special file to tell them system particulars, so be prepared for it.

That's about it, so welcome to the world of modeming. Drop by and say hello on the BHEC system (301-661-2175), or just have fun on your local one. Be with you next time with some good words on the CIS system. In the interim, by all means, do have fun!

Acknowledgements

My thanks goes to all of the vendors who supported this project and demonstrated their commitment to the Heath community. Also, my particular appreciation for helpful inputs on these articles goes to program authors Dr. William Parke, Bill Pearson, Will Pierce, and to Tim J. Pugh, Jr. Finally, thanks to my fellow BHEC system operator Charlie Schnepf, for insight and encouragement.

In This Issue Read About:

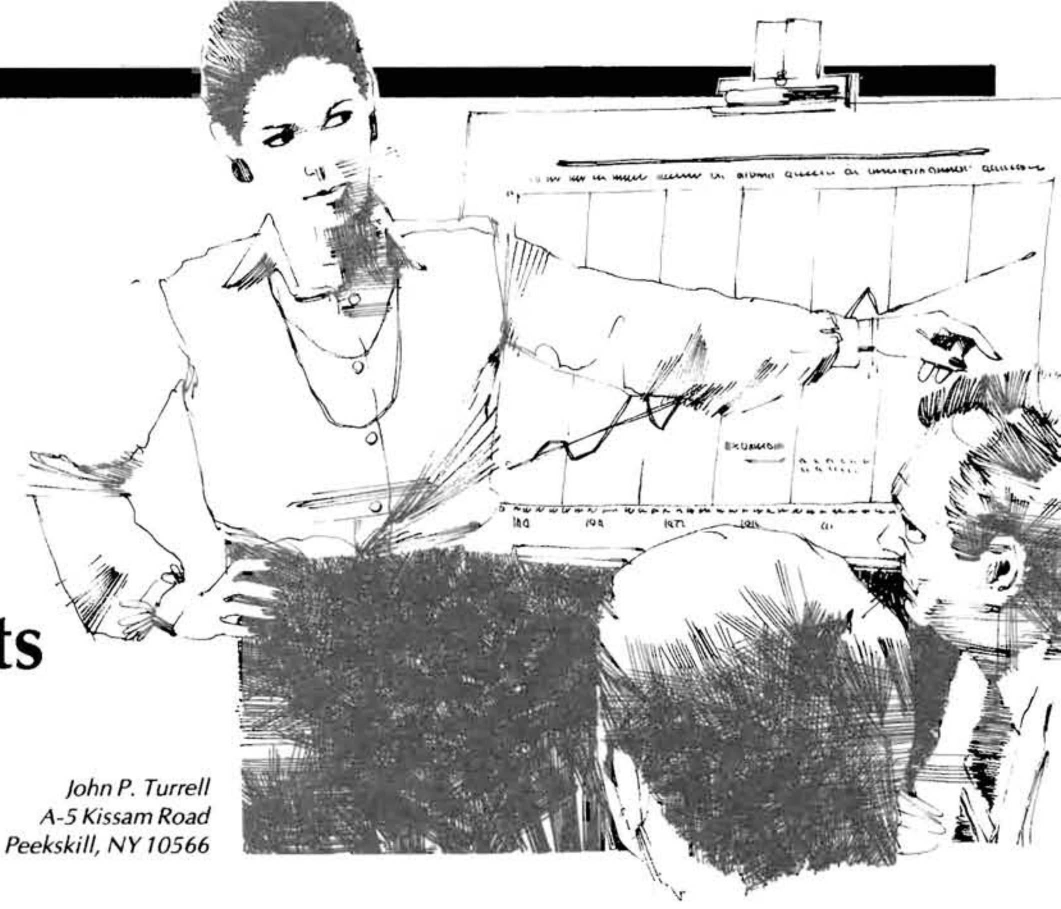
The HDOS Directory Structure

by David Pelowitz see page 74

And Coming In The Next Issue:

The CP/M Directory Structure

Three Usable Spreadsheets



John P. Turrell
A-5 Kissam Road
Peekskill, NY 10566

Picture a large sheet of paper that has been ruled to allow many rows and columns forming boxes. Within these boxes can be placed numeric quantities. Now develop a system of commands that will complete mathematical functions among the boxes. Add the ability to label the boxes, show the entire sheet on a CRT, save and recall the entered sheet, print the sheet, and you have an electronic spreadsheet. The uses of sheets can vary from simple house budgeting to tax preparation to rather elaborate business or scientific sheets.

The use of paper, pencil, and calculator can do the same job at less cost. The advantage of spreadsheets is that a single numeric value can be changed and all the calculations redone, allowing for multiple variations of the same mathematics without redrawing the whole sheet as would be done with paper and pencil. The sample sheet (Fig. 1) shows how common charges, which must be done yearly, are calculated for a condominium group. Each unit charge is based on percentage owned multiplied by the total of Group I share plus insurance to which is added one twenty fourth of the reserve fund.

The Heath/Zenith H/Z-89/90 and H8 users have several potential software packages that can be used on their systems. This article reviews Multiplan, Ver. 1.05 (MicroSoft), SuperCalc, Ver. 1.06 (Sorcim), and ZenCalc, Ver. 3.0 (The Software Toolworks), three electronic spreadsheets available for the Heath/Zenith systems.

Each of these sheets are of approximately the same size (Multiplan 255 x 63, SuperCalc 254 x 63, ZenCalc 255 x 52) with varying abilities as to what they accomplish, but all with the necessary mathematical basics. Their list prices vary, $49.95 (ZenCalc), $250 (Multiplan), and $295 (SuperCalc). They each are available in CP/M versions and in the case of ZenCalc, also HDOS. System requirements vary: Multiplan - 64K memory and disk capacity of 107K for its software, SuperCalc - 48K memory and disk capacity of 45K, and ZenCalc - 48K memory and disk capacity of 40K. The disk space is only for the working programs and overlays and does include space for saved files or the install files used on Multiplan and SuperCalc. Multiplan can be used without its 40K help file lowering the disk requirement to 67K. Multiplan and SuperCalc support a variety of terminals (Heath/Zenith 89/19 included) while ZenCalc has

```
        1         2          3          4          5          6          7
 1
 2
 3                     Condominium Group I       1983
 4
 5                     Group I share          $2,596.70
 6                     Insurance                $100.00
 7                     Reserve                  $240.00
 8
 9
10
11            A-1        4.18375                $122.82
12            A-2        4.16375                $122.28
13            A-3        3.95375                $116.62
14            A-4        4.49375                $131.18
15            A-5        4.18375                $122.82
16
17            B-1        4.62375                $134.69
18            B-2        4.39375                $128.49
19            B-3        4.18375                $122.82
20            B-4        4.18375                $122.82
21            B-5        4.18375                $122.82
22            B-6        4.18375                $122.82
23            B-7        4.18375                $122.82
24            B-8        4.18375                $122.82
25            B-9        4.18375                $122.82
26            B-10       4.31375                $126.33
27
28            C-1        4.18375                $122.82
29            C-2        3.95375                $116.62
30            C-3        3.95375                $116.62
31            C-4        3.95375                $116.62
32            C-5        3.95375                $116.62
33            C-6        3.95375                $116.62
34            C-7        3.95375                $116.62
35            C-8        4.18375                $122.82
36            C-9        4.31375                $126.33
37                                            $2,936.70
```

Example Spreadsheet (Multiplan)

Figure 1

been written specifically for Heath/Zenith systems.

Spreadsheets are used by positioning the cursor box (generally backlit) to the cell that is to have data entered and then typing the entry. The entry is made on a "bottom line" entry field and is not placed into the cell until commanded to (usually by pressing the RETURN). After this, the user positions the cursor box to the next desired position and repeats the above procedure. Each spreadsheet has its own rules for indicating what kind of entry is being entered (numeric, formula, or string). Since these are numeric spreadsheets, numerics are entered without a prefix. Formulas may require some first character to indicate that it is a formula. The same is usually true for string data (ZenCalc does not require a prefix). Commands, such as GOTO,SAVE, LOAD, BLANK, DELETE, etc. require a prefix in SuperCalc and ZenCalc but require just the first letter of the command in Multiplan. Multiplan and ZenCalc allow cursor direction to cause entry of data and move the cursor, a feature which allows for one less keystroke.

The sheets use coordinates to designate the cells, in ZenCalc and SuperCalc numeric rows and lettered columns. Multiplan uses numerics for both columns and rows. The designation of cells for two of the sheets (SuperCalc and ZenCalc) is column/row. The first cell becomes A1 (first column "A" first row "1", "a1" in ZenCalc). Multiplan uses row/column for designating cells, therefore the first cell becomes R1C1 for the same cell. This adds additional key strokes for addressing Multiplan cells, but it is easier to remember the twenty fifth column as C25 rather than as "Y".

While ZenCalc saves its sheet in ASCII, SuperCalc and Multiplan save theirs encoded. Both of these programs can save in ASCII for use by other programs (text editors) but the encoded form is rather difficult to understand. Multiplan can also save a symbolic file for exchange with other programs and read a VisiCalc file (prepared with VisiCalc).

The sheets can vary the number of digits right of the decimal point; Multiplan from 0-15, SuperCalc 0,2, or floating, ZenCalc 0-13.

Special Features

The three programs have features (see chart) that vary one to the other. Described here are some of the features that are enough different to warrant special mention.

Multiplan has the ability to sort data by row or column, a feature unique to this spreadsheet. The sort will sort numeric and/or alpha in ascending or descending order with equal ease. This can be helpful in instances where order of magnitude is important such as in budgets.

Multiplan has the ability to link data from a stored spreadsheet onto the current spreadsheet (external reference). This is useful if detail sheets have been prepared and a top sheet is to show the summary of the various detail sheets.

I have used this for a financial worth program. The detail sheets contain stock and bond information on one, savings on another, checking on a third, retirement on a fourth, and personal property on a fifth. These detail sheets can be updated as needed while the top sheet is not disturbed. When the top sheet is called, it reflects the most current data from the detail sheets. Multiplan's method of doing this is the use of named cells. A cell or a group of cells can be named with an alpha string (such as CURRENT TOTAL) and saved. Within the current sheet, an external reference is requested and the sheet name (file name) and cell name are entered. Multiplan will then open and read that data, and enter it into the current sheet.

The naming of cells also allows for easily understood formulas. If a column is named SALES and another OVERHEAD, the total sales

Comparative Features

	Multiplan	SuperCalc	ZenCalc
Mathematical			
Add	●	●	●
Subtract	●	●	●
Multiply	●	●	●
Divide	●	●	●
Exponential	●	●	●
Percent	●		
Mod	●		
Mathematical calculations			
Common log	●	●	●
Natural log	●	●	●
Square root	●	●	●
Average	●	●	●
Absolute value	●	●	●
Integer	●	●	●
Sum	●	●	●
Max	●	●	●
Min	●	●	●
Net percent value	●	●	●
Count (of items)	●	●	
Standard deviation	●		
Round	●		
Trigonometric functions			
Sine	●	●	●
Cosine	●	●	●
Arctangent	●	●	●
Tangent	●	●	
Arcsine		●	
Arccosine		●	
Logical			
Greater than	●	●	●
Less than	●	●	●
Equal to	●	●	●
Lookup (table)	●	●	●
PI	●	●	
IF	●	●	
NOT	●	●	
AND	●	●	
OR	●	●	
True	●		
False	●		
Index	●		
Sign (n)	●		
String Functions			
Len ()	●		
Mid ()	●		
Value ()	●		
Repeat ()	●		
Concatenation	●		
Formatting			
Scientific notation	●	●	●
Graphic display (using ***)	●	●	
Choice of numeric alignment	●	●	
Choice of text alignment	●	●	
Comma in numbers (1,100)	●		●
Dollar sign in numerics	●		
Commands			
Blank cell contents	●	●	●
Copy cell to	●	●	●
Delete row or column	●	●	●
Edit cell	●	●	●
Format	●	●	●
Help	●	●	●
Goto	●	●	●
Insert row or column	●	●	●
Load from disk	●	●	●
Lock cells	●	●	●
Save to disk	●	●	●
Print	●	●	●
Clear screen	●	●	●
Move row or column	●	●	
Split screen	●	●	
Name cells	●		
Sort	●		
External reference	●		
Remove borders		●	
Screen/Cursor movement			
Uses H/Z-19/89 keys	●	●	●
Uses shifted keypad	●	●	●
Page up/down/left/right	●		●
Move to top/bottom one keystroke	●		●
Move to right/left one keystroke			●
Unshifted keypad		●	●

could be named TOTAL SALES and calculated by the formula SUM(SALES). The same could be done with the OVERHEAD (SUM(OVERHEAD)). A calculation for profit would be TOTAL SALES–TOTAL OVERHEAD. These formulas are very easily understood.

Copying of cells (formulas) is also different with Multiplan, the copy command allowing duplication down or right of the cell a specified number of times (limit the sheet size). Although SuperCalc has a replicate command, it is not as easy to use.

ZenCalc uses a technique to copy or duplicate cell content with the use of "pick" and "put". The "picked" cells are placed in a buffer and the cursor moved to the area where they are to be "put", and placed there by single keystroke. This makes it much easier to duplicate partial rows or columns than in the other two spreadsheets. This buffer also allows for "goof" correction as deleted cells, rows, or columns are stored in the buffer and can be put back (if no other deletion has been performed before the discovery of the "goof").

Relative cursor addressing is another Multiplan feature. By stating a formula as being equal to some manipulation of cells referenced to the current cursor position, duplicating formula is very simple. In the sample spreadsheet (Fig. 1), the unit charge is calculated as:

(R5C5+R6C5)*RC[-2]%+(1/24*R7C5)

(Group I share + Insurance) * number two cells to right /100 + (Reserve /24). The use of relative reference within this formula allowed the copying down of the formula 25 times to complete the worksheet after blanking the two cells (in rows 16 and 27) that are the separation of groups.

Although all of the spreadsheets allow insertion of a row or column, SuperCalc and ZenCalc allow only one insertion at a time while Multiplan allows insertion of any number at one time. This is a saving of time if the spreadsheet has to be expanded to any great extent.

SuperCalc has the ability to load or save partial sheets from disk rather than the whole sheet. It may not be as useful as it appears since the coordinates must be known to use the load feature and it is not usual for the operator to remember these coordinates.

Multiplan allows cell content alignment either right, left, or centered which allows more appealing printouts. SuperCalc allows right or left alignment while ZenCalc does not give an option.

Multiplan allows for the sending of a control string to the printer that could set printer spacing or type of print. SuperCalc also allows for this function but it is done at installation and can not be changed for an individual print request. Additionally, Multiplan allows selection of the margin top and left so that offset sheets can be centered.

SuperCalc has the ability of removing the coordinate borders from the CRT so that the sheet can be seen as it will be printed. It also has more trigonometric functions than either of the other two spreadsheets.

SuperCalc and Multiplan allow split screen whereby a portion of the screen can be divided into a separate worksheet. SuperCalc allows one split either horizontal or vertical while Multiplan allows up to eight splits. Split screens can be used to keep a cell visible while entry is made on other portions of the screen. Such a use might be to keep the tax due and the refund due visible while one enters data into a tax preparation program. The user can instantly see the effect of various deductions on the taxes being paid. Multiplan allows the window created by the split to be bordered (enclosed in lines) using graphic capabilities of the terminal.

All the programs can adjust the column width upon user command and each has a different maximum width (32, 126, 77 - Multiplan, SuperCalc, ZenCalc, respectively).

All can plot graphic data for histograms. Multiplan and SuperCalc use a string of "*" characters while ZenCalc uses a graphic symbol. The width of the column will determine the sensitivity of the plot.

SuperCalc and Multiplan can write protect cell contents. SuperCalc must protect the cell on an individual basis while Multiplan can protect all formulas on the sheet via a single command. Multiplan has a command (↑ F) that instructs movement to the next unlocked cell, a convenience that allows rapid movement to cells that require data entry.

The legend on the bottom of SuperCalc displays the cell contents and the attributes of the cell (protected, text, memory, last column/row, etc.) as well as the current content if any. This is helpful in editing the cell contents, particularly if they are protected. Multiplan does not indicate that a cell is protected and if edited, a beep and message on the terminal after the attempt is made indicates that the cell is protected.

Editing of formulas can be done on all three sheets while SuperCalc and Multiplan also allow editing of text.

Multiplan has string manipulation abilities similar to BASIC.

Use of H/Z-89/19 Keyboard

SuperCalc uses it the least, basically the arrows for changing from cell to cell and the blue (keypad toggle), red (help), and white (recalculate) keys. Its strong point is that the keypad can be unshifted for numeric entry. This allows for extremely rapid numeric entry once the cursor is set to move in the correct direction.

Multiplan utilizes the shifted keypad, the function keys, and the colored keys. Some of the keys are assigned functions not designated by the keycap. IC and DC are character left and right respectively while IL and DL are left and right word. These are not easy to remember and make for some confusion. Multiplan uses the arrows as they are designated, while the home key returns to the upper left most cell.

ZenCalc fully utilizes the special keys as one would expect of a program written with our terminals in mind. If the reader is familiar with the PIE editor, he will recognize some of the key usage as this spreadsheet uses all the f-keys, blue, red, white, and all the keys on the keypad. To move a cell right, press right arrow, to move 6 cells right press ENTER, type a numeric argument, and press the right arrow key. To move all the way to the right side of sheet press shift and right arrow. This full utilization of the H/Z-89/19 keys makes rapid movement possible. The keypad can be unshifted by a command for numeric entry while the "ENTER" key will move the cursor downward. The red key does "PICK" and the white key does "PUT".

Help

All three programs have a help "key" that will bring some information to the screen to assist the user in understanding the possible choices he can make. Multiplan's help outshines them all. It is 40K on disk and reads like a book. It is capable of being called by a particular topic and then paged forward and backward, changing topics, or resuming the spreadsheet entry. It covers all the subject matter that is within Multiplan.

SuperCalc's help is a brief description of what each command does. Much of it is only available after the initial command is entered. This allows the use of help only when the operator is "stuck". It's briefness along with an inadequate index in the manual makes it harder to solve questions when they arise.

ZenCalc's help is similar to SuperCalc's but is divided into subject matter and can be called at any time, not just when stuck.

Manuals

The manual that is instructive can make a program useful or useless, depending on its ability to present the necessary information.

The respective manuals that are with each product are generally good. SuperCalc's is one of the best from the standpoint of programmed learning with the text leading the first timer step by step through the fundamentals. It also includes some very good examples of more advanced techniques (with disk copy). It however lacks an index for quick reference. Multiplan has an excellent index and comes with a quick reference card. Referencing, when using a spreadsheet, is very necessary as it is difficult to recall all the possible commands and usage. This is particularly true when the "help" command shows only the limited prompts of what the commands mean.

ZenCalc's manual also has examples of programs that reside on the disk and it contains a good reference. The manual's examples are not as well explained as those of SuperCalc.

Memory Usage

All the spreadsheets expect the user to start the sheet at the upper left and work evenly downward or toward the right maintaining a rectangular shape. As a cell is addressed or formatted, that cell and any cell above and to the left not previously used is assigned space in memory if not already assigned. If the user makes entries spread out leaving many empty cells (particularly whole rows or columns) causing a large rectangle to be formed, a memory full situation will occur. This will surprise the user who sees many free cells. In all spreadsheets it is wise to keep the entries in a rectangle.

A problem is encountered with Multiplan and ZenCalc if an entry is attempted at the last cell on the spreadsheet in that a memory full message will result. The reason for this is that all the cells previous to this cell are automatically formatted (allocated), thereby using all memory available plus. Doing the same with SuperCalc causes no problem. It is not because this program has that much more memory available, but because the program does not initially allocate as much space for each cell as does Multiplan or ZenCalc.

The amount of memory used by a spreadsheet is important. If a large amount of data is to be entered, will one spreadsheet handle more data than the others? A test sheet was written with each program to fill the sheet with ASCII until the memory full message appeared (using 64K CP/M). The results show the following: Multiplan with 1506 cells, SuperCalc with 1584 cells, and ZenCalc with 1275 cells. Different types of data (numeric or formula) can take more or less cells than ASCII and result in more or less cell usage. An additional problem occurred with Multiplan in that all the memory was used and the sheet could not be saved. One would suspect that the overlay for saving cannot be loaded. The only option is to delete a row or column to free up memory with the possible loss of much data. It is wise to keep on eye on the bottom line that indicates memory remaining. All three spreadsheets have some way of indicating memory left. Multiplan uses percent while the other two sheets use numeric size, SuperCalc in "K" (21K) while ZenCalc indicates bytes (21254). Different types of data (numeric or formula) can take more or less room than ASCII and result in more or less cell usage.

Disk Usage

In a test of disk usage, a sample spreadsheet was entered into each program and saved. Multiplan required 5K, SuperCalc 8K, and ZenCalc 6K. The test program to fill maximum system memory when saved required 13K for Multiplan, 29K for SuperCalc, and 19K for ZenCalc.

SuperCalc adds a file extension (CAL) to files if one is not stipulated at save and supplies this extension if one is not given on loading. This allows the user to forget the worry of using extensions while at the same time protecting him when he reads a disk directory from erasing an apparently unknown file.

Weaknesses

Multiplan's calulating time is slow. In a test of calculating speed, a sheet 6 columns by 16 rows that averaged the columns and rows was made. One value was changed on the spreadsheet and timed to see the speed of calculations. Multiplan took 18 seconds to complete the calculation, SuperCalc took 2 seconds, and ZenCalc took 4.8 seconds. SuperCalc is not able tp properly handle backward references (a cell that references another cell further down on the sheet). If cell A1 contains a reference to cell A5 that contains the formula B2+B3, a change in B2 that changes A5 (via recalculation) will not change A1.

The use of the arrows for rapid cursor movement frequently will result in some other command or an alpha character appearing. The reason for this was discovered in the ZenCalc manual which included a nice explanation. The CP/M operating system 2.2.02 does not have a type-ahead buffer so that subsequent commands lose the leading character of the multi key command if they are entered before completion of the previous command. HDOS and CP/M 2.2.03 have type ahead buffers and this will not occur. Multiplan has the most frequent occurence as it appears to respond slower than SuperCalc or ZenCalc. Because of this fault, I find it easier when using Multiplan to use the control codes (CTL X, etc. which respond faster) to move the cursor. These are the same as those used in WordStar thereby making it necessary to only remember a single set of commands (these are workable in SuperCalc also).

Summation

The three spreadsheet programs are different, each having its strong and weak points. Summarized:

	Strengths	Weakness
Multiplan	sorting external references naming of cells string manipulations good "help" good disk space utilization easy to write formulas split screens	slow calculations slow response
SuperCalc	most trigonometric functions fastest calculations split screens	poor disk utilization poor backward reference
ZenCalc	good use of keyboard fast calculations good disk space utilization good duplicating HDOS version lower price	less features

About the Author:

John Turrell *is s senior systems engineer for Technicon Instruments Corporation, Tarrytown, N.Y., a major manufacturer of automated biochemical analyzers. He has a B.S. in biochemistry and has worked in the biomedical instrument field for 16 years. His initial computer use was with an IBM System 7, used as an industrial controller in QC testing and its 16 bit assembly language. This sparked an interest and an H8 was built to learn the basics, it was added to, and then an H89 was purchased second hand. Computer uses have involved learning BASIC, 8080 assembly, word processing, and any other useful software. Application has been in data reduction, algorithm design for his company's systems, and general personal uses including games, music, voice synthesis, word processing, electronic spreadsheets, and data management.*

NEW FROM FLOPPY DISK SERVICES, INC.

the H-89 TWOET SYSTEMS

We've got a great idea for your H-88, 89 or 90. It's a dual *internal* half height drive system. Two of our half height 5¼" drives can replace your built-in disk drive, doubling your information storage capacity.

Floppy Disk Services provides you with everything you need. That's two double-sided, double or single density, half height drives in either 48 or 96 tpi format, ail hardware, cables and power connector adaptors. And most important, you get easy, step-by-step instructions, in the Heath/Zenith tradition of good, clear documentation.

We've thoroughly tested the TWOET/Heath set-up. Remember that a double sided 48 tpi will work perfectly as a single sided drive right out of the box! Hard or soft sectored—so you can even use this system with your H-17 controller. And of course we have the software drivers (additional cost) to run 48 or 96 tpi double sided, single density drives on the H-17.

Model TWOET 455
2 Shugart SA-455 half height
48 tpi double sided
All hardware
Metal, shielded mounting plates
Data cable with chassis connector
Power 'Y' connector
Complete documentation
Price** .... **$595.00 complete

Model TWOET 465
2 Shugart SA-465 half height
96 tpi double sided
All hardware
Metal, shielded mounting plates
Data cable with chassis connector
Power 'Y' connector
Complete documentation
Price** .... **$695.00 complete

Wondering what to do with your internal drive if you buy this system? Here's the solution. If you purchase a dual half height system for your Heath computer from Floppy Disk Services, just include an extra $50.00 plus shipping and receive a single 5¼ case with power supply *and* data cable ready to receive your SIEMENS internal drive! The case with data cable is normally a $70.00 item. And the cable that comes with your TWOET system includes the external chassis disk I/O connector.

Due to production deadlines, prices in this ad are 2 months old, so we encourage you to call us for current prices and new product info. Prices and specs subject to change without notice.

Dealer inquiries invited.

PAYMENT POLICY — We accept MasterCard, VISA, personal checks and Money Orders. We reserve the right to wait 10 working days for personal checks to clear your bank before we ship. All shipping standard UPS rates plus shipping & handling. NJ residents must add 6% tax.

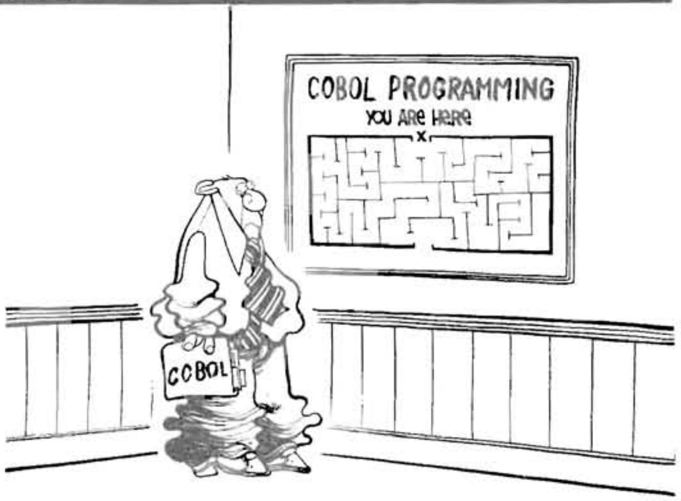
Toll Free Order Line:
(800) 223-0306
Tech Help or Info:
(609) 799-4440

H-88/89/90 are registered trademarks of Heath Corp.

FLOPPY DISK SERVICES™ INC.

741 Alexander Rd. Princeton, NJ 08540

COBOL Corner I



H. W. Bauman
493 Calle Amigo
San Clemente, CA 92672

Introduction

Beginning this month, there will be a regular feature in REMark to help programmers work with Structured COBOL. Why COBOL? In the "Real World", if you are out and working with computers, you will most likely come into contact with COBOL. Just read the help-wanted ads in your local paper for jobs in the Computer Science Field. Most require knowledge of COBOL. It is used by the U.S. government and large companies because it is a computer language that has standards! We will cover the equivalent of three (3) semesters, at the college level, of COBOL study in this series.

The first computer language used by most micro-computer owners is BASIC. BASIC is not necessarily the easiest language to learn but it is the one furnished with or purchased with the first computer. When BASIC is an Interpreter, not a Compiler, it provides an easy-to-use programming environment. After learning a few "key word" statements (usually on a hit-or-miss basis), you can be running a simple BASIC program. So, the programmer feels BASIC is easy to learn, but is he/she writing "good" and useful programs? As this "COBOL Corner" series progresses, we are going to learn what constitutes a "good" program!

Background

After using BASIC and learning some of BASICS's limitations (Is it capable of providing portable and upgradable programs that can be maintained on other computers at some other time?), many programmers begin to look for a higher level programming language that will provide them with greater power over the computer. Which high-level language should you choose? Programming languages are complex and difficult to evaluate objectively. Even experienced experts have not agreed on a criteria to evaluate a language. What should you do? The best policy I have found is to learn many languages well, and then through your own experiences, determine which one or more languages meet your need. I have studied BASIC, FORTRAN, PASCAL, and COBOL through the Computer Science Department at my local college. I then integrated these languages into my Investment Business and narrowed my interest to two languages, as of this time. BASIC (especially CBASIC) for my short length, short life, quick and dirty programs, and Structured COBOL for my complex, long life, easy to DEBUG, change, and maintain programs. I have really enjoyed working with COBOL and like the idea of sharing my programs with others, even if they have different brands or models of computers.

Structured COBOL

COBOL, like the English language, is a dynamic language that is continuously changing and evolving. Similarly, computers and data processing are dynamic fields that are also going through continual changes. American National Standards (ANS) issued COBOL standards in 1968 with revisions in 1974, and they are working with future upgrades.

"COBOL Corner" will be working with Contemporary, Business-Oriented COBOL using most of ANS-74 Level-I and some of the Level-II Implementations. This will provide you with a "portable" source code program; thus, if your friend or associate has a brand "X" computer running ANS-74 COBOL, you can share program source code that may have taken you months to design, debug, and upgrade.

COBOL is a compiled language; that is, (1) you create the source code with an editor, (2) you compile the source file with a COBOL compiler into a relocatable object file, (3) you link the object file using the COBOL Linking Loader into an Absolute file that is executable, and (4) you execute the program. Do not let this worry you. We will explore the procedure in great detail in "COBOL Corner". What this means is that unlike assembly language programs that generate a runable machine code, COBOL requires four steps to obtain a fast, executable program.

"COBOL Corner" is going to use a "Tandem" presentation of programming concepts and coding, using sample programs that will increase the program complexity and provide improved program structure. I believe that the best and only way to learn to work with a high-level language is to review the language reserve words and the syntax by WORKING and READING sample programs. This will hold your interest and imbed the language in your mind. Any programmer seriously wanting to learn COBOL will not learn by just READING "COBOL Corner" articles. He MUST WORK with them on his computer! I will provide many differing, as well as increasing complexity, sample programs that will develop a structured programming style and expand on possible applications of COBOL. This will take many months, but it will demonstrate how easy it is to learn a new language and "Good" programming techniques.

Hardware and Software

One of the best arguments for COBOL's use by HUG members is the availability of high quality Heath/Zenith COBOL software for Heath/ Zenith computers using either HDOS or CP/M operating systems at a reasonable cost. Both the Heathkit and the Heath/Zenith Computer Catalogs list the Microsoft COBOL-80 Software which exceeds the ANS-74 minimum implementation level for $395.00. This software will run on the H-8/H-19, H/Z-89, or Z-90 computers with at least 48K RAM (I recommend 64K RAM), two (2) disk drives (I use three (3) drives), and either the HDOS or CP/M operating system (I use CP/M). The larger RAM and disk access allows you to run the larger "COBOL Corner" programs with 600 to 2000 lines of COBOL coding as we expand our knowledge. To work along with "COBOL Corner", you will need a printer capable of 132 columns; such as, the H/Z-25, MX-100, or MX-80 using compressed print.

Note: In the January through May, 1983 issues of BYTE magazine, Ellis Computing, 3917 Noriega St., San Francisco, CA 94122, (415) 753-0186, advertised their Nevada COBOL Software for $29.95 and stated that it is based on ANS-74 Standards with some Level-II features. They now have it available on Heath/Zenith soft or hard 10 sector diskette format. It is also available on CP/M 8" disks. I will be reviewing this Software to determine whether it has the implementations that we need to run our more complex sample programs.

There are other COBOL software suppliers for 8 and 16 bit computers available, but I do not have enough experience with them at this time to recommend them. I have added a Z-100 computer to my office and will review its usage of COBOL in future articles.

You will also need an editor to create your source files and data files. CP/M comes with ED.COM which will do the job. However, my preference is PAGED, short for PAGed EDitor, sold by Scott Witt, 79 Old Haverstraw Rd., Congers, N.Y. 10920, (914)268-6162. It is available for either HDOS or CP/M for $25.00 and well worth the price. Of course, there are many others. It depends on which one you get use to.

Closing

The next "COBOL Corner" will explain how to set up COBOL-80 on a computer, describe the general COBOL program component structure/order, and start with a simple COBOL program to show you the program format. So, if you do not have a COBOL system, be sure to obtain the required hardware and software before the next "COBOL Corner".

I am by no means a COBOL "expert"! I find that I just keep learning more and more about Structured COBOL all the time. I know some HUG members could teach me many things about Structured COBOL; so, let's all input your suggestions and questions! Together, we will make "COBOL Corner" work for everyone. We will start with the basics and hopefully expand into programs that will help even the experienced programmers as they work along with us.

The following is a list of some of the COBOL programs we will be designing, coding, keying, and executing:

1-READ-PRINT

a) Read a Customer Transaction File and Print a Customer List.

b) Read an Employee Record File and Print Employee Address/Telephone List.

2-READ-COMPUTE-PRINT with TOTALS

a) Read an Order File, Compute the Sales Tax Amount and the Transaction Amount, and Print a Sales Report with a Total Transaction Amount and an Average Purchase Amount.

b) Read an Order File, Compute the Discount Amount and the Net Amount, and Print a Discount Report with two total lines. The first would Total the Number of Transactions, Total Purchase Amount, Total Discount Amount, and Total Net Amount. The second total line would show the Average Purchase Amount, Average Discount Amount, and the Average Discounted Purchase Amount.

3-READ-COMPUTE-PRINT with HEADINGS and TOTALS

(Note: This is not the longest, but it may be one of the hardest programs to design and understand.)

a) Read an Accounts Receivable File and Print an Accounts Receivable Report with a Report Heading, Page Heading, and Report Totals. We will count the report lines and direct the printer to start a new page when required with its own Page Heading.

4-READ-COMPUTE-PRINT with HEADINGS and TOTALS using nested IF STATEMENTS

a) This program will read an Employee Payroll Records File and print an Earnings Report Line for each employee. The program will compute the Total Hours, Shift Differential, Regular Earnings, and Total Earnings for each employee. Again we will have the computer determine when a new page will be required.

5-READ-PRINT with CONTROL BREAKS

a) This program reads an Employee Hours Record File and prints a Department Report and a Total Company Report with necessary new pages and headings.

6-READ-TABLE HANDLING-PRINT

a) Read a Payroll File and prepare an Employee/Department List using a hard-coded table of department numbers/names using Indexing and the Search Statement.

b) Read a Parts Number Record File and print a Parts Data List using a hard-coded table of Part Numbers/Units of Counting/ Part Descriptions/Price using Subscripts.

7-READ and EDIT a Transaction File and prepare a MASTER FILE

a) Read a Sales Transaction File, Edit the data to determine valid data only. Prepare a Sales Report using only the valid data. This will be done by using a Table and the Indexed By clause.

b) Again, read the Sales Transaction File, Edit the data to determine the valid data, and prepare a Sales Report using only valid data and use of a Table Handling method with Subscripts.

The above program list represents about one-half of the programs we will be working with, but this will give you some idea of the range of programming we will get into. We will be doing more with Editing Data Files, External and Internal Sorting, Upgrading Programs, Maintaining Programs, and Random Files.

But, I think I hear you saying, that is enough words; lets start doing something with COBOL! Will do! Next month!

SPECIAL NOTICE:

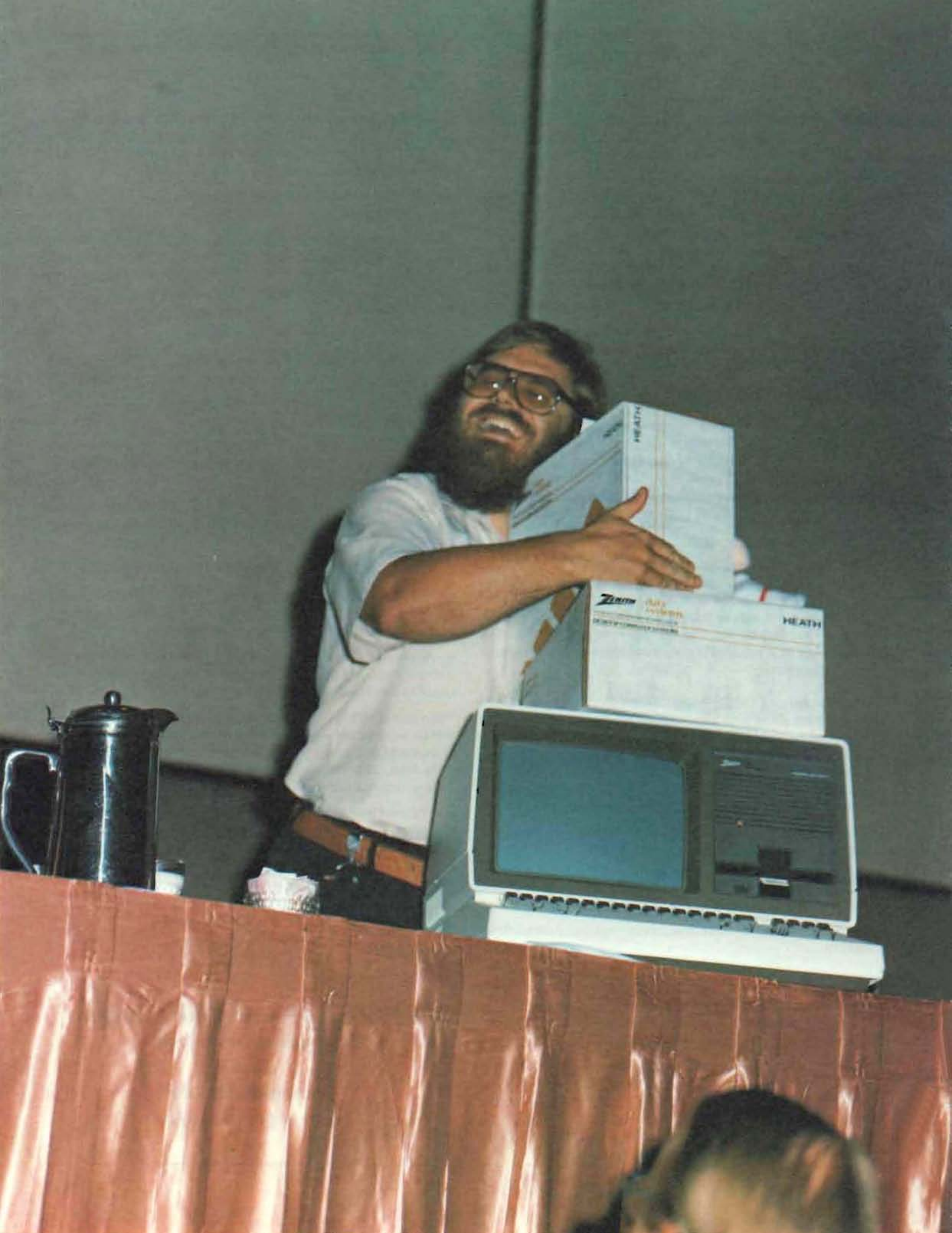
HDOS 2.0 Source Code Listings

are no longer available

and

are not scheduled for

another production run.


HEATH
ZENITH data systems
HEATH

2nd Heath Users' Group National Conference

Walt Gillespie
REMark Editor

Successful!

If any one word could be used to describe the Second National Heath/Zenith Users' Group Conference held recently at the Hyatt Regency O'Hare in Chicago, it would be 'Successful'. It was a success for the HUG staff, as many months of meticulous planning paid off. It was also a success for the 46 vendors who had brought their related Heath/Zenith products to the Hyatt that hot August weekend. But, it was an even bigger success for almost 1500 HUG members and their families who attended the three day conference and exhibit.

The weekend started for the National HUG staff around noon on Thursday, August 18, as we arrived at the hotel. As this early vanguard began their check-in, they were greeted by a few vendors and some HUG members who just couldn't wait for things to get rolling. With the arrival of trucks bearing show materials and additional vendors appearing on the scene hourly, preparations for the long weekend ahead began to take shape.

This year the entire downstairs area (International Level) was put to use for the HUG Conference. A group of six rooms with movable walls were opened up for the vendor exhibits. Rendering over double the square footage as the previous year, the Vendor Exhibit Area became the focal point of the Conference. Adjoining the central entrance hall to the exhibit area were four large conference rooms that were set up to accommodate the speaker and panel sessions scheduled for Saturday morning.

On the upper (entrance) level, HUGhers, staff wives, and family prepared the registration booth under the direction of Margaret Bacon, HUG Secretary. A room directly behind the Registration Booth, United A, was made ready for the NUA (Non-User Attendee) Lounge. This lounge would be put to good use over the following three days as a resting place for many weary feet and a haven from the hectic pace one floor below. Also on this level of the Hyatt, the Rosemont Ballroom was prepared for the Opening Ceremonies Friday evening and the Banquet and Grand Prize Drawings scheduled for Saturday evening.

Vendors from all over the country came prepared to "deal" ... and deal they did. A vendor who had exhibited items for three different computers at the West Coast Computer Faire earlier this year commented Sunday that, with only the items for the Heath/Zenith computers, he came within 5% of matching his total gross sales at the WCCF. But, I'm getting ahead of myself.

Late Friday morning and early Friday afternoon, the finishing touches were being put on the vendor booths. The Heath Users' Group booth was being completed by Nancy Strunk, Software Coordinator, and Donna Melland, REMark Assistant Editor. With boxes of HUG Conference 'T' shirts , HERO license plates, and numerous copies of HUG Library software marked at 30% off, the girls set about making things ready. Pat Swayne, HUG's Software Engineer, prepared and installed demos for HERO and an H/Z89 color board. John, Kurt, and Chris, three of my sons, were also on hand to assist vendors in moving their many boxes to their booth areas. As materials moved in, and booths took shape, the hall began to take on a festive atmosphere.

Gerry Kabelman, on loan from Zenith Data Systems, put the finishing touches on a random number generator program he had written. This particular program would post eight winning door prize numbers on color monitors placed at the back of the HUG booth. These numbers were displayed starting at 8 a.m. Saturday morning with new numbers posted every two hours until closing. The vendor response to the door prize request proved so great that it was necessary to increase the numbers drawn to 16 every two hours. Even this did not prove adequate to dispose of all the prizes, so special drawings and 'scavenger hunts' needed to be held.

During all this preparation downstairs, HUGhims and HUGhers were gathering upstairs waiting for the registration booth to open at 1:00 p.m. As each attendee signed-in they received a HUG Conference notebook, an "I'm a HUGGER" button, a hotel map with complete Conference schedule plus their personalized Conference name tag. (These name tags would prove invaluable for the next three days as attendees, vendors, and HUG staff tried to put unfamiliar faces together with some very familiar names.) After registration, everyone moved to the large staircase and lobby leading to the Vendor Exhibit Area which had not yet opened. Many new friendships were made as the excitement and anticipation of the coming events took place.

The vendors were not quite prepared for the onslaught that came when the exhibit area was opened at 3:00. Not that they didn't have plenty of merchandise, or some really super deals; it was the eagerness of the HUGgers to buy that took them by surprise. Record first day sales were encountered by many of the support vendors as attendees snatched many great bargains. After composing themselves, the sales representatives moved into the aisles to work on a one-on-one basis. The crowd was not as large as it would prove to be Saturday, thus giving attendees a chance to get answers to many questions.

The closing bell was sounded and the hall cleared at 7:00 p.m. This gave each attendee and the vendors a chance to freshen up and relax before the start of the evening festivities.

At 9:00 p.m. the Grand Opening and Award Ceremonies got under-way in the Rosemont Ballroom. These activities had been preceded by a cash bar warm-up at 8:00 p.m. Bob Ellerton, HUG Manager, officially opened the Second National Heath Users' Group Conference by hanging out a large "Yes We Are OPEN" sign. After a few remarks and a hand showing by local HUG clubs represented at the Conference, he proceeded to introduce the National HUG Staff. At this time, special awards were presented to persons who had made outstanding contributions to the Heath/Zenith community. Tom Jorgenson (Software Wizardry, St. Louis, MO) was presented the Vendor Award for his contribution as a hobbyist, a support vendor, and ZDS dealer. Pat Swayne (HUG Software Engineer) was presented the User Award for his many years of fine programming contributions and for furthering the programming education of many HUGgies. Mike Cogswell, President of CHUG (Capital Heath Users' Group), was presented with the Local HUG Club Award for CHUG's effort in sponsoring the First Regional Conference. Also, Larry Sites, of CHUG, was presented with a lifetime membership to HUG for being the longest active Heathkit customer currently a member of the Heath Users' Group.

Following these ceremonies Bob introduced Pat McNamara, Director of Marketing Communications for Zenith Data Systems. Pat's talk on the scope of ZDS advertising is covered elsewhere in this issue.

Bowing to a unanimous round of applause, ZIGGY, the ZDS mascot, entered the hall and helped bring the evening's festivities to a close. Later, many vendors and attendees were seen sitting around various locations in the hotel looking quite worn out. And there were still two days to go!

Kick off was scheduled for 8:00 a.m. Saturday morning. The HUG staff had a 6:30 strategy meeting (yes, 6:30 in the morning) to iron out any problems that might have come up the previous day. At 7:30, the Registration Booth opened with the Exhibit area opening at 8:00. It was at this point that the Conference activities really began to pick up steam.

Terry Jensen, HUG Software Developer, introduced the speakers of the various discussion groups. (Terry reports elsewhere in this issue on these talks.) All during the day, at least four discussion groups held meetings simultaneously. This approach was different than that used for the First HUG Conference which had featured consecutive speakers.

The action in the Vendor Exhibit Hall was much the same as Friday, except it was a much larger gathering. The display of winning door prize numbers had been timed to coincide with the changing of the discussion groups. As the time came near for new numbers to be displayed, the crowd in front of the HUG booth would swell to well over 300. Silence would fall on the area as excitement and anticipation covered the faces. And, as the new numbers came up, winners would step forward to claim their prizes and the groans of the less fortunate could be heard across the room. This scene would be repeated over and over both Saturday and Sunday. At 5:00 p.m., the closing was again announced.

Many HUGgers gathered at 6:30 p.m. in the United A & B rooms for a cash-bar preceding dinner. Members gathered at the entrance to the Rosemont Ballroom eager to deposit their Grand Prize Tickets into the drawing box. At 7:30 p.m. the doors were opened and the crowd surged into the very large dining room. This was the big event. The room filled quickly as those eligible for the prizes nervously waited that part of the program. After a fine meal of salad, prime rib, and dessert, Bob Ellerton called the gathering to order. Introductions of special guests and staff seated on the platform were given. They were, Tom Dornback, ZDS Vice President Software Development, Walt Gillespie, REMark Editor, and his wife Mary, Joe Schulte, President of Veritechnology (Heathkit Stores), Bill Johnson, President of Heath Company, Don Moffet, President of Zenith Data Systems, and his wife Sally, Bob, his wife Jill, and Mike Brenner, ZDS Product Line Director.

Now for the big event of the evening, the drawing for the Grand Prize Winners. Don Moffet drew the first ticket from the box and it was placed into an appropriate envelope for later announcement. Likewise, Bill Johnson and Joe Schulte drew tickets and placed them in envelopes.

It was now that Bob took the time to make a special award. He placed on the podium a small galvanized pail filled with red and white marshmallows. The bucket was labeled with large red letters ... 'BIT BUCKET'. This special award was presented to Mike Caddy, winner of last year's Grand Prize. Bob then recognized individuals who had traveled great distances to attend the Conference. From the Canary Islands, Spain was Luis Teja; from France, were Ambassador and Mrs. Yusif Kamal Zada; from Switzerland, Mr. Hans Van Velsen; and from South Africa, Mr. Peter Briggs. It was noted then that the National HUG Conference had truly become the International HUG Conference.

Bill Johnson, President of Heath Company, was introduced as the Keynote speaker for the evening. Bill's talk was centered on the history, growth, and future of micro-computers in the Heath/Zenith community. Special credit was given to individuals in the HUG community who had contributed to this growth. At the conclusion of Bill's presentation, ZIGGY was introduced to the gathering of HUGgers.

Now came the special time that had many twitching in their seats. But first, a special presentation was made to Bob Ellerton as this was his 34th birthday. Bob was presented with an authentic 'blubber cutter' from the HUG Staff. It was said that he should have received a harpoon for use on his charter boat (he is quite the fisherman in his spare time) but he might poke a hole in the floor during one of his (fishing) parties.

Bob again regained the podium and asked that Joe Schulte open his envelope and read the name of the winner of an H110 computer kit. The winner was Glenn C. Davidson of Oconomowoc, Wisconsin. Glenn was so excited after receiving his prize that he disappeared from the hall before a picture could be taken. Evidently he rushed to his room to start assembly that night.

Next Bill Johnson was asked to open his envelope and announce the name of the winner of a HERO I Robot kit. Bob Crowley of Parma, Ohio was announced as the proud new owner of HERO.

Then it was time to let everyone know just who would take home a ZW-120 (Winchester All-In-One) computer with two operating systems. Don Moffet moved to the podium and clearly announced that Ronald Armstrong of Waukesha, Wisconsin was the Grand Prize winner. A yell went up from the back of the hall and people along the path to the platform quickly moved aside. Ron moved immediately to the podium to claim his prize. Upon receiving congratulations from Don and Bob, a very excited Ron hoisted his new posession onto his shoulder and proceeded to maneuver through the crowd, back to his table. With approving applause, the evenings festivities were brought to a close.

Sunday morning things got off to a much slower start, but with pretty much the same schedule as Saturday. The big difference for this day would be the HUG General Meeting held at 2:00 p.m. in the Rosemont Ballroom. A very happy (sales wise) but very tired group of vendors sighed in relief when it was announced that the exhibit area would close one hour early.

At 4:00 p.m. it was over! Most were ready to go home, but the big question that had been asked all weekend, was still asked, "Is there going to be a HUG Conference III?" The answer is YES. The plans are already being made, so watch REMark for the announcement of the International Heath/Zenith Users' Group Conference III.

I cannot let any comments on the 2nd HUG Conference pass without noting some special individuals who helped make everything work: Jill Ellerton (Bob's wife), Teresa Jensen (Terry's wife), Laura Swayne (Pat's wife), Mary Gillespie (my wife) and sons, John, Kurt, and Chris; and Kris Young (Margaret's daughter). All these girls and guys worked very hard in keeping the Registration booth functioning and filling in where needed. I would also like to thank Jon Falkner (Heath Quality Assurance Dept.) for doing such a fine job as our official photographer. Jon took some 300 photos, many of which grace the pages of this issue.

So until next year .... Happy Computing!

With the Conference over, Terry and Pat pack up Donna to head home.

Heath/Zenith Users' Group

Another Conference View

Donna Melland
Assistant Editor

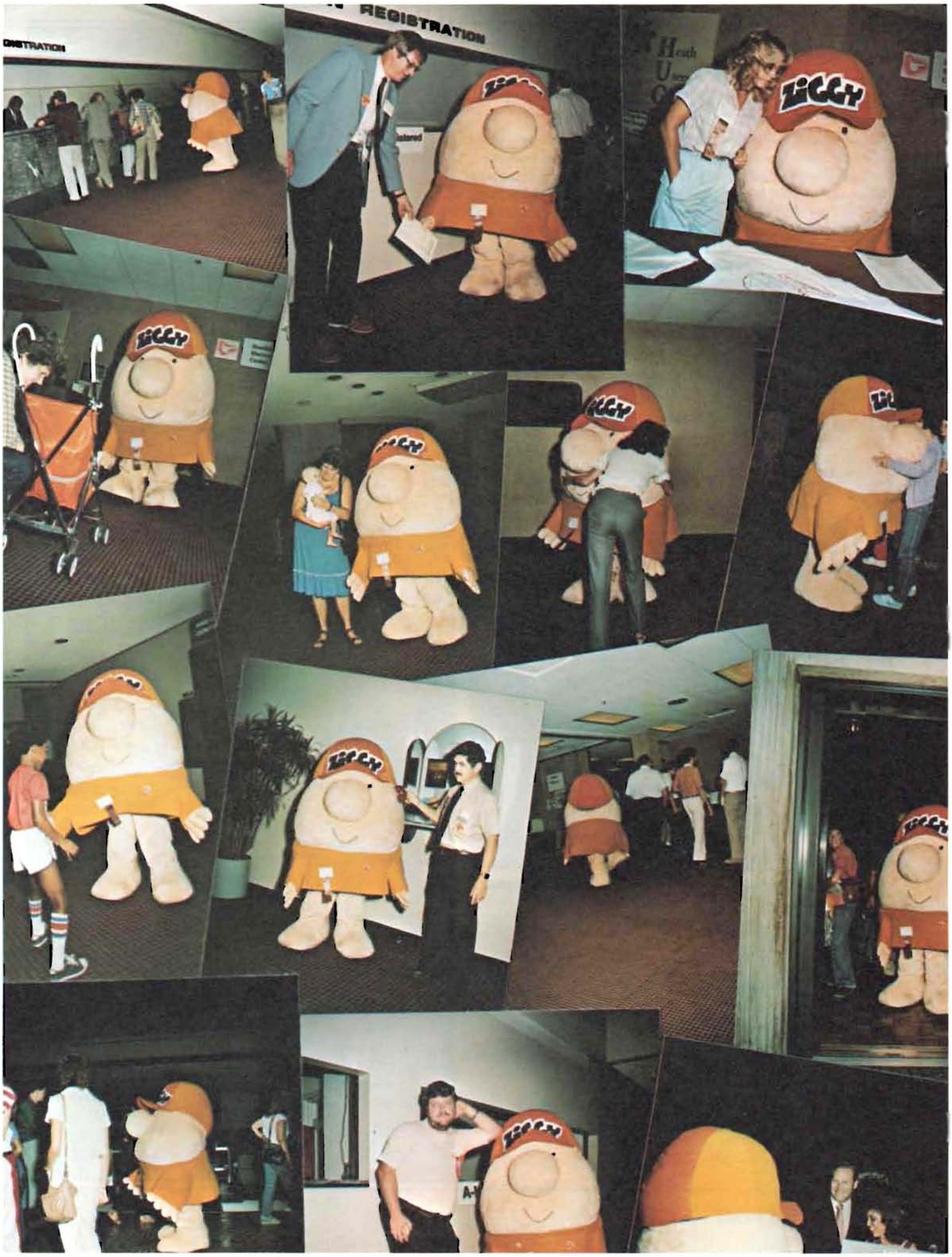
And now for a few words on the Conference from the rookie of the HUG staff. The rest of the group already had one HUG Conference under their belts, so they were all confident and couldn't wait for it to get here. But not me — I was scared to death! They were telling me I'm going to be working in the HUG booth selling software, and I'm wondering how I'm going to learn all of these programs so I can at least appear as though I know what I'm doing. It did calm my nerves knowing Pat and Nancy would be there to bail me out when I got in over my head (which by the way, did happen once or twice, or ten or twenty times). Poor Nancy had me tagging at her heels every waking moment of the day, she was my security blanket! But by Sunday, I was an old pro at this conference stuff and even ventured out on my own a couple of times.

When the vendor area finally opened on Friday, I discovered what everyone meant when they said I'd never find time to use the Hyatt's swimming pool. All of those happy HUGgers kept Pat, Nancy, Gerry, and I so busy selling software that we barely had time to break for lunch. I never did get a chance to look around at all of the vendor's booths, so it was a good thing Jon Falkner took so many pictures. It wasn't until after we got back home that I saw what the whole vendor area looked like and who all was there. I must admit that I've never enjoyed working so hard as much as I did at the Conference. I'm still amazed at how much you can do in so short a time and come out smiling and alive! Pat, Nancy, and I do have one request though, we have put in an order for a couple of cots and lounge chairs for our HUG booth next year.

All of the people there were so nice and having such a good time, it was really a great experience. One of the best parts of working the booth was that we got to give out the vendor prizes to the winners who's numbers were drawn. I'm sure it was great being on the winning end of it, but it was so much fun being the one to give the prizes out (except when an H-8 owner would win Z-100 software, etc.). But Nancy and I were easy to get along with and we just kept digging until we came up with good prizes for everyone. We were ready to kidnap Walt when he started running his blue light specials. The whole place went crazy with people running around trying to find whatever it was he had them looking for.

The weekend was over much too fast, although I'm not so sure any of us could have kept the hectic pace up too much longer. But I came home with a lot of good memories and I was very proud to be a part of the HUG staff who worked so hard to make the Conference such a great success. Many thanks to all who participated and helped in this success. See you all next year!

P.S. We're already working on the '84 Conference and believe me, it is going to get even better!!!


REGISTRATION
REGISTRATION
ZIGGY

H19/H89

THE ULTRA ROM

ULTRA is an enhanced ROM for your H19 terminal display which retains compatibility with present software (except ANSI sequences), yet adds new capabilities that you probably thought were impossible: self-test, second page display (with option), operation to 38.4K baud, user function keys, and much, much more. Comes with a pair of ROM's, binder, and complete documentation.

ULTRA	$49.95
Page Two Option	$14.95

H89/H100

ZLYNK/II

ZLYNK/II is a CP/M or CP/M-85 modem system like nothing else in the entire industry!ZLYNK/II actually contains its own text editor and language (ZLINGO). It can be used directly by the operator, or else programmed to operate completely independently. Eight protocols are supplied with ZLYNK/II, but more can be added via overlays at any time.

ZLYNK/II (hard sector)	$59.95
ZLYNK/II (soft sector)	$59.95

Something for Everyone

H100

PALETTE

An interactive graphics system for artists and home users alike. With PALETTE you can draw superb color or monochrome graphics displays and even dump to compatible printers. PALETTE currently supports both a lightpen and interlace mode, and it comes with a complete tutorial! NOT to be confused with others that SEEM to be similar. PALETTE is truly one-of-a-kind!

PALETTE (ZDOS)	$69.95
Z100 Lightpen	$150.00

H8/H89

THE HDOS TOOLKIT

The HDOS Toolkit is a group of utilities for both the comfort of the average user, and for the education of developing assembly-language programmers. We supply full source code with the TOOLKIT with comments to unlock truly 'Wizard-Class' tricks. Any single program in the package is worth the price of the entire group. If you like HDOS, you'll LOVE this!

Toolkit (hard sector)	$29.95
Toolkit (soft sector)	$29.95

All of the above items (and a LOT more!) are available direct from Software Wizardry, Inc. as well as from most Heathkit Electronic Centers and many Zenith Data Systems dealers and distributors as well. Please add $2 minimum (or 2%, whichever is greater) shipping/handling plus sales tax if shipped to a Missouri address. Software Wizardry, Inc. also has over 450 other items of interest to Heath/Zenith users as well. Call or write for our FREE price list. Dealer inquiries and HUG group purchases welcome!

VISA MasterCard

ZENITH data systems
AUTHORIZED SALES & SERVICE

Software Wizardry
THE MAGIC TOUCH

122 Yankee Drive
St. Charles, MO 63301
(314) 946-1968

And Speaking Of The Conference...

Terry L. Jensen
Software Developer

The Second National HUG Conference was highlighted by many speaking events throughout the weekend. The general meetings in the Rosemont were informative and "fun". The sessions on the lower level which were scheduled throughout Saturday and Sunday, proved to be successful with many of the attendees taking advantage of the opportunity to learn from the guest speakers.

I had the honor of contacting, preparing for, and introducing most of the speakers at this year's convention. If I may say, this was one of the most rewarding "responsibilities" I have ever been assigned! Every one of the speakers were well qualified for participating in their respective meetings and it was indeed a privilege for me to have a part in each session.

Friday

Friday night as part of the opening for the Second National HUG Conference, Pat McNamara, Director of Marketing and Communications for Zenith Data Systems (ZDS), started the weekend off on the right foot. His vibrant attitude and excitement for Zenith computers made and makes him the right person, at the right place, at the right time.

Having only seen Pat's presentation once, I was delighted that he could and would come as our guest and it was truly great to meet him personally. He treated me like an old friend. His personality is one of "Hey, those are people that buy Zenith computers, let's (ZDS) do something about it". He is a tough man in a tough position, and he means business.

Pat presented the marketing success of ZDS through such accounts as Clarkson College, Martin Marietta, Boeing, and others. From there he moved into the marketing plans for ZDS. Of these plans, Pat explained how and why ZIGGY has become a part of the advertising for the Zenith computer line. The highlight of his presentation was the showing of a 16 mm rough draft of a ZDS 15 second advertisement for national television to be merged with 15 seconds of Zenith T.V. advertising, which is the first step in using this type of advertising media.

Pat's entire presentation displayed his optimistic mood that Zenith Data Systems is here to stay. At the conclusion of his presentation, ZIGGY (as seen in the photos of this issue) received a warm round of applause.

The sessions for the most part were filled including HUGgies seated on the floor.

Saturday/Sunday

Saturday morning started the beginning of the classroom style sessions which touched on a number of different subjects. The sessions were scheduled so that an attendee could attend six of the eight classes of his/her choice. Each presentation was informative for all who attended.

Communication/Modem

Dale Lamm of North Canton, Ohio was scheduled as the first speaker and began at eight o'clock Saturday morning. Dale was invited to speak on Commmunication and Modems. He was one of our guest speakers who was not a Heath/Zenith engineer or technician.

Dale has used and programmed micros since 1975, the H89 since 1979. He now uses both the H89 and Z100 at his home. He is employed as a systems programmer at Novar Controls Corporation, working primarily with Z80 designs used for process control applications. His specialization is in real-time networking.

Pat McNamara kicks off the HUG Conference.

Bill Johnson (center) talks with Bob Todd (left) and another HUGgie.

Terry Jensen introduces Bruce Denton.

Dale prefers assembly language, but finds the C compiler adequate when there is a lot of time given to write a piece of source code. The first languages he was familiar with were FORTRAN and BASIC. He writes software for the hobbiest marketplace (e.g. ZYLINK II) as a pleasant diversion from the "real world". His software packages are sold through Software Wizardry of St. Louis, Missouri.

The Communication/Modem sessions by Dale covered, in general, the growing popularity of computer communication via modems. This included personal computer to personal computer, personal computer to local Bulletin Boards, and personal computer to multiuser host systems such as CompuServe. (CompuServe is one of many mainframe systems that provide a large data base to personal computer users. The National HUG Bulletin Board is on the CompuServe timeshare system.)

Z100/S100 Interfacing

The S100 interfacing to the Z100 session was lead by Jim Buskiewicz of Heath Company. Jim is not only a big man physically but a "big" man when it comes to knowledge of hardware, including the S100 bus.

Jim received his technical training from two colleges and also from the University of Notre Dame. He became interested in microcomputers in 1977 when he built his first microprocessor based system—a PDP-8. Later that year, Jim pieced together his first S-100 system using George Morrow's hardware. That basic system is still what he uses today.

Jim started working for Heath in April 1979 as a computer techician. He became a consultant 6 months later. He is currently a supervisor and computer consultant for Heath Company's hardware group.

The Z100/S100 interfacing sessions covered the basics in understanding the S100 bus. Jim had charts to show the IEEE standard bus. He covered some technical areas and answered questions regarding specific applications of the S100 bus.

ZDOS 2.0

Brian Barnes, a software engineer for Zenith Data Systems, led the sessions on ZDOS 2.0 or MS-DOS 2.0. Brian is an excellent programmer and he knows how to apply himself. Brian and his wife Katie are personal friends of my wife and I, and it was truly a pleasure to introduce him as a guest speaker.

Brian, originally from the state of Washington, started working for HUG in June of 1980 as a software developer. In January of 1981, he transferred to the operating systems group of ZDS. Since that time, he has been involved with HDOS 2.0, as well as the original release of ZDOS. Brian has been responsible for the COBOL and BASIC compilers under HDOS.

Brian presented his thoughts and knowledge concerning generic MS- DOS 2.0, throwing in his ideas of what he would like to see in a Zenith release of MS-DOS 2.0. He also answered questions on the released versions of ZDOS.

Robotics/Educational Products

It is obvious that in a session on robotics, HERO 1 would be present as one of the guest speakers. Ron Johnson, a Senior Educational Product Developer of Heathkit/Zenith Educational Systems, was the main speaker.

Ron and HERO 1 have appeared together on the ABC Evening News, The Phil Donahue Show, the Hour Magazine, and numerous affiliate stations regarding HERO 1 and robotic fundamentals. Ron has lectured on, written articles on, and done numerous other functions on the topics of robotics.

Ron made a comment after one of his sessions that the group of attendees to the HUG Conference were one of the most attentive and enthusiastic groups he has ever lectured to.

Ron talked about the general history of robotics, touching on the different areas of robotics. He also talked specifically about HERO I and presented a short demonstration featuring HERO I.

Introduction to Computing

Many of the "attendees" to the HUG Conference are spouses to a computer hacker. These "attendees" have become known as the NUA's, Non User Attendees. (This was coined by a NUA at the First

Brian Barnes of ZDS talks on MS-DOS 2.0.

Ron Johnson and Hero I talk on robotics.

Dale Lamm conducts the Communication/ Modem session.

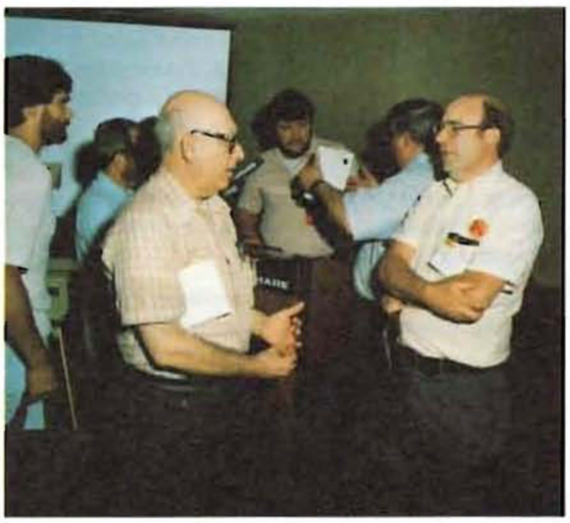
Dennis Hamilton leads CP/M Panel Discussion.

Bill Zurney answers questions on CP/M-86.

Gerry Kabelman leads in an introductory session.

The members of the CP/M Panel. Left to right: Al Dallas, Dean Gibson, Ray Livingston, Bob Mathias, and Bill Parrott. Missing from photo is Doug Sauby.

Missing photos for Arthur Seeback, Herbert Friedman, and David Perkins.



Jim Buszkiewicz (back center) answers questions on S100 bus.

National HUG Conference.) The two sessions were presented to NUA's to aid them in understanding computers and their "lost" spouses.

Gerry Kableman of ZDS held the two sessions. Gerry in many respects does not need an introduction because most of us are familiar with Gerry's past. He has been with Heath Company for about eight years, three of those years were spent here at HUG. We do not feel that HUG has lost Gerry. (Don't be misled, Gerry is very active at ZDS.) He is always there to lend a hand at the HUG Conferences.

Gerry is super at conversing with beginners and was an excellent choice for leading the sessions. He talked about very general areas of computers, including a history of computers. He showed slides at one of the sessions of areas in Europe where he has been for Heath Company. He also fielded questions to aid those with specific questions.

Academic Computer Uses

The Academic Computer Users session also included NUA's. This session was led by Arthur Seeback of St. Olaf College. Arthur is a mathematician and part time computer scientist in the Mathematics Department of St. Olaf College which is located in Northfield, Minnesota.

Although Arthur admits to being able to do a pretty good imitation of a computer hacker, his principal interest in computers is using them as tools for his writing as an editor of several journals; as a source of examples and ideas for elementary computer science courses; and as a graphics aid for advanced mathematics courses. (Of course, his wife's business use of her H89 for her typesetting and antique car services business is what payed for their trip to Chicago.)

During the period 1980-1982, when microcomputers first arrived in quantity at St. Olaf, Arthur was supported by a grant to help organize the explosive growth of Heath products on campus. The current count is something like (50) H89's, (30) H19's, and (40) H29's. The first H100 has just recently arrived.

Arthur discussed a different area with the attendees. His interest and area of concern was the practical use of the computer in an academic institutional environment. He projected some of his ideas. (His experience as an instructor was evident as he addressed his audience.)

CP/M Panel

The CP/M Panel discussion group was the most unique session presented. Six guest panel members were asked to participate on the panel, each having 10 minutes to expound on the area of CP/M in their particular area of expertise. At the conclusion of the presentations, the floor was open for questions to any of the panel members.

The idea for such a panel was suggested by Dennis Hamilton (familiar to many of you as "orcmid" on the HUG Bulletin Board). I felt it proper to have Dennis as the moderator for the discussion group. Dennis introduced each of the panel members and fielded the questions and discussions.

Dennis has twenty-five years experience in software development, including operating systems, programming languages, and general applications. He is currently in the process of setting up a software publishing firm that will emphasize convenient use on CP/M-based systems. Dennis was an excellent user to moderate the CP/M Panel.

Each of the six panel members have had experience in one or more areas of the CP/M operating system. Each member is a respectable programmer in his own right. Many of the members you will undoubtly recognize. The distinguished six members of the panel are as follows:

Bill Parrott - has worked with applications software, languages and operating systems since 1975. He has worked on a number of projects for D-G Electronics. It is quite obvious that programming comes natural to Bill.

Bill discussed ZCPR-2, what it provides, and what is available for installing it.

Bob Mathias - has worked as a systems programmer on main frames for eleven years. Bob was probably the first person (outside of Heath) to use CP/M on the H-8. His overall knowledge of CP/M truly makes him a CP/M "heavy weight". (He has submitted several programs to the HUG Library.)

Bob talked on the subject of Concurrent CP/M.

Al Dallas - an architect by trade and part time writer for the BUSS Newsletter, learned what software was about by disassembling programs. He first started with the H-8 computer. (Dennis teased Al that he got his programming ability from his mother, who has 20 years experience with IBM.)

Al talked about the development of vertical (industry oriented) applications and the change of attitude that writers of software need to have.

Ray Livingston - is a Mechanical Engineer and is currently working for the California Institute of Technology. Ray has his own company, Livingston Logic Labs. He has produced a number of products. His best known product is BIOS-80.

Ray discussed the real differences between disks, disk drives, and controllers.

Dean Gibson - has been employed as a computer software specialist for more than 20 years. He has experience on many mainframes, minicomputers, and microcomputers. Dean is a consultant for his company, UltiMeth Corporation. His specialty is systems programming. For the H89, he has written a number of hardware support packages including the HUG SY: Device Driver.

Dean talked about protecting your investment in disk data. He emphasized systemic backup procedure for keeping out of trouble.

Doug Sauby - had been a broadcasting engineer for a number of years. He began working with micros in 1977. He has been a Microcomputer Design Engineer for Magnolia Microsystems for three years and has been responsible for a number of projects. He has an interest in networking between personal computers.

Doug discussed CP/NET environment for computer networking.

H8/H89 Hardware

The H8/H89 Hardware presentation by Bruce Denton was one of the most popular sessions. Bruce has the kind of personality that makes it a treat to listen to him speak. (It was fun to talk to him before each session, because he would comment that he was wondering what he was going to talk about. He, of course, knew.)

Bruce is the President of D-G Electronics and his main function at D-G is in research and development. He is the designer of the D-G 32K dynamic memory board, the D-G Z80 CPU board, the D-G 64K dynamic memory board, and the D-G 64K static memory board all for the H-8 computer. Bruce also designed the D-G Super 89 board and the new D-G computer, the Heartbeat.

Bruce was kept busy by the Standing Room Only attendence at his lectures on hardware. He discussed the very basics of computer hardware and how it all fits together. For those interested, he also touched on the more technical areas, in which he, of course, has expertise. He was excellent at fielding questions from the attendees.

Z100 Interlace Mode

The Z100 Interlace Mode session was led by David Perkins of ZDS. Dave was the only ZDS software engineer who was a speaker that I had not met prior to this year's Conference. I had heard much about him and had seen his work, so it was indeed a pleasure to officially meet and introduce him as our guest speaker.

Dave has been with ZDS for two years. Previously, he was employed by Data General Corporation where he took part in many projects including COBOL, BASIC, and PL/I languages; an RPG compiler; system utilities; and design and implementation of an operating system. Dave is expecting to finish his Masters degree in Computer Science this December.

At ZDS, he was project leader of ZDOS for the Z100. After its release, he helped on the 2.x series of the monitor ROM for the Z100.

Dave discussed the features of the interlace mode for the Z100 screen display. He talked in some detail about how the interlace mode works and how it is implemented. (The interlace mode increases the vertical pixel resolution of the Z100 screen display.)

Software - Heath/Zenith

Bill Zurney of ZDS was the guest speaker for the Software for Heath/Zenith Z100 computers session. I have known Bill for some time and understand the caliber of software writer that he is. (Away from his work, Bill is a real joker.)

Bill has a BSEE from Purdue University. He has several years of experience before coming to ZDS. His experience includes applications of remote sensing and also information and signal processing. In both cases he was a systems programmer.

Bill has been with ZDS for five years and is currently a Senior Software Engineer for the operating systems and languages group. One of his projects include the implementation of the CP/M BIOS for CP/M 2.2.03. He is presently working on the implementation of CP/M-86 for the Z100 computer.

Bill fielded questions about and expounded on areas of CP/M-86. One of the interesting points is that CP/M-86 will use the 8 bit processor and run the existing CP/M-85 software.

Sunday — General Meetings

Local Club Presentation

At the First National HUG Conference, the Local HUG presentation was by Bill Johnson of the Capital Heath Users Group (CHUG) from the Washington D.C. area. This year the guest speaker was from the other side of the country; down in Southern California.

Herbert Friedman, M.D. is a urologic surgeon currently practicing in Southern California in the Kaiser Permanente Medical Care Program. Herb has been a kit builder since 1971 and has literally built dozens of Heathkits. His interest in electronics led to computers and from there into HUG. Herb is currently president of the Pomona HUG (California).

Dr. Friedman outlined the formation and growth of the Pomona HUG. His presentation included slides and a topic entitled "How to Form a Successful Local HUG". He fielded questions from the floor and also opened the meeting up for comments. The response from the floor was from a number of attendees representing locations from around the country.

Public Domain Software

Throughout much of the weekend Bob Todd from Bensalem, Pennsylvania had equipment set up in the NUA lounge to copy public domain software. Bob also was a speaker for a session discussing public domain software.

Bob's full time employment is a systems engineer at the First Ann Arbor Corporation. He has been in microcomputing since 1976, where his original computer was a MIL-MOD-8, an 8008 based system. He has been or is currently a member of several groups or societies including Philadelphia Area Computer Society, Amateur Computer Club of New Jersey, Central Jersey Computer Club, Frazer User Group (Heath), and Philadelphia Heath Users' Group. Bob also is the SIG/M distribution coordinator. He is the president of Extended Technology Systems. He does Heath format copy down for any and all users of CP/M and MS-DOS public domain software.

Bob discussed many areas of public domain software. He presented overlays to aid in his presentation. Bob opened the session up for questions from the floor.

Final Comments

The entire weekend of the Conference was full of talks and sessions that were there to encourage the exchange of ideas relating to the Heath/Zenith computer line. The Heath Users' Group is grateful to each guest speaker for taking the time to prepare and present their thoughts and share the knowledge that they have acquired through the past several years.

HOME FINANCE SYSTEM VERSION 2

—An extensive Home Finance System that keeps track of checking, asset accounts (cash, savings, IRAs, CDs), and regular bill payments. Let your printer write your checks for you on **any** business-sized check (design your own check format).
—Checks have user defined codes and a separate flag for tax deductible items.
—Many reports, including listing all checks, or checks by codes or tax flag.
—System consists of 100 page users manual with 5 program disks (5-¼") and a sample data disk.

Hardware: H8/H19 or H-Z89/90 with 64K RAM and two disk drives. Printer strongly recommended (any Heath®- Zenith® or other printer).
Software: HDOS 2.0 and Microsoft® MBASIC 4.82 for HDOS.
—Complete system: $89† (specify hard-soft sector 5-¼", or 8"). Manual alone $21†.

Master Card/Visa accepted, please include your phone number.

Jay H. Gold, M.D.
Jay Gold Software
Box 2024, Des Moines, IA 50310
(515) 279-9821

†Prices include shipping

Margaret's Crew .... You mean we have to sleep here too?

Is that a fly in my drink!?

Early registration as the HUGgers arrive.

Jill looks

Walt Gillespie, Bob Ellerton, and Pat McNamara prepare for the Grand Opening.

Mike Cogswell indicated that he needed two chairs.

What did you say about HDOS 3.0!!??

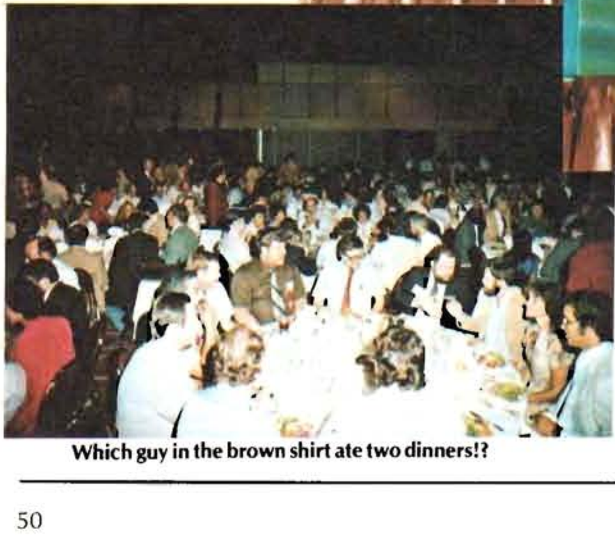
Activity at the H weekend.

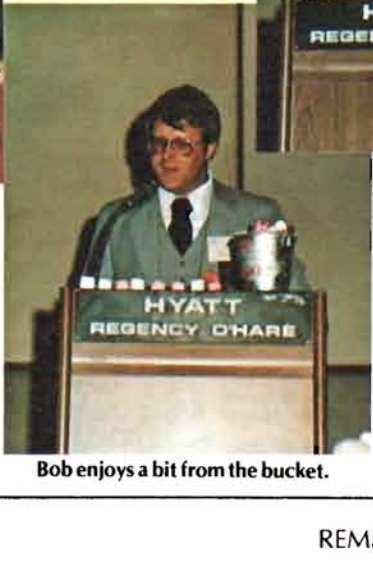
Mike Brenner, Product Line Manager from ZDS, does his impression of Fidel.

Tom Dornback, Vice President of Software from ZDS, indicated that his chair was too hard.

Joe Schulte, Pres. of VEC (Heathkit Stores), announces the winner of the H-100 computer.

Guess who's

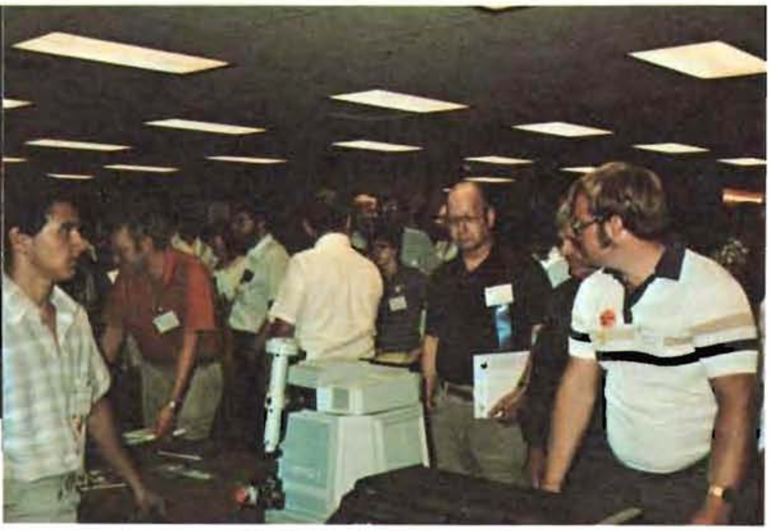

Which guy in the brown shirt ate two dinners!?



Bob enjoys a bit from the bucket.

n as Margaret "batches" at Walt.

Awards were presented to Mike Cogswell of CHUG, Pat Swayne of HUG, and Larry Sites by Bob Ellerton. Missing was Tom Jorgenson of Software Wizardry.


Ready? .... Ready for what!?

athkit (VEC) display was heavy all

Bill Johnson, President of Heath Company, discusses the products offered by Tom Jorgenson of Software Wizardry.

Luis Teja, our visitor from Spain, takes timeout with Bob Ellerton, Manager of the Heath Users' Group.

Ziggy meets friends from France.

oming to dinner?

Bill Johnson, President of Heath Company, thanks both users and vendors for continued support of Heath/Zenith products.

Don Moffet, President of Zenith Data Systems, announces the winner of the ZW120 during the Grand Prize Drawing.

Ready for what!? .... To build HERO 1 naturally!

What!!??.....HUGCON III already!!???

And The Winners Are......

James Good, Kutztown, PA	Home Finance System Vers.2	Jay Gold Software, Des Moines, IA
Robert Gallagher, Springfield, OH	S100 Sound Clock	Software Wizardry, Inc., St. Charles, MO
David Garwood, Red Cloud, NE	Z100 RAM Upgrade	Software Wizardry, Inc., St. Charles, MO
Melville Berry, Pros. Hts., IL	Ultra ROM	Software Wizardry, Inc., St. Charles, MO
Christian Kessler III, Sperry, OK	Crash Disk Recovery Pkg.	Software Wizardry, Inc., St. Charles, MO
Marvin Cohen, Des Plaines, IL	ZLYNK/II Modem System	Software Wizardry, Inc., St. Charles, MO
Perry Straw, Chicago, IL	Palette Graphics Design System	Software Wizardry, Inc., St. Charles, MO
Bruce Saunders, Lake in the Hills, IL	$100 Gift Certificate	Magnolia Microsystems, Seattle, WA
Jerry Lowery, Bouronnais, IL	$100 Gift Certificate	Magnolia Microsystems, Seattle, WA
Steven Beyer, Bay City, MI	$100 Gift Certificate	Magnolia Microsystems, Seattle, WA
Birnie Speltz, Lake Forest, IL	$100 Gift Certificate	Magnolia Microsystems, Seattle, WA
Don Leabo, Silver Spg., MD	$100 Gift Certificate	Magnolia Microsystems, Seattle, WA
Linda Faircloth, Otathe, KS	$100 Gift Certificate	Magnolia Microsystems, Seattle, WA
Thomas Kesler, Grove City, OH	$100 Gift Certificate	Magnolia Microsystems, Seattle, WA
Evelyn Pelan, Xenia, OH	$100 Gift Certificate	Magnolia Microsystems, Seattle, WA
Dick Dunbar, Wilmette, IL	$100 Gift Certificate	Magnolia Microsystems, Seattle, WA
Marvin Gino, Barrington, IL	$100 Gift Certificate	Magnolia Microsystems, Seattle, WA
Judy Dziubek, Menomonee Falls, WI	Instant Help Software	Studio Computers, Inc., Birmingham, MI
Peter Briggs, South Africa	Instant Help Software	Studio Computers, Inc., Birmingham, MI
Curt Gramlich, Washington, D.C.	Micro-Menu Software	Studio Computers, Inc., Birmingham, MI
Philip Wu, Rochester, MN	Terminal Modem Software	Studio Computers, Inc., Birmingham, MI
David Dolgin, Granada Hills, CA	Disk-Tran Program	Computer Consultants to Business, Cumberland, MD
Bill Whitson, West LaFayette, IN	Disk-Tran Program	Computer Consultants to Business, Cumberland, MD
Karl Bockholt, Delavan, WI	Disk-Tran Program	Computer Consultants to Business, Cumberland, MD
Mitch Wesolowski, Portage, MI	Disk-Tran Program	Computer Consultants to Business, Cumberland, MD
Sammye Hager, Crevecoeur, MO	Disk-Tran Program	Computer Consultants to Business, Cumberland, MD
James Leverett, Tonawanda, WI	Disk-Tran Program	Computer Consultants to Business, Cumberland, MD
Wolfgang Wiedemann, Rome, NY	$500 Gift Certificate	D-G Electronic Developments Co., Denison, TX
Mike Couch, W. Des Moines, IA	Imaginator I-100 Graphics Retrofit Board	Cleveland Codonics, Inc., Cleveland, OH
Roger Fields, Rock Island, IL	24 issues of BUSS	Sextant Publishing Co., Washington, D.C.
Walter Janowski, Valparaiso, IN	12 issues of Sextant	Sextant Publishing Co., Washington, D.C.
Betty Rogers, Carbondale, IL	$100 Software Certificate	The Software Toolworks, Sherman Oaks, CA
Don Hall, Ventura, CA	AUDIT Data Base Program	Generic Software, Marquette, MI
Michael Wolf, Xenia, OH	$25 Software Certificate	Husker Systems of Nebraska, Omaha, NE
Alex Crinzi, Orchard Park, NY	$50 Software Certificate	Husker Systems of Nebraska, Omaha, NE
Lee Rogers, Carbondale, IL	$25 Software Certificate	Husker Systems of Nebraska, Omaha, NE
Donald Wright, Ventura, CA	SYMED	New Horizon Software Systems, San Antonio, TX
James Good, Kutztown, PA	WatchWord or EDIT19	S & K Technology, San Antonio, TX
Tim Winslow, Milwaukee, WI	COBOL	Sigmasoft & Systems, Dallas, TX Tex-Matics MicroSystems, Plano, TX
Anita Kimery, Wheaton, IL	One Free Product	The Software Subscription. San Pablo, CA
John Robbins, St. Joseph, MI	One Free Product	The Software Subscription, San Pablo, CA
Gary Gaul, Sterling, VA	Gravitron	Apogee Software, Savannah, GA
Tom Tcimpidis, Granada Hills, CA	One copy of Dezign	Zeducomp, Stirling, NJ
Dorothy Rogers, Sault Ste. Marie, MI	1 Box of BASF 5 1/4" SSDD Floppies	Northwest Digital Systems, Seattle, WA
Jim Dorsey, Edmonton, Canada	$100 Gift Certificate	ARTRA, Inc., Arlington, VA
Frank Rozanski, Niles, IL	RCCP	Ultimate Computer Systems, Inc., Hastings, MI
Robert Presbrey, St. Paul, MN	Quizzer	Interactive MicroSystems, Columbus, OH
David Shaw, Elmhurst, IL	Super 19 ROM	Extended Technology Systems, Bensalem, PA
Donald Gregg, McHenry, IL	64K HDOS Patch Prog.;Extended SYSCMD; XP Printer Device Driver	Extended Technology Systems, Bensalem, PA
Monte Rubenstein, St. Charles, MO	One of any Sunflower Software Packages	Sunflower Software, Shawnee, KS
Amedee Hachey, Camp Bellton, Canada	FDC-880H Disk Controller	C.D.R. Systems, Inc., San Diego, CA
James Suttie, Springfield, IL	ZT Terminal Emulation Program	Codewright, St. Louis, MO
Lois McGovern, Arlington Hgts., IL	ROOTS89	Commsoft, Palo Alto, CA
Robert Hovey, Minneapolis, MN	ROOTS89	Commsoft, Palo Alto, CA
John Gibnatasio, Gainesville, FL	Extended Screen Editor for Z100	Cherry Engineering, Laurel, MD
Ross Towbin, Scarborough, NY	2S+2P+RTC	W.I.S.E., Baraboo, WI
Linda Gambill, Chesterfield, MO	TELPAC Modem Control Software	U.S. Robotics & Hampton Business Machines, Chicago, IL
Jim Colby, Bloomington, MN	1 Box of Accu-cut Paper	Access, Norristown, PA
Mary Howard, Cottage Grove, MN	Typesetting Manual & HUGCON Disk + PIE	MicroWorld Publishing, Northfield, MN
Robert Kan, Towson, MD	Replacement Backplate for H89	Kres Engineering, Irvine, CA

Margie Dolgin, Granada Hills, CA	Scooter Guard-It Control Center	Valley Data Sciences, Mountain View, CA
Leanord Geisler, Ann Arbor, MI	Choice of one - BIOS-80, CDRDVD, CDRBIOS	Livingston Logic, Pasadena, CA
Adolf Fejfar, Sierra Vista, AZ	QUIZZER	Interactive MicroSystems, Columbus, OH
Herman Cohen, Culver City, CA	$250 Gift Certificate	Technical Micro Systems, Inc., Ann Arbor, MI
Peter Seebach, Northfield, MN	H81-AK or H891-AK	Micro Widget Works, Inc., Tustin, CA
Joseph Gonzalez, Santurce, PR	dBASE II	Ashton-Tate, Culver City, CA
John Hoffman, New York, NY	MBASIC CAI (or $50 Software Credit)	Newline Software, Littleton, MA
H. J. Rosevear, Munster, IN	Z-100 HDOS 2.0 (or $69 Software Credit)	Newline Software, Littleton, MA
Lee Rogers, Carbondale, IL	Text Processor (or $50 Software Credit)	Newline Software, Littleton, MA
Jack Kerns, Grand Rapids, MI	Touch Typist (or $25 Software Credit)	Newline Software, Littleton, MA
George Martin, St. Louis, MO	Check Processor (or $25 Software Credit)	Newline Software, Littleton, MA
William Moore, Jr., DeLand, FL	Super Math 1 (or $33 Software Credit)	Newline Software, Littleton, MA
R. Page Carter, Bethesda, MD	TM100-1 Disk Drive	H-Scoop & Quik Data, Inc., Sheboygan, WI
Robert Bernier, St. Paul, MN	H-Scoop Disk Offer Set	H-Scoop & Quik Data, Inc.,Sheboygan, WI
Dave Lennstrom, Utica, MI	PASCAL/MT+ FOR HDOS	New Orleans General Data Services, New Orleans, LA
Judy Knie, Niles, IL	Graph-Pac-I Graphics Support Pkg.	Micro-Doc, Omaha, NE

ATTENTION:

Local Clubs and Heath/Zenith Vendors

In an effort to bring new members of the Heath/Zenith Users' Group up to date in this fast paced world of personal computers, we are now beginning to compile the January 1984 issue of REMark. The Heath/Zenith Users' Group will be providing this issue as a cross-reference of all REMark articles, all Local HUGs that wish to participate, all vendors that desire mention in REMark, and finally, a brief description of the HUG software released during 1983.

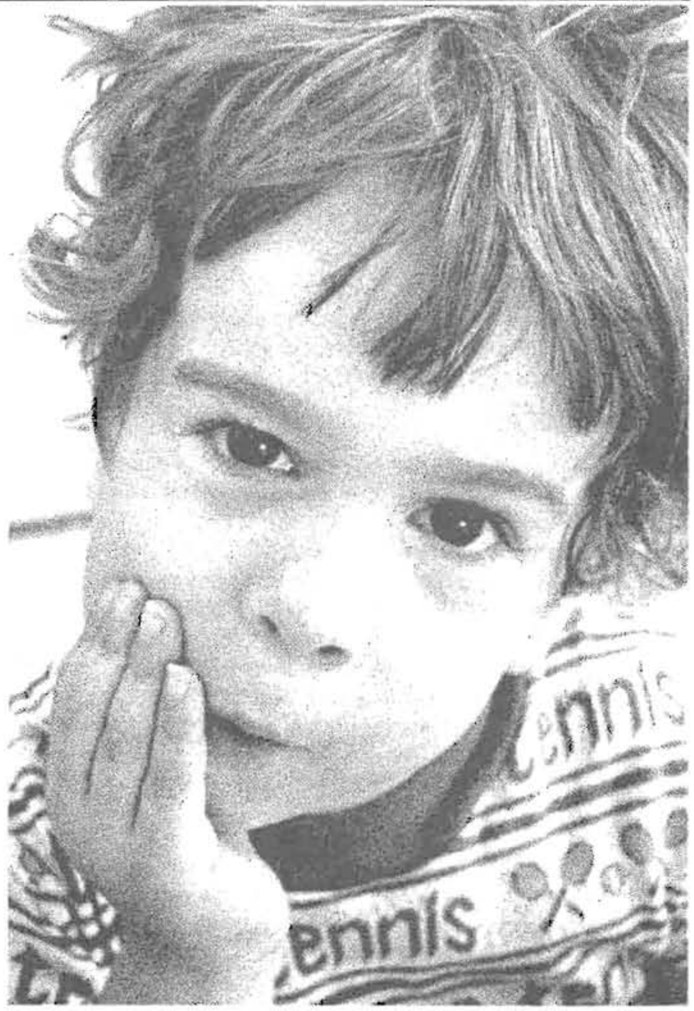
We would like to supply the latest information on your club via the club officers and/or the latest that your company has to offer in the way of Heath/Zenith related hardware or software. If your club or company wishes to be listed, please send us the following information.

Clubs:

Club Name:
Club Address:
Contact Individual: Phone Number:
Group Size:
Meeting Time and Place:
Bulletin Board: Yes/No Phone Number:
Special Notes:

Heath/Zenith Related Vendors:

Company Name:
Company Address:
Contact Individual: Phone Number:
Hardware: Yes/No Software: Yes/No
Consultation Available: Yes/No
Description of Product(s):
(A total of 25 words or less for all products.)
Special Notes:
(A total of 15 words or less.)



To accomplish our goal of providing the necessary information for our members in the January 1984 issue of REMark, we will need your information no later than **December 1, 1983**. Please send your information TYPED in a neat format that will allow for ease of entry.

For the Local Clubs: If the information appearing in the August 1983, Issue 42 of REMark is correct, we will not require further contact until changes are necessary.

Send your information to:

Heath/Zenith Users' Group
Attn: Donna Melland, Asst. Editor
Hilltop Road
St. Joseph, MI 49085

HUGCON GOODIES II

Pat Swayne
Software Engineer

Last year, following the first National HUG Conference, I told you about some of the new hardware and software products that I saw there. This year, your HUG Manager Bob Ellerton saw fit to provide much more space for the vendors of such products, and so there was much more to see. It was enough to make you wish you had an oil-well somewhere, and I guess some of the attendees acted like they did, because the vendors reported exceptionally good sales.

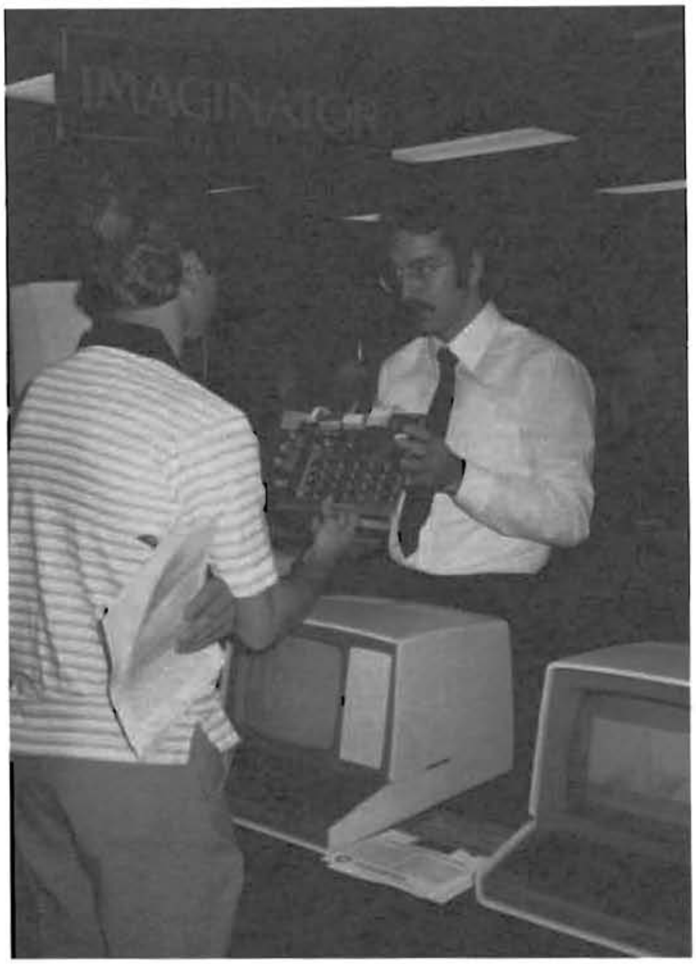
Maybe it is because I work with software every day, but when I go to something like the HUG Conference, it's the hardware that excites me the most. And there was more of it to see this year...

Super Graphics

Cleveland Codonics, Inc., who presented the first decently priced graphics mod for the H/Z-19 and H/Z-89 last year brought their new Imaginator 2 for the H/Z-29. It is a single board modification that provides 672 by 500 pixel resolution graphics, 500,000 pixel per second line drawing, panning and zooming, and even an optional parallel port for high speed communications with the host computer. With this board, you not only get super high resolution graphics, but at speeds that make animation possible. The Imaginator 2 has its own processor and a graphics instruction set that should make drawing your own graphics fast and easy. For more information, contact Cleveland Codonics, Inc., 18001 Englewood, Cleveland, OH 44130, (216) 243-1198.

Shirt Pocket Modules

A new Modular Board System was shown by Kres Engineering, P. O. Box 17328, Irvine, CA 92713, (714) 559-1047 or (213) 957-6322. This system consists of a mother board the size of a standard H/Z-89 right-hand side board on to which you can plug one to 4 daughter boards, each of which can perform a different function. The daughter boards are 1.8" (4.6 cm) by 4" (10 cm) in size. The daughter boards shown at the conference were a serial interface board (compatible with the Heath serial interface), a parallel interface board, a four-channel D/A board, and a color graphics board. You can place any combination of the daughter boards on a mother board, so if you wanted 2 serial ports, a parallel port, and color graphics, you could get it all on one H/Z-89 right hand card. Each daughter board can be addressed to any of the 4 right hand slot port banks, and if a daughter board does not use the whole bank, it is possible for more than one board to share a bank. For example, the parallel board and color board could both be addressed to the bank at port 362Q, freeing ports 330Q and 340Q for use with serial ports. In the works are other daughter boards, including a math processor, sound effects, synchronous serial port, calendar clock, A/D, audio



filter (for the sound effects board), BSR home controller interface, and hardware interval timer boards. Larger daughter boards are also planned, including a SASI hard disk interface (for the Z-67 and others). A half-height mother board is planned for use with the left hand slots on D-G Super 89 boards, and H8 and S-100 mother boards are also in the works.

The Widget That Never Forgets

Micro Widget Works, P. O. Box 15185, Santa Ana, CA 92705, (714) 544-8252 were showing their family of Micro Widget boards, which contain a clock calender with battery back-up, a Centronics compatible parallel port (works with Epson's), three user configurable parallel ports, provision for the AMD9511/9512 math chips, support for the Optimal Technology EPROM programmer, and expansion for the system bus. These boards are available in versions for the H8, H/Z-89, and S-100 bus (H/Z-100) computers. I am currently evaluating the H/Z-89 version, and will have a report on it soon. I can tell you at this point that it looks good. Sure is nice to boot up in HDOS and not only have the date already available from power up, but also the time and the day of the week.

The Computer With a Heart

It was amusing to see Henry Fale of H-SCOOP ask "What is that?" when he first saw D-G's new Heartbeat computer, because we at HUG had known about it for two or three months. I guess he hadn't received his August issue of REMark yet. The Heartbeat is built around D-G's Super 89 board. The board is mounted in a cabinet with an efficient switching power supply, hard and/or floppy disk drives, and plenty of room for Heath/Zenith H/Z-89 compatible boards. The whole thing is amazingly compact, and will easily fit on a shelf under a computer work station desk. It is color coordinated to fit in with the H/Z-29 terminal and H/Z-207 disk drives. With room for up to 256k of RAM on the main board, it can handle up to 4 users with Digital Research's MP/M-II, and can also run CP/M, CP/M Plus, and good old HDOS. Sounds like an ideal small business system. See ad in this issue for more information.

The Ultimate '89

At the conference, I got my first look at the H1000 board by Technical Micro Systems, Inc., P. O. Box 7227, Ann Arbor, MI 48107, (313) 994-0784. The H1000 is a replacement CPU board for the H/Z-89 that has a 2 or 4 MHz Z80 and an 8 MHz 8086 processor. Although it does not provide the fancy graphics of the H/Z-100, the faster clock speed coupled with the use of a true 16 bit processor versus the 8/16 bit 8088 should make it a real speed demon, and the ideal system for engineers, scientists, and software developers. On board memory can be expanded up to 1 megabyte, providing plenty of memory for multi-user operation. For graphics, TMSI recommends that you add the Imaginator by Cleveland Codonics. The H1000 looks like it has plenty of potential, and I hope TMSI will fully develop the system. I would like to see Concurrent CP/M developed for it, Digital Research's GSX graphics standard implemented for the Imaginator, and Xenix or another Unix-type system developed.

There were many other interesting hardware developments at the conference, and if I hadn't been so busy in the HUG booth getting bombarded with questions and making sure HERO didn't walk off the table, I would probably have more to say in this article. I had even less time to look at new software (since I looked at the hardware first), but I did manage to see demonstrations of two excellent new word processors for the H/Z-100.

Watch Your Words

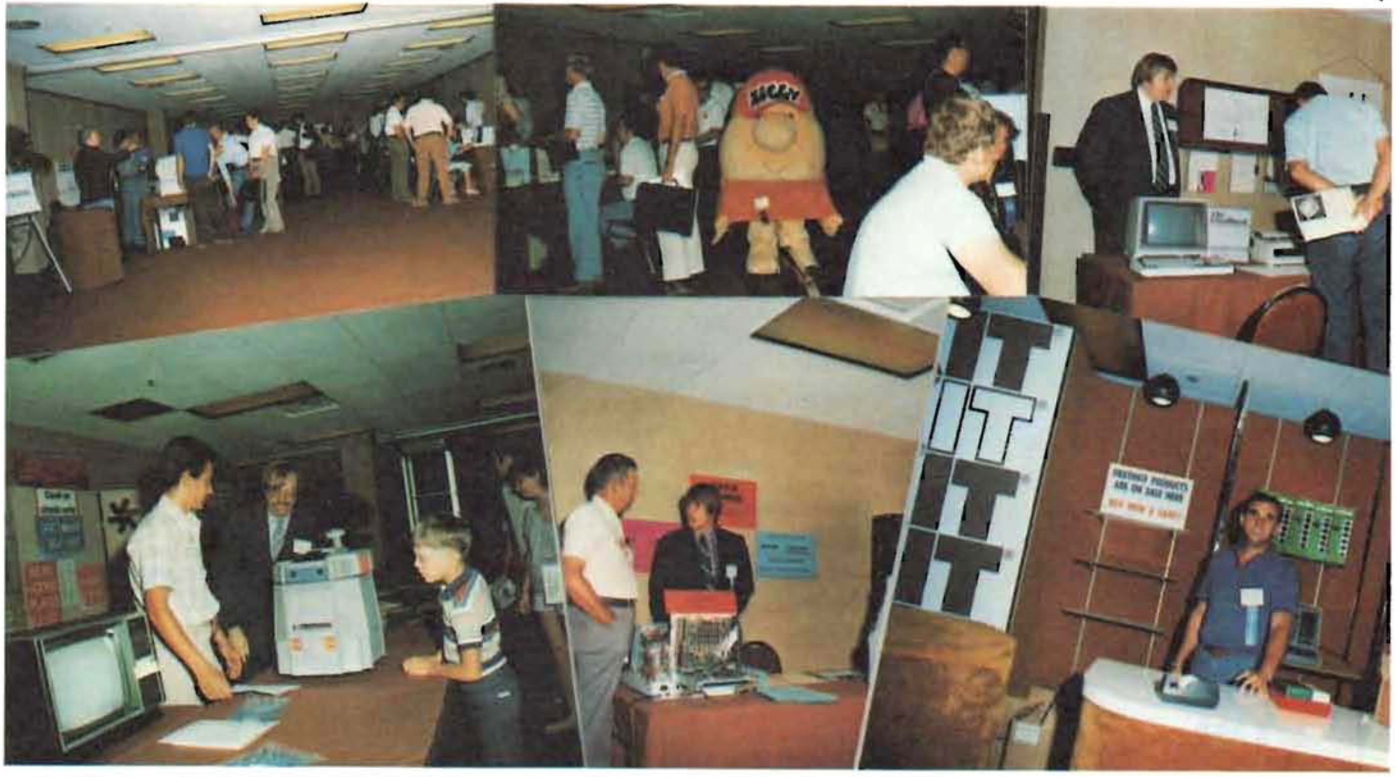
WatchWord is a word processor and editor from S and K Technology, Inc., 4610 Spotted Oak Woods, San Antonio, TX 78249, (512) 492-3384. It is unique among such programs that I have seen in that it writes to the video RAM in the H/Z-100 directly to produce characters on the screen. This means that video display changes are quick and smooth, including forward and backward scrolling. Horizontal scrolling is also provided, and lines can be any length, up to 65535 characters! With direct screen writing, WatchWord is also able to show superscripts, subscripts, underlining, and special characters directly on the screen. You can also create your own fonts and characters.

Watchword has a split screen feature that allows you to edit two different files at the same time, and transfer text from one to the other. You can also edit different sections of the same file. Powerful macros are available, to accomplish whole jobs at the stroke of a key. With all that it can do, WatchWord is not expensive ($100), and a demo disk can be obtained for only $3.00.

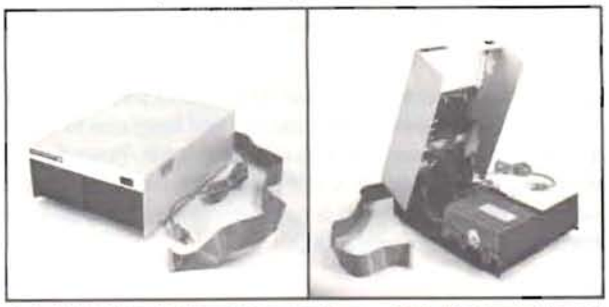
Perfect Words

WordPerfect is a professional wordprocessing system from Satellite Software International of Orem, Utah, that is available for several 16-bit machines including the H/Z-100. It was being shown at the convention by Hampton Business Machines, 5559-61 North Elston Ave., Chicago, IL 60630. It is what you would call a "full featured" word processor, that can do just about anything you would expect a word processor to do (automatic everything, pagination, headers, footers, etc.), and it can also perform math operations right in the text. Every function and keypad key is used for special purposes in WordPerfect, and a template to place on the keyboard that defines the keys is provided. An optional speller is available to do spelling checks from within the word processor, with context checking, etc., and an optional sorter lets you sort selected records alphabetically or numerically.

I don't think that anyone who went to the conference can say that there is not a lot of hardware and software support available for Heath/Zenith computers. It was great to see all of those goodies under one roof (even if it was under repair), and I just wonder how we are going to top it next year. Meanwhile, watch future issues of REMark for in-depth evaluations of some of the many hardware and software products available for Heath/Zenith computers.



NOW 12 MEGABYTE
(CDR-10M) $2995



WINCHESTER SYSTEM
For the Heath/Zenith Computer

Systems complete with software, case, power supply, signal cable and interface.
Runs with CP/M, on the H/Z89 & H8 (with Z80 card).

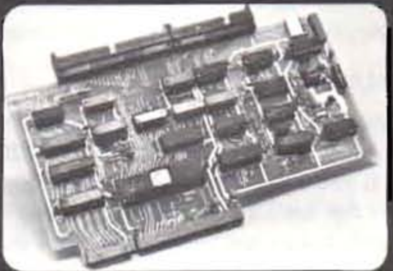
- Switching power supply
- Expansion for backup installations
- Auto attach BIOS
- Hard disk utilities
- Formatting program
- 1 year parts & workmanship warranty

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

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

Contact:
C.D.R. Systems Inc.
7210 Clairemont Mesa Blvd.
San Diego, CA 92111
Tel. (619) 560-1272

1 CONTROLLER



FOR 8" & 5.25" DRIVES

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

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

NEW PRICE $495

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

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

Contact:
C.D.R. Systems Inc.
7210 Clairemont Mesa Blvd.
San Diego, CA 92111
Tel. (619) 560-1272

PASCAL/MT+™
FOR HDOS

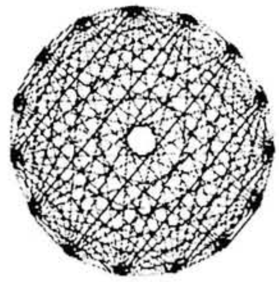
- VERSION 5.5
  [LATEST VERSION]
- USES STANDARD HDOS DEVICE DRIVERS
- ISO STANDARD PASCAL
- EMITS NATIVE 8080 OR Z80 CODE
- ALL HDOS SYSTEM CALLS CAN BE MADE FROM PASCAL
- AUTOMATIC CHANNEL ALLOCATIONS
- MODULAR COMPILATION
- CHAINING FROM ONE PROGRAM TO ANOTHER SUPPORTED
- UP TO 225 OVERLAYS SUPPORTED
- SELECTABLE BCD REALS [FOR BUSINESS APPLICATIONS] OR FLOATING POINT REALS [FOR SCIENTIFIC APPLICATIONS]
- FLOATING POINT REALS MAY BE EITHER SOFTWARE OR HARDWARE
  [AMD 9511 / INTEL 8231]
- SUPPORTS PACKED RANDOM FILES
- RELOCATING LOADER
- SYMBOLIC DEBUGGER
  [INCLUDES BREAKPOINTS]
- LIBRARIAN
- DISASSEMBLER
- INTERRUPT LEVEL PROCEDURES
- GENERATES ROMABLE CODE
- INLINE MINI-ASSEMBLER ALLOWS YOU TO WRITE ASSEMBLY LANGUAGE CODE AS PART OF THE PASCAL/MT+ SOURCE CODE
- PROCEDURES MAY BE DECLARED EITHER RECURSIVE OR STATIC
  [STATIC PROCEDURES EXECUTE CONSIDERABLY FASTER THAN RECURSIVE PROCEDURES]
- NO ONGOING CHARGES FOR RUN TIME PACKAGES

PRICE **$ 350**
SPECIFY H-17/37/47

PASCAL/MT+ IS A TRADE MARK OF DIGITAL RESEARCH

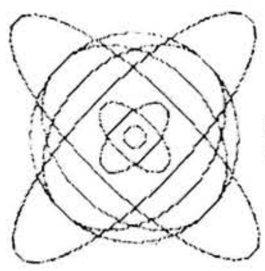
NEW ORLEANS
GENERAL DATA SERVICES, INC
7230 CHADBOURNE DR
NEW ORLEANS LA 70126
(504) 241-9495

Graphics for any EPSON Printer!



Have your EPSON plot an unlimited number of artistic sketches, geometric designs, and business graphs, which would otherwise not be possible without MXGRAPH. MXGRAPH will turn your EPSON printer into a graphics plotter for personal or business needs.

Runs under **CP/M-80** or **HDOS** operating systems on the H/Z-89 and H/Z-90 computer systems. Requires an EPSON printer equipped with bit plot capabilities. Specify HDOS or CP/M, hard or soft sectored 5.25" disk. **MXGRAPH is ONLY ~~$79.95~~ $59.97!**



"NOVA" The WordStar™ Enhancer!

- NOVA provides a series of extremely powerful enhancements to your copy of WordStar.
- **Enable full use of ALL your keypad** AND special function keys to greatly increase the speed and ease of editing documents.
- **Full use of special function keys** labeled at all times on the 25th line of your terminal.
- **Speeds text entry** by replacing those functions which would otherwise require multiple control codes to be entered.
- **Requires** Heath/Zenith distributed WordStar and CP/M-80 (Version 2.2.03) for operation on an **H-8, H/Z-89,** or **H/Z-90**; or CP/M-85 for operation on an **H/Z-100.**
- Specify hard or soft sectored 5.25" disk.

NOVA is ONLY ~~$39.95~~ $24.97!

ENHANCE YOUR WORDSTAR TODAY!

MICRO INNOVATIONS
2455 Sylvania, Toledo, Ohio 43613
VISA and MASTERCARD Orders Welcome!
Phone Orders CALL: (419) 471-1245

DO NOT DELAY - CALL NOW!

H-1000

A Z80/8086 UPGRADE FOR THE H89/Z89

HARDWARE

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

SOFTWARE

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

TMSI
Technical Micro Systems Inc.
P.O. Box 7227, Dept. H
366 Cloverdale • Ann Arbor, Michigan 48107
(313) 994-0784 **We accept MasterCard and VISA.**

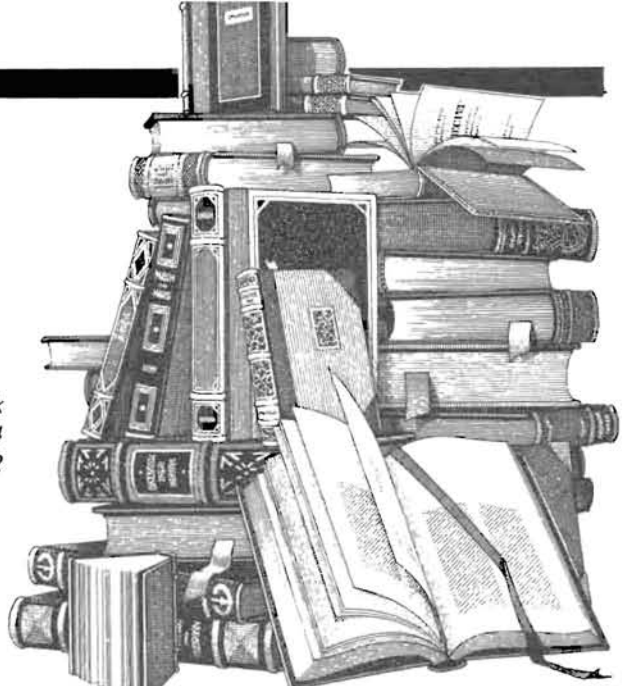
The H-1000 is a quality product of Technical Micro Systems, Inc., manufacturers of innovative microcomputer systems since 1979.

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

Heath
Users'
Group
Studio
Computers

BASIC Computing

Completing Our Game of Words



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

This article completes a series which develops a game that required some lengthy and at times complex programming. We've made the job easy by using modular programming and flow charts.

Figure 1A, 1B, and 1C provide the chart to complete this game, and the program lines follow. I won't explain these as they're very similar to functions of the input section. Here goes.

```
210 E11$=E$+"J"            'ERASE TO END OF THE PAGE
1300 W=A-1                 'NUMBER OF LETTERS IN WORD
1310 P=72-INT(J/2)         'LEFT EDGE OF WORD ON SCREEN
1320 FOR Z=1 TO W          'LOOP FOR WORD LENGTH
1330 X=46:Y=50             'LINE 15, COLUMN 19
1340 GOSUB 5150:GOSUB 5080:GOSUB 5020:GOSUB 5100         'PICK
1350 X=X+4:GOSUB 5000:X=X+4     '(SPACE) A (SPACE)
1360 GOSUB 5110:GOSUB 5040:GOSUB 5190:GOSUB 5190:GOSUB 5040:
GOSUB 5170      'LETTER
1370 A$=INPUT$(1)          'INPUT 1 CHARACTER
1380 PRINT E1$;CHR$(49);CHR$(45);E11$             'ERASE MESSAGE
1390 L=ASC(A$)-64          'CONTROL VALUE
1400 X=68:Y=44             'LINE 13, COLUMN 37
1410 ON L GOSUB 5000,5010,5020,5030,5040,5050,5060,5070,5080,5090,
5100.5110,5120,5130,5140,5150,5160,5170,5180,5190,5200,5210,5220,
5230,5240,5250             'PRINT LETTER
1420 IF A$=L$(Z) GOTO 1500      'IF INPUT IS CORRECT LETTER
1430 X=40:Y=50             'LINE 19, COLUMN 9
1440 GOSUB 5220:GOSUB 5170:GOSUB 5140:GOSUB 5130:GOSUB 5060  'WRONG
1450 X=X+8:GOSUB 5190:GOSUB 5170:GOSUB 5240      '(SPACE) TRY
1460 X=X+4:GOSUB 5000:GOSUB 5060:GOSUB 5000:GOSUB 5080:
GOSUB 5130                 'AGAIN
1470 FOR Q=1 TO 300:NEXT Q      'DELAY
1480 PRINT E1$;CHR$(44);CHR$(32);E11$             'ERASE MESSAGE
1490 GOTO 1330             'BACK FOR NEXT INPUT
1500 X=50:Y=50             'LINE 19, COLUMN 19
1510 GOSUB 5190:GOSUB 5070:GOSUB 5000:GOSUB 5190:GOSUB 5500:
GOSUB 5180                 'THAT'S
1520 X=X+4:GOSUB 5170:GOSUB 5080:GOSUB 5060:GOSUB 5070:
GOSUB 5190                 '(SPACE) RIGHT
1530 X=P:Y=36              'LINE 7, POSITION LETTER
1540 ON L GOSUB 5000,5010,5020,5030,5040,5050,5060,5070,5080,5090,
5100,5110,5120,5130,5140,5150,5160,5170,5180,5190,5200,5210,5220,
5230,5240,5250             'PRINT LETTER
1550 P=P+C                 'ADVANCE RIGHT EDGE OF WORD
1560 FOR Q=1 TO 300:NEXT Q      'DELAY
1570 PRINT E1$;CHR$(44);CHR$(32);E11$             'ERASE MESSAGE
1580 NEXT Z                'DO IT AGAIN TILL ALL LETTERS USED
1590 PRINT E2$             'ALL LETTERS USED, ERASE SCREEN
1600 X=43:Y=32             'LINE 1, COLUMN 11
1610 GOSUB 5030:GOSUB 5140:X=X+4:GOSUB 5240:GOSUB 5140:
GOSUB 5200                 'DO YOU
1620 X=X+4:GOSUB 5220:GOSUB 5000:GOSUB 5130:GOSUB 5190:X=X+4:
GOSUB 5190:GOSUB 5140   'WANT TO
1630 X=53:Y=38             'LINE 7, COLUMN 21
1640 GOSUB 5150:GOSUB 5110:GOSUB 5000:GOSUB 5240:X=X+4  'PLAY (SPACE)
1650 GOSUB 5000:GOSUB 5060:GOSUB 5000:GOSUB 5080:GOSUB 5130  'AGAIN
1660 X=54:Y=44             'LINE 11, COLUMN 23
1670 GOSUB 5240:X=X+4:GOSUB 5050:GOSUB 5140:GOSUB 5170  'Y FOR
1680 X=X+4:GOSUB 5240:GOSUB 5040:GOSUB 5180      'YES
1690 X=58:Y=50             'LINE 19, COLUMN 27
1700 GOSUB 5130:X=X+4:GOSUB 5050:GOSUB 5140:GOSUB 5170  'N FOR
1710 X=X+4:GOSUB 5130:GOSUB 5140                 'NO
1720 A$=INPUT$(1)          'INPUT 1 CHARACTER
1730 IF A$="N" GOTO 1760        'GO TO END
1740 G=0:J=0               'RESET VARIABLES AS REQUIRED
1750 IF A$="Y" GOTO 590 ELSE GOTO 1720  'PLAY AGAIN OR INPUT NOT N OR Y
1760 PRINT E2$;E4$;E6$;E8$:END          'RESET TERMINAL AND END
```

Now type lines 210 and 1300 thru 1760 and save them in ASCII format as we've done in the past. Then load our old program "WORDS" and merge our new addition. Finally, save the whole program as "WORDS". You can now type RUN and play to your heart's content. The game is complete and will give lots of fun as it is. Just remember to keep the CAPS LOCK key down and type only letters. We'll remedy these shortcomings and add some frills in the coming paragraphs, but for now, have fun. Get your family or friends together and see if you can stump each other.

You must be ready for more programming if you're reading this paragraph, so let's go. In the remainder of this article, we'll make a good program great. We'll convert lower case letters to upper case and provide a message to use "LETTERS ONLY PLEASE" if other than a letter is struck. Then we'll decode the red function key during the guessing portion of the game so it can complete the whole word if a player gives up or knows the word but doesn't want to type each of the letters. We'll add a feature to remove the letters from the jumble as they're used, then we'll provide back spacing during the input in case a player types the wrong key.

It's important to note that the program works just fine as it is. It does what we set out to do. This is typical of many of the programs we write, and in a lot of cases is sufficient to meet our needs. If a program is written for a one-time or occasional use, you won't want to spend more time adding frills than the amount of time the program will save you. If the programmer is the only user, a lot of the prompts and explanations of what's going on or what input is required won't be needed, so these features can be left out. However, if someone other than the programmer is to make use of the program, the computer must be made as invisible as possible. The user shouldn't even know it's there. Mistakes in input and operation, if they occur (and they will), should be self-correcting or should provide an error message explaining what is wrong and how to make it right.

Figure 2 is the routine to convert lower-case to upper-case letters, and to provide an error message if the input is not a letter. When the error message is printed, the program cycles back for re-entry of the input. We'll put this routine in the program every time a keyboard entry is requested.

Our flow chart shows part of the existing program which asks for the input, checks for certain characters like the return key, then calculates the ASCII value of the input character. This is the value we'll

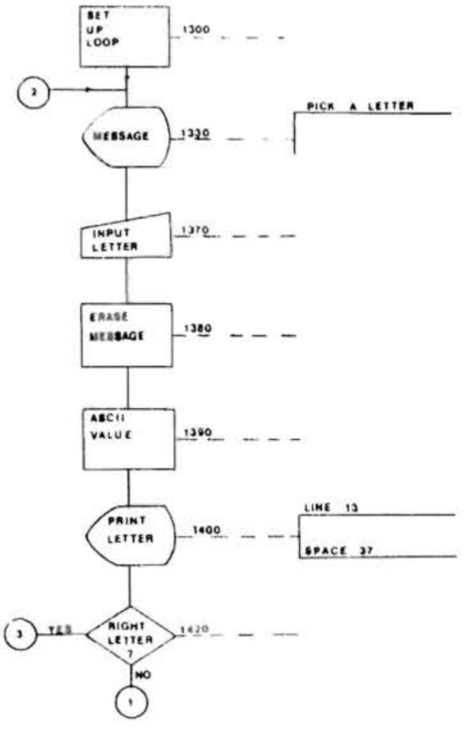


Figure 1-A

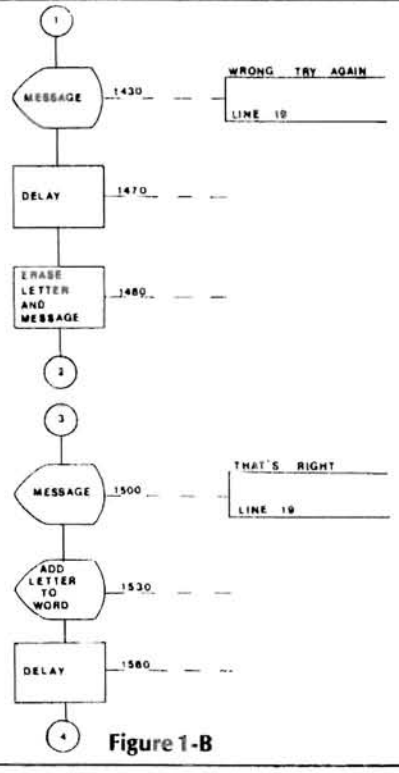


Figure 1-B

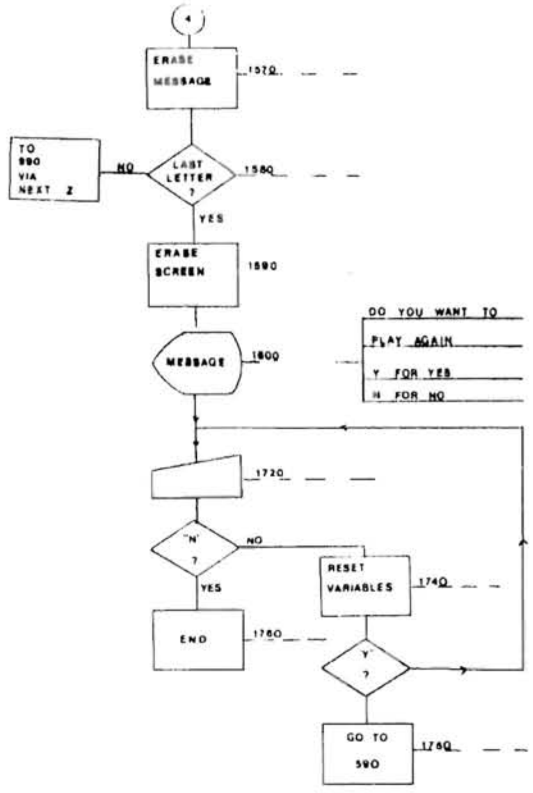


Figure 1-C

use to determine whether we have a capital letter. Remember that the number we calculated with our program is 64 less than the ASCII value, so that's what we must check for and use. Let's write programming for Figure 2 to check our initial input. To test for upper case, we must check the number assigned to variable L in our program. If it is a capital letter, the value will be in the range of 1 to 26. Then we can just continue execution. From the flow chart, we see that we must add this check between lines 820 and 830. The test for capitals looks like this:

```
822 IF L>0 AND L<27 GOTO 830          'CAPITAL LETTER?
```

A test for lower case is similar, but what range of numbers should we be looking for? You knew all along I'd get you to look in your Operating Manual sooner or later and now's the time. Lower case "a" has an ASCII value of 97 and each letter increases by 1 just as the upper case did. We can't test for that value, however, because we subtracted 64. Therefore, if an "a" was struck, the value 97 minus 64, or 33, was calculated. If a "z" was struck, the ASCII value was 122 and L was 58. Now all we must do is check for a value of 33 to 58 and we'll know whether we have a lower-case letter. So what? Now that we've got it, how do we change it to what we want? If we have a 33 (a), we want a 1 (A). If we have a 58 (z), we want a 26 (Z). A little simple math shows that the difference is 32 in every case. All we've got to do is check our variable L again. If it's in the range of a lower-case letter, subtract 32 and continue the program at line 830. If it's not in the correct range, it's not a lower-case letter. Line 822 showed us it wasn't upper-case either. It's time to put a message on the screen to tell the player who struck the wrong key that we programmed this thing for "LETTERS ONLY PLEASE", hold that message long enough to read it, and go back for another input at line 800. Because the error-message routine takes quite a few lines, we'll write it at line 5600. It looks like this:

```
824 IF L>32 AND L<59 THEN L=L-32:I$=CHR$(L+64):GOTO 830  'CONVERT
LOWER TO UPPER CASE
826 GOSUB 5600            'PRINT ERROR MESSAGE
828 GOTO 800              'GO BACK FOR NEXT INPUT
5600 Z2=X:Z1=Y:X=35:Y=50                    'COLUMN 4, LINE 19
AND SAVE CURSOR POSITION
5610 GOSUB 5110:GOSUB 5040:GOSUB 5190:GOSUB 5190:GOSUB 5040:
GOSUB 5170:GOSUB 5180   'LETTERS
5620 X=X+4                '(SPACE)
5630 GOSUB 5140:GOSUB 5130:GOSUB 5110:GOSUB 5240        'ONLY
5640 X=X+4                '(SPACE)
5650 GOSUB 5150:GOSUB 5110:GOSUB 5040:GOSUB 5000:GOSUB 5180:
GOSUB 5040                'PLEASE
5660 FOR X=1 TO 450:NEXT X       'DELAY
5670 X=32:Y=46            'COLUMN 1, LINE 15
5680 PRINT E1$;CHR$(Y);CHR$(X);E11$                 'ERASE MESSAGE
5690 X=Z2:Y=Z1:RETURN   'RESET CURSOR TO WORD AND RETURN
```

To add the same checks for inputs during the guessing part of the

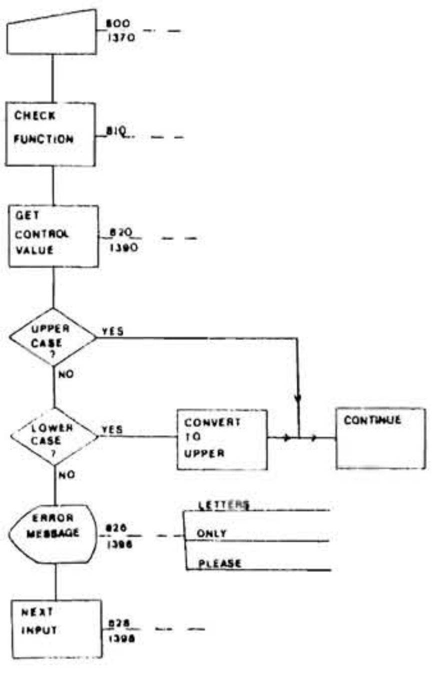


Figure 2

game, we've got to add four lines between existing lines 1390 and 1400 as follows:

```
1392 IF L>0 AND L<27 GOTO 1400          'CAPITAL LETTER ?
1394 IF L>32 AND L<59 THEN L=L-32:A$=CHR$(L+64):GOTO 1400  'LOWER-
CASE LETTER
1396 GOSUB 5600          'PRINT ERROR MESSAGE
1398 GOTO 1370           'GO BACK FOR THE NEXT INPUT
```

Now add lines 822 - 828, 1392 - 1398, and 5600 - 5690 to your program. Be sure to make an updated backup copy, and play the game with these new additions. Try entering incorrect keys to see what happens. It's not perfect yet, but it's coming along fine.

What happened when one of the keys F1 to F5, or red, blue, or white was struck? Getting the error message was OK, but then a letter was added to the word. How did it get there? Go back to the manual again to see how a function key works. Look in the appendix (Chapter 12) for SPECIAL FUNCTION KEYS. It's page 12-9 in my manual. You'll find a table of keys and HEATH ESCAPE CODES. For example, the F1 key shows ESC S, but what does this mean? This is an area of the manual which I feel is sorely inadequate, so here's my explanation of how it works.

Most of the keys on your terminal send the computer one character. The function keys are different. When they're pressed, they first send an ESCape (ASCII character 27), then they send a second character. In the case of the F1 key it was the letter S. That's the same as holding down the ESC key and pressing the letter S. We do that in our program with the variables E1$ thru E11$. The difference here is that the combinations sent by the special function keys are not recognized as anything special by our terminal or computer. They're called USER-DEFINED FUNCTIONS. That means we must add instructions to our program if we want them to do anything. I'll cover that in detail in a future article. You can also refer to REMark Issue 28, page 6 for my prior article on the subject.

Our program took the first character (ESC) and found that it wasn't a letter. So, it did what it was supposed to do. It printed the error message and asked for the next input. That's where the problem occurred. The second character from the function key was there waiting, and it was a letter. If you run the program and press the F1 key, you'll see that an S is added to the word. The other keys produce the letters shown in your manual. To prevent this we must first see if the input character is an ESCape. If it is, we can assign the second character to a string variable that won't be used for anything else. That's right. We'll input the unwanted character just to get it out of the buffer, then we can forget about it. According to our flow chart in Figure 2, we do this after the input and before calculating the control value. Add these lines to your program.

```
805 IF I$=CHR$(27) THEN R$=INPUT$(1)  'IF FUNCTION KEY THROW OUT LETTER
1372 IF A$=CHR$(27) THEN R1$=INPUT$(1)  'IF FUNCTION KEY STORE LETTER
```

Now see how the program works. I think you'll find it much improved.

If you've been playing our game, some of the players were sure to guess the word before they typed all the letters, or maybe they gave up and wanted to see what it was. It would be nice if we could press a single key and print the whole word correctly, rather than having to type each letter in order. Let's use the red key for this function. According to our Operating Manual, that key sends ESC Q. If the red key is pressed while solving the jumble, line 1372 which we just added will store the letter Q in variable R1$. All we've got to do is check R1$. If it's a Q, we can send program execution to a subroutine to print the whole word. Figure 3 is a flow chart of what we must do. We'll begin our subroutine at line 6500. Add these lines to your program.

```
1374 IF R1$="Q" THEN R1$="":GOTO 6500  'TEST R1$,RESET R1$,PRINT WORD
6500 X=32:Y=38             'CHARACTER 1, LINE 7
6510 PRINT E1$;CHR$(Y);CHR$(X);              'POSITION CURSOR
6520 PRINT E11$            'ERASE TO END OF PAGE
6530 X=72-J/2:Y=38         'CENTER WORD ON LINE 7
6540 FOR G=1 TO A-1        'FOR EACH LETTER OF WORD
6550 L=ASC(L$(G))-64       'CALCULATE CONTROL VALUE
6560 ON L GOSUB 5000,5010,5020,5030,5040,5050,5060,5070,5080,5090,
5100,5110,5120,5130,5140,5150,5160,5170,5180,5190,5200,5210,5220,
5230,5240,5250
6570 NEXT G                'CONTINUE TO END OF WORD
6580 FOR X=1 TO 750:NEXT X       'DELAY
6590 GOTO 1590             'RETURN TO "DO YOU WANT TO PLAY AGAIN"
```

Now when you're solving the jumble, if you press the red key, the whole word will be printed.

While solving the jumble, especially on long words, many of those who played my first version of this game said it would be nice if the used letters were removed from the jumble. We'll add that feature now. Figure 4 is the flow chart. The first thing we must do is save the screen position of each letter in the jumble as we print it. We'll use an array "I". It will have to be dimensioned.

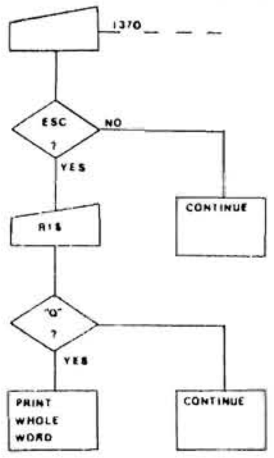


Figure 3

```
250 DIM L$(20),C(20),W$(20),W(20),M$(20),M(20),I(20)  'DIMENSION ARRAYS
1255 I(K)=X                    'SET I(LETTER) TO SCREEN POSITION OF LETTER
```

Next we must erase the letter when it's been picked. We know that one of the remaining letters has been chosen if the "THAT'S RIGHT" message is printed. Then we find the letter in the jumble and print spaces over it to erase it from the screen.

```
1522 B=1                    'SET CONTROL VALUE
1524 IF M$(B)=A$ GOTO 6700        'GOTO ERASE SUBROUTINE
1526 B=B+1                  'NOT FOUND, INCREMENT COUNTER
1528 GOTO 1524              'LOOK AT NEXT LETTER IN JUMBLE
6700 X=I(B)                 'SET HORIZONTAL SCREEN POSITION OF LETTER
6710 FOR Y=32 TO 35         'FOR EACH LINE OF LETTER
6720 PRINT E1$;CHR$(Y);CHR$(X);           'POSITION CURSOR
6730 FOR I=1 TO M(B)        'FOR WIDTH OF LETTER
6740 PRINT " ";             'PRINT A SPACE OVER GRAPHIC CHARACTER
6750 NEXT I                 'CONTINUE FOR WIDTH OF LETTER
6760 NEXT Y                 'CONTINUE FOR EACH LINE OF LETTER
6770 PRINT                  'CLEAR SEMI-COLON
6780 M$(B)=""               'ERASE LETTER FROM JUMBLE IN MEMORY
6790 I=0                    'RESET VARIABLE
6800 GOTO 1530              'CONTINUE PROGRAM
```

Now add lines 250, 1255, 1522 - 1528, and 6700 - 6800 to your program. Play the game a while and see how you like the new features.

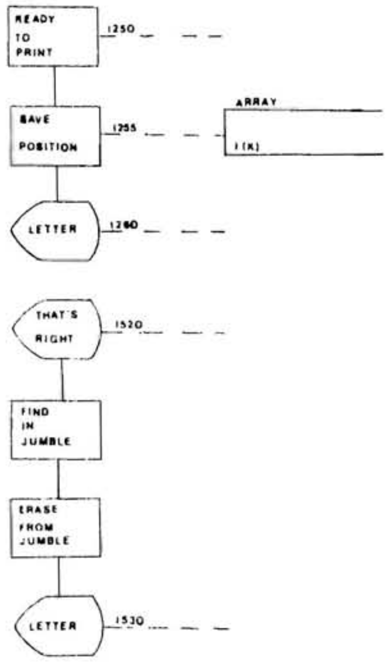


Figure 4

The last addition we'll make allows you to back up and change letters while inputting the word to be jumbled. The backspace key will erase the last letter of the word and can be used as often as necessary to correct an input. Figure 5 shows what we must do. A look at the Operating Manual shows us that the backspace key sends ASCII character number 8. We'll test our input for that and complete our program.

```
815 IF I$=CHR$(8) GOTO 7000      'TEST FOR BACKSPACE
7000 A=A-1                  'DECREMENT LETTER COUNTER
7010 IF A<1 GOTO 710        'BACKSPACE PRESSED WITH NO LETTERS IN WORD
7020 X=X-C(A)               'MOVE CURSOR POSITION BACK
7030 FOR V=Y TO Y+3         'FOR EACH LINE OF THE LETTER
7040 FOR H=X TO X+C(A)      'FOR EACH SPACE IN LETTERS WIDTH
7050 PRINT E1$;CHR$(V);CHR$(H);" "        'ERASE ONE SPACE
7060 NEXT H                 'CONTINUE ACROSS LETTER
7070 NEXT V                 'CONTINUE DOWN LETTER
7080 GOTO 800               'BACK FOR NEXT INPUT
```

Add line 815 and the backspace subprogram lines 7000 - 7080 to your program. This completes our game. The articles which describe it are copyrighted by the author, and all programming is copyrighted by Applied Computing. This just means you can't sell it or give it away. Feel free to play with family and friends. You'll be surprised how quickly they forget this thing is a computer. You may even be shocked at how much fun this game is and how much time can be spent at it.

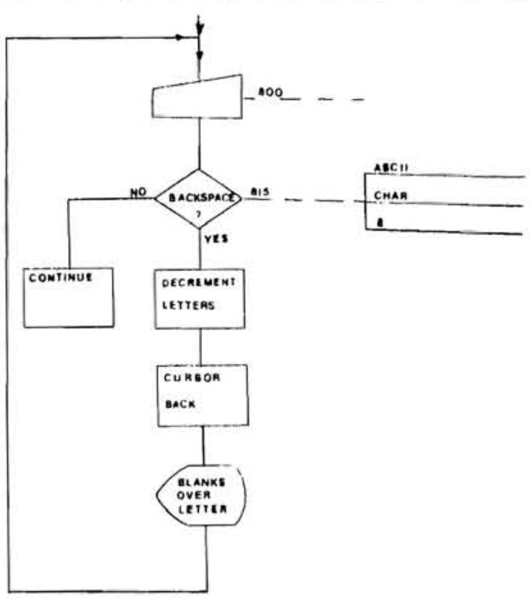


Figure 5

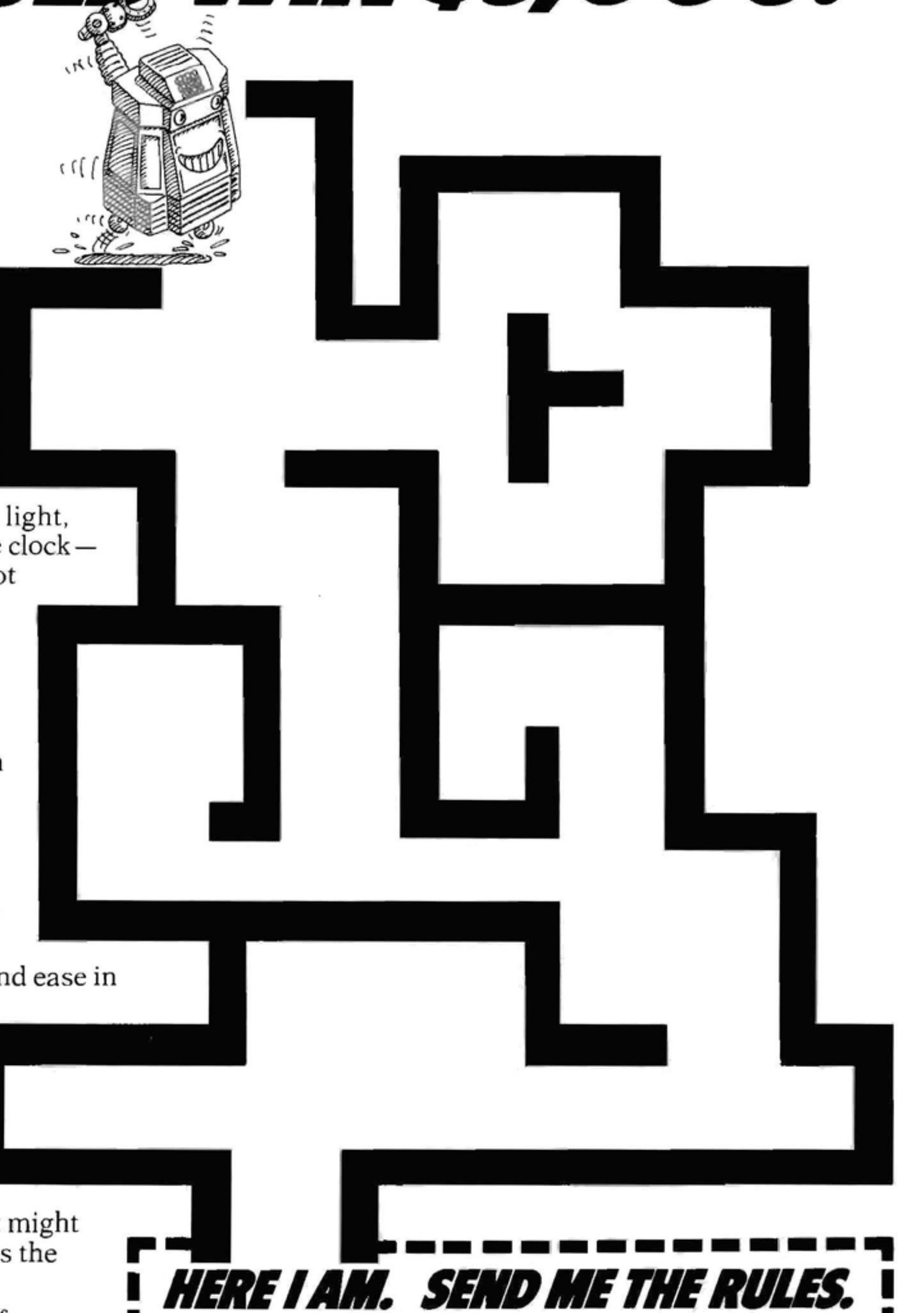
As you look over this program and what you've learned while writing it, try to apply modular programming and flow charting to your own work. We wrote a lot of steps and several complex functions. By breaking them into small sections (modules) and using flow charts to keep track of it all, an otherwise difficult job was fairly easy. Building our program in modules and supplying temporary lines as needed permitted us to test our work in small pieces. If there had been an error in the program, it would have been isolated to speed debugging.

Look over this game. There are lots of other features which can be added. How about a counter to tell you how many tries it took to solve the jumble, or a function key to give just the next letter for a hint on the tough ones? Make a backup copy of this program, then experiment with your own modifications. Write me with your ideas when you've got them working and I'll try to include several of them in a future article with the proper credit for who developed them. Have fun.

Compare the original program with the final version. Was it adequate the way we first wrote it? Did it do the job we intended it to do? Were the additional features we added necessary or even desirable? Were they worth the effort required to write and add them? What other features can we add to improve the operation or the ease of use of the program? These are the questions we should ask ourselves as we begin each program we write, and again before we decide that it's complete. Write and get the basic program working. Once this is done, add the features you desire as modules to the program. When your pride and joy is working the way you want, sit down and write an article for REMark. It's very rewarding and may help your fellow HUGgies enjoy their hobby.

See you next month.

IF YOUR ROBOT CAN MAKE IT TO THE COUPON, YOU COULD WIN $5,000.



Announcing the Virtual Devices 1st Annual Robot Roll-Off.

The race is on to find the smartest, fastest robot in the U.S.

It's the 1st Annual Robot Roll-Off sponsored by Virtual Devices, Inc., manufacturer of the most advanced remote control hardware and software for robot hobbyists.

To win, you have to build a robot that can maneuver a simple course on its own, open doors, detect sound, light, and motion. And do it all against the clock—faster and better than any other robot around.

Robot Roll-Offs are scheduled in cities across the country during the summer of '84. So robot enthusiasts can meet and compete. The Roll-Off grand prize winner walks away with $5,000. And there's cash and merchandise for the runners-up.

And because you'll want every advantage, you'll want to check into Virtual Devices radio remote control products. They let you use your personal computer for greater control and ease in programming.

So if you have or plan to build a HERO, Androbot, RB Robot, other comparable product, or homebuilt of your own design, get ready to roll. You don't have to use Virtual Devices equipment to enter the Roll-Off, but it just might give you the control you need to cross the finish line first.

For more information and Roll-Off rules, mail the attached coupon today to: Virtual Devices, Inc., P.O. Box 30440, Chevy Chase, MD 20814. Or call 1-800-762-ROBO.

VIRTUAL DEVICES INC.

HERE I AM. SEND ME THE RULES.

Yes! I'd like to enter the Virtual Devices 1st Annual Robot Roll-Off. Please send me the rules and details. And tell me all about Virtual Devices products for robot hobbyists.

Name ______________________

Address ______________________

City ______________ State ________ Zip ________

Telephone (   ) ______________________

Mail to: Virtual Devices, Inc., P.O. Box 30440, Chevy Chase, MD 20814.

HERO, Androbot and RB Robot are Trademarked names.

COLOR GRAPHICS
SOUND EFFECTS
VOICE SYNTHESIS
ALL ON ONE BOARD
FOR THE HEATH-ZENITH 89 COMPUTER

COLOR GRAPHICS
- USES TMS 9918A
- 256 x 192 DOTS
- 16 COLORS
- 16 K ON-BOARD VIDEO RAM

A/D CONVERTER
- 8 CHANNELS
- 8 BIT RESOLUTION

PARALLEL I/O
- 2, 8 BIT PORTS
- EACH PORT EITHER INPUT OR OUTPUT

COUNTER-TIMERS
- USES 8253
- 3, 16 BIT COUNTERS
- PROGRAMMABLE

SOUND EFFECTS
- USES AY-3-8910
- 3 TONE AND 1 NOISE CHANNEL
- PROGRAMMABLE
- ENVELOPE CONTROL

PRIORITY INTERRUPTS
- USES 8259
- 8 MASKABLE INTERRUPTS

VOICE SYNTHESIS (OPT 1)
- USES VOTRAX SC-01A
- PHONEME ORIENTED SYNTHESIS

ARITHMETIC PROCESSOR (OPT 2)
- USES 9511A/9512/8231A/8232
- INTEGER AND FLOATING POINT
- TRANCENDENTAL FUNCTIONS

D/A CONVERTERS (OPT 4)
- 2 CHANNELS
- 12 BIT RESOLUTION
- PRECISION REFERENCE

SOFTWARE SUPPORT
- ROUTINES FOR ALL FEATURES
- COMPATABLE WITH HA-8-3 SUPPORT ROUTINES

PASCAL/MT+™
- ALSO AVAILABLE FROM **NOGOS**
- PROGRAM DEVICES WITH PASCAL

OPTION NO 1 **$130** ·
OPTION NO 2 **$210** ·
OPTION NO 4 **$110** ·

·PRICES SHOWN FOR OPTIONS PURCHASED WITH BASIC BOARD. ADD $15 IF PURCHASED AT A LATER DATE

BASIC BOARD **$399** P/N HA-89-3
SPECIFY H-17/37/47

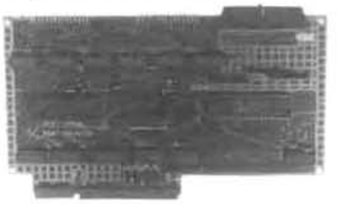
PASCAL/MT+ IS A TRADE MARK OF DIGITAL RESEARCH

NEW ORLEANS
GENERAL DATA SERVICES, INC
7230 CHADBOURNE DR
NEW ORLEANS LA 70126
(504) 241-9495

H/Z89 PERIPHERALS from SECURED COMPUTER SYSTEMS

2/3 PORT SERIAL CARD I/O PORT PARALLEL

" . . . not your typical vanilla-flavored serial and parallel interface . . . "



Features:

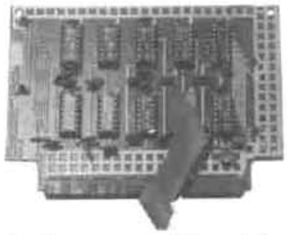
Chip independent design • Reduces computer data buss loading from 3 to 1 • Choice of Centronics or Epson parallel drivers for HDOS or CP/M • Complete documentation and installation instruction.

- 2 Serial Ports
- Supports: Ring Input, External Clock, Auto Dialer
- 3 Port Parallel with 2 Level Interrupt Control
- Fully compatible with all models of H/Z 88, 89, 90 using CP/M or HDOS.
- Fully tested, 90 day warranty, two serial cables and a parallel cable (internal to computer) and software driver.

PRICE $199.00
Shipping & Handling $10.00

16K RAM EXPANSION CARD

Expands your H/Z89 RAM Memory capacity to a FULL 64K!



Fully compatible with:
H/Z 89 • H/Z 88 • Magnolia Microsystems CP/M and disk drive I/O interface cards

NOW INCLUDING SUPPORT MOUNTING BRACKET

Featuring:
Complete installation instruction • 90 day Warranty
Field reliability record now entering its 21st month

Now Only $65.00

HDOS is a registered trademark of Heath Company
CP/M is a registered trademark of Digital Research

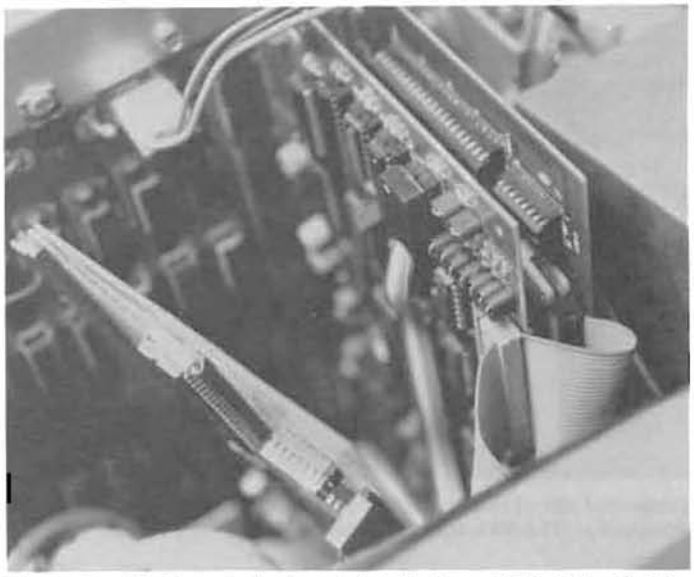
Shipping & Handling $5.00

PRICES ARE LESS SHIPPING AND TAX IF RESIDENT OF CALIFORNIA
MAIL ORDER: 12011 ACLARE, CERRITOS, CA 90701 (213) 924-6741
TECHNICAL INFO/HELP:
8575 KNOTT AVE., SUITE D, BUENA PARK, CA 90620 (714) 952-3930
Terms and specifications subject to change without notice.

ZENITH data systems
SERVICE CENTER

H-17/H-37 Dual Controller Modification

Robert H. Todd, Jr., President
Extended Technology Systems
1121 Briarwood
Bensalem, PA 19020


The H-17 and H-37 controller boards installed in an H-89. Notice that the drive cable starts on the H-17 board and jumps to the H- 37 board, then on to the drives.

Introduction

The H-17/H-37 dual disk controller modification is a modification to the H/Z-89 disk drive controllers which allows use of both hard sectored and soft sectored disks in the same set of drives. As presented here, this article assumes the user has up to three disk drives and both the H-17 and H-37 disk drive controllers.

Hardware modifications require several pins bent out and jumpers installed on the H-37 disk controller card only. Software modifications require minor modification to the CP/M BIOS for CP/M operation and a SET option for HDOS operation. Note that Heath/Zenith does not support hardware modifications on Heath/Zenith parts. You install this modification on your H-37 disk controller card at your own risk.

These modifications are based upon modifications defined by David Granz and modified by George Deffendall. David Granz owns Sterling Software and is the author of the public domain improved H-37 soft sectored device driver for HDOS and its improved follow-on, the Super-37 Device Driver, currently offered by Extended Technology Systems. George Deffendall is an officer of Extended Technology Systems.

Modification Discussion

This modification is designed to allow the user to use a single cable to connect both disk controllers with up to three disk drives. Both controllers can access the same disk drives only if they do not interfere with each other electrically. Therefore, hardware and software modifications must be designed to assure there is no chance that interference can happen.

To understand how this controller card modification works, let's look at the drives and cables to see how disk drives get information.

First, disk drives don't care what kind of disks are inserted; they just attempt to read or write what the disk controller tells them to. They also report other events such as track 00, door open, and the presence of an index or sector hole. It is up to the disk drive controller to make sense of what the disk drive is trying to say.

The disk drive cable is the pipeline through which the data passes. Popularly, flat cables are used with every other line on the cable tied to ground to reduce the chance of cross talk or interference in the signals being passed back and forth between the disk drives and the disk drive controllers.

The information passing through the remaining wires in the cable consists of control signals and data sent to the disk from the disk controller and status signals and data sent from the disk to the disk controller. Signals sent are said to use negative logic. That is: a zero (0) signal is indicated by high voltage (+5 volts) and a one (1) signal is indicated by low voltage (0 volts). Negative logic is used because it is easier and requires less logic to connect more than one device to the same line.

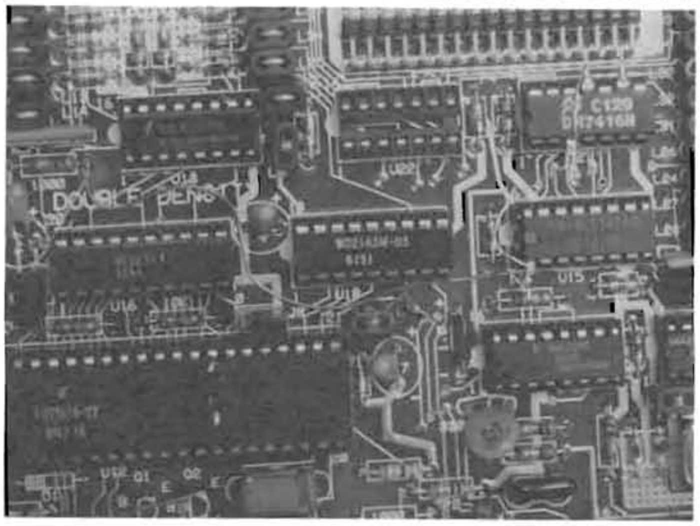
What is necessary, then, to prevent the two controllers from conflicting with each other, is (1) to make sure that a controller is prevented from sending a signal while the other is in operation, and (2) that the control signals are always off (high) when not in use. The first requirement will be accomplished via a software change, and the second requirement will be accomplished via a hardware change.

The disk drive status signals and data lines do not have to be modified. This is because both controllers simply read what is on these lines and do not try to modify these lines at any time. Similarly, the signal write data line through which the disk drives receive data from the controllers is only pulled low when the appropriate controller is writing data. Therefore, we can leave this signal alone.

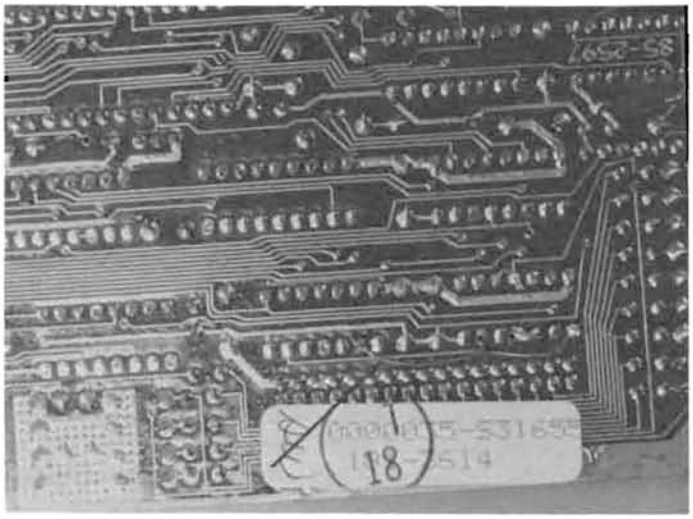
The disk control signals from the disk drive controllers, which must be tied high when not in operation, consist of the disk drive select lines, the write enable line, the head step and direction lines, and the side select line. We will steal some logic on the H-37 controller card to make sure that these lines are pulled high (+5 volts) or turned off when not in use.

A side effect of this modification will be the disabling of the lower plug (P4) of the H-37 card. Since this drive modification requires all drives to be connected to the same cable, this should cause no major difficulty. We will get our extra signals necessary to tie the disk drive control lines high from the logic which formerly was connected to this drive cable plug.

The software modifications to prevent the two controllers from interfering with each other are fairly simple. For HDOS, a set option is necessary to force each disk controller to reset itself when the other controller is in operation. For CP/M, the BIOS is modified to include a call to a controller reset routine when the opposing controller is in operation.



Component side of H-37 controller board showing jumper wires tack soldered to ICs U15 & U21, and to 1000 ohm resistor.



Foil side of H-37 controller board showing jumper wire installed (arrow).

As a consequence of this modification, the disk drive identification varies as a function of which controller is used. This is because Heath/Zenith selected different methods of addressing disks in their hard and soft sectored disk controller cards. We will explain this at the end of this article.

Now let's get to the modifications.

Hardware Modifications

On the H-37 soft sectored disk controller card, make the following changes:

☐ 1. Remove the 7416N from IC socket U22.

☐ 2. Remove the 74LS367 from IC socket U15. Bend pin 1 out at right angles and replace in the socket.

☐ 3. Remove the 7416N from IC socket U21. Bend pins 3, 9, and 11 at right angles and replace in the socket.

☐ 4. Using wire wrap wire, or other fine wire, solder wire from the four pins bent out, above to the end of the 1K resistor at R8. A single wire can be used to connect the pins bent out on the IC at socket U21. Solder the wires to the end of the resistor at R8 NEAREST THE CENTER OF THE CIRCUIT BOARD. The other end is +5 volts which could damage both controllers and the disk drives electrically. BE CAREFUL.

☐ 5. Turn the H-37 controller card over so that the solder side faces up.

☐ 6. Using wire wrap wire or similar wire, connect a jumper between pins 3 and 4 of IC socket U21.

☐ 7. Using wire wrap wire or similar wire, connect a jumper between pins 10 and 11 of IC socket U21.

☐ 8. Using wire wrap wire or similar wire, connect a jumper between pin 8 of IC socket U21 and pin 9 of IC socket U22.

☐ 9. Turn the H-37 controller card component side up and install jumper plugs at pins J4, J5 and J6.

☐ 10. Carefully inspect the modifications you have made to your H-37 disk controller card with these instructions. Be sure that you have connected the jumpers properly to the resistor at R8 AWAY FROM THE FIVE VOLT SUPPLY SIDE.

☐ 11. Install the H-37 disk drive controller into the center I/O slot of the processor board.

Make the following change to the internal drive cable:

☐ 12. Add a 34 connector female ribbon connector to the drive cable.

Finally:

☐ 13. Plug the drive cable back into the H-17 disk controller and to the H-37 disk controller. Use connector P3 (top connector) on the H-37 controller card.

Software Modifications

To operate both controllers simultaneously in HDOS, if you are using the stock Heath/Zenith controllers, there are no changes required. If you use the Ultimeth SY: driver, it is necessary to use the SET command as follows:

SET SY: SELECT ZERO

There, that was simple!

In CP/M, the modification is a bit more complex. It will be necessary to add a call to each of the disk controller routines to reset the opposite drive controller. The reset calls are added as follows:

```
Old H17 Code:

        RDYH17D:
                MVI     A,D$E$UNR
                CALL    H17E
        RDYH17E:
                JMP     XIT
        ONH17:  EI
                LXI     H,0
                SHLD    DLYMO
                LDA     CTLPRT
                ANI     0FFH-040H
                STA     CTLPRT
                OUT     H88CTL
                LHLD    HSTDPB
New H17 Code:

        RDYH17D:
                MVI     A,D$E$UNR
                CALL    H17E
        RDYH17E:
                JMP     XIT
        ONH17:  EI
                IF      H37T                ;MOD TO RESET H37
                CALL    RESH37              ;MOD TO RESET H37
                ENDIF                       ;MOD TO RESET H37
                IF      NOT H37T            ;MOD TO RESET H37
                LXI     H,0
                ENDIF                       ;MOD TO RESET H37
                SHLD    DLYMO
                LDA     CTLPRT
                ANI     0FFH-040H
                STA     CTLPRT
                OUT     H88CTL
                LHLD    HSTDPB
Old H37 Code:
                JC      RST37
                MVI     A,FD$TS
                OUT     FD$INT
                MOV     A,M
                OUT     FD$TRK
                RET
        ONH37:
                LXI     H,0
                SHLD    DLYM037
                LHLD    HSTDPB
                MOV     A,M
                ANI     DPEDD
                JZ      ONH37A
                MVI     A,CONMFM
        ONH37A: INX     H
New H37 Code:
                JC      RST37
                MVI     A,FD$TS
                OUT     FD$INT
                MOV     A,M
                OUT     FD$TRK
                RET
        ONH37:
                IF      H17T                    ;MOD TO RESET H17
                CALL    RESH17                  ;MOD TO RESET H17
                ENDIF                           ;MOD TO RESET H17
                IF      NOT H17T                ;MOD TO RESET H17
                LXI     H,0
                ENDIF                           ;MOD TO RESET H17
                SHLD    DLYM037
                LHLD    HSTDPB
                MOV     A,M
                ANI     DPEDD
                JZ      ONH37A
                MVI     A,CONMFM
        ONH37A: INX     H
```

Note that the LXI H,0 is replaced with the call.

Loose Ends

When booting from a soft sectored disk, CONFIGUR does not find all hard sectored drives. This is because it does not use this part of the BIOS code, but uses its own internal code. The equivalent code within CONFIGUR (and CONFIG80) must be found and patched to include this reset call. When that is done, CONFIGUR (and CONFIG80) will work properly.

In the meantime, it is suggested that the user run CONFIGUR on a hard sectored disk and copy the BIOS modified to the soft sectored disk. This procedure should be used until a patch is defined for CONFIGUR (and CONFIG80).

Disk Drive Identification

As we said previously, Heath/Zenith did not choose the same addressing logic for the hard sectored controller as they did for the soft sectored controller. In the hard sectored controller, Data Select line 3 is defined to be Drive 0, Data Select line 2 is Drive 1, and Data Select line 1 is Drive 2. In the soft sectored controller, Data Select line 1 is defined to be Drive 0, Data Select line 2 is defined to be Drive 1, and Data Select line 3 is defined to be Drive 2. The soft sectored drive selection logic is in accordance with industry standard.

The following table should aid in determining how the drives are identified.

	Normal Boot on H-17			Normal Boot on H-37		
	DS1	DS2	DS3	DS1	DS2	DS3
HDOS						
H-17	SY2:	SY1:	SY0:	DK2:	DK1:	DK0:
H-37	DK0:	DK1:	DK2:	SY0:	SY1:	SY2:
CP/M						
H-17	C:	B:	A:	F:	E:	D:
H-37	D:	E:	F:	A:	B:	C:

When booting from other than the low order disk in either system, the drive designations rotate for the boot controller only. This can get tricky, so be careful of your drive designations until you get used to the way they change depending on how you boot.

Using Alternate Disk Drives

One of the questions which frequently arises when discussing the modification of disk drive hardware or software is the feasibility of changing disk drives. This is particularly relevant when considering the fact that the H-37 soft sectored card and BIOS software will support both 40 and 80 track drives and both single and double sided drives.

As most of you know, Heath/Zenith selected 40 track, double sided drives for the H/Z-100 rather than 80 track, double sided drives. This reflects industry concern that the floppy disk media is not acceptably reliable in 80 track, drives due to thermal expansion properties and reduced error tolerances. This is also why I am using 40 track, double sided as my 5 inch soft sectored format for mastering the public domain software disks, which I personally make available to the rest of the Heath community for a copying donation.

With the large number of disk drive manufacturers currently offering a large variety of full height and half height disk drives including both 40 and 80 track and both single and double sided models, you generally won't go wrong if you get a new tested unit from a reliable name manufacturer. You, however, must make the decision between 40 and 80 track formats. If your system is not moved around very much and is not subject to wide temperature variances, 80 track will probably be fine. I have chosen 80 track double sided for my 5 inch hard sectored public domain software masters.

Most readers know, of course, that Heath/Zenith supports only 40 track single sided drives in hard sectored format. Therefore, the user must go outside Heath/Zenith to get disk driver software to support 80 track and double sided formats. I personally recommend, and use, the Ultimeth SY: driver for HDOS and Livingston Logic Labs BIOS80 for CP/M. *ED) HSY.DVD is also available from HUG on disk part #885-1121, this 2 disk set is supported by the Heath Users' Group.* Henry Fale is also one of two master distributors for Ray Livingston's BIOS80, the other is Ray Massa of Studio Computers, in Birmingham, Michigan.

We at ETS are a dealer for BIOS80 (we buy from Henry Fale) and also offer a version of the HDOS SY: driver for those customers of ours which require the SY: driver to use the capabilities of our other products.

My system, which uses the modifications described in this article, consists of an H-89 with two 40 track, double sided Qume Model 142 drives and one 80 track, double sided Tandon Model 100-4 drive. George Deffendall, my business partner, uses two 80 track, double sided Tandon Model 100-4 drives and a 40 track, single sided Siemens Model 100-5 drive. Both systems work fine. (My other system, also an H-89, uses the hard sectored controller with a 40 track, single sided Siemens Model 100-5b drive and two 80 track, double sided Tandon Model 100-4 drives, along with the H-47 controller with two Remex soft sectored 8 inch drives.)

Epilog

This concludes the modifications and discussions necessary to use both hard and soft sectored controllers with the same disk drives. As you can see, it's really not too hard and quite powerful once you get used to it.

Happy computing, Bob Todd.

(All rights reserved, Copyright (c) 1983, Todd Enterprises)

PERSONAL ROBOTS

Peripherals and Software for Personal Robots

MICROMATION introduces a new line of products designed to enhance Heathkit's® HERO-I robot. These products eliminate the current problem of having to hand code programs into the robot.

HERO MEMCOM BOARD

This product provides a means to develop programs for the robot using a personal computer, and expands the robot's memory with an additional 30K of RAM. A large wire-wrapping area is also available on the board for user experimentation. The product includes:

- Two 8-bit bi-directional parallel ports with handshaking lines for superfast data transfers between the robot and a computer (connects directly to our APPLE-HERO COMMUNICATOR board), plus two 16-bit timers.
- An RS232 serial port for two-way communications between the robot and any computer having an RS232 serial port.
- Serial communications software in an onboard EPROM which allows uploading and downloading of programs via the RS232 port.
- Sixteen 2K RAM chips in sockets so 2K EPROM chips may be substituted, if desired. "Memory Protect" and "Ram Supply" are used for each RAM chip so programs will not be lost when robot enters "sleep" mode.
- A large (2" × 5") wire-wrap area with CPU address, data and control lines, ±5V, ground, and others readily available.
- All hardware to mount board on rear door of robot and to connect board to the robot's CPU board.
- Complete instruction manual and schematics.

PRICE $295.00

APPLE-HERO COMMUNICATOR

This product provides the hardware and software necessary to implement two-way high speed parallel communications between an APPLE® computer and a HERO-I robot equipped with our HERO MEMCOM BOARD. The product includes:

- A peripheral card for an APPLE that contains two 8-bit parallel ports with handshaking lines, and two 16-bit timers.
- A ribbon cable (about 5 feet long) with connectors for connecting the parallel ports on the card to those on the HERO MEMCOM BOARD.
- Data transfer software for the APPLE board and for the HERO MEMCOM BOARD burned into two 2716 EPROMs. These programs provide ultra fast two-way communications for rapid downloading and uploading of programs.
- A disk containing heavily commented 6808 and 6502 source codes for the communications software. These source codes are compatible with the S-C MACRO ASSEMBLER and the S-C 6800 CROSS ASSEMBLER available for the APPLE from the S-C SOFTWARE CORPORATION. Programs developed using the S-C 6800 CROSS ASSEMBLER can be immediately downloaded and run on the robot using this product and our HERO MEMCOM BOARD.

PRICE $159.00

STORYTELLER

This is an artificial intelligence program that turns HERO into a teller of galactic tales. The program uses an advanced self-programming technique which allows the robot to speak randomly generated, but grammatically correct, sentences concerning his space voyages. The program is both amazing and amusing. After listening to the robot for a while one gets the impression that it has just returned from a space battle with alien invaders.

PRICE: Tape (machine code) **$15.00** Disk (source code) **$25.00**

ROBOT LANGUAGE MACRO

This program consists of an S-C 6800 CROSS ASSEMBLER macro source code which completely implements the HERO Robot Language opcodes. This macro provides the user with a very fast means of developing programs for the robot when used in conjunction with our APPLE-HERO COMMUNICATOR and HERO MEMCOM BOARD. The S-C 6800 CROSS ASSEMBLER is required in order to use this product.

PRICE $35.00

6808 CROSS DISASSEMBLER

This program allows 6808 machine code resident in an APPLE's memory to be disassembled in a format similar to that used by the APPLE monitor. Disassembly can be performed optionally into text files for later EXECing into the 6800 CROSS ASSEMBLER. Machine code resident in HERO's memory (like the robot's monitor) can be made into source codes by first transferring the code to the APPLE's memory (using the APPLE-HERO COMMUNICATOR and HERO MEMCOM BOARD) and then disassembling to a text file. This text file can then be EXECed into the S-C 6800 CROSS ASSEMBLER and meaningful comments added. Source code for the program is also available.

PRICE: Program only **$35.00** Program and source code **$55.00**

Send check or money order to:

MICROMATION INC.
9104 Red Branch Rd.
Columbia, MD 21045

MasterCard VISA

For information call:
(301) 730-1237
9 am-5pm Monday through Friday
MasterCard & Visa welcome

The Single Step Approach to Recovering Deleted Files

C. F. Webber
34 Mills Street
Morristown, N. J. 07960

The subject of recovering deleted files in the HDOS environment can be dealt with in several ways. Some of them have already been discussed here in REMark. Those discussed have involved the use of dump utilities of some sort and a copy operation to another disk. I offer here a different philosophy on the subject and a small utility program that will restore a deleted file in one command.

My philosophy is this. More often than not, the files that get deleted in error are usually those that are currently in use; mostly source code for programs in development. Also, I believe that most of these will fit into the available user RAM. I know that this is not always true, but when it is true the UNDELETE program will prove very useful.

Once it is assembled, this program takes up only three sectors. Therefore, it can be placed on your disks as easily as your device drivers and you need not even care that it is there until you need it. To use it you simply type:

UNDELETE -<drive #> filename.ext

As you can see, you need only to provide two arguments. The first one, the drive number, is optional. If you use it, it must be prefixed with a '-'. The second one, the deleted filename, naturally is required.

Once the command line is typed, your computer will go through pretty much the same procedure that you would have to go through if you were to use a dump utility and then do a copy. The steps are outlined below:

1. Make adjustments for the optional drive # if provided.
2. Find out how much memory you can have for the deleted file.
3. Store the filename in two places, one of them with the first character split off.
4. Read DIRECT.SYS, one sector at a time, searching for a match of the value 0FFH followed by the deleted filename minus the first character.
5. In the buffer, replace the 0FFH with the first character of the deleted filename.
6. Save that sector and byte for later.
7. Perform a routine to allow a write to this read-only file. This routine is from an article by P. Swayne in REMark #28 'Patch Mystery Revealed'.
8. Write the buffer to DIRECT.SYS and close the file.
9. Open the deleted file, read it into memory, and close it.
10. Reopen DIRECT.SYS, restore it to the way we found it, and close it.
11. Open a brand NEW file using the name of the deleted file, write the buffer to it, and close it.
12. Exit to HDOS and it is done!

If this fails, you can still go the way of the dump and copy because DIRECT.SYS will be as it was before this program.

A listing of the program follows. I suggest that you create a test disk and only use the test disk until you have this program working properly. Any program that WRITES to DIRECT.SYS is dangerous if it is not working properly. In testing, use different file names for each test because this program only works if a write has NOT been executed subsequent to the delete. Also, use file names of different lengths (up to 8 characters).

Known Restrictions:
-Files with one character file names can't be recovered.
-If more than one file has been deleted with the same file name but different ext's., the results are unpredictable.

```
*PROGRAM NAME: UNDELETE
*By C. F. Webber 01/12/83
*Morristown, New Jersey
*

         XTEXT   HDOS

S.CFWA   EQU     040352A
$HLIHL   EQU     030211A
IOC.FLG  EQU     000004A

         ORG     USERFWA

START    LXI     H,0      *Clear H&L
         DAD     SP       *Get the STACK
         MOV     A,L
         CPI     200Q     *Are there any arguments
         JNZ     CHKARG   *Yes, go get them
         SCALL   .CLOSE
         JMP     EXIT     *All done so EXIT to HDOS

*****************
*The ERROR Stuff*
*****************

*FATAL ERROR!  No further processing.
ERROR    MVI     H,07Q
         SCALL   .ERROR
         JMP     EXIT

*Error requiring the restoral of direct.sys
*but kill the program before an attempt is
*made to recreate the deleted file
ERROR1   MVI     H,07Q
         SCALL   .ERROR
SETKILL  XRA     A        *Move 0 to A
         STA     KILLFLG
         JMP     CONTINU

ERCHK    CPI     EC.EOF
         JNZ     ERROR
```

```
        CALL    $TYPTX
        DB      012Q,'SORRY, I can not find that on','e'+200Q
        JMP     EXIT

*************
*Sub Routines*
*************

*Send D&E on a wild goose chase*
LOSED   LXI     D,BUFFER
        RET

*Open the DIRECTORY file*
OPEND   LXI     H,DIRECT
        LXI     D,DEFAULT
        XRA     A       *Move 0 to A
        SCALL   .OPENR
        JC      ERROR
        RET

*Set the End of Search Source Flag
SETFLG  XRA     A       *Move 0 to A
        STA     FLAG
        RET

*Allow WRITES to Channel 0
*Which is only open for READ
*and is Write Protected
**from REMARK #28 pg 36 in
**the article 'PATCH Mystery
**Revealed' by P. Swayne.
CFU     LHLD    S.CFWA
SYNTAX  CALL    $TYPTX
        DB      'Useage is UNDELETE -(drive #) filename.ext',012Q
        DB      'Example:  UNDELETE -1 EDIT.AB','S'+200Q
EXIT    XRA     A       *Move 0 to A & clear flags
        SCALL   .EXIT   *Go back to HDOS

CHKARG  INX     H
        MOV     A,M     *Get character
        CPI     '-'     *Is the drive # specified?
        JNZ     FILENM  *No so use the default & go on
        INX     H       *Point to drive #
        MOV     A,M     *Get it
        STA     DEFAULT+2 *Change default to specified #
        INX     H       *Look at next character
        MOV     A,M
        CPI     ' '     *Is it a space?
        JNZ     SYNTAX  *If not, syntax error
        INX     H       *Point to filename
        PUSH    H       *Save it
*Find out how much memory you can have
        XRA     A       *Move 0 to A
        SCALL   .LOADO  *Load overlay 0
        JC      ERROR
        MVI     A,1
        SCALL   .LOADO  *Load overlay 1
        JC      ERROR
        LXI     H,-1    *Put maximum value in H&L
        SCALL   .SETTOP *Ask for maximum
        LXI     D,-10
        DAD     D       *Subtract 10 for slop over
        SHLD    MAXMEM  *Save the maximum
        SCALL   .SETTOP *Reserve it
        JC      ERROR
        POP     H       *Get the filename back
FILENM  LXI     D,FSTCHR *Point to split name
        LXI     B,FNAME *Point to whole name
RDFLNM  MOV     A,M     *Get a char from filename arg
        ORA     A       *Test for zero
        JZ      OPCALL  *End of filename go on
        CPI     '.'     *Have we reached the EXT?
        CZ      LOSED   *Split filename doesn't need it
        STAX    D       *Save char in split filename
        STAX    B       *Save char in whole filename
*Bump All of the pointers up one
        INX     D
        INX     B
        INX     H
        JMP     RDFLNM  *Do it again
OPCALL  CALL    OPEND   *Open direct.sys
*Read one sector at a time & count them in POSIT
READD   LXI     H,POSIT
        INR     M       *Count one read
        LXI     D,BUFFER
        LXI     B,256   *One sector
        XRA     A       *Move 0 to A
        SCALL   .READ
        JC      ERCHK
        MVI     C,0     *Set up counter for bytes checked
        LXI     D,BUFFER
SEARCH  MVI     B,1     *Set up counter for matches found
*Search for the value 0FFH or end of sector
FINDFF  LDAX    D
        INR     C       *Count one checked
        JZ      READD   *If 0 then need another read
        CPI     0FFH
        INX     D
        JNZ     FINDFF  *Keep looking
FINDFN  LXI     H,REST  *Point to 2nd part of split filename
CKNXT   XRA     A       *Move 0 to A
        CMP     M       *End of split filename?
        CZ      SETFLG  *Mark it
        LDAX    D       *Get char from direct.sys
        CMP     M       *Same as split filename?
        JNZ     SEARCH  *Nope search some more
CKFLG   LDA     FLAG    *Get the flag
        ORA     A       *Is it 0?
        JZ      GOTIT   *Yes, Got a match
*Bump all up one
        INX     D
        INX     H
        INR     B
*Check if 8 chars. have been matched
        MVI     A,8
        CMP     B
        JZ      GOTIT
        INR     C       *Count one checked
        JZ      READD   *If 0 need another read
        JMP     CKNXT   *Keep checking
*Back up the number of chars that matched
GOTIT   DCX     D
        DCR     B
        JNZ     GOTIT
        LDA     FSTCHR  *Get the first char of filename
        STAX    D       *Put where 0FFH was found
        PUSH    D       *Save
        POP     H       *    That
        SHLD    BYTE    *        Address
*Get the last read sector & save that too
        LXI     H,POSIT
        MOV     C,M
        DCR     C
        MOV     M,C     *SAVE FOR LATER
        XRA     A       *Move 0 to A
        SCALL   .POSIT  *Set up that sector
        JC      ERROR
        CALL    CFU     *Allow write
        CALL    WRITE'  *Write it
*Load deleted file into memory if enough space
LODFIL  LXI     D,DEFAULT
        LXI     H,FNAME
        XRA     A       *Move 0 to A
        SCALL   .OPENR
        JC      ERROR1
        LHLD    MAXMEM
        LXI     D,-LASTBUF
        DAD     D
        MOV     B,H
        MOV     A,B
        STA     COUNT   *Save the max sectors in COUNT
        MVI     C,0
        LXI     D,LASTBUF
        XRA     A       *Move 0 to A
        SCALL   .READ
        JC      ERCHK1
        CALL    CLOSE1
        CALL    $TYPTX
        DB      12Q,'SORRY, file does not fit in memory,'g'+200Q
        JMP     SETKILL

ERCHK1  CPI     EC.EOF  *Is it end of file?
        JNZ     ERROR1  *If not, Bad error
        LDA     COUNT   *Get the max sectors
        SUB     B       *Subtract not read
        STA     COUNT   *Store the # read
        CALL    CLOSE1
*Restore direct.sys to original status with
*our file shown as deleted
CONTINU CALL    OPEND   *Open it
        CALL    CFU     *Allow writes
        LHLD    BYTE    *Get the byte back
        MVI     M,0FFH  *Restore the 0FFH value
        LXI     H,POSIT *Get the sector back
        MOV     C,M
        XRA     A       *Move 0 to A
        SCALL   .POSIT
        JC      ERROR
        CALL    WRITE`  *Write it back out
        LDA     KILLFLG *Get KILLFLG
        ORA     A       *Test for 0
        JZ      EXIT    *Kill prog here
*Write the deleted file from memory as if
*it was a brand new file *%*RESTORED*%*
        LXI     H,FNAME
        LXI     D,DEFAULT
        XRA     A       *Move 0 to A
        SCALL   .OPENW
        JC      ERROR
        LXI     D,LASTBUF
        LDA     COUNT   *Get the number of sectors to write
        MOV     B,A     *Put it in B
        XRA     A       *Move 0 to A
        MOV     C,A     *Move 0 to C
        SCALL   .WRITE
        JC      ERROR
        XRA     A       *Move 0 to A
        CALL    $HLIHL
        LXI     D,IOC.FLG
        DAD     D
        MVI     A,014Q
        ORA     M
        MOV     M,A
        RET

*Write to direct.sys
WRITE`  XRA     A       *Move 0 to A
        LXI     B,256   *One sector
        LXI     D,BUFFER
        SCALL   .WRITE
        JC      ERROR
        XRA     A       *Move 0 to A
        SCALL   .CLOSE
        JC      ERROR
        RET

*Close file without EXIT on an ERROR
CLOSE1  XRA     A       *Move 0 to A
        SCALL   .CLOSE
        JC      ERROR1
        RET
*********
*STORAGE*
*********
DEFAULT DB      'SY0SYS'
DIRECT  DB      'DIRECT',0
FSTCHR  DB      0
REST    DB      0,0,0,0,0,0,0,0
FNAME   DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
FLAG    DB      -1
KILLFLG DB      -1
BYTE    DS      2
MAXMEM  DW      0
COUNT   DB      0
POSIT   DB      0
BUFFER  EQU     *
LASTBUF EQU     BUFFER+256

        END     START
*
*               EOF
```

SPECIAL NOTICE

If you would like an UNDELETE program for HDOS that is easy to use, order HUG disk no. 885-1120.

Sunflower Software

13915 MIDLAND DRIVE
SHAWNEE, KANSAS 66216
(913) 631-1333

NEW SOFTWARE

ZDOS:	ZDUMP—Disk Utility $29.95
	ZBMAP—ZBasic Cross Reference $29.95
	SPELGUD—Spelling Checker $45.95
	COLOR GAMES—5 Old Favorites $24.95
HDOS:	HDOSx2 (Multi-user) $199.95
	TSAP (Thermal Systems Anaylsis Program) $149.95
CP/M85:	Mailmate, Easy, S-Basic, General Ledger, Spelgud, Fireman
CP/M 80:	S-Basic

WRITE FOR OUR FREE CATALOG

Patch Page

Pat Swayne
Software Engineer

KEYMAP — WordStar Fix

The pre-configured version of KEYMAP for use with WordStar that is included on the KEYMAP disk (HUG P/N 885-1230[-37]) has one of the keys configured incorrectly. The '7' key on the keypad is supposed to produce Control-R in the normal mode and Control-Q-R in the shifted mode, but instead produces Control-U and Control-Q-U. The easiest way to fix the problem is to run through the KEYMAP configuration procedure and make the change. First, copy KEYWS.COM to your system disk (if it is not already there), and rename it to KEYMAP.COM. Then enter

```
A>KEYMAP S
```

to enter the keymap setup mode. When asked for a status line file, enter KEYWS.MSG. Press return after this and any entry you make while configuring KEYMAP. When asked for a function shift key, press ESC and H. The characters " ↑ [H" (less the quotes) should appear when you do this. After you press RETURN, KEYMAP will prompt for the individual key entries. Hit RETURN in response to each prompt until you get to the one that says "IC or 7", and enter SHIFT-6 and R. The line should look like this after your entry:

```
IC or 7 (^U): ^R
```

Press RETURN after this entry and at each prompt until you get to a second "IC or 7" prompt. This time, enter Control-U and R. The line should look like this:

```
IC or 7 (^QU): ^QR
```

Press RETURN after this entry and at all prompts (including "Want keypad shifted?") until you get to "Enter on/off toggle code". At this prompt, enter a backslash (\), and press RETURN. The re-configured KEYMAP will be written to your disk, and you can rename it back to KEYWS.COM.

DIR19 Fix

The HDOS version of DIR19 (HUG P/N 885-1126) assumes that your console driver is set to allow TAB characters. If it is not, when DIR19 prints more than one column of files, it will erase previous columns from the screen. The easiest way to fix the problem is to enter

```
>SET TT: TAB
```

to fix your console driver. You must have SET.ABS (supplied with HDOS) on your system disk to do this. If for some reason you prefer to have TAB characters expanded to spaces and want to keep the NOTAB setting, you can alter DIR19 to work under this condition with the following changes to DIR19.ASM. First, locate the label DIR and add the code shown in bold print.

```
DIR     MVI     A,3
        MVI     C,0
        SCALL   .CONSL          GET USER'S CONSOLE WIDTH
        STA     CONWI           SAVE IT
        MVI     A,3
        LXI     B,0FFFFH
        SCALL   .CONSL          SET MAX WIDTH
        MVI     A,1
        MVI     C,0
        SCALL   .CONSL          GET USER'S CONSOLE TYPE
        STA     CONTY           SAVE IT
        ORI     1               SET TAB SUPPORT
        MOV     B,A
        MVI     C,0FFH
        MVI     A,1
        SCALL   .CONSL          SET CONSOLE TO SUPPORT TABS
        XRA     A
        LXI     B,8080H
```

Next, locate the label EXIT and add the new code shown below.

```
EXIT    XRA     A
        SCALL   .CLOSE          CLOSE DIRECTORY FILE
        LDA     CONWI           GET USER'S CONSOLE WIDTH
        MOV     B,A
        MVI     C,0FFH
        MVI     A,3
        SCALL   .CONSL          SET IT
        LDA     CONTY           GET USER'S CONSOLE TYPE
        MOV     B,A
        MVI     C,0FFH
        MVI     A,1
        SCALL   .CONSL          SET IT
        XRA     A
        SCALL   .EXIT           RETURN TO HDOS
```

Locate the data area near the end of the file, and add a line as follows.

```
CONWI   DB      0               USER'S CONSOLE WIDTH
CONTY   DB      0               USER'S CONSOLE TYPE
DIRPTR  DW      0               DIRECTORY POINTER
```

After you make these changes, you can re-assemble DIR19.ASM to produce a new DIR19.ABS file. You will need to copy the files HOSDEF.ACM, HOSEQU.ACM, ESINT.ACM, and DIRDEF.ACM from your HDOS Software Tools disk to either the disk containing the edited DIR19.ASM or your system disk. Then you can run the assembler with a command line like this one.

```
>ASM SY1:DIR19=SY1:DIR19
```

This example assumes that DIR19.ASM is on SY1:, and ASM.ABS is on SY0:.

ONECOPY Modification

The following patch will cause ONECOPY to terminate a copy operation with the destination disk mounted instead of the source disk. This eliminates the need to enter /MOU and change disks in order to use the files you have just copied when you are running with STAND-ALONE set. Make this patch with the PATCH.ABS program supplied with HDOS.

```
>PATCH
Patch Issue #50.06.00
File Name? ONECOPY.ABS
Patch ID? IFOJIC
Prerequesite Code? IFBEIADPGEFFCF
Address? 44067
044067 = 100/107
044070 = 044/        (Hit RETURN where
044071 = 021/         no entry is shown.)
```

```
044072 = 226/
044073 = 044/
044074 = 107/
044075 = 315/72
044076 = 224/1
044077 = 046/63
044100 = 315/247
044101 = 277/312
044102 = 044/137
044103 = 072/44
044104 = 001/315
044105 = 063/224
044106 = 246/46
044107 = 312/315
044110 = 137/277
044111 = 044/^D            (Control-D)
Address? ^D
Patch Check Code? MFALAAHF
```

Be sure that you retain an unpatched copy of ONECOPY on one of your disks in case you want to go back to normal operation.

Tiny Pascal Fix

The CONFIGUR program included with Tiny Pascal (HUG P/N 885-1086) breaks one of the rules I have discussed in my "Getting Started with Assembly Language" series. It alters the console width setting and does not restore the user's setting upon exiting. To fix it, first add the following Function to the file PROCLIB.TPI.

```
FUNCTION READWIDTH;      { Read user's console width  }
  BEGIN     { READWIDTH }
        MEM[%63200] := %076;   MEM[%63201] := %003;   {  MVI   A,3    }
        MEM[%63202] := %016;   MEM[%63203] := %000;   {  MVI   C,0    }
        MEM[%63204] := %377;   MEM[%63205] := %006;   {  SCALL .CONSL }
        MEM[%63206] := %062;                          {  STA          }
        MEM[%63207] := %212;   MEM[%63210] := %063;   {        %63212 }
        MEM[%63211] := %311;                          {  RET          }
        CALL(%63200);                                 { Do actual call }
        READWIDTH := MEM[%63212]                      {  Get width    }
  END;  {  READWIDTH  }
```

A good place to add it is just before the Procedure SETWIDTH. Next, edit the file CONFIGUR.TPS and add the variable WIDTH to the variable list as shown below.

```
VAR     ANSWER,                 {  Answer to menu  }
        VALIDANSWER,            {  Valid answer flag  - 0 = continue  }
                                   {                  - 1 = good answer, exit }
        DONE,                   {  Done with program flag  }
        TPASCALOPEN,            {  UPDATE file flag - 0 = TRANSLAT open  }
                                   {                - 1 = TPASCAL  open  }
        WIDTH,                  {   To save user's console width  }
        I        : INTEGER;     {  Counter used on FOR loops  }
```

Now, just before the use of the procedure SETWIDTH, add a line to save the current console width.

```
CLEARDISPLAY;
WIDTH := READWIDTH;             {  Save user's console width  }
SETWIDTH(255);                  {  Prevents HDOS from generating NLs  }
ENTERGRAPHICS;
```

The last change to CONFIGUR.TPS is to change the last call to SETWIDTH so that it restores the user's width instead of setting an arbitrary width of 80 characters. Locate the line containing SETWIDTH(80) and change it to SETWIDTH(WIDTH), as shown below.

```
  SETWIDTH(WIDTH);               {  Restore user's console width  }
  CLEARDISPLAY
END.  {  Main  }
```

After making these changes, run TPASCAL and re-compile the CONFIGUR program.

QBD

HELP FOR DECISION MAKERS
BUSINESSMEN • HOMEMAKERS • STUDENTS

QBD IS A DECISION SUPPORT SYSTEM THAT USES MANY OF THE QUANTITATIVE TECHNIQUES TAUGHT IN BUSINESS SCHOOLS.

ONLY **$49.95** **Requires MBasic**

PICK'EM

PREDICTS NFL GAMES COMPLETE WITH POINT SPREADS.

EASY TO USE ONLY **$19.95**

BOTH PROGRAMS REQUIRE H/Z-89, H-8/H-19 OR Z 100-SERIES

SPECIFY DISK FORMAT AND HDOS OR CP/M.
Add $2.00 S/H per order
(CA residents add appropriate sales tax)

STUART SOFTWARE
25381-G ALICIA PARKWAY
SUITE 316
LAGUNA HILLS, CA 92653
(714) 855-4753

DESIGN & IMPLEMENT PROGRAMMERLESS SYSTEMS... EASILY

Utilizing **SoftTouch-Z80** a non-programmer can design and implement forms oriented applications software quickly and easily. This is possible by the use of fourth generation software techniques from the mainframe computer world.

Some of the features of **SoftTouch-Z80** are:

— Screen design at your finger tips provided by the integration of Display Manager—Z80
- Fields referenced by name.
- Immediate field validation.
- Support of video attributes.

— Data input, output and manipulation easily expressed via 15 keywords.

— Multiple files and random data access support.

— Ideally suited for many data management applications.

- Inventory systems.
- Accounting systems.
- Client information systems.
- And more. . . you design the application.

SoftTouch—Z80 is available with documentation for the low cost of $175. Documentation packages are available for $15 per copy. **Display Manager—Z80** and **TOOLKIT—Z80** are also available for programmers of **FORTRAN—80.**
Inquiries for further information and orders should be sent to **Systems Design Network, Inc.,** PO Box 31232, Independence, Ohio 44131. Orders should be accompanied by check or money order. Ohio residents should include 6.5% sales tax.

The Heath Disk Operating System Directory Structure

David G. Pelowitz
11614 So. 35th Street
Omaha, NE 68123

Have you ever had a disk that HDOS won't mount? Have you ever deleted a file on a disk by mistake and then wondered how to get it back? Are you curious how HDOS keeps track of the files on a disk? The answers to these questions are not as complex as you might think. After all, a computer is really just a dumb chunk of silicon, plastic, and metal and it keeps track.

Let's take a look into the files which HDOS uses to keep track of the disk. To do this we need one of the disk dump type programs available through either HUG or the local Heathkit Electronics Center. Two common programs are SZAP.ABS and DUMP.ABS. Both of these utilities are easy to use and appear to be relatively bug free. I prefer SZAP so that is the one we'll be using. I'll also limit the discussion to HDOS 2.0, earlier versions manipulate the directory differently. **If you are not using 2.0, this discussion will lead you astray.**

I'm going to be taking you by the hand through a typical hard sectored, 100k, H17 type disk. You will need a bootable HDOS disk with SZAP (DUMP or UDUMP). If you have only one drive, set HDOS STAND-ALONE so that you may reset SYO:. Further, you need a freshly initialized disk. Do not SYSGEN this second disk! You should also put SYSCMD and PIP on it. This is going to sound odd, but bear with me for a while and you'll see why. Copy any other file onto the disk, then delete that file. This second disk is the one we will be examining.

From this point on, some paragraphs will have an action line such as "PREPARE THE SECOND DISK". The text which follows will provide information about what will be occurring and what you should see as you perform the indicated task. In each case read the paragraph before you continue.

PREPARE THE SECOND DISK

When you INIT an HDOS disk, INIT fills all 400 sectors with "GL"s. Why GL? Does the name J. G. Letwin ring a bell? I understand he wrote much of HDOS. I guess I too would like my initials on everyone's disk. At this time, track 0 has some extra information put on it. We will get into some of the contents of track 0 in a few minutes. When INIT asks for bad sectors, give it sector 100. You'll see why shortly. The first files to be put on a disk will be put there by INIT. They are GRT.SYS, RGT.SYS, and DIRECT.SYS. Normally, all disks always have all three of these files. Collectively, they are the directory system of HDOS. You can verify their existence by mounting the second disk and performing CAT/S.

BOOT HDOS AND EXECUTE SZAP (DUMP)

Follow SZAP's prompts and select direct disk access. We know about the three directory files and can find them using the directory, but how does HDOS find the directory without using the directory first? Sound like a catch 22 situation? Easy problem; simple answer! Track 0, sector 9 contains bytes pointing to the first sector of both the GRT.SYS and the DIRECT.SYS files.

EXAMINE SECTOR 9

You may step one sector at a time until you get to sector 9 or you may go directly to sector 9. Refer to figure 1 if you don't have SZAP or DUMP. Sector 9 is the last sector of track 0. There is some interesting information stored in this sector. The first byte is the volume number of this disk. Don't change it! If you do, HDOS will not allow you to mount the disk. Each sector contains information outside of the 256 bytes you normally see. There are synchronization, track and sector number, checksum, and volume number bytes all stored outside the user accessible 256 bytes and none of them are easy to change. If the hidden volume number does not match the sector 9 volume number, HDOS will not allow you to mount the disk and will flag an error.

```
                        Track 0 - Sector 9
HEXADECIMAL
ADDRESS   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F        ASCII
------------------------------------------------------------------------------
 0000     8C 94 15 DE 00 EE 00 02 00 16 00 00 00 00 00 00      ................
 0010     00 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20      .
 0020     20 20 20 20 1B 70 20 20 2A 20 2A 20 2A 20 45 44          .p  * * * ED
 0030     2D 41 2D 53 4B 45 54 43 48 20 2A 20 2A 20 2A 20      -A-SKETCH * * *
 0040     20 1B 71 20 20 20 20 20 20 20 20 20 20 00 00 20       .q           ..
 0050     20 20 20 20 0D 0A 53 59 53 54 45 4D 20 43 4F 50          ..SYSTEM COP
 0060     59 52 49 47 48 54 20 48 45 41 54 48 20 43 4F 2E      YRIGHT HEATH CO.
 0070     2C 20 31 30 2F 31 39 37 37 2C 20 37 39 2F 34 0D      , 10/1977, 79/4.
 0080     0A 20 42 59 20 4A 47 4C 2C 20 31 30 2F 31 39 37      . BY JGL, 10/197
 0090     37 2F 67 63 20 28 67 68 74 20 28 43 29 20 48 45      7/gc (ght (C) HE
 00A0     41 54 48 20 43 4F 2E 2C 20 31 39 37 39 0A 20 28      ATH CO., 1979. (
 00B0     62 79 20 47 41 43 28 69 6E 20 72 65 6D 65 6D 62      by GAC(in rememb
 00C0     72 61 6E 63 65 20 6F 66 20 4A 47 4C 29 29 0A 0A      rance of JGL))..
 00D0     04 0A 0A 43 6F 70 FE 00 00 00 00 00 00 00 00 00      ...Cop..........
 00E0     00 00 00 00 00 00 00 00 00 00 00 00 00 FE 00 00      ................
 00F0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      ................

8C(hex) = Volume number        DE(hex) = 222(dec) = First SECTOR of DIRECT.SYS
0011 - 004D = Disk Label       EE(hex) = 238(dec) = First SECTOR of GRT.SYS
```

Figure 1

The byte at address 0003 (hex) contains a pointer to the first sector of the file DIRECT.SYS. Convert this byte to decimal and display that sector to verify your conversion. Now return to sector 9. In the same way the byte at 0005 (hex) is a sector pointer to the file GRT.SYS. Convert this byte and display the first, and only sector of the GRT.SYS. When HDOS mounts a disk, track 0 sector 9 is examined to find these two files. While we are here, there is another interesting item in this sector. Look at what starts at address 0011 (hex). Recognize the disk label? You can safely change the label by editing this sector. The label can continue through the remainder of the sector, but it must end with a null byte (00). Next we will be accessing the disk by file rather then by direct track and sector, so exit SZAP now.

BOOT HDOS AND EXECUTE SZAP (DUMP)

Follow the prompts of SZAP and access RGT.SYS from the second disk. Figure 2 is a typical RGT.SYS. This is the Reserved Group Table. It is a single sector file with a flag for each cluster on the disk. As you may already know, HDOS accesses the disk in clusters instead of sectors. With the standard 100K disk each cluster contains two sectors. Why? That's an easy one to answer. There are 40 tracks of 10 sectors each on this size of disk. That accounts for 400 sectors.

To save time and to simplify things, HDOS uses one byte values as addresses into the disk. But if we want to address any sector on the disk with only one byte, we are limited to 256 sectors. That falls considerably short of the 400 required. But if we group the sectors into two sector clusters, we can address up to 512 sectors (256 clusters). With this scheme and this size of disk we have 400 sectors divided into 200 clusters, well within our limits.

```
                     Reserved Group Table
                            RGT.SYS
HEXADECIMAL
ADDRESS   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F        ASCII
-------------------------------------------------------------------------
 0000     00 00 FF FF FF 01 01 01 01 01 01 01 01 01 01 01     ................
 0010     01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01     ................
 0020     01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01     ................
 0030     01 01 FF 01 01 01 01 01 01 01 01 01 01 01 01 01     ................
 0040     01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01     ................
 0050     01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01     ................
 0060     01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01     ................
 0070     01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01     ................
 0080     01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01     ................
 0090     01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01     ................
 00A0     01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01     ................
 00B0     01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01     ................
 00C0     01 01 01 01 01 01 01 01 FF FF FF FF FF FF FF FF     ................
 00D0     FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF     ................
 00E0     FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF     ................
 00F0     FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF     ................
```

Figure 2

```
                        CLUSTER TO TRACK AND SECTOR

                            CLUSTER LOW NIBBLE
C   | 0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F
L   +--------------------------------------------------------------
U 0 | 000 002 004 006 008 010 012 014 016 018 020 022 024 026 028 030
S 1 | 032 034 036 038 040 042 044 046 048 050 052 054 056 058 060 062
T 2 | 064 066 068 070 072 074 076 078 080 082 084 086 088 090 092 094
E 3 | 096 098 100 102 104 166 108 110 112 114 116 118 120 122 124 126
R 4 | 128 130 132 134 136 138 140 142 144 146 148 150 152 154 156 158
  5 | 160 162 164 166 168 170 172 174 176 178 180 182 184 186 188 190
H 6 | 192 194 196 198 200 202 204 206 208 210 212 214 216 218 220 222
I 7 | 224 226 228 230 232 234 236 238 240 242 244 246 248 250 252 254
G 8 | 256 258 260 262 264 266 268 270 272 274 276 278 280 282 284 286
H 9 | 288 290 292 294 296 298 300 302 304 306 308 310 312 314 316 318
  A | 320 322 324 326 328 330 332 334 336 338 340 342 344 346 348 350
N B | 352 354 356 358 360 362 364 366 368 370 372 374 376 378 380 382
I C | 384 386 388 390 392 394 396 398 TTS TTS TTS TTS TTS TTS TTS TTS
B D | TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS
B E | TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS
L F | TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS TTS
E
    FORMAT: TTS = TRACK/SECTOR      FORMULA: TTS = ((HIGH*16)+LOW)*2
    EXAMPLE:100 = TRK 10 / SEC 0    EXAMPLE: 100 = ((03[hex]*16+2[hex])*2
```

Figure 3

Again, each byte in the RGT.SYS file is a flag for each cluster on the disk. The first byte is associated with the first cluster. The second byte is associated with the second cluster, and so on. But what are these cluster flags and how are they used? Ignore the first five flags for a few minutes and examine the remaining bytes. You will see two types of entries, 01 (hex) and FF (hex). The 01 (hex) flags the cluster as usable by HDOS. The FF (hex) tells HDOS not to use that cluster. Look at the address of the last 01 (hex) in the file. Its address is C7 (hex) and consequently it points to cluster 199, corresponding to sectors 398 and 399, the last sectors on a 400K disk. No, I didn't make a math error. We started counting the sectors and clusters with zero instead of one. You can do the track and sector to cluster conversion yourself or look it up in figure 3. All of the remaining cluster pointers are set to "not usable" because they are beyond the end of the disk. Remember the bad sector we told INIT about. See if you can identify which cluster INIT marked as unusable. Hint: Sector 100 (dec) is the first sector in cluster 50 (dec), that converts to 32 (hex). Did you find the FF (hex)?

EXIT FILE DISPLAY MODE AND DUMP DIRECT.SYS

This file is the meat of the directory, hence its name, DIRECT.SYS. Figures 4, 5, 6, and 7 comprise a complete four sector directory.

```
                               DIRECT.SYS
                               SECTOR  00
HEXADECIMAL
ADDRESS   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F        ASCII
-------------------------------------------------------------------------
 0000     50 49 50 00 00 00 00 00 41 42 53 00 00 03 00 00     PIP.....ABS.....
 0010     08 11 01 DA 1A DA 1A 53 59 53 43 4D 44 00 00 53     .......SYSCMD..S
 0020     59 53 00 00 03 00 00 14 1C 02 DA 1A DA 1A FF 4E     YS.............N
 0030     49 54 00 00 00 00 41 42 53 00 00 03 00 00 20 2E     IT....ABS..... .
 0040     01 DA 1A DA 1A FF 00 00 00 00 00 00 00 00 00 00     ................
 0050     00 00 00 00 00 00 00 00 00 00 00 00 FF 00 00 00     ................
 0060     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 0070     00 00 00 FF 00 00 00 00 00 00 00 00 00 00 00 00     ................
 0080     00 00 00 00 00 00 00 00 00 00 FF 00 00 00 00 00     ................
 0090     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 00A0     00 FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 00B0     00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00     ................
 00C0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF     ................
 00D0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 00E0     00 00 00 00 00 00 FF 00 00 00 00 00 00 00 00 00     ................
 00F0     00 00 00 00 00 00 00 00 00 00 00 00 00 FF 00 00     ................
```

Figure 4

```
                               DIRECT.SYS
                               SECTOR  01
HEXADECIMAL
ADDRESS   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F        ASCII
-------------------------------------------------------------------------
 0100     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 0110     00 00 00 00 FF 00 00 00 00 00 00 00 00 00 00 00     ................
 0120     00 00 00 00 00 00 00 00 00 00 00 FF 00 00 00 00     ................
 0130     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 0140     00 00 FF 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 0150     00 00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00     ................
 0160     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 0170     FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 0180     00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00 00     ................
 0190     00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 00     ................
 01A0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 01B0     00 00 00 00 00 FF 00 00 00 00 00 00 00 00 00 00     ................
 01C0     00 00 00 00 00 00 00 00 00 00 00 00 FF 00 00 00     ................
 01D0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 01E0     00 00 00 FF 00 00 00 00 00 00 00 00 00 00 00 00     ................
 01F0     00 00 00 00 00 00 00 00 00 00 00 17 84 00 88 00     ................
```

Figure 5

```
                               DIRECT.SYS
                               SECTOR  02
HEXADECIMAL
ADDRESS   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F        ASCII
-------------------------------------------------------------------------
 0200     FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 0210     00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00 00     ................
 0220     00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 00     ................
 0230     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 0240     00 00 00 00 00 FF 00 00 00 00 00 00 00 00 00 00     ................
 0250     00 00 00 00 00 00 00 00 00 00 00 00 FF 00 00 00     ................
 0260     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 0270     00 00 00 FF 00 00 00 00 00 00 00 00 00 00 00 00     ................
 0280     00 00 00 00 00 00 00 00 00 00 FF 00 00 00 00 00     ................
 0290     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 02A0     00 FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 02B0     00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00     ................
 02C0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF     ................
 02D0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 02E0     00 00 00 00 00 00 FF 00 00 00 00 00 00 00 00 00     ................
 02F0     00 00 00 00 00 00 00 00 00 00 00 00 00 FF 00 00     ................
```

Figure 6

```
                               DIRECT.SYS
                               SECTOR  03
HEXADECIMAL
ADDRESS   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F        ASCII
-------------------------------------------------------------------------
 0300     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 0310     00 00 00 00 FF 00 00 00 00 00 00 00 00 00 00 00     ................
 0320     00 00 00 00 00 00 00 00 00 00 00 FF 00 00 00 00     ................
 0330     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 0340     00 00 FF 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 0350     00 00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00     ................
 0360     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 0370     FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
 0380     00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00 00     ... ............
 0390     00 00 00 00 00 00 00 00 00 00 00 00 00 00 52 47     ..............RG
 03A0     54 00 00 00 00 00 53 59 53 00 00 00 F0 00 05 05     T.....SYS.......
 03B0     01 DA 1A DA 1A 47 52 54 00 00 00 00 00 53 59 53     .....GRT.....SYS
 03C0     00 00 00 F0 00 4A 4A 01 DA 1A DA 1A 44 49 52 45     .....JJ.....DIRE
 03D0     43 54 00 00 53 59 53 00 00 00 E0 00 42 44 02 DA     CT..SYS.....BD..
 03E0     1A DA 1A FE 47 41 43 20 2F 20 48 45 41 54 48 20     ....GAC / HEATH
 03F0     43 4F 2E 4A 47 4C 29 29 0A 0A 00 17 88 00 00 00     CO.JGL)).......
```

Figure 7

Normally the directory consists of 18 sectors. Step through this file, a sector at a time to familiarize yourself with it. There will always be entries for the RGT, GRT, and DIRECT files in DIRECT.SYS. If you put SYSCMD and PIP on this disk, you should be able to spot those entries also. Now return to the first sector. This sector should have the SYSCMD and PIP entries. There will also be the remanents of an entry for the file you put on this disk, then deleted.

Each entry in DIRECT.SYS consists of 23 (dec) bytes. Refer to Figure 8. The first eight bytes contain the name of the file. If the name of the file is less than eight bytes, the remainder of this area is filled with null (00) bytes. The next three bytes are the file name extension. It too will be padded with nulls to fill the field.

```
             T Y P I C A L   H D O S   D I R E C T O R Y   E N T R Y

                               23 bytes per entry

                                                      R
                                       P   V   C      E
                                       R   E   L      S            S   C
                                       U   R   U  F   E   FG   G   E   R      A
               N                       J   S   S  L   R   IR  LR  LC   ED     LD
               A                E      E   I   T  A   V   RO  AO  AT   AA     TA
               M                X      C   O   E  U   E   SU  SU  SO   TT     ET
        •      E                T      T   N   R  S   D   TP  TP  TR   EE     RE

ASCII D I R E C T . .  S Y S  .   .   .  .   .   o   q   .   • .  • .

BYTES 1 2 3 4 5 6 7 8  1 2 3  1   1   1  1   1   1   1   1   1 2  1 2
  HEX 4449524543540000 535953 00  00  00 E0  00  6F  71  02  231A 231A

        SLWC1234                     MMMDDDDD YYYYYYYM   JAN 3, 1983
  FLAGS = 11100000 = E0        DATE = 00100011 00011010 = 231A

          • = FF if available, FE if after DIRECT entry and available
```

Figure 8

The next two bytes, Project and Version, are not used. The following byte, Cluster, usually contains a value one greater than the number of sectors in each cluster size. In this case we should find 03 (hex). Notice that the three system file's cluster byte is 00.

The Flags byte is next. Each bit in this byte stands for a flag. If the bit is on, then the flag is set. There are a total of eight flags which may be set and cleared: S, L, W, C, 1, 2, 3, and 4. The 1, 2, 3, and 4 are user flags. HDOS and all its utilities ignore those bits. The C flag means the file uses consecutive sectors on a disk. You can display this flag when you do a catalog by using the /JGL switch. Normally, HDOS and both of its overlays, HDOSOVL0.SYS and HDOSOVL1.SYS will have this flag set. Have you ever wanted to clear an L flag? Its simple, just change the flag's byte so the corresponding bit is off. The next byte is reserved and consequently not used in this file.

The following three bytes tell HDOS where to find the file. The first of these is called the First Group entry (I told you this was going to be easy). It contains a pointer to the first cluster of a file. The next byte contains a pointer to the last cluster of a file, consequently it is called the Last Group entry. The third byte tells HDOS which sector in the last cluster is the end of file. A 01 (hex) indicates the first sector in the cluster is the end, whereas a 02 (hex) indicates the second sector ends the file. Simple, right?

The remaining four bytes comprise the creation date and the last altered date. Each of these dates uses two bytes and is structured with low order byte first. Those of you who are assembly language programmers will recognize why, but that's for another article some day. The key to decipher the date bits is "MMMDDDDDYYYYYYYM". The last five bits of the first byte store the day of the month in binary. For example, for Jan. 3, 1983, the day bits would be "00011". The first three bits and the last bit of the next byte are the month, also in binary. In the example above, Jan. 3, 1983 would set the first date byte to "00100011" and the last bit of the second byte to "0". The remaining bits represent the year. Although it is also a binary value, it is based on 1970. In the example we have been using the year 1983. The year bits must therefore contain the binary equivalent of 13 (dec) or "0001101". Put all this together and Jan. 3, 1983 ends up "00100011 00011010". Both dates are encoded in the same fashion.

Now that we know what is in a directory entry, lets look at how HDOS uses this information. You can see from the first sector of the directory that there are entries containing FF (hex) and all null bytes. These are unused entries and are consequently available any time a file is created or copied onto this disk. Take a close look at the entry for the file you copied onto the disk and then deleted. The first character of the directory entry has been changed to FF (hex). That's a flag to HDOS meaning the entry is available for use just like the unused entries are. Note that the entire remaining entry is intact. This is important if you intend to recover a deleted file. Here is your quiz for the day. What would happen if after we deleted a file, we copied another file onto the disk and then wanted to recover the deleted file? Obvious, isn't it? In copying another file onto the disk, HDOS may have chosen the deleted file entry area in DIRECT.SYS to create an entry for the new file! In that case we would have lost the old file information stored here. Further, we probably would also have lost the old file contents on the disk. You'll see why later.

There is one other thing we need to look at while examining DIRECT.SYS. Advance to the second sector of the file. This sector is the last sector of a cluster. It also contains some special pointers which allow HDOS to access the directory faster than normal disk file accesses. Count five bytes backwards from the end of the sector. If you are on the correct sector, you will find a 17 (hex). A 17 (hex) is 23 (dec). That's a familiar number, the length of each DIRECT.SYS entry! The following byte is a pointer to the previous SECTOR of DIRECT.SYS. Skip the 00 byte, the next to last byte is a pointer to the next sector of the DIRECT.SYS. These are sector pointers, not cluster pointers like you saw in the Reserved Group Table, RGT.SYS. What that means is that DIRECT.SYS will always be totally in the first 256 sectors of a disk. If you don't understand why, go back to the explanation of why HDOS uses clusters instead of sectors. Follow these pointers through the entire DIRECT.SYS and write them down. Remember, they will only be in the last sector of each cluster. Therefore, look in alternate sectors. HDOS uses these pointers to rapidly find each sector of the DIRECT.SYS file. We now know how to find the starting location and the ending location of a file. But what about the stuff in between?

EXIT FILE DISPLAY MODE AND DUMP GRT.SYS

Ever wonder why HDOS had a file with the alphabet in it? And further, why that alphabet was goofed up? Well, here is your answer. It just looks like the alphabet, and here is why. This single sector file contains the information that tells HDOS which clusters belong with which files. Figure 9 is a typical GRT.SYS. Let me digress for a moment. Remember the Reserved Group Table and how each entry in it referred to a specific cluster on the disk? The first byte indicated the status of the first cluster. The second byte indicated the status of the second cluster and so on. The same relationship exists here. The address of each byte in this table is the same as the number of the cluster it references. Wow, that was a mouthful! Again, there are C7 (hex) clusters, starting at 00, each of which must either belong to a file or be available to HDOS via the freepool. The freepool is a list of the clusters which are not being used by any file.

The first five bytes in this file are pointers to track 00 (five clusters yielding 10 sectors). These sectors have special HDOS functions and consequently are not, under any circumstances, available for use.

HDOS uses the first byte of this sector as a pointer to the start of the freepool chain. These pointers each indicate the next cluster which is in the chain. For example, follow the freepool chain. The first byte, address 0000, contains a pointer which is the address of the first empty cluster on the disk. That address in this sector contains a value which points to the second empty cluster. It in turn contains a value of the third and so on. The chain continues in this fashion until a null byte is encountered which indicates the end of the chain. When a file is deleted, its clusters are placed in order back into the freepool.

```
                        Group Reservation Table
                                GRT.SYS
HEXADECIMAL
ADDRESS    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F        ASCII
--------------------------------------------------------------------------------
0000       06 00 FF FF FF 00 07 12 09 0A 0B 0C 0D 0E 0F 10       ................
0010       11 00 13 1D 15 16 17 18 19 1A 1B 1C 00 1E 1F 2B       ...............+
0020       21 22 23 24 25 26 27 28 29 2A 00 2C 2D 2E 2F 30       !"#$%&'()*.,-./0
0030       31 33 FF 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40       13.456789:;<=>?@
0040       41 43 44 45 00 46 47 48 49 4B 00 4C 4D 4E 4F 50       ACDE.FGHIK.LMNOP
0050       51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60       QRSTUVWXYZ[\]^_`
0060       61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70       abcdefghijklmnop
0070       71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F 80       qrstuvwxyz{|}~..
0080       81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F 90       ................
0090       91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F A0       ................
00A0       A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF B0       ................
00B0       B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF C0       ................
00C0       C1 C2 C3 C4 C5 C6 C7 00 FF FF FF FF FF FF FF FF       ................
00D0       FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF       ................
00E0       FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF       ................
00F0       FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF       ................
```

Figure 9

Like the freepool, all files on the disk have a chain associated with them. The address of the null byte in any chain should correspond with the Last Cluster value stored in the DIRECT.SYS entry for that file. The start of a chain is taken from the First Cluster entry in DIRECT.SYS. Find the chain for DIRECT.SYS and follow it from beginning to end. There will be nine clusters used. This is the only file on the disk which leapfrogs back and forth like you just discovered.

In all files other than the DIRECT.SYS, each element in the chain will point to a higher numbered element. All good rules must have at least one exception. It is possible for a file to start close to the end of the disk and then wrap around to lower numbered sectors on the disk. Either way, no file other than DIRECT.SYS will leapfrog around itself. Further, in no case will a file's chain intersect with the chain of another file.

Conclusion

You now have sufficient information to totally rebuild the entire directory structure. Even if all three of the system files were destroyed, the only information you must have is which file each cluster belongs to. If that happens, you can examine all 400 sectors to figure out which file they each belong to. Figure 3 will help you convert back and forth between sector number and cluster number. Ideally, you will have much more information available, particularly in the case of a crashed disk.

The most common disk crash can be simulated by mounting a disk, then removing it without telling HDOS. In its place insert the disk you intend to crash, then tell HDOS to dismount it. Of course most of us don't need any help with this procedure, it comes natural. What happens in this case is the GRT from the mounted disk is inadvertently written to the inserted, unmounted disk, destroying the good linked lists in the GRT. To recover from this type of problem, examine the directory and jot down the first and last cluster for each of the files. At this point you must decide if you want to recover the entire disk or just a specific file or two off the disk. Either way, the disk must have good entries for RGT.SYS, GRT.SYS, and DIRECT.SYS. Delete all file entries in the DIRECT.SYS file which you will not be recovering. FF (hex) in the first byte of each will accomplish this. Careful, don't delete those three system files. While you are here follow the sector pointers in the DIRECT.SYS. Keep track of the order of the clusters so you can recreate the DIRECT.SYS chain in the GRT.SYS. Don't give up now, we are almost there.

The next step is to rebuild the GRT. You will need to edit nearly all of the bytes in it. This step is the most prone to error and the most critical part of recovery. Set the first byte to 0005 (hex), the next to null, and the next three bytes to FF (hex). Now starting with the sixth byte, address 0005 (hex), insert a value of the next higher address. For example, address 0005 (hex) will contain 0006 (hex), address 0006 (hex) will contain 0007 (hex), and so on. Continue through address 00C6 (hex). In 00C7 (hex) enter a null byte. All the following bytes should be FF (hex). At this stage we have the entire disk, excluding track 0, in the freepool chain.

Now we must add the chains for each of the files. A good starting place is DIRECT.SYS. Again sector 9 will show us which sector DIRECT.SYS starts in. Don't forget, this is a sector pointer, not a byte pointer. We would be safer to get the initial sector from the entry in DIRECT.SYS. Using the information from DIRECT.SYS internal sector pointers, modify the GRT cluster pointers. Remember the last cluster pointer must be 00 to indicate an end of file. Also, don't forget that no chain may intersect with any other chain including the freepool chain. The DIRECT.SYS chain is by far the most difficult of them all, be very careful. Once the DIRECT.SYS chain is patched in, then patch the freepool chain around it.

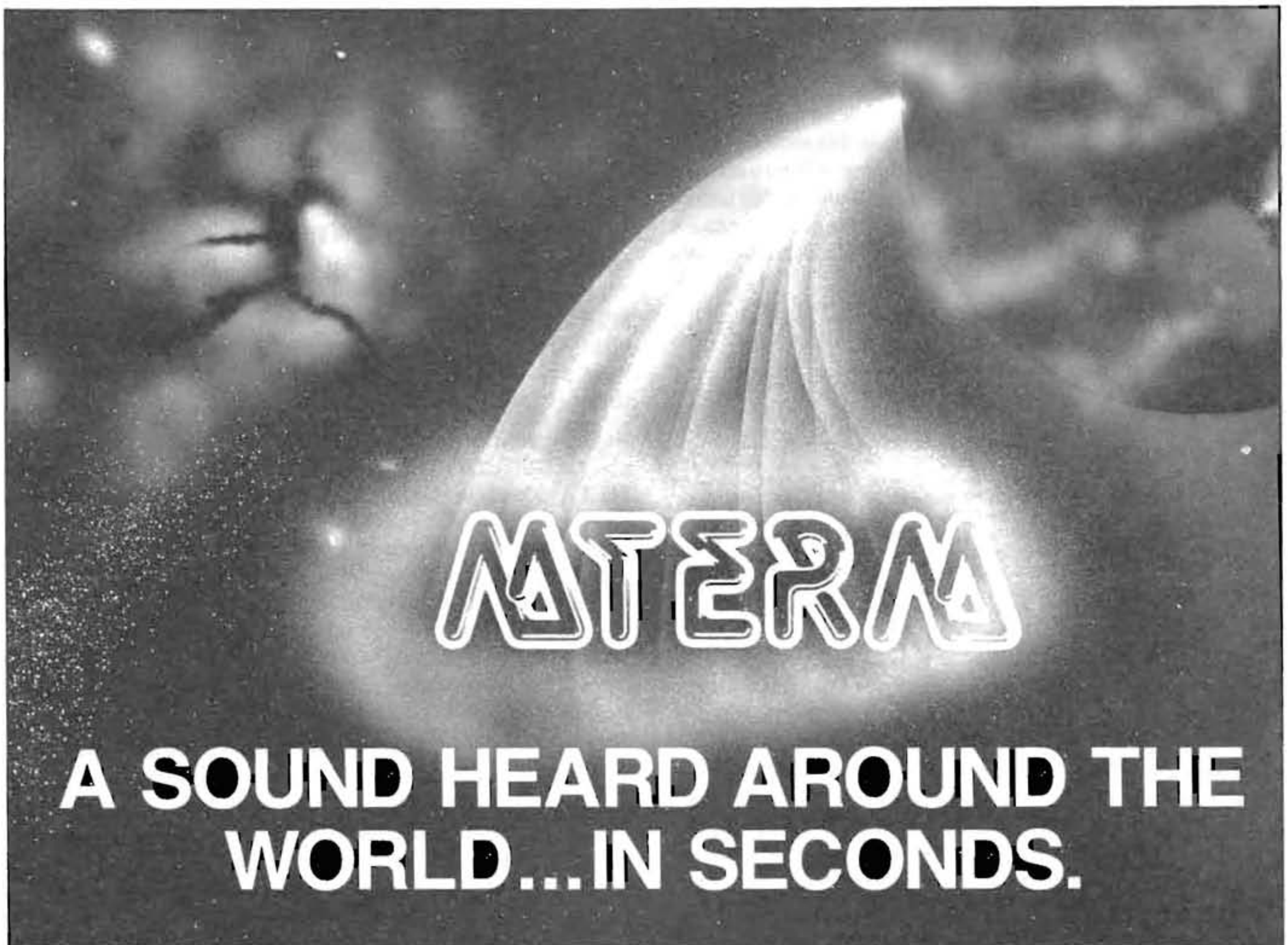
Now patch the RGT and GRT files into the Group Reservation Table and out of the freepool. Usually RGT uses cluster 05 (hex). To patch it in, the byte at 00 must be changed to 06 (hex) and the byte at 05 (hex) must be changed to 00. For each of the files you must start at the initial cluster and step forward through all of the intervening sectors to discover if they belong to that file. In some cases a good guess and a lot of luck is all you have to go on. Look for hints like text messages and keep in mind that you are patching in a cluster but viewing a sector. Look at the second sector if you can't find a hint in the first. Again jot down all the intervening clusters which belong to the file you are recovering. When you reach the cluster listed as the Last Group, you have identified all the clusters belonging to that file. You may now patch them into the GRT. Yes, this can take hours. Continue in this fashion until all of the files are patched back into the disk.

When retrieving a deleted file, you may be able to get by with changing the FF (hex) character in the DIRECT.SYS entry and then mounting the disk and copying the file to another disk. This works only if the file happens to reside contiguously on the disk. Otherwise you will have to patch the GRT.

All considered, mucking with the directory in any way is tricky and prone to error. After all, isn't that what the operating system is supposed to do for us? Generally, keep backup copies of critical data, save your edits often, never delete a file, and never work past 2 in the morning and you can't go wrong.

About the Author:

Dave Pelowitz *is a Captain in the U.S. Air Force. He has been working on or with computer systems for 14 years and is currently managing a software development and maintenance office. Both computer hardware and software comprise the bulk of his hobby time, but programming language syntax and specialized I/O are his favorite topics.*



Micro-System Software presents MTERM – for Z-100, H-100 – the high speed smart terminal – for today and tomorrow's telecommunications. MTERM is the first truly high speed terminal program. Most versions can run at speeds up to 4800 baud without the insertion of null characters. And with the advent of 1200 baud modems and information services, this becomes an important consideration.

But MTERM is more than just a terminal program. It's a complete communications package. Supporting file transfer in both the traditional ASCII mode and the new "error-free" direct file mode.

Not only does MTERM surpass the competition in features offered, but it also offers ease of operation second to none. MTERM's powerful translation tables let you interface MTERM to any variety of communications service. Its unique MacroKey function allows you to have ten "user-defined" keys that transmit up to 64 characters at the touch of a key. MTERM even goes so far as to allow you to dial the phone and transmit the buffer at a specified time completely un-attended by the operator. MTERM is supplied with a user's manual that clearly explains program operation and answers many questions before they are asked.

MTERM will open up the world of telecommunications for you as never before. By using this program, you can do more in less time. The serious computer user cannot afford to be without a MTERM in their library.

And although MTERM geometrically outperforms the others, it's priced at only a fraction of their cost. MTERM is supported through the Micro-Systems technical support network, giving it support previously unavailable for a terminal program. Whether it is for the features, the performance, the documentation, the support, or the price, you just can't lose with MTERM.

CHECK THE FEATURES:

1. Spooled printer output and screen print function.
2. Easy to use translation tables with ready made default files for most applications.
3. Maximum auto dial support with user definable tables to allow dialing from a list of numbers at the touch of a key.
4. MacroKey function allows you to have up to ten "user-defined" keys with a maximum of 64 characters per key. Can be used for multiple "auto-logons" or any of hundreds of other applications.
5. Return to menu and execute MTERM command while still receiving data. Screen will be updated upon return. Never any lost data with MTERM!
6. Supports file transfer with either the traditional ASCII mode or new "error-free" direct file mode.
7. DOS commands from menu without exiting program.
8. Large capture buffer for uploading and downloading.
9. MTERM supports most major brands of modems, with full auto-dial capabilities where applicable.
10. MTERM fully supports some of these modem's advanced features that the competitors won't touch.
11. Simple and easy to use program with clear users' manual.

Now Available at Heath Kit Stores.

MICRO-SYSTEMS SOFTWARE, INC.

4301-18 Oak Circle, Boca Raton, Florida 33431, Telephone: (305)983-3390
Toll Free 1-800-327-8724

Using a Speech Synthesizer Under CP/M

Bob Brownlee
1129 Lantern Square, Apt. 1
Waterloo, IA 50701

I recently purchased the Vortax Type and Talk speech synthesizer from Heathkit and the special hook-up cable. It came with directions on how to modify an HDOS driver but nothing on CP/M. After some experimenting, I found that the Type and Talk is treated just like a printer using a high signal, RTS handshaking. I found that altering the BIOS for a new device would also require the modification of the CONFIGUR program. The next best option was to modify the TTY driver of the CP/M BIOS. This modification can also be used if you have 2 printers that require handshaking without having to change cables. There is one drawback to this modification, that is you have to run CONFIGUR.COM every time you cold boot to initialize the TTY UART for handshaking.

Copy the file BIOS.ASM and ED.COM from your distribution disk. If you are using an H17 drive (2 drives are required), use the following set up: Disk A files ED.COM, Disk B files BIOS.ASM. Enter the following:

A>ED B:BIOS.ASM

Find the label TTYOUT:, it should be line 3355, provided you are using CP/M 2.2.03 and you have not made any previous changes. Change TTYOUT to the following:

```
TTYOUT: LDA     DCLPOS
        ORA     A
        JNZ     TTYOU2
TTYOU1: CALL    TTYOS
        ORA     A
        JZ      TTYOU1
TTYOU2: LXI     H,H84PT2
        LXI     D,TTYCTS
        XRA     A
        STA     DCLPOS
        JMP     UO
```

Now find label TTYOS:, line 3407, and make the following changes:

```
TTYOS:   LXI    H,H84PT2
         LXI    D,TTYCTS
         CALL   UOS
         JZ     TTYOSB
         MVI    A,6
         CALL   PINX
         MOV    B,A
;        MOV    A,B      ;REMOVE SEMICOLONS IN FRONT OF THESE LINES IF YOU
;        CMA             ;USING A LOW READY SIGNAL
;        MOV    B,A
         LDA    010H     ;CHANGE TO 020H IF USING DTR  HANDSHAKE
         ANA    B        ;I HAVE NOT TESTED THIS PROGRAM USING A LOW SIGNAL
         JZ     TTYOSB   ;OR DTR SO I CANNOT PROMISE THAT THESE CHANGES
         LDAX   D        :WILL WORK
         ORA    A
         JNZ    TTYOS1
         DCR    A
         STA    DCLPOS
         RET
TTYOS1:  DCR    A
         STAX   D
         PUSH   B
         MVI    C,NULL
         CALL   UO
         POP    B
TTYOSB:  XRA    A
         RET
```

Once these changes have been made, exit the editor using the E command, and rename BIOS.BAK to BIOS.ASM. Now use the MAKEBIOS facilities to create the new BIOS.SYS (for those unfamiliar with MAKEBIOS, see REMark Issue 26, March 1982, "Making Sense of MAKEBIOS"). Now that you have the disk SYSGENed with the modified BIOS, use the pip command to copy CONFIGUR.COM to the new disk, if you have not done so already. Put the disk in drive A, reset your computer, do a cold boot, and answer <n> to standard system. Set the TTY for proper baud rate and port number, and make the other changes that are needed for your system. Also, on the automatic program control, change cold boot to false. After you have finished with the CONFIGUR program, use the pip command and type in the following:

*TTY:=X:FILENAME

where X:FILENAME is any ASCII file. The speech synthesizer or device should now work. If not, double check to see if you're using the proper baud rate and port. If you are not using the Vortax Type and Talk, make sure that you are using the proper handshaking and signal polarity. Once again, I will repeat that after a cold boot, you must use the CONFIGUR option to initialize the TTY UART for handshaking. Once the CONFIGUR parameters have been set up, just answer <n> for standard system and then enter <x> or <y>. I should point out that you only need to configure if you are using the TTY. An inconvenience I'll admit, but until someone comes up with an article on how to overcome this problem, you will just have to live with it. Looking on the bright side, you can now run 2 printers without having to change cables, and you are not giving up your printer when using a speech synthesizer.

Call For Hero I Articles

REMark Magazine and the Heath Users' Group are very interested in your experiences with your Hero I Robot.

Share these experiences with other HUGgies, submit your ideas today!

A SuperCalc Worksheet Program

Elbert Rufus Rogers
4860 S. Rosette Pl.
Tucson, Az. 85730

The following program is one I wrote to assist me in developing SuperCalc spreadsheets. It was designed to print SuperCalc worksheets on an MPI 99G printer via Stylewriter, however, it should work on other dot matrix printers if they are capable of printing the graphics set of the H/Z-89 computer. I find it less frustrating if I design my spreadsheets away from my computer on a worksheet. At first I had designed a blank form encased in acetate treated plastic on which I could write and erase rather easily. However, this proved to be rather awkward because there were many designs I wished to keep for further reference. Then I developed this short program to print the worksheets for me. This works out very well because I can print out a series of worksheets to cover a wide area and then work on them out of my briefcase whenever time permits.

I could have spent forever developing this program by crunching code and designing prompts that would knock the phosphorous right into your lap, but I fought my hacker instincts and said, "Enough is enough! Now use it.". So I'll leave any enhancements up to other hackers.

To promote a better understanding of how the program functions, it will be explained line for line as much as possible. Who knows, you may learn something.

Lines 10 through 250 are self-explanatory.

Line 280 sets up the sequence to erase the screen by first putting the escape code in a string variable called ESC$. Then I concatenate ESC$ with E and call it ERA$. I normally don't use string variables this long but for the sake of clarity, I have used some long names for some of the variables in this program indicative of their functions.

Line 290 establishes the string variable GRO$ as the sequence to turn on the graphics function.

Line 300 establishes the string variable GRX$ as the sequence to turn the graphics off.

```
10 '      ************************************************
20 '      *                                              *
30 '      *        Supercalc Worksheet   CALSHT.BAS      *
40 '      *                                              *
50 '      ************************************************
60 '
70 '              Written by: Elbert Rufus Rogers
80 '                             7/15/83
90 '
100 '
110 '    This program was designed to provide users of Supercalc with
120 '    a hardcopy worksheet to lay out complex reports on prior to
130 '    entering them onto the CRT. This will help you produce more
140 '    readable and reliable reports. It is written in MBASIC for
150 '    the MPI 99G printer via STYLEWRITER. However other printers
160 '    may work if they will recognize the escape codes and graphics
170 '    character set of the H89 computer.
180 '
190 '    RULES OF THE GAME:       Remember, in Supercalc, the rows
200 '    are numbered from 1 to 254 and the columns from A to BK
210 '    (i.e., A-Z, AA-AZ, and BA-BK). This program will not accept
220 '    anything outside these parameters.
230 '
240 '    ---------------------------------------------------------------
250 '
260 '    DEFINE ESCAPE CODE SEQUENCES
270 '
280 ESC$=CHR$(27):ERA$=ESC$+"E"
290 GRO$=ESC$+"F"
300 GRX$=ESC$+"G"
310 '
320 '    INITIALIZE SYMBOL ELEMENTS
330 '
340 A$="faaaaaaaa"
350 B$="saaaaaaaa"
360 C$="saaaaaaaac"
370 D$="`        "
380 E$="`"
390 F$="vaaaaaaaa"
400 G$="baaaaaaaa"
410 H$="baaaaaaaat"
420 I$="eaaaaaaaa"
430 J$="uaaaaaaaa"
440 K$="uaaaaaaaad"
450 PRINT ERA$
460 '
470 '    PROMPTS AND ERROR TRAPPING SEQUENCES
480 '
490 INPUT"STARTING NUMBER PLEASE ";F
500 IF F < 1 THEN PRINT "NUMBER IS TOO LOW; TRY AGAIN":GOTO 490
510 IF F > 254 THEN PRINT "NUMBER IS TOO HIGH; TRY AGAIN":GOTO 490
520 INPUT"ENDING NUMBER PLEASE ";L
530 IF L < 1 THEN PRINT "NUMBER IS TOO LOW; TRY AGAIN":GOTO 520
540 IF L > 254 THEN PRINT "NUMBER IS TOO HIGH; TRY AGAIN":GOTO 520
550 IF L < F THEN PRINT " ENDING < STARTING NUMBER; TRY AGAIN":GOTO 520
```

```
560 INPUT"STARTING LETTER PLEASE ";ALPHA$
570 IF LEN(ALPHA$) > 2 THEN PRINT"TOO MANY LETTERS; TRY AGAIN":GOTO 560
580 IF LEN(ALPHA$)=2 THEN CHK$=ALPHA$ ELSE GOTO 600
590 IF CHK$ > "BK" THEN PRINT"LETTER BEYOND BK; TRY AGAIN":GOTO 560
600 IF ASC(ALPHA$) > 90 THEN PRINT"MUST BE A LETTER; TRY AGAIN":GOTO 560
610 IF ASC(ALPHA$) < 65 THEN PRINT"MUST BE A LETTER; TRY AGAIN":GOTO 560
620 LPRINT GRO$
630 PRINT ERA$
640 '
650 '   LETTER TO NUMBER CONVERSION PRIOR TO LOOPING
660 '
670 Q=ASC(LEFT$(ALPHA$,1))
680 P=ASC(RIGHT$(ALPHA$,1))
690 GOSUB 990
700 '
710 '   TOP OF WORKSHEET
720 '
730 TOP1$=A$+B$+B$+B$+B$+B$+B$+C$
740 TOP2$=D$+D$+D$+D$+D$+D$+D$+D$+E$
750 LPRINT TAB(6)TOP1$
760 LPRINT F;
770 LPRINT TAB(6)TOP2$
780 '
790 '   BODY OF WORKSHEET
800 '
810 BOD1$=F$+G$+G$+G$+G$+G$+G$+H$
820 FOR Q = F TO L-1
830 IF Q > 253 THEN GOTO 890
840 LPRINT TAB(6)BOD1$
850 LPRINT Q+1;
860 LPRINT TAB(6)TOP2$
870 NEXT Q
880 '
890 '   BOTTOM OF WORKSHEET
900 '
910 BOT$=I$+J$+J$+J$+J$+J$+J$+K$
920 LPRINT TAB(6)BOT$
930 LPRINT GRX$
940 END
950 '
960 '
970 '   ALPHABETICAL HEADING PRINT ROUTINE
980 '
990 FOR I = 1 TO 8
1000 T$=CHR$(Q)+CHR$(P)
1010 IF "BL" = T$ THEN LPRINT SPC(8);"*";:GOTO 1060
1020 IF P > 90 THEN Q=Q+1
1030 IF P > 90 THEN P=P-26
1040 IF LEN(ALPHA$)=2 THEN LPRINT SPC(7);CHR$(Q);CHR$(P);:P=P+1
1050 IF LEN(ALPHA$)=1 THEN GOSUB 1120
1060 NEXT I
1070 RETURN
1080 '
1090 '
1100 '   SINGLE ALPHABET TO DOUBLE ALPHABET TRANSITION ROUTINE
1110 '
1120 LPRINT SPC(8);CHR$(Q);:Q=Q+1
1130 P=P+1
1140 IF Q>90 THEN ALPHA$="AA":Q=65:P=65
1150 RETURN
1160 END
```

Lines 340 through 440 are the graphics elements. I found that the worksheet I had designed contained many similar parts that were very repetitive, so I came up with a list of the common graphics elements. Later as you'll see, I further utilize these graphics elements in designing the unique parts of the worksheet through concatenation.

Lines 490 through 610 are the prompts for input as well as qualifications for proper inputs. I did this to save paper. These lines force the user to conform to SuperCalc protocol but always gives the user another chance to input the proper response.

Lines 490 through 510 request a starting number. SuperCalc does not use negative numbers nor does it use numbers larger than 254, the program will not accept numbers out of this range. However, it will continue to request a valid input.

Lines 520 through 550 are similar to the previous lines with one exception. In line 550, it insures that you do not end your worksheet before you even start it by testing to see that your ending number is not smaller than your starting number.

Lines 560 through 610 are the prompts and tests for alpha inputs. SuperCalc will only accept the letters A-Z, AA-AZ, and BA-BK. These qualifications must be met. Sounds simple doesn't it? However, there are many things that one can attempt to enter unintentionally.

Line 570 prevents the user from entering a string larger than two characters long.

Lines 580 and 590 prevent the program from printing any further than BK. However it will print asterisks over the cells after BK until it reaches the eighth column. This will give the user a little scratch pad area if he desires to use it as such. You can not start with a letter higher than BK but if the program encounters BK during a loop, it will print the rest of the headings with asterisks.

Lines 600 and 610 insure that inputs will only be capital letters by limiting the input to ASCII codes 65 to 90 inclusive.

Line 620 turns on the graphics followed by line 630 erasing the screen.

Lines 670 and 680 convert our string input to numerical form to enable looping across our alphabetical heading printout with the desired information.

Lines 730 and 740 develop the top of the worksheet. As you can see, this is done by concatenating the string variables of the symbol elements as were established in lines 340 through 440.

Lines 750 through 770 prints not only the top of the worksheet but prints the row number as well which is contained in the variable F. Notice the semicolon after F. This keeps the row number on the same line as the print statement in line 770.

Lines 810 through 870 are the body of the worksheet and consequently are the most repeated part of the program depending on what was requested when prompted for input. Notice in line 860 that the bottom of the body portion is the same as the bottom of the top portion.

Line 810 develops the upper section of the body portion of the worksheet in the same manner as line 730.

Lines 820 through 870 will print the body until it exceeds the SuperCalc limit of 254. It appears in line 830 that it will stop at 253, but notice it has already printed one row prior to this point in lines 750 to 770; the top of the worksheet. It can only print a maximum of 253 more.

Lines 910 through 940 are pretty straight forward. Line 910 develops the bottom of the worksheet, 920 prints it, 930 turns the graphics off, and line 940 of course terminates program execution.

Lines 990 through 1070 are the alphabetical heading print routine.

Line 990 establishes a loop, not to exceed eight columns. This works out very well because SuperCalc CRT displays have a default of eight columns with nine characters per cell.

Line 1000 sets up a test variable called T$. This was necessary because the contents of T$ changes as the loop is executing. T$ will actually contain each alphabet heading string before it is to be printed across the top of the worksheet.

Line 1010 checks to see what string is contained in the variable T$. If it contains a BL, then it will not print it but will print a series of asterisks across the top of the worksheet instead. This informs the user that these are invalid column headings and are to be used for scratch pad only.

Line 1020 insures that if an alphabet to be printed in the heading exceeds the letter Z (the number 90 is the decimal ASCII code for the letter Z), then it will increment the letter in front of it by one letter (et.al. AZ to BZ).

Line 1030 works in conjunction with the previous line. After it has incremented the letter in front of it, it must decrement from the letter Z to the beginning as the letter A (et.al. BZ to BA).

Line 1040 prints a two character column heading, then increments the second part of the two character column heading by one alphabet.

Line 1050 tests for single character headings and will look for a special situation in the subroutine at line 1120.

Lines 1060 and 1070 are self-explanatory.

Lines 1120 through 1150 are the single alphabet to double alphabet transition routine as well as the single alphabet print routine.

Line 1120 prints the single character followed by line 1130 incrementing it by one alphabet.

Line 1140 tests to see if the character is a Z (ASCII 90), and if it is, then the next character must be an AA. Thus it has made the two character transition.

Writing this program was very educational to say the least. It was an interesting method of mixing numbers and strings and getting some productive results. There are perhaps many ways to modify this program to produce some tailor made worksheets. I'm sure I will be doing some of these modifications or updates as the need arises. I'll leave it up to you to tailor make some of your worksheets. Here is a list of items you may wish to do to make this program perform by adding to it or modifying it.

(1) Change from the default of eight columns to a selectable amount of columns to accommodate larger or smaller cells.

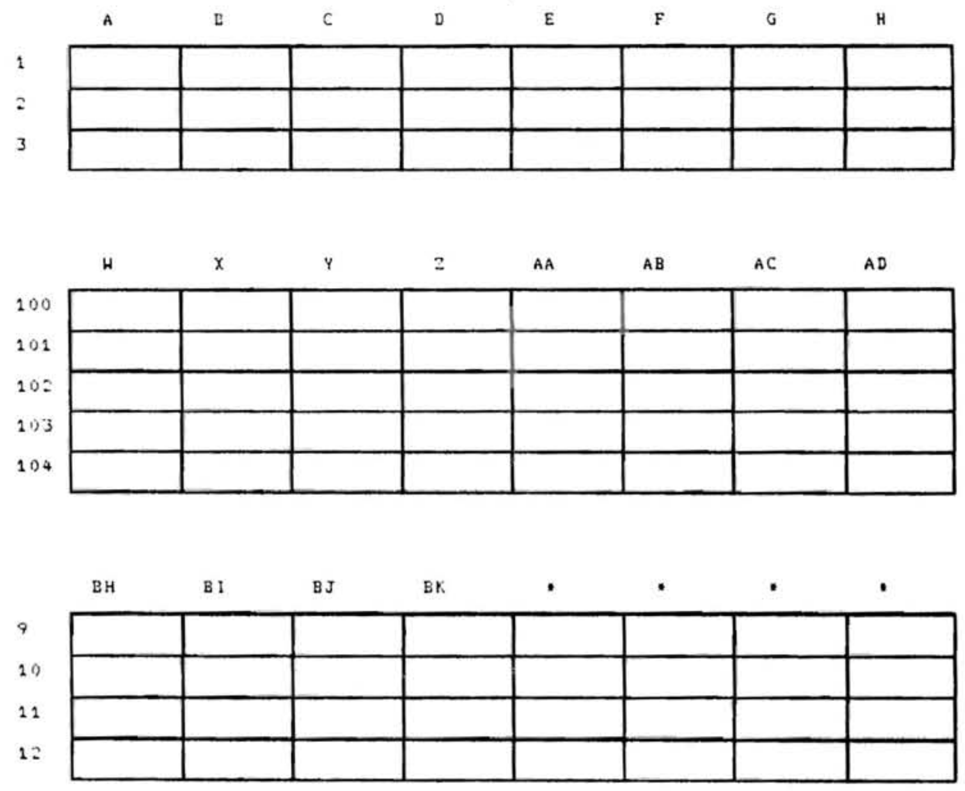
(2) Have the program automatically print the correct amount of columns depending on a cell size selection.

(3) Make provisions to type in commonly used headings in selected cells.

(4) Enable the program not only to print a worksheet vertically but horizontally as well. For example, if you wanted two worksheets, AA-AH and AI-AP, modify the program to accept a range of sixteen letters, then print one eight column worksheet after the other. This could be further modified to accept even larger ranges.

As you might imagine, this program lends itself to a great deal of enhancement. This is what makes programming a challenge as well as a considerable amount of self satisfaction and enjoyment; it can always be tailor made to your demands. So knock this program around in the ol' bit bucket and see what happens. I hope you have as much fun with it as I have had writing it. Even if you don't plan on using this program, you may find some techniques in it that may be useful in some programs you might write in the near future. Happy bytes in the future.

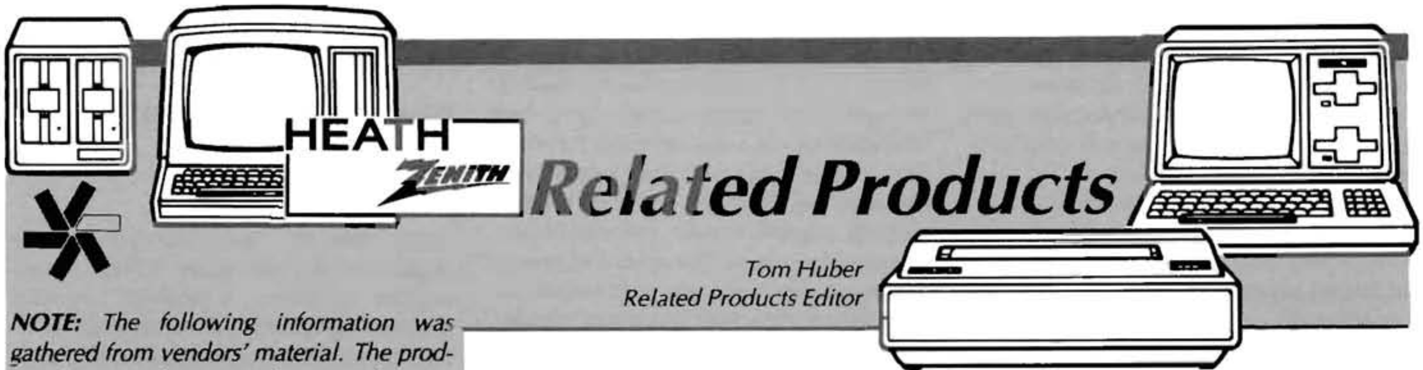
Here are some sample runs.



```
        A     B     C     D     E     F     G     H
  1
  2
  3

        W     X     Y     Z     AA    AB    AC    AD
100
101
102
103
104

        BH    BI    BJ    BK    *     *     *     *
  9
 10
 11
 12
```



Related Products

Tom Huber
Related Products Editor

***NOTE:** The following information was gathered from vendors' material. The products have not been tested nor are they endorsed by HUG. We are not responsible for errors in descriptions or prices.*

This month, I am taking time out to give you a quick look at the HUG convention from my point of view. Having attended and participated in a number of conventions and fairs over the years, several things stuck out in this show that made it something special.

First, I was impressed with your behavior— those of you that attended. You were polite, courteous, and willing to participate. The special workshops and sessions were filled to overflowing; the exhibition area was not dominated by those few who seem to hog every booth; you were looking everywhere; and (I think most of the exhibitors will agree with me) you were spending lots of that green and plastic stuff.

Next, the vendors (exhibitors) were great. They took the time to patiently explain their wares to anyone that was interested. Those who were doing the "fill-in" thing for lunch breaks were doing their darndest to keep up with the questions being fired at them. Some of these vendors are running their businesses in an "infancy" state—real "garage" and "family room" operations (remember, that's how Apple and Pet got their starts). All-in-all, a great job.

Finally, the organizers did a fantastic job laying out the exhibition area. There was plenty of room and it was air- conditioned perfectly. Of course part of the credit also has to go the Hyatt Regency at O'Hare International outside Chicago— their facilities are some of the best available—anywhere. The only real short-coming appeared to be the constant problem of not enough free coffee for those attending and inadequate restaurant facilities.

So much for praising the show. Here are some of the products, new and old, that caught my eye.

DEZIGN is a structured program designer, based on the Jackson Program Design Method. All data and program structures are defined under three basic constructs: sequence, iteration, and selection. DEZIGN enables the user to build data and program structure diagrams, build executable operations, and generate and obtain listings of structured (SBASIC, C, PASCAL, or PL/I) source code. For CP/M-2.2 or 85, one disk drive, and 64K. Documentation includes a tutorial (on the use of the system, not the languages) and a reference guide.

Vendor: Zeducomp
P.O. Box 68
Stirling, NJ 07980
(201) 755-2262
Price: $70.00

Wisconsin Intelligent Systems Engineering (W.I.S.E.) is still offering the 2S+2P+RTC board for the H/Z-89 series of computers. The board features 2 serial ports, 2 parallel ports (true I/O, not Centronics printer ports), and a real time clock. An optional battery (can be rechargeable) will keep the clock on time for two months (minimum). Contact the vendor for full pricing and details on other H/Z-89 products.

Vendor: W.I.S.E.
P.O. Box 344
422 Third St.
Baraboo, WI 53913
(608) 356-9432

Ashton-Tate was offering dBASE II (an alternative to Condor's rDBMS package) and a new program called FRIDAY!. FRIDAY! is a file handling system built around a circular system that allows the user to go directly to the function that is desired. The report system interfaces with a number of popular packages including WordStar, Lotus 1-2-3, and is available under CP/M-80, CP/M-86, PC-DOS, and MS-DOS. Contact the vendor for full details.

Vendor: Ashton-Tate
10150 West Jefferson Blvd.
Culver City, CA 90230
(213) 204-5570
Price: $295.00

Cleveland Condonics, Inc., haven't left the new H/Z-29 terminals out in the cold for high-resolution graphics. The Imaginator 2 is a high resolution (672 x 500 pixel) upgrade for the H/Z-29 that features an on-board graphics processor, up to 6 memory pages for true animation, a very complete instruction set (including get/put and rotation), and a number of options, such as Tektronix 4010/4014 emulators, printer port for hard-copy capabilities, and source code. Contact the vendor for pricing.

Vendor: Cleveland Condonics, Inc.
P. O. Box 45259
Cleveland, OH 44145

Sunflower Software offers a wide range of software for HDOS, CP/M, and Z-DOS. The software includes WordPro-2, a word processing system with a full-screen text editor and file handler; Spelgud and SuperFog for document spell-checking and readability testing; the structured BASIC language, S-BASIC; States, a graphics educational state program; and games and utilities. For a complete brochure, contact the vendor.

Vendor: Sunflower Software
13915 Midland Drive
Shawnee, KS 66216
(913) 631-1333

Valley Data Sciences is offering the 150 Intelligent Memory Programmer, a microcomputer-based, software-driven PROM programming work station employing a screen-oriented, menu-driven system that interfaces quickly and easily with the user. This professional system supports checking, editing, programming, verifying, and maintaining programs for most commonly available PROM, EPROM, and EEPROM devices. The hardware features ZIF (zero insertion force) sockets and supports 16, 18, 20, 24, and 28-pin memory devices. The 150 Intelligent Memory Programmer is available as a complete system with a 64K Z-89; or without the 64K Z-89, if one is available to the user. Contact the vendor for complete details and pricing.

Vendor: Valley Data Sciences
Charleston Business Park
2426 Charleston Road
Mountain View, CA 94043
(425) 968-2900

The Kres Expansion is a six-layer PC board that plugs into (and in parallel with) the H/Z-89 CPU card inside the case. A total of seven "right-hand" slots are available with four doubling as either right- or left-hand slots. Kres Shadow Operation software eliminates conflicts with cards (such as soft- and hard-sectored controllers) that require the same hardware ports. Included on the board are 16K memory sockets to expand the H/Z-89 to a full 64K without having to purchase an additional board. The Expansion and most added boards can operate from the existing Heath power supply. However, if your boards require more power, Kres offers an optional power supply upgrade and fan; or the expansion board can be powered by a secondary supply via a second power connector on the Expansion's board. Contact the vendor for more information on this and other products.

Vendor: Kress Engineering
P.O. Box 17328
Irvine, CA 92713
(714) 559-1047;
(213) 957-6322; or
Computerized Bulletin Board:
(714) 559-8579

Generic Software and dotted i SoftStuf were showing the early development of a new General Ledger system at the show. For those who are interested in receiving early documentation (available sometime after November 15, 1983) should contact the vendor.

Vendor: dotted i SoftStuf
70 Old Queens Blvd.
Englishtown, NJ 07726

Studio Computers Inc. offers a line of hardware and software for the Heath/Zenith computers, including the MPI Printmate 150 and supporting software for dot-addressable graphics. For a complete catalog and prices, contact the vendor.

Vendor: Studio Computers Inc.
999 South Adams
Birmingham, MI 48011
(313) 645-5365
9:30-5:30 Eastern Time Zone

Apogee Software is now offering Gravitron II, a sequel to the popular Gravitron, featuring tanks, fighters, fuel-steeling hovercraft, plasma field launchers, disrupter beams, and more. Available for HDOS and CP/M for the H/Z-89 series or H-8/H-19 combination and color CP/M-85 for the H/Z-100.

Vendor: Apogee Software
Box 15124
Savannah, GA 31416
(912) 925-3765
Price: $19.95 + $1.50 S&H
(GA residents add 4% sales tax)

New Horizon Software was showing SYMED, a color graphics editor for the H/Z-100 series. Basic design appeared to be built around the idea of a size-definable (in pixels) color graphics character generator. Once a character was defined it could be oriented (flipped), copied, stored, and placed anywhere on the screen. The result was impressive, especially in the area of schematic designs and an interesting "dungeon" display. Contact the vendor for more information.

Vendor: New Horizon Software Systems
7115 Blanco Rd. Ste. 114-177
San Antonio, TX 78216

That's all I have room for from the convention. Obviously, with over 40 vendors at the convention, I didn't cover them all. Those of you who weren't mentioned in this column need to note that we publish this information from literature that you send to Heath Related Products, in care of HUG, Hilltop Road, St. Joseph, MI 49085.

Other Related Products

Microservices is offering two new products for the H/Z-100. ZPATTERN is a program that generates a set of video test patterns useful for checking, aligning, and maintaining color monitors. Most patterns contain several test parameter options such as the number of vertical lines of resolution. All user entries are menu selected. ZPALETTE is a color palette program that allows the user to paint simple and complex shapes in any of 92 color hues. The program has a built-in editor for outlining the shape to be painted or it can use a shape generated by other means (ZBASIC color commands, for instance). Contact the vendor for more information.

Vendor: Microservices
P.O. Box 7093
Menlo Park, CA 94025
(425) 851-3414
Prices: ZPATTERN: $29.95
ZPALETTE: $79.95
(CA residents add state sales tax)

The Bit Srubber is a diskette residual noise eraser that can be used with all formats of 5.25- and 8-inch floppy disks including hard- and soft-sectored, and single- and double-sided disks. All densities—single, double, and extended—are also handled. In most instances, the bit scrubber can recover disks that have been made unusable due to residual noise build-up, magnets, or other non-physical causes. For more information, contact the vendor.

Vendor: Techstar, Inc.
8651 N.W. 56th Street
Miami, FL 33166
(305) 592-0201
Price: $49.95 + $4.00 S&H
(FL residents add 5% sales tax)

SPRINTER-2 is a text processor for large documents that runs under SofTech Microsystems' p-System. It features automatic footnote placement and numbering, multicolumn formats, headers and footers, compiling indexes, tables of contents, lists of figures, and maintaining forward and backward references. It makes use of a Macro Formatting Language to assure consistency throughout a document. It supports the popular daisy wheel printers including Diablo, NEC, Printmaster, Starwriter, Transtar, and Qume. Additional drivers for other printers, photo typesetters, and laser printers are under development. An optional spelling checker is also available. For more information, contact the vendor.

Vendor: Scenic Computer Systems, Inc.
14852 NE 31st Circle
Redmond, WA 98052
(206) 885-5500
Prices: SPRINTER-2 . . . . . $350.00
Spelling option . . . $125.00

A free copy of the Programmer's Pipeline checklist is available for those interested. Mention REMark in your request and include a large (#10), self-addressed, stamped envelope.

Vendor: Programmer's Pipeline
P.O. Box 666
Glendore, CA 91740
(213) 914-4317

Steven Meyerson is offering the dBASE II Programmer's Notebook (46 pages, 8-1/2"x11") that contains many tips and techniques for using Ashton/Tate's dBASE II data based manager. In addition to the hints and ideas, there are also a number of useful routines for error checking, two-column printouts, and using '89 graphics. Included is the complete listing for S-Mail, a mailing list and label program. S-Mail is also available on disk, but requires dBASE II to run. Disk versions are available on CP/M(-85) 5.25- inch '100 or '89 formats (soft- or hard-sectored) and 8-inch standard.

Vendor: CompuTech
P.O. Box 2027
Poquoson, VA 23662
(804) 868-8055
Price: $12.95 (add $7.00 for disk, specify version)
(VA residents add 4% sales tax)

More On Using a Hard-Sectored 96 TPI Drive

David A. Sandage
808 Oakland Ave. #107
Urbana, IL 61801

I have been using HUG's SY.DVD and a Siemens FDD-221-5 high capacity drive with my HDOS system for about 5 months and have been very pleased with the results. So, when I decided recently to experiment with CP/M, I very much appreciated Mr. Baker's article in issue 32 of REMark explaining how to use a high capacity drive with the H17 controller under CP/M. While performing the BIOS modifications given in Mr. Baker's article, I discovered a few interesting items that may be of use to anyone who has done the modifications, or anyone who is thinking about it. I have also written a short utility program which copies files between 2 disks using a single high capacity drive. This will be most useful on a system with one 96 TPI drive and two standard drives, since there will not usually be an 'imaginary' drive available for this purpose.

The first item of note is the placement of the BIOS.SYS file on a 96 TPI disk. Mr. Baker states that it must be put on the disk before it is half full. This will insure that the BIOS loader can find it during cold boot. However, I have discovered that if it is not placed on one of the first several tracks of the disk, the CONFIGUR program will not recognize the existence of the drive. Since I do not have access to the source code of CONFIGUR, I am only speculating as to its operation, but here is my explanation. When CONFIGUR is run, it reads the first part of the BIOS from disk to make sure that the version number is correct. This leaves the drive head positioned over the BIOS.SYS file. It then does a hardware verification. I think that the way it can tell whether or not a disk drive exists is by stepping the drive toward track zero. If the drive reaches track zero after a certain number of steps, it will send a TRACK0 signal to the controller, and CONFIGUR will know that the drive exists. So, if the drive head on the 96 TPI drive happens to be positioned further away from track zero than CONFIGUR looks, then CONFIGUR will flag that drive as non-existent. You can of course specify a step time for it, in which case CONFIGUR will recognize it. The moral of all this is that you should try to put BIOS.SYS on a newly sysgened 96 TPI disk as the first file so that you can be sure that CONFIGUR will recognize all of your drives.

When Mr. Baker wrote his article, he used a system that consisted of one 96 TPI drive and one standard H17 drive. This allowed him to use the BIOS' imaginary drive capability to copy files from one high capacity disk to another using a single 96 TPI drive. My system, however, consists of one 96 TPI drive and two standard drives. Since the BIOS only supports 3 hard-sectored drives, this appears to preclude the use of an imaginary drive. The main purpose of this article is to present two ways around this apparent problem.

While I was experimenting with the CONFIGUR utility, I discovered a feature which does not seem to be mentioned in any documentation. The disk options menu of CONFIGUR lets the user change the step times for each disk drive among other things. If you specify a step time of zero for any drive, CONFIGUR will flag that drive as undefined and the BIOS will think that the system has one less drive than it actually does. So, by flagging one of the standard drives as non-existent, and booting from the 96 TPI drive, logical drives A: and C: will both be mapped to the 96 TPI drive. You can then use two different 96 TPI disks to copy files or whatever, by calling one A: and the other C:. Later, when you need three drives again, just run CONFIGUR, and during hardware verification, it will see your third drive and reset its step time to the previous value. Then just write the changes back to disk. IMPORTANT: You must reset the computer and perform a cold boot after you reconfigure the number of drives you are using. This is because the logical to physical drive mapping takes place only at cold boot time. So, if you don't perform a cold boot, your drive mapping tables will not be correct after running CONFIGUR and you could get some strange results. This method solves the problem stated above at the expense of temporarily losing the use of a disk drive.

If you don't need all of the features of an imaginary 96 TPI drive and you just want to copy files between high capacity disks, then you may find the next solution preferable. It is a very simplified version of the HDOS utility ONECOPY. Rather than re-defining drive mappings and the like, it just reads in part of the file from the source disk, resets the disk system, then prompts for a disk change, and writes the buffer out to the destination disk. The program could very easily be expanded to perform all of the functions of the HDOS version of ONECOPY, but this particular version just copies files.

The program itself is written in the "C" programming language and was compiled using AZTEC C II from Manx Software Systems. This is one of the most complete implementations of a C compiler that I have seen for microcomputers. With very few changes, ONECOPY could also be compiled using C/80 from The Software Toolworks. Although not as complete as the Manx compiler, C/80 is probably one of the best values for the money in a C compiler. Since there have been some recent articles in REMark about using C, there are probably quite a few HUG members who are learning this language. Therefore, I will present ONECOPY in a manner which I hope will help these members learn a little more about programming in C.

It is probably most instructive to look first at the overall structure of the program and leave the details until later. This method is often referred to as the "top down" approach. I have already given a brief description of what the program does, but will now make the description a bit more formalized. The first thing that ONECOPY must do is ask for the name of the file to be copied. After receiving a response to that question, it prompts the user to insert the disk containing the file (the SOURCE disk). Since this disk may not have been on the system before, the disk system should be reset before continuing so that CP/M can recognize the new disk. This is not absolutely necessary, since CP/M will notice that the disk has been changed, and will flag the new disk as READ ONLY, but it is good practice to do it anyway. After the source disk has been logged, ONECOPY will look for the file to be copied. When the file has been found, it is opened for reading, and as much of it as will fit into a buffer is read in. The user is then prompted to insert the destination disk. This time it is absolutely necessary to reset the disk system since we want to write on the new disk. ONECOPY then opens a file on the new disk and writes the buffer to that file. If not all of the source file would fit into the buffer, the above procedure must be repeated until the entire file has been copied. When ONECOPY is done copying a file, it will ask the user if there are more files to be copied. If there are, then it will copy them in the manner described above, otherwise, it will exit to CP/M.

Now that we have a specific outline of the program, we can attend to the details. The first line of the program is an INCLUDE directive which tells the compiler to get the file specified (LIBC.H in this case) and include it in the source file. LIBC.H is a file that defines certain constants and data types that are used by the various I/O routines that the program calls. This file is called STDIO.H in some implementations of C. Next comes the main program. In C, the main body of a program is always a procedure called 'main'. A procedure definition in C consists of the name of the procedure followed by a list of arguments. Since 'main' has no arguments, we indicate this with empty parentheses, thus: main(). Next, some variables are defined. Variables are defined in C using lines of the form:

TYPE VAR1,VAR2,...,VARn;

where TYPE is any legal data type specifier, and VARn is a variable name that you wish to declare as type TYPE. An asterisk in front of a variable name means that the variable will be a pointer to an object of type TYPE. So, the first variable in ONECOPY is called 'file' and is a pointer to an object of type FILE. Notice that upper and lower case letters are distinct in C. The variable 'file' will be used to access the file that we want to copy.

The next thing to do is allocate a buffer to read the file into. You will notice that there is a variable called 'buffer' which is a pointer to character. So far no space has been allocated for the character, so 'buffer' points to nothing useful. In order to keep the .COM file small, the buffer space is allocated dynamically. This means that while the program is running (NOT at compile time), we ask the system to set aside some memory to be used as a buffer. That is exactly what the function 'alloc' does. It accepts as an argument the number of bytes that you want, and returns a pointer to the first byte in the block that it allocates. If there is not enough free memory to satisfy the request, then 'alloc' returns a pointer called NULL. I have requested a buffer size of 30K bytes for the file. If you run this program and get the 'Not enough memory' message, you can reduce the number assigned to 'buffsize' and recompile until it runs. Of course, the larger the buffer, the fewer disk swaps will have to be made when copying large files.

At this point, the program prints a sign-on message and enters a loop which will be executed once for every file to be copied. ONECOPY prompts for, and reads in the name of the file. It then sets a variable called 'pos' to zero. This variable is used to keep track of the current position within a long file that will not fit into the buffer all at once. So, after the buffer has been written to the destination disk, we can position the input file right where we left off last time, and read in another buffer full. Since none of the file has yet been read, the position is zero. The program now enters a loop that will be executed as long as the buffer was completely filled during the last read. If the buffer was filled, then there is probably more of the file still to be copied. The input file must be opened before it can be read. This is done in C by a call to 'fopen'. The function 'fopen' will return a pointer to a file descriptor if the file was opened successfully, or NULL if it was not. If the file could not be opened, ONECOPY jumps out of the loop it just entered, and asks the user if there are more files to be copied. The 'break' statement in C causes a jump out of the smallest enclosing loop.

If the file was opened successfully, then the file is positioned to the point described by the variable 'pos' and a buffer full is read in from the file. The function 'fread' reads from the file into the buffer and returns the number of bytes actually read. By comparing 'count' to 'buffsize', we can tell whether or not the buffer got filled up. After the buffer is written to, the program prompts for a disk change and resets the disk system using a call to the BDOS. The output file is opened and positioned, then exactly as many bytes as were read into the buffer are written out to the file. If the buffer was filled on the last read, then ONECOPY loops back to read more, otherwise, we know that the file has all been copied. After the file has been copied, the user is asked if there are more files to copy. If there are, then the main loop is started again, otherwise the program exits with a warm boot to CP/M.

As stated earlier, this program could be expanded to do more of the things that the HDOS ONECOPY program does. Probably the most useful addition would be allowing a disk directory listing from within the program. That way you wouldn't need to remember the names of all the files that you wish to copy. Despite its simplicity, ONECOPY has proven to be a very useful program to me since I don't really like to reconfigure my system every time I want to copy a file from one high capacity disk to another. I hope that both of these methods of copying 96 TPI disks will prove useful to others who have made the BIOS modification.

```
/****************************************************************
 *  ONECOPY - Single Drive Copy Utility v1.1
 *
 *  David A. Sandage
 *  8-Jul-83
 *
 *  Compiled with AZTEC C II
 ****************************************************************/

#include "libc.h"

main()
{
    FILE *file;                         /* file to be copied */
    char *buffer;                       /* pointer to buffer */
    int buffsize,count;
    char string[80],filename[20],ch;
    long pos;                           /* file position */

    buffsize = 30720;                   /* size to allocate for buffer */
    buffer = alloc(buffsize);
    if (buffer == NULL){
        fputs("Not enough memory\n",stderr);
        exit(1);
    }
    fputs("\nONECOPY - Single Drive Copy Utility v1.1\n\n",stdout);
    ch = 'Y';
    while (ch == 'Y'){                  /* while there are files to copy */
        fputs("File to copy: ",stdout);
        gets(filename);
        pos = 0;                        /* beginning of file */
        count = buffsize;               /* fill buffer */
        while (count == buffsize){      /* there is more of file to get */
            fputs("\7Put SOURCE in drive and press RETURN",stdout);
            fgets(string,80,stdin);
            bdos(13,0);                 /* reset disks */
            if ((file = fopen(filename,"r")) == NULL){  /* open file for read */
                fputs("Unable to open ",stderr);
                fputs(filename,stderr);
                fputs(".\n\7",stderr);
                break;
            }
            fseek(file,pos,0);          /* position input file */
            count = fread(buffer,1,buffsize,file);
            fclose(file);
            fputs("\7Put DESTINATION in drive and press RETURN",stdout);
            fgets(string,80,stdin);
            bdos(13,0);                 /* reset disks */
            file = fopen(filename,"a"); /* open file for write on dest disk */
            fseek(file,pos,0);          /* position output file */
            if((fwrite(buffer,1,count,file)) != count){
                fputs("Error during copy.\n\7",stderr);
                fclose(file);
                break;
            }
            pos = pos + count;          /* update file position */
            fclose(file);
        }                               /* end while there is more of file */
        fputs("\nDo you have more to copy? (y or n) ",stdout);
        fgets(string,80,stdin);
        ch = toupper(string[0]);        /* get first char, make upper case */
    }                                   /* end while there are more files */
    fputs("\nInsert a bootable disk and press RETURN",stdout);
    fgets(string,80,stdin);
    bdos(0,0);                          /* warm boot */
}
```

☜ Vectored from 6

my stupid typos!) and impresses all I demonstrate it to.

And, *IF*IT'S*OK*WITH*Miss*McGraw*, I'll be delighted to copy her MBASIC program to anybody's disk sent to me (with return address and postage) FREE! I've even included her REM statements from REMark #38. NOTE: Only 5 1/4" 10-sector hard HDOS disks, please! INIT, number, and title the disk or I'll do that for you if it is a new, blank disk.

Leonard E. Geisler
895 Starwick Drive
Ann Arbor, MI 48105

ESCape Command Equates

Dear HUG,

Again the greatest plague of the computer industry, lack of standardization, raised its head in the June 1983 issue of REMark - no less than three sets of ESCape command equates were shown. My personal invention was E$=CHR$(27), then all upper case ESCapes ended with a U and all lower case with an L, so that JU$=E$+"J", JL$=E$+"j", etc. Then the 'odd-balls' could be assigned to unused letters like IL$=E$+">". I went so far as to define all possible ESCapes and place them into a subroutine that could easily be merged into a BASIC program. But, when I began using this subroutine, I discovered that my programs ran slower!

So, I'd like to caution those who don't already know it: BASIC has a look-up chart and there's no sense in making it look through a bunch of stuff it doesn't need. Enter only the equates into your program that you really need. If you have a lot of variables or a large array, you may be better off to use E$+"J".

Now I've got a question. I'm using an Epson MX-80 printer, with the MX80.DVD supplied by the Heath store where I bought the printer. It worked fine. Recently, however, I added Graftrax Plus, and, of course, modified the driver according to instructions. Now, whenever I open the printer, it immediately prints an "H" at the left edge of the line. It does this whether I OPEN in MBASIC, SAVE"LP:",A IN MBASIC, TYPE filename in HDOS, SAVE LP: in PIE, or NeWOut LP: in EDIT, and any other program. The only way I have found to stop it is in MBASIC to command *OPEN"0",2,"LP:":OUT224,127*. Apparently the delete character put out through the port is fast enough to kill the "H". (*OPEN"0",2,"LP:":PRINT 2,CHR$(127)* will not stop it.) Likewise, *MVI 127* and *OUT 224* in assembly language will delete the

"H". Does anyone know the cause or a solution? Is there somewhere in HDOS or the driver where I could PATCH the output of the 127 through port 224 to stop the "H" every time?

I might add that the HDOS 1.6 I use is modified to allow the printer to slave after the screen. When this is done, the "H" is not printed. But don't blame the modifications, it does this with unmodified versions, as well as with HDOS 2.0

Also, does anyone out there have a solution to HDOS's refusal to read a NULL from a disk? Epson likes NULLs for superscripts and turning underlining off. I have found that they may be read from a file as a random file, but not sequentially. PIE allows all control characters except 0 and Walt told me he has no patch for it because HDOS doesn't like them. (I know what the CP/M boys' answer will be!) My solution for text such as this is to use a CTRL ↑ instead of a NULL, then run through a BASIC program to change that to a NULL. That seems like the hard way!

William E. White
25845 S. Federal Hwy.
Homestead, FL 33032-6698

Dear Mr. White,

You did not say where you got the instructions that you used to modify your Epson driver for use with Graftax, but if you did not get them from HUG, you should make the patch in REMark issue #32 to the original version of your driver. If you still experience the problem, it probably is not in the driver.

Pat Swayne
HUG Software Engineer 31

Patches For Matrix Printers

Dear HUG,

The following patches are the compliments of the Peachtree Software Technical Support Group and apply to Matrix Printers. I have used them on the C.Itoh 1550B Prowriter 2 Printer and can verify they work.

Spool-print problems: (Magic Wand Version 1.12)

1. A>DDT EDIT.COM
2. S10E (Change from 80 to 00)
3. ↑C (Control-C)
4. A>SAVE 67 EDIT.COM

Turn on the underscore feature: (Magic Wand Version 1.12)

1. A>DDT PRINT.COM
2. S2C7F (Change from 08 to 00)
3. ↑C (Control-C)
4. A>SAVE 86 PRINT.COM

To print boldface and underscore:

1. \OUT27,33\ (Turn on boldface)
2. \OUT27,34\ (Turn off boldface)

Example:

\OUT27,33_This is a test message._\OUT27,34\

Will print in boldface and underscore the sentence.

Note:

These Escape Sequences are for the C.Itoh Prowriter 2. Substitute appropriate ASCII codes for your particular printer. If you have changed the Magic Wand default value for underscoring, substitute the new value in the above test message. You can turn on elongated type, or any other special feature of the printer by "tacking" on the appropriate code to OUT command string.

Larry Lankston
3475 St. Catherine St.
Florissant, MO 63033

Introducing Random Access

Dear Bob,

Just a short note to introduce you to Random Access.

Random Access is a weekly newspaper dedicated to the advertisement of used microcomputer hardware and software. The main intent of this newspaper is to offer the owner of used computer items a forum to advertise the sale or availability of the item. This means that we are open to accept classified ads from private individuals. It is not our intention to place nor allow to be placed classified ads for businesses. We will accept paid display ads for any computer business that wishes to do so. We want to afford people greater access to computer items more suited to their needs and desires. It is everybody's chance to shop at home and at their leisure for an inexpensive first system or a desperately needed expansion.

The current subscription rates for Random Access are:

1 year (52 issues) $18.50
2 years (104 issues) $28.00
3 years (156 issues) $39.00

The address is:
Random Access
54 Purcell Drive
Danbury, CT 06810

We hope that this publication will help everyone to find a computer item that they are searching for but have not found, or to sell a used item which is no longer useful to them but may be useful to someone else. Both parties can benefit from the sale or the purchase of a used item. We wish to impress on our readers that a used computer item is not totally useless.

Gerard P. Donohue
Comma Co.
54 Purcell Drive
Danbury, CT 06810

Assembly Language

Dear HUG,

I recently became interested in assembly language programming and found that the distribution disks for the HA-8-2 and HA-8-3 have the HDOS.ACM files on them. This can save a considerable amount of typing. These files also contain a list of address constants for useful routines in the disk controller ROM. However, I found one very small mistake. The comments on the $DADA and the $DADA. equates are reversed. The correct lines are as follows:

```
$DADA   EQU   030072A
(HL)=(HL)+(A) REG A IS NOT DESTROYED

$DADA.  EQU   030101A
(HL)=(HL)+(A) REG A IS DESTROYED
```

I hope this is of some use to someone.

Fred Hochschild
1214 Deutz Ave.
Trenton, NJ 08611

Using the Anchor Signalman, MK I Modem

Dear HUG,

I recently purchased and installed an Anchor Automation Signalman Mark I modem for use with my H-89 and the MD: software package from Newline Software. It works well, but it took a little work to set it up. In the process, I discovered a simple way to install it without buying anything else, at the sacrifice of some I/O flexibility.

The modem comes with a male 25-pin connector installed, and directions to install it to an RS-232 connector port. Using port 320, where it would fit nicely, won't work because it is a DCE port. (The modem comes with instructions on how to construct a special cable, but I'm not sure they'll work exactly as written.) The DTE port at 330, which has a male connector, also won't work without an adapter cable.

My solution was to interchange the 15-pin connectors at P604 and P605 on the I/O board. This connects the DTE port (IC U604) to the female output connector (now mislabeled) on the back of the computer and the DCE port (IC U603) to the male connector. No further rewiring is required, and you can connect the modem directly to the H89.

In order to use the modem, set the port to

330. You must also use the interrupt setting for the DTE port on the I/O board if your software uses interrupts, as MD: does.

Dick Healy
14 Tamarack Lane
Peabody, MA 01960

Assembly Language

Dear Walt,

I've just had the opportunity to take a look at the Heath Assembly Language Course EC-1108 and, since its reputation preceded it, I could not help but chuckle when I saw the troublesome "Type Your Name Ten Times Program". The course is so nicely written that it is a shame that no one put any real effort into updating these programs so that they would run on HDOS and CP/M. But since I've seen the fix to this program in REMark and in other places, I'll ignore it and get on to another more important problem area.

First, the WISE program. This program represents a novel idea; write a program that teaches and reinforces the text material. Unfortunately, this program can only work on an 8080 system and I am surprised that the publisher has chosen to ignore the vast body of users who have Z80 equipment. Luckily the program can be made to work even though the results obtained will differ slightly from the examples in the text. The main point, however, is that the results will be correct for a Z80.

An assumption, widely held, is that a Z80 performs just like an 8080 when fed the same instructions. Not always true! In relationship to this course, the main point of difference is in the handling of the flags. The Auxiliary (Half Carry in Z80) and the Parity flag work differently in two chips and the result can be very confusing to the novice.

Since I have not seen anything mentioned about it until now, please let me offer the following information about these two flags and tell how to fix the WISE program so the student can finish the course with his sanity intact.

The parity flag in the Z80 is different than in the 8080. In fact, it is termed the Parity/Overflow flag. In some instructions it will indicate if the result of an operation has overflowed +127 or -128 in the accumulator. To keep this brief, I will make a sweeping generality and say that rotates and boolean operations affect the parity flag and that addition and subtraction affect the overflow flag. This is not the full story but a Z80 manual will spell out exactly which instructions change what flags. Read one if you really want to learn.

Knowledge of the operation of the parity flag is needed in Unit 10 of the course when the C Register is successivley decremented below zero and the parity flag is seen to change (in the text). This flag will NOT change with this instruction with a Z80 computer. There may be other erroneous references to this in the course. The main thing is that YOU know the difference.

Along the same line is the problem with the WISE program itself as listed on page 9-45,6. The flags will get us into trouble again. Under the label "DOINSTR" notice the two instructions "STA PSWSTOR" and "RAR". The idea of this is to preserve the carry flag by rotating it into the accumulator so that it will not be changed by the following DAD instruction before it can be saved. Unfortunatly, the RAR instruction will wipe out our Auxiliary Carry flag so that now it will be wrong. The Z80 always resets the Auxiliary flag in an RAR instruction and the 8080 does not.

Now, thanks to our wonderful Z80 processor, we have a problem. How can we change this program so that we can still perform the DAD instruction and not change this program so that we can still perform the DAD instruction and not change our flags? Well, since the Z80 got us into this we should let the Z80 get us out of it. Replace the STA PSWSTOR and RAR instructions mentioned above with DB 008H. Do the same thing again 6 lines down the RAL and LDA PSWSTOR instructions, making sure to replace the two lines, again, with a single DB 008H instruction.

The DB 008H instruction will assemble a Z80 instruction into the program that will perform an EX AF,AF' that will swap the primary AF register with the secondary AF register and then swap it back a few lines down when we want our A and F register back intact. The Z80 is incredibly powerful in respect to its ability to move and store data as compared with the 8080. In fact, the entire program could have been rewritten to emulate an 8080 with much less code using the Z80 instructions but then it wouldn't work on an 8080 and that would be pointless.

In summation, if you have found yourself falling for that old line that the Z80 works just like the 8080 then forget it. It certainly can ACT like one, however, with a little care in programming. I suggest that if you want your code to be transportable to both microprocessors that you watch that parity flag and perform DAA operations as soon as possible before the A flag can be changed. Keep conditional calls that depend on the parity flag to a minimum (luckily, they are seldom used anyway).

The information presented in this letter may be a mystery to those HUGgies that steadfastly stick to higher-level languages. It is to this crowd that I suggest you try to learn something about assembly language by whatever means available. Except for some minor points, the EC-1108 is an excellent course, written by a masterful instructor. It takes work, but when your code finally executes you will experience one of the greatest thrills possible in computing; like the way you felt when you finally got out ot the Repository alive.

James Meyers
13A Riggs Parkway
Las Vegas, NV 89115

Patches For Autoscribe Needed

Dear Hug,

I drive a 56K H-89 under HDOS, using (3) hard sector drives. The primary application is word processing. I started with Autoscribe, moved to PIE, and for the last year have been using Wordpro 2.00 from Sunflower Software.

Anyone Understand Autoscribe?

Has any HUGgie discovered a patch for Autoscribe to enable boldface printing on a Daisywriter, Diablo 630, or similar? How about some code to produce headers and footers? Except for these limitations, it is an excellent package and I am sorry to have to leave it on the shelf. Wordpro 2 is very good, but lacks decent block move capabilities.

Anyone For Plotters?

Neither Heath nor HUG offer driver software for a plotter. Anyone aware of a source? In particular, how about something which can drive a (relatively) inexpensive multi-pen plotter such as from Radio Shack?

For that matter, given that my friendly neighborhood HEC always has sold printers, modems, etc., not of Heath manufacture, how come Heath has never marketed an RS-232C plotter package? That can't be more exotic than log-splitters and roll-top desks...and let's hope that comment gets under someone's skin!

There's a profit motive. My employer pays $40 a pop for the color artwork for charts and graphs and the quality is inconsistent. Selling him (and others) 8 1/2 X 11 plotter-produced overhead transparencies at $30 each seems worth exploring.

Warren L. Robinson
155 Stanley Avenue
St. Lambert, P.Q.
J4R 2R4
CANADA

Dear HUG,

The attached program is submitted for inclusion in REMark for those new to programming in B. H. BASIC. It is intended to be improved upon, as it could be classed as a boneheaded way of getting results. For instance, how can trailing zeros be added, why must "A" be multiplied by 100 on line 565, how can the increments be improved, etc.

The program rights are granted to HUG.

James C. Starbird
City Heights Elementary School
301 Mt. Vista Avenue
Van Buren, AR 72956-3399

```
00100 REM COMPOUND INTEREST PROGRAM by J.C. STARBIRD 72956-3399
00110 INPUT "ENTER THE INTEREST RATE AS A PERCENT <8%.=.08>";A
00120 PRINT
00130 INPUT "WHAT AMOUNT IN DOLLARS IS TO BE COMPUTED ?";B
00140 PRINT
00150 PRINT "THE ANNUAL INTEREST PAID ON $";B; "IS $";A*B:REM MULTIPLY INT-PRIN
00160 PRINT
00170 INPUT "DO YOU WISH TO COMPOUND THE INTEREST BY THE MONTH? <Y/N>";C$
00175 PRINT
00180 IF C$="Y" THEN PRINT ,,"WORKING":PAUSE 1000
00190 IF C$="N" THEN 600
00200 LET A=A/12:REM CONVERT INTEREST TO A MONTHLY BASIS <.08/12=.006666+>
00210 PRINT ,"FIRST MONTH",,"$"B+(B*A)
00220 LET C=B+(B*A):REM ADD INTEREST TO PRINCIPAL AMOUNT--INCREMENT EACH MONTH
00230 PRINT ,"SECOND MONTH",,C+(C*A)
00235 LET D=C+(C*A)
00240 PRINT ,"THIRD MONTH",,D+(D*A)
00250 LET E=D+(D*A)
00260 PRINT ,"FOURTH MONTH",,E+(E*A)
00270 LET F=E+(E*A)
00280 PRINT ,"FIFTH MONTH",,F+(F*A)
00290 LET G=F+(F*A)
00300 PRINT ,"SIXTH MONTH",,G+(G*A)
00310 LET H=G+(G*A)
00320 PRINT ,"SEVENTH MONTH,,H+(H*A)
00330 LET I=H+(H*A)
00340 PRINT ,"EIGHT MONTH",,I+(I*A)
00350 LET J=I+(I*A)
00360 PRINT ,"NINTH MONTH",,J+(J*A)
00370 LET K=J+(J*A)
00380 PRINT ,"TENTH MONTH",,K+(K*A)
00390 LET L=K+(K*A)
00400 PRINT ,"ELEVENTH MONTH",L+(L*A)
00410 LET M=L+(L*A)
00420 PRINT ,"TWELFTH MONTH",,"$"M+(M*A):PAUSE 5000
00421 PRINT
00422 INPUT " DO YOU WANT A PRINTOUT ? <Y/N> ";D$:IF D$="Y" THEN 430
00423 IF D$="N" THEN 600
00430 OPEN "LP:" FOR WIRTE AS FILE #1:REM PRINT IT.'LET' STATEMENTS IN REGISTER
00440 PRINT #1,"FIRST MONTH",,"$"B+(B*A)
00450 PRINT #1,"SECOND MONTH",,C+(C*A)
00460 PRINT #1,"THIRD MONTH",,D+(D*A)
00470 PRINT #1,"FOURTH MONTH",,E+(E*A)
00480 PRINT #1,"FIFTH MONTH",,F+(F*A)
00490 PRINT #1,"SIXTH MONTH",,G+(G*A)
00500 PRINT #1,"SEVENTH MONTH",,H+(H*A)
00510 PRINT #1,"EIGHTH MONTH",,I+(I*A)
00520 PRINT #1,"NINTH MONTH",,J+(J*A)
00530 PRINT #1,"TENTH MONTH",,K+(K*A)
00540 PRINT #1,"ELEVENTH MONTH",,L+(L*A)
00550 PRINT #1,"TWELFTH MONTH",,M+(M*A)
00560 PRINT #1," "
00562 LET A=A*12
00565 PRINT #1,"PRINCIPAL AMOUNT WAS $";B;". INTEREST RATE WAS ";A*100;"%"
00570 PRINT : REM ADD A DISCLAIMER!
00580 PRINT #1,"THE COMPOUND FIGURES MAY NOT BE EXACT, BUT ARE CLOSE."
00590 CLOSE #1
00600 END
```

A BDOS ERROR Problem Is Solved

Dear HUG,

I wrote to you previously regarding the problems I was having with the 96 TPI Remex drive I bought from Software Support.

The problem does not appear to be with the drive but appears to be my fault.

I have been using the Z89-37 board for some time. I was operating one 48 TPI drive on the Z89-37 board and put the three 48 TPI drives all back on the H88-1 board.

I had made a number of soft sectored disks on the 48 TPI drive and had no trouble reading them on the 96 TPI drive. In order to copy them to the new 96 TPI disks, I first had to copy the soft sectored disk back to the H88-1, change disks in the 96 TPI drive, and then copy it back. This is when I would get BDOS errors and couldn't copy.

I finally found that in the CP/M manual #595-2776, Chapter 7, Section IV, Page 114, that this potential problem is covered. The instructions state that any time the density of drives is changed that the system should be rebooted by a warm boot.

Since I have been doing this, my BDOS errors have disappeared. This problem was not mentioned in the Z89-37 operation manual that I had been using since I installed the board.

I don't know if any one else has encountered this problem but thought you would like to know how it turned out.

Robert E. Carson
Rt. 2, Box 927
Alva, FL 33920

Cataloging Your Disks the Easy Way

Dear HUG,

Bob Sutherland provided further insight into the uses of the IOBYTE in MBASIC running under CP/M 2.2.03 (REMark #32, September). His simple PEEK/POKE commands provide an easy and direct way to send information to the printer instead of the screen.

I've used this idea to develop the disk cataloging program shown here. This program will title each disk and write a complete directory on the label.

Some comments on the program:

Line 10: Width 60 is set so that the directory only prints 4 entries across my 3.5 inch labels. If you use longer labels, change the WIDTH accordingly. Notice that I have not used WIDTH LPRINT 60 because this would cause the printer to wrap at other than a full entry. Since POKE 3,171 merely shifts output from screen to printer, the wrap will be the same as the screen, i.e., 4 entries across. The rest of line 10 assigns values my OKIDATA 82A recognizes to print bold, normal, and condensed fonts at eight lines per inch.

Line 30: This line directs you to mount the disk you wish to catalog. The next lines show the directory so you can decide what to label the disk (line 80).

Line 90: This line ensures you replace the disk from which you ran this original disk label program. It then RESETs the disk and ends the program. It exits to MBASIC to allow you to go to another program or end.

Line 100: This prompt allows time to align the labels.

Line 120: This line centers the disk title on the label. Change the values to center on your label if necessary.

Line 150: This calls the directory from the disk in place and sends it directly to the printer. An interesting thing happens on my printer. There must be some encoding (probably to indicate SYSTEM FILE and R/O) on the SY of the file BIOS.SYS, because these two characters print out on my printer as graphics characters. Since this only happens on the SYS files and it doesn't affect the rest of the label, I ignore it.

Disk Label Program

```
10  WIDTH 60:E$=CHR$(27):
    CLS$=E$+"E":NORMAL$=CHR$(30):
    LINESPERINCH$=E$+"8":
    CONDENSED$=CHR$(29):
    BOLD$=CHR$(29)+CHR$(31)
20  PRINT CLS$;TAB(23)
    "LABEL PRINTING PROGRAM":
    PRINT
30  PRINT TAB(10)"INSERT DISK
    YOU WISH TO LABEL IN DRIVE A:."
40  PRINT TAB(15)"PRESS RETURN TO
    CONTINUE";:LINE INPUT ANS$
50  PRINT:PRINT"HERE ARE THE
    FILES ON THIS DISK: ":PRINT
60  FILES
70  PRINT:PRINT " (ENTER 'END'
    TO STOP)"
80  LINE INPUT "ENTER
    THE TITLE OF THIS DISK:";LABEL$
90  IF LABEL$="END" THEN PRINT:
    PRINT"REPLACE ORIGINAL DISK
    IN DRIVE A:-PRESS RETURN TO END.";:
    LINE INPUT ANS$:RESET:END
100 PRINT TAB(10)"LINE UP LABELS IN
    PRINTER; PRESS RETURN TO PRINT";:
    LINE INPUT ANS$
110 LPRINT LINESPERINCH8$;BOLD$;
120 LPRINT TAB(16-LEN(LABEL$)\2)
    LABEL$
130 LPRINT CONDENSED$;
140 POKE 3,171:     'OUTPUT TO LST:
150 FILES
160 PRINT CHR$(13):'RETURN NEEDED
    TO CLEAR BUFFER OF PARTIAL LINES
170 POKE 3,169:     'OUTPUT TO CRT:
180 LPRINT STRING$(3,10)
190 PRINT:GOTO 30
```

John L. Dove III
37 Tern Road
Groton, CT 06340

Comments on 4MHz Mod

Dear Mr. Swayne,

The following comments relating to your 4MHz Mod to the H89 will appear in the August Issue of >CHUG, our local HUG newsletter:

***On Pat Swayne's
4MHz H89 Modification**

Pat Swayne at HUG has established himself as an excellent programmer. With the November 1982 issue of REMark, he has also displayed impressive hardware talents. The article described 4MHz modification to the H89. I have made the modification on my machine, and wish to add a couple of pointers. Be sure to read the March 1983 4MHz Update article. In the Update, Pat suggests adding a 330 ohm pull-up resistor to the clock line feeding the Z80A to replace the one taken out by a cut line. However, the old 330 ohm resistor, R506, was left to pull up the clock line feeding U1 on the mod board. Since U1 is a TTL device, no pull- up resistor is needed or desired here. In fact, leaving it in caused my modification to be unreliable. Simply clip the top end of R506 (found just to the left of U502) to correct this problem. It is also possible to make the modification reversible (generally a good practice!) by adding a fourth wirewrap socket in the space left on the mod board and use that to plug in U506. The wirewrap pins of this socket can be inserted into the socket for U506. The same piggy back socket arrangement can be used for U1 on the mod board inserted into a new socket below on the CPU board.

My system continues to work perfectly at 4MHz (1 mo. of daily operation) under HDOS. By the way, instead of reassembling the SY: driver (I am using Dean Gibson's latest driver, which he supplied without source code), I found I could patch the driver where it sets the disk timing constants. I could shorten the old code here by using the floppy ROM routines $MOVE and $ZERO, then added the 4MHz port-out and General Purpose Port HDOS byte setting. Of course, I also had to correct the relocatable address table for this part of the code.

Thanks for your very skilled and dedicated work for HUG and its members.

William C. Parke
1820 S. Street NW
Washington, D.C. 20009

Dear HUG,

Thank you for a very fine publication. Although I have been a Heath user for two years, I still consider myself a novice and the information I get in REMark is very useful. My subscription paid for itself the other day after reading Bill Simpson's article "BDOS ERROR" in the July issue.

After editing a lengthy file with Magic Wand, I looked at the directory, and the file and it's backup were not on the disk. I'm not sure how it happened, I may have exited Magic Wand with "QUIT" instead of "END". Since I had no other copies of this file, I decided to try the approach outlined by Mr. Simpson. It worked! I discovered Magic Wand stores the file being edited in memory beginning at address location 3C00 (Hex). I then located the end, added a page of zeros, moved the contents of the block of memory to the location starting at 100 (Hex), calculated the number of pages in the block, exited DDT, and saved the file. Thank you Bill for the good instructions, and thank you HUG for printing the article.

I also want to thank Pat Swayne for his articles on the 4MHz modifications for the H-89. I have been running for several days now at 4MHz and think it's wonderful. I must admit I was a little fearful of doing the modifications because of my limited electronics background. Thank you for your good instructions.

I also wanted to comment on the CP/M version of MAPLE (Modem Applications Effector) written by Dr. William Parke and distributed through HUG. First let me say that after using the program for several months, I really like it. It is easy to use and has a large number of options available. I especially like the ability to send parts and pieces of the screen activity or the text in the buffer to the printer.

There are, however, two things about the program that are disappointing. The first is the most serious and I pass this on to other users (you can avoid the frustration it caused me). The program uses the terminal in the alternate keypad mode and doesn't seem to change the keypad back to normal mode when the program is exited. If another program is executed without first changing the terminal characteristics back to their defaults (either through a cold boot or through entering the correct escape sequences) and the keypad is used, the program will probably do unexpected things. I noticed the biggest problem in running dBase II after MAPLE. dBase II would crash when the keypad was used for entering the date, corrupting the dBase II program so that a backup copy had to be used. My advice to anyone using MAPLE, do a cold boot after exiting.

The second disappointment is that I wish an option for transmitting files existed which stripped a line feed, carriage return sequence for transmitting to university type computers such as the local campus' PDP-11. Except for these two items, I'm very happy with the program.

I hope this information may be of some help to other HUG members.

Daniel Gilbertson
P.O. Box 158
Platteville, WI 53818

Inkey Routine for Benton Harbor BASIC/HDOS 2.0

Dear HUG,

Listed here is a "Quickie" Inkey routine for Benton Harbor BASIC/ HDOS 2.0 that may be of interest to the few HUGgers like me that are still using Benton Harbor BASIC. The routine bypasses the HDOS SCALL .CONSL and changes the console mode directly from BASIC by poking into the S.CSLMD at 8406. The routine sets a single TT: character without waiting for a LF/CR.

```
01000 REM ********************************
01010 REM * B.H. BASIC INKEY/GET ROUTINE *
01020 REM * EXIT I = NUMERIC VALUE       *
01030 REM *      I$= ASCII CHARACTER     *
01040 REM ********************************
01050 REM                                * 8406 IS HDOS 2.0 S.CSLMD
01060 POKE 8406,129                     :REM CHARACTER/WITHOUT ECHO MODE
01070 I=CIN(0):IF I<0 THEN GOTO 1070    :REM WAIT FOR CHARACTER
01080 I$=CHR$(I)                        :REM MAKE IT A STRING VARIABLE
01090 POKE 8406,0                       :REM LINE INPUT WITH ECHO
01100 RETURN
```

Dick Livingston
413 Kennerly Road
Springfield, PA 19064

Transferring Files From ZDOS To CP/M

First boot the ZDOS disk containing DEBUG and the file to be transferred. Then call upon DIR to make sure the file you want is there and make a note of the size of the file. (The number between the file extension and the date is the size in bytes.)

Then:

A:DEBUG this invokes the debug utility, which says...

DEBUG version 1.08 and uses an arrow prompt like...

>NFILENAME.EXT note no space between N and the name of your file.

>L77DF now your file is loaded in RAM with an offset of 77DF.

Next, hit control and reset together to reset the computer and boot up with the CP/M disk to which you want to transfer the file. This disk should have DDT on it.

A:DDT puts you in DDT, whose prompt is a -.

-M1500,(1500+filesize in bytes),100 this moves the file from 1500H where we loaded it to 0100H where SAVE expects to find it. (I know this sounds wrong but if we load it under ZDOS with an offset of 77DF, CP/M will find it at 1500. Trust me.)

-G0 (Gzero) now we are back out of DDT.

A:SAVE (filesize in pages) FILENAME.EXT
divide the filesize in bytes of 256. If there is a remainder, add one to the result. This is the number of 256 byte pages you need to store.

The file is now on your CP/M disk under the name you specified.

This is a cumbersome process whose only virtue is that it works. When one of you gets so irritated with this complicated operation that you write a nice program to read ZDOS files directly from CP/M, I would appreciate a copy.

Lynwood H. Wilson
Jamestown Star Route
Boulder, CO 80302

Patches to Heath's ASM.ABS From HDOS 2.0

Dear HUG,

The following patches are for publication:

"These software patches are copyrighted 1983 by UltiMeth Corporation. Permission is granted to reproduce them without charge, provided this notice is given with each copy."

The following patches to Heath's ASM.ABS from HDOS 2.0 allow the assembly language programmer to reference the current date during the assembly process. This allows the programmer to automatically create in his or her program an ASCII string which contains the date of assembly. The patches also allow the time of day to be obtained in a similar manner, if Steven Robbins' time patch (or a compatible modification) has been applied to HDOS 2.0.

```
File    Old     New     -- Program --------
offset  value   value   loc     instruction

0004    87      FE        (--- low-order byte of program length)

0006    6C      60        (--- low-order byte of entry point loc)

0AE2    2A      00      2D5A    NOP
0AE3    B2      11      2D5B    LXI     DE,427EH
0AE4    3A      7E
0AE5    EB      42

18E8    20      2A      3960    LHLD    20CAH   ;Get time of day
18E9    28      CA
18EA    43      20
18EB    29      22      3963    SHLD    427CH   ;Place in symtab
18EC    20      7C
18ED    48      42
18EE    45      2A      3966    LHLD    20C8H   ;Get date
18EF    41      C8
18F0    54      20
18F1    48      22      3969    SHLD    4273H   ;Place in symtab
18F2    20      73
18F3    43      42

1FF4    72      2E      426C    DB      '.DATE','.'+80H,2
1FF5    69      44
1FF6    67      41
1FF7    68      54
1FF8    74      45
1FF9    20      AE
1FFA    28      02
1FFB    43      00      4273    DW      0
1FFC    29      00
1FFD    20      2E      4275    DB      '.TIME','.'+80H,2
1FFE    48      54
```

```
1FFF      45        49
2000      FE        4D
2001      01        45
2002      50        AE
2003      48        02
2004      54        00        427C      DW        0
2005      2F        00
```

The following program uses the above feature to create an ASCII string during assembly which contains the date and time that the program was assembled.

```
*         "This software is copyrighted 1983 by Ultimeth Corpor-
*         ation.  Permission is granted to reproduce it without
*         charge, provided this notice is given with each copy."

YEAR      EQU       .DATE./200H+70D
MONTH     EQU       .DATE.*80H/1000H
DAY       EQU       .DATE.*800H/800H

HOUR12    EQU       .TIME.*4/400H*4097/16*2561/256
MINUTE    EQU       .TIME.*100H/100H*4097/16*2561/256
AM.PM     EQU       .TIME.*2/8000H*15+'A'

TIME24    EQU       .TIME.*4/4800H*0C000H+.TIME./4000H-2*1200H+.TIME.*4/4

          DB        'This program was assembled on '
          DB        MONTH/10+'0',MONTH/10*0FFF6H+MONTH+'0','/'
          DB        DAY/10+'0',DAY/10*0FFF6H+DAY+'0','.19'
          DB        YEAR/10+'0',YEAR/10*0FFF6H+YEAR+'0',', at '
          DB        HOUR12/10+'0',HOUR12/10*0FFF6H+HOUR12+'0',':'
          DB        MINUTE/10+'0',MINUTE/10*0FFF6H+MINUTE+'0',' ',AM.PM,'M'

          END       0
```

This type of feature is quite common on assemblers on larger computer systems. It may be used for a variety of purposes, but the most common use is in a sign-on message which contains the date (and sometimes the time) that the program was last modified. Such a sign-on message may be used by both the program creator and the program users to ascertain which version of a program is being used.

Dean K. Gibson
UltiMeth Corporation
24025 Fernlake Drive
Harbor City, CA 90710

Improvements to the Program "Draw and Print a Picture in MBASIC"

Dear HUG,

I would like to submit the following improvements to Ron White's program titled "Draw and Print a Picture in MBASIC" that was published in REMark Issue 42, July, 1983.

```
4   PRINT CHR$(27)+"E"
35  P$=E$+"p"              'INVERSE VIDEO
36  Q$=E$+"q"              'NORMAL VIDEO
55  Y$=E$+"Y"              'DIRECT CURSOR ADDRESSING
65  DEF FN C$(X,Y)=Y$+CHR$(X+31)+CHR$(Y+31)     'CURSOR POSITION
131 GOSUB 132:GOSUB 134:FOR X=0 TO 500:NEXT X:GOSUB 132:GOTO 135
132 PRINT E$+"x1";: PRINT FN C$(25,1);"NORMAL  ";:FOR Z=0 TO 32:
    PRINT XG$;CHR$(94+Z);G$;CHR$(94+Z);:NEXT:PRINT E$+"k";:
    S=0:RETURN
134 PRINT E$+"x1";: PRINT FN C$(25,1);"INVERSE ";:FOR Z=0 TO 32:
    PRINT XG$Q$;CHR$(94+Z);G$P$;CHR$(94+Z);:NEXT:
    PRINT Q$;E$+"k";:S=1:RETURN
180 I$=INPUT$(1):Q1$=" ":Q1$=INKEY$:IF Q1$="Q" THEN PRINT E$+"j";:
    GOSUB 134 ELSE IF Q1$="R" THEN PRINT E$+"j";:GOSUB 132
340 IF S=0 THEN PRINT GR$; ELSE GR$=P$+GR$+Q$: PRINT GR$;
380 CLOSE: PRINT XG$: PRINT E$+"z":END'
380 '
```

With these improvements, now the 25th line will display either "NORMAL" or "INVERSE" graphics, depending on which one you would like to use. It can be switched using the GRAY FUNCTION key for "NORMAL" and the RED FUNCTION key for "INVERSE" graphics. This is accomplished in lines 132, 134, and 180. Line 340 is used for storing of the "NORMAL" or "INVERSE" graphics or characters.

John G. Hoover
19 So. Stockdale St.
DuBois, PA 15801

Members Needed For New HUG Club in Midland, Texas Area

Dear HUG,

I am interested in finding other Heath/Zenith users in the area who might be interested in forming a Users' Group or trading ideas. My phone number is (915) 689-7964.

Mark W. Byrd
5216 Tremont #510
Midland, TX 79707

More on BDOS Error

Dear HUG,

I greatly enjoyed Bill Simpson's article in the July issue regarding recovery from a BDOS error on a save command through not having performed a warm boot on a new disk under CP/M.

Having had the same problem, and recovering from it in much the same manner, I later mentioned this to John Secor, the computer guru at the Anaheim Heathkit store. He then gave me a much simpler solution to the problem, which is also presented in Pat Swayne's column in REMark of June 1983 (page 43).

After the BDOS error, type any character to return to CP/M. Then type SAVE 0 G.COM (or any filename), and RETURN. Then type G (or the filename you chose), and RETURN.

This will load the zero length program at 100H and run it. Which puts you right back where you started, with one exception. The bomb back to CP/M has performed a warm boot, so now you can go ahead and save whatever you were trying to save.

Hopefully, this solution will work with almost any program, it has only been tried by me with MBASIC and PIE. With PIE, it will not work, as PIE comes back up, but then prompts you for a filename. When you type it, it loads the file from disk, so you have lost what was in memory.

If someone has found a way around this problem with PIE, I would appreciate the solution, since I anticipate having that particular problem. Such is Murphy's Law.

Derek Picken
11521 West St.
Garden Grove, CA 92640

Info on the DumBurner II

Dear HUG,

First, I would like to thank all the HUG Staff for the fine magazine and HUG services. I can truly say that REMark is the most sought-after mail delivery at my house!

Second, I would like to tell your readers of the pleasant dealings I had with one of REMark's advertiser.

Back around the beginning of July, I ordered the DumBurner II Bare Boards with documentation and software from Ross Custom Electronics. The delivery time was a little over a week and what I received in the mail was a wealth of documentation and software in addition to excellent quality circuit boards. The way things are today, it is refreshing to deal with a firm who provides fast service and quality goods. I was especially impressed with the software. The source code was included which was heavily annotated. This makes it easy to both understand its operation and find the correct areas for possible personalization. Also, there was plenty of information in the parts list and construction manual to satisfy the most demanding home builders who decided to build the DumBurner from scratch.

Don't let the price of the DumBurner or DumBurner II fool you! These products are professional quality EPROM/EEROM programmers which are provided with very powerful software to team-up with your H/Z-89 or H/Z-100 computers with CP/M or HDOS formats available (H/Z-100 format is CP/M only).

If you need an EPROM programmer and you have a computer, I would highly recommend the DumBurner line of products from Ross Custom Electronics.

Richard H. Swenton
19 Allen Street
Bristol, CT 06010

Dear HUG,

I'll bet most HUG members are kit builders that are comfortable working in electronics and their funds come out of the family budget. Instead of letting ABC, CBS, and NBC insult our intelligence, we pound on computers, both software and hardware.

Pat Swayne's column is my favorite. His HDOS assembly language articles are great. I just ordered CP/M. I'm confident Pat's articles on CP/M are of equal quality. Alot of the time I read between the lines. I am new at assembly language, but I picked out a ROM subroutine called $TYPTX. For a one time print, it's better than SCALL .PRINT. Also,

```
SCALL  .SCIN    is better than    LABEL  SCALL  .SCIN
JC     *-2                               JC     LABEL
```

because no label is needed.

I'm looking forward to David Warnick of Applied Computing's new column. It sounds promising for people that like to do more than run canned programs on a computer.

My favorite column in BYTE is Steve Circia's Circuit Cellar. I would like to see a hardware column in REMark. You don't have to build and test the projects. It would be nice if someone could, work load permitting, check it out on paper to say that it should not smoke your system. Limit it to projects that will fit on a 6 x 6, or whatever, PC board.

I suppose that Heath could, and probably does, have a lot of say about the format of REMark. It's nice to see they are not afraid of, but welcome, the companies that make enhancements for their computers. The competitors do more help than harm. Those who want all the cake usually end up with none.

BYTE has its BOMB, a $100 award and recognition of the person's achievements printed in their magazine. This BOMB goes to the person the readers select as having the best article in the last issue. Applause makes an entertainer feel good and appreciated. It's also good feedback. It tells the person, and his manager, what the people like. We could call our award winner HUG'S HERO.

Last, how about a questionaire. Each new member gets one as part of his membership package. It could ask his expectations of HUG and REMark, his interests, and his system. Old members could be asked about our interests, our system, things we like and dislike about HUG and REMark, changes needed, and new columns. This could be done annually or semi-annually and sent out in an issue of REMark.

Mike Bronosky
922 Alvin Street
Flatwoods, KY 41139

More On Morse Code Program

Dear HUG,

Thank you for your help in getting me in contact with Robert Horn regarding his Morse Code program in the June REMark.

Bob sent me a nice letter explaining that the problem with the program was in the CP/M MBASIC version 5.21, which I have. He had written the program for MBASIC 4.82 or earlier. With that information, my son Tod was able to figure out what to do to correct the problem. The following shows the corrections that have to be made to overcome the loop cancellation.

Change Line 380 to read:
```
380 L$(K)=" THEN GOTO 465
```

Change Line 410 to read:
```
410 FOR U=1 TO LEN(M$(D)):R$=MID$(M$(D),U,1):
IF R$=CHR$(32) THEN FOR A=1 TO N4:NEXT A:GOTO 445
```

Add Lines 445 and 465 as follows:
```
445 NEXT U
465 NEXT K
```

The remainder of the program is as published.

The program is running fine and I am enjoying relearning the morse code again.

Henry Milam, Ph.D.
4026 Kingman Blvd.
Des Moines, IA 50311

Looking To Form HUG Club In Des Plaines, Ill. Area

Dear HUG,

I am looking for other Heath/Zenith users in the Des Plaines, Ill. area who would like to start a Heath Users' Group. Anyone interested, give me a call at 699-6944 or write to me at the address below.

Michael Cohen
10029 Linda Ln.
Des Plaines, IL 60016

ARE YOU MOVING?

Don't leave your REMark behind.

Send your change of address in now to:

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

SPOOLDISK™89

FOR THE H89/Z89

TAKE THE WAIT OUT OF COMPUTING!

Is your floppy slowing you down? Does your computer go 'out to lunch' when you use your printer? Well, take a look at our Spooldisk Emulator and Printer Spooler card. Not just memory but a tiny microcomputer in its own right, the Spooldisk emulates a 128K byte disk plus it allows any files stored on itself to be output to a printer without intervention of the host H89/Z89. Don't let your disk or printer slow down your computing, add a Spooldisk to your system today. For more information on the Spooldisk or our other H89/Z89 products, write or call.

FBE
FBE RESEARCH COMPANY, INC
P.O. BOX 68234, SEATTLE, WA 98168
206-246-9815 (EVENINGS)

FLOPPY DISK
CONTROLLER

Controls Any Combination Of Up To Four
8" and 5¼" Drives

This easy to install plug in board can control any combination of single or double sided, single or double density drives.

Designed especially for H88/H89 users.

- Fully compatible Bios supplied for your CP/M 2.2 operating system
- Easy to follow instructions
- Contains controller board with boot prom
- Order cables for connection $15 (HFDC-110)
- ***Introductory Offer $395,*** *Order HFDC-100*

NORTH COAST INTELLIGENCE INC.
1201 Cherokee Trail
Willoughby, Ohio 44094
Phone: 216-946-7756

Check, COD, VISA or MC — 90 Day Warranty

16 BIT SOFTWARE FOR YOUR H/Z 100 SYSTEM

CHEM101 CHEM101 accepts simple molecular formulas up to 18 elements long then computes the molecular wt., % of composition and atomic wt. and number. A touch of a key will send a copy to the printer. Another touch will make a color 3-D graph of % of composition. REQ: ZBASIC, 128K.

OTHELLO/WARI Two games for the price of one. Both games use hi-resolution color graphics and touch-key (the return key is not needed) response to bring a fresh and lively challenge to these games. (Impressive to watch also). REQ: ZBASIC, COLOR, 128K.

SCREEN POTPOURRI Grap hold of some added power for your H/Z 100. This package includes 5 font files to dress up your display as well as a program to control your terminal functions from the ZDOS command line. Clear your screen, set your fore/back ground and many more with simple Basic-like commands. REQ: ZDOS, 128K.

Z-ZAP (VERSION 2.0) A powerful utility to access a disk by both absolute sector and direct file. Display/edit in hex, oct, dec, ASCII formats. Edit/swap/copy/fill data in a flexible buffer work area. A buffer window displays all four formats at once while a larger page display shows 256 bytes at a time. Select any displayed page to dump to a printer with a touch of a key. REQ: ZDOS, 128K.

Z-ZAP	$29.95
CHEM101	$16.95
OTHELLO/WARI . .	$16.95
SCRN POTPOURRI .	$16.95

*Shipping/Handling: $2.00.
*Ca. Res. please add your appropriate sales tax!
*All orders sent UPS.

*All check payments must be in U.S. funds drawn on a U.S. bank.

*All orders are filled on a SSDD 5¼" disk with proper documentation (unless otherwise noted).

Westcomp
POST OFFICE BOX 494
TEMPLE CITY, CALIFORNIA 91780

A Plug-In 4 MHz Modification For The H89A

Gary Wintergerst
1459 Laramie Avenue
Redlands, CA 92373

Clock Circuit

In REMark #38, p21, Pat Swayne asked for a clever way of installing the 4 MHz modification (REMark #34) on an H89A. The method I will describe in this article has two attractive features: (1) it requires no permanent modification of any standard H89A parts, and (2) it seems to work fine. The essence of the method is depicted in Figure 1 which shows a plug-in piggy-back board (similar to the one Pat described for the H89) which plugs into the U502 socket and also covers U506 and U507.

Building the plug-in module requires a couple of tricks which are best described with the aid of the sketches in Figures 2 and 3. Figure 2 shows the circuit pattern in a typical etched format. While the circuit could be etched if extreme neatness or great quantities were required, I wired mine point-to-point on a piece of perfboard using a combination of insulated and non-insulated wire. This is a backside view so that the circuit positions and pin locations are reversed with respect to Figure 1. Also, while the proportions are generally correct, the drawing should not be scaled directly, especially for the mounting hole near the bottom edge. I also added a .01 mfd decoupling capacitor which Pat did not show—probably not necessary but I like to put them in every few IC's. Otherwise the circuit is electrically equivalent to Pat's.

Sockets for the U1-U3 aren't really needed but they are cheaper than the IC's so I used them—they also make it a whole lot easier to change a bad chip. Since U502 is moved from the main CPU board to the piggy-back, its socket requires some special attention. Pins 8 and 14 of U502 must be connected together (to preserve the 2 ms clock) but must be isolated from their original connections on the CPU board. I did that by carefully bending those socket pins out to the side where a jumper could be soldered between them; this jumper then passes through the board for connection to U1 pin 13. The

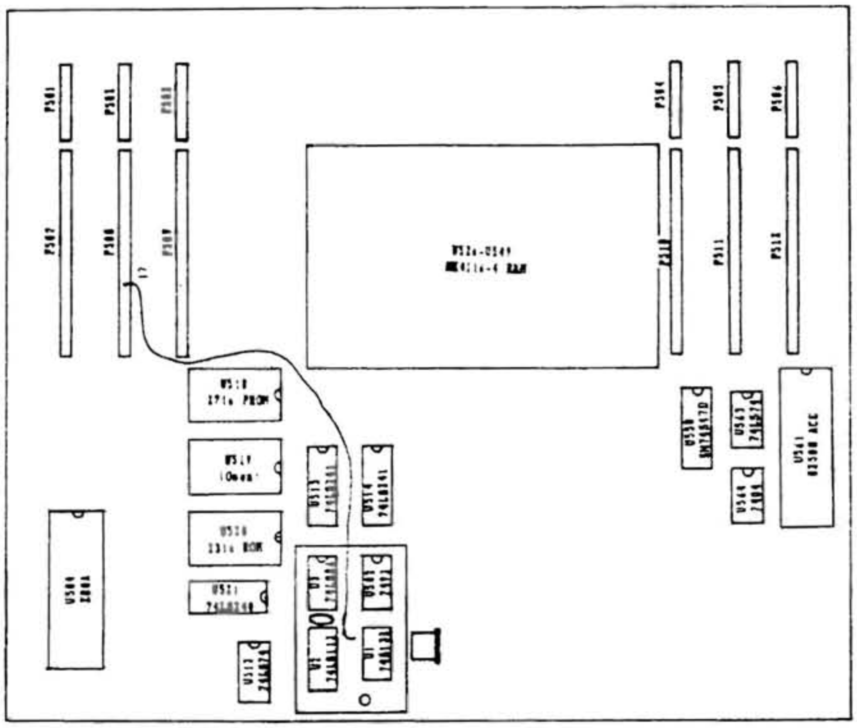


Figure 1 - H89A CPU Board With 4MHz Plug-In

piggy-back board must set high enough off of the CPU board to clear the IC's below it but not so high that the modified board hits the video circuit boards when installed in the chassis. I soldered extension pins on the socket pins as shown (make sure the pins will fit the CPU board socket), then made a spacer-brace for the pins out of another socket from which the pins had been pulled out. In fact, I used two low profile sockets but that's really a bit too much; a slightly smaller spacer would be better. The pins can be glued in the spacer to brace them but be sure to keep the contact ends clean. Also note that the new pin 8 does not go through the board but connects to U1 pin 8 only, the new clock signal. Some of the pins on U502 are unused and need not be extended—the fewer pins to plug in, the easier.

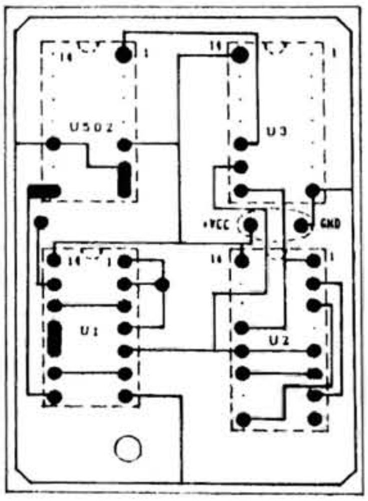


Figure 2 - Underside of Plug-In Board

There is a wire which connects to pins U1-1, 2, and 4 and then goes to P507, P508, or P509 pin 17. This replaces the connection to U552-6 since they are a common connection and is perhaps a little neater since I had both P507 and P508 available. I found a jumper plug in the parts box which I used to plug onto the pin.

The plug-in can be mechanically secured by replacing the short 4- 40 screw below U502 with a longer one using an appropriate spacer. One other thing I did was to tape the bottom of the plug-in module to reduce the possibility of a short to the components below.

Now, about U563 and U564 (REMark #38, p21). A similar non-destructive mod could be made for the designated change using another socket and an IC header. I won't describe this in detail since I haven't done it. The reason I haven't done it is that in my ignorance I installed the modification without it and I believe it is working fine! Maybe Pat or someone can explain to us why my change is or is not working fine.

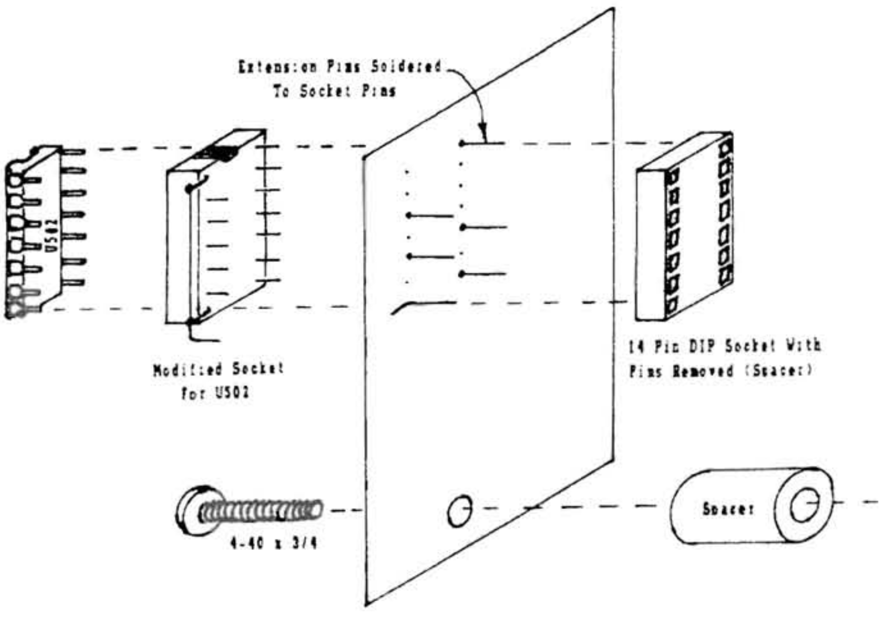

Extension Pins Soldered To Socket Pins
U502
Modified Socket for U502
14 Pin DIP Socket With Pins Removed (Spacer)
4-40 x 3/4
Spacer

Figure 3 - Module Assembly

Power Supply

I will describe this minor modification (call it a fix or enhancement) because of problems which I have encountered and which might be associated with the increased power requirements of 4 MHz operation. It appears that the current through the rectifier bridge BR-1 is sometimes too great for the connectors in P101 and P103, resulting in heating of the contacts, softening of the plastic insulators, and finally loosening of the connection. Unsure of the rating on BR-1, I first replaced it with one of those big 25 amp block jobs you can get at neighborhood electronics stores. I also tapped the AC lugs for some small screws and removed the AC lines from P101-4,-5 and ran those yellow wires to lugs screwed directly to the bridge. I could (and probably should) have tapped the DC lugs as well and then soldered the P103-2 and -4 filter wires directly to the power supply circuit board (the P103-1 and -3 wires were removed). As it is, I only recently found a problem with the DC lines and fashioned contacts out of some small brass tubing, which hopefully are of higher capacity, going to P103. Thus I eliminated the original plastic plug and contacts.

Software

One last item which may be of interest is my clock set routine. Some of you may be reluctant to modify your operating system or simply haven't gotten around to trying it (I have tested and implemented Pat's suggested BIOS modifications for Z-37 soft sectored disks, and they work fine). My simple solution is a short 8080 routine which prompts

```
                    Routine to Set System Clock to 2 or 4 Mhz
0100                      ORG    0100H
0100 114801     START     LXI    D.PROMPT
0103 0E09                 MVI    C.09H
0105 CD0500               CALL   0005H              .Display Prompt
0108 0E01                 MVI    C.01H
010A CD0500               CALL   0005H              .Read Input Char
010D FE32                 CPI    32H
010F CA2201               JZ     TWOMHZ             .Select Two Mhz
0112 FE34                 CPI    34H
0114 CA3501               JZ     FOURMZ             .Select Four Mhz
0117 0E09                 MVI    C.09H
0119 11A101               LXI    D.ERRMSG
011C CD0500               CALL   0005H              .Send Error Message
011F C30001               JMP    START
0122 3A0D00     TWOMHZ    LDA    000DH              .Get Control Byte
0125 E6FB                 ANI    0FBH               .Reset Bit 2
0127 320D00               STA    000DH              .Set Clock Rate
012A 0E09                 MVI    C.09H
012C 117101               LXI    D.TWOMSG
012F CD0500               CALL   0005H              .Display 2 Mhz Message
0132 C34501               JMP    EXIT
0135 3A0D00     FOURMZ    LDA    000DH              .Get Control Byte
0138 F604                 ORI    04H                .Set Bit 2
013A 320D00               STA    000DH              .Set Clock Rate
013D 0E09                 MVI    C.09H
013F 118901               LXI    D.FORMSG
0142 CD0500               CALL   0005H              .Display 4 Mhz Message
0145 C30000     EXIT      JMP    0000H              .Exit
0148 0D0A537973PROMPT     DB     0DH.0AH.'System Clock Rate Desired (Mhz
                                   2/4)? $'
0171 0D0A32204DTWOMSG     DB     0DH.0AH.'2 Mhz Clock Rate Set $'
0189 0D0A34204DFORMSG     DB     0DH.0AH.'4 Mhz Clock Rate Set $'
01A1 0D0A556E72ERRMSG     DB     0DH.0AH.'Unrecognized Input$'
01B6                      END


 D0100.01BF
0100 11 48 01 0E 09 CD 05 00 0E 01 CD 05 00 FE 32 CA  H          2
0110 22 01 FE 34 CA 35 01 0E 09 11 A1 01 CD 05 00 C3 "  4 5
0120 00 01 3A 0D 00 E6 FB 32 0D 00 0E 09 11 71 01 CD         2     q
0130 05 00 C3 45 01 3A 0D 00 F6 04 32 0D 00 0E 09 11     E       2
0140 89 01 CD 05 00 C3 00 00 0D 0A 53 79 73 74 65 6D              System
0150 20 43 6C 6F 63 6B 20 52 61 74 65 20 44 65 73 69  Clock Rate Desi
0160 72 65 64 20 28 4D 68 7A 20 20 32 2F 34 29 3F 20 red (Mhz  2/4)?
0170 24 0D 0A 32 20 4D 68 7A 20 43 6C 6F 63 6B 20 52 $  2 Mhz Clock R
0180 61 74 65 20 53 65 74 2E 24 0D 0A 34 20 4D 68 7A ate Set $  4 Mhz
0190 20 43 6C 6F 63 6B 20 52 61 74 65 20 53 65 74 2E  Clock Rate Set
01A0 24 0D 0A 55 6E 72 65 63 6F 67 6E 69 7A 65 64 20 $  Unrecognized
01B0 49 6E 70 75 74 24 70 2B 71 2A 67 1E EB 0E 13 CD Input$p+q*g
```

Listing 1.

for 2 or 4 MHz operation, then sets the clock accordingly. Stored as a CP/M .COM file, the routine is quick, easily called, and takes little disk space. Source and object are given in Listing 1.

Finally, I would like to thank Pat Swayne for his article. I feel it's the most important article I've yet read in REMark and certainly erased any doubts I may have had about renewing my subscription.

About the Author:

Gary Wintergerst is a Sr. Engineer/Scientist with Merritt CASES Inc., Civil, Structural, and Geotechnical Consulting Engineers, in Redlands, CA. He received his BS in Civil Engineering from California Polytechnic University and an MS in Engineering from the University of California. Gary has had experience with several large computer systems (mainly IBM), but most recently has been involved in software development and hardware interfacing for micros (Apple II, Osborne) used in data acquisition. His own H89A is his favorite micro so he greatly enjoys projects like the 4 MHz upgrade which Pat inspired. Gary is married and has two children.

Run a Hands Off Demo With This Simple Program

Vincent Arias, Jr.
501 Arvida Pkwy.
Miami, FL 33156

This program will let you prepare text, graphics, even animation, depending on your imagination, and show it on the screen of your computer. For instance, you have designed a program and want to show how it works. Prepare text describing your program and run it with DEMO.BAS. Your hands off demo will show on the screen, delete upwards, delete downwards, type words in specific quadrants, and even do graphics and reverse video.

The text is first prepared with a word processor and saved by whatever filename you intend to put on line 90. For our example, lets call it TEXT.

Line 30 defines several variables. ES$ is ESC, CL$ is Clear Screen, PO$ is setting up the cursor position. X and Y are used in loops to cause delays in acquiring the next Input line from the TEXT.

Line 50 OPENs our file called TEXT.

Line 60 inputs the TEXT line by line. The contents of each line is processed by the rest of the program.

The important part of the program is between lines 80 and 190 of the program DEMO.BAS. These are called dot commands. The two letters preceded by a dot are arbitrarily chosen by the designer of the TEXT. I chose to use abbreviations of what these commands do. Example, .cl means Clear Screen, .in means Indent, etc.(see the remarks).

If the INSTRing function is satisfied by the Input from line 60, then the command is performed. Line 90, .zc, zeros the line counter. A line counter is necessary in order to advance the cursor down the screen as each line from the TEXT is input. Lines 100 and 110 are two one-shot delay constants. You can have as many as you want. These are used at the end of a paragraph to allow the reader to catch up. Lines 120 and 130, .rd and .in, define the down and in axis where you want to position the cursor for the next input line which usually would be to print text. In the text itself, you would enter this dot command with a number allowing a space between the command and the number. Example, .rd 12, and on the next line, .in 40. This would put the cursor 12 rows down and 40 columns in, or, on the middle of the screen.

Line 140 gives D a value that is used in the loop at 240 - 250. This value is usually in the hundreds. It is used to delay the printing of the next line. This delay remains constant until changed or zeroed.

Line 150 prints letters as if being typed manually. The typing speed may be varied by changing the value in the FOR...NEXT loop in line 300 from 100 to whatever you desire.

Line 160 deletes upwards. It first recovers the value given .du (never greater than 24), then branches to Line 290 where it applies the value of E to the loop I. The cursor posi-

```
10  '       DEMO.BAS    Hands off demo - 23-MAY-83
20  '       Contributed by VINCE ARIAS - Miami, Fla.
30  ES$=CHR$(27):CL$=ES$+"E":PO$=ES$+CHR$(89):X=1000:Y=2000:PRINT ES$"x5"
40  ON ERROR GOTO 310
50  OPEN"I",1,"TEXT"
60  LINE INPUT#1,Z$:IF EOF(1) THEN 280
70  '       DELAY, ERASE  AND CURSOR POSITION
80  IF INSTR(Z$,".cl") THEN PRINT CL$:K=0:GOTO 60           ' CLEAR SCREEN
90  IF INSTR(Z$,".zc") THEN K=0:GOTO 60:                    ' ZERO LINE COUNTER
100 IF INSTR(Z$,".d1") THEN FOR J=1 TO X:NEXT:GOTO 60:      ' DELAY X (1000)
110 IF INSTR(Z$,".d2") THEN FOR H=1 TO Y:NEXT:GOTO 60:      ' DELAY Y (2000)
120 IF INSTR(Z$,".rd") THEN B=VAL(MID$(Z$,5,3)):GOTO 60:    ' n ROWS DOWN
130 IF INSTR(Z$,".in") THEN C=VAL(MID$(Z$,5,3)):GOTO 60:    ' n LEFT MARGIN
140 IF INSTR(Z$,".ld") THEN D=VAL(MID$(Z$,5,3)):GOTO 60:    ' n LINE DELAY
150 IF INSTR(Z$,".tp") THEN GOSUB 300:                      ' TYPE LETTERS
160 IF INSTR(Z$,".du") THEN E=VAL(MID$(Z$,5,3)):GOTO 290: ' n DELETE UPWARDS
170 IF INSTR(Z$,".rv") THEN PRINT ES$+"p":GOTO 60
180 IF INSTR(Z$,".nv") THEN PRINT ES$+"q":GOTO 60
190 IF INSTR(Z$,".el") THEN PRINT ES$PO$CHR$(32+B)
CHR$(32+C)ES$"K":K=0:GOTO 60:                               ' ERASE TO END OF LINE
200 IF INSTR(Z$,".ep") THEN PRINT PO$CHR$(32+B)CHR$(32)
:K=0:PRINT ES$"J":GOTO 60:                                  ' ERASE TO END OF PAGE
220 PRINT PO$CHR$(32+K+B)CHR$(32+C)Z$:                      ' PRINT TEXT
230 IF EOF(1) THEN 280
240 FOR I=1 TO D:                                           ' LINE DELAY LOOP
250 NEXT
260 K=K+1:                                                  ' LINE COUNTER
270 GOTO 60
280 PRINT ES$"y5"CL$PO$"*7""WANT ANOTHER RUN? (Y/N) - ":Q$=INPUT$(1)
:IF Q$="Y" THEN CLOSE:GOTO 30:ELSE SYSTEM
290 FOR I=1 TO E:PRINT PO$CHR$(31+E-I)
:FOR L=1 TO 50:NEXT:PRINT ES$"1":NEXT:GOTO 60:              ' DELETE UPWARDS SUB
300 LINE INPUT#1,Z$:IF INSTR(Z$,".stp") OR EOF(1) THEN
 60 ELSE PRINT PO$CHR$(32+B)CHR$(32+C)Z$:C=C+1
:FOR F=1 TO 100:NEXT:GOTO 300:                              ' TYPING ROUTINE
310 IF ERR=62 AND ERL=60 THEN 280
```

tion is determined by (30+E-I), where E is the start of the row to be deleted, and I the decrement upwards of the cursor position. The L loop is merely a delay. ESCape "1" is clear line. These ESCape commands can be found in the Computer Operation Manual, or in the Key Board Studio's THING. Lines 190 and 200 use ESC "K" and "J", respectively, for erase to end of line and erase to end of page. This is accomplished by first positioning the cursor with the .rd and .in commands in the TEXT. Add as many dot commands as you want after 200, such as graphics, etc.

Line 220 prints the input TEXT. Line 260 increments K by one each time a line is input at 60. This in turn increments the downward position of the cursor on line 220. In your TEXT, once you define the cursor axis with B and C (line 120 and 130), they will remain constant until changed within the TEXT. Line 240 simply puts a line delay equivalent to the value of D which was the value attached to .ld in line 140. Example, .ld 100.

The program keeps looping to line 60 until finally the TEXT is exhausted and the EOF(1) branches to line 280. This line may be changed to produce a continuous running display by simply changing it to: 280 CLEAR:GOTO TO 30.

The following sample TEXT shows the arrangement of the dot commands and how to form a simple demo. Play with it awhile and use your imagination.

```
.zc      zero line increment count
.rd 12
.in 29
Now let us type the next line

   My name is
.rv      reverse video
.rd 14
.in 44
.tp      type letters below
J
O
H
N

P
.

D
O
E
.stp     stop typing
.nv      normal video
.d1
.d2
```

Using the 'Sector Zapper'

William H. Craig
P. O. Box 311
Grayson, KY 41143

In issue #23, Richard L. King described a simple circuit to prevent the soft sector controller board from 'seeing' the ten sector hole pulses in a hard sector disk, without affecting the index pulse.

He had breadboarded it on his H17 controller board, but did not present any way of connecting it to the WH-8-37 soft sector controller board. This simple circuit works perfectly with my WH-8-37 and can be added with NO modification to the controller board required.

The required INDEX pulses can be intercepted at IC U29B (443-791) (74LS244), pin #4. To extract the signal, construct a 'socket adapter' built of a 20 pin low-profile DIP socket soldered on top of a 20 pin low-profile DIP plug. Connect all pins of the IC socket to the corresponding pins of the plug EXCEPT pin #4. Bend the pin #4 solder tangs of both the socket and the plug outward at 90 degrees and attach the input lead to the 'Sector Zapper' to pin #4 of the DIP plug, and the output lead to pin #4 of the socket. Plus 5 volts can be obtained from pin #20 and ground from pin #10. I used an 8 inch length of 6 conductor ribbon cable (grounding every other conductor) to connect from the 'Sector Zapper' PC board to the adapter plug. Carefully remove U29 from the controller card and plug it into the socket of the adapter plug, and then plug the adapter into the U29 socket of the controller board.

The only change I made to the original circuit was to delete the .001 Mfd capacitor between T1 and T2, as there was less pulse 'jitter' with the two timers D.C. coupled.

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

CUT ALONG THIS LINE =

HUG MEMBERSHIP RENEWAL FORM

When was the last time you renewed?

Check your ID card for your expiration date.

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

Name ______________________________

Address ______________________________

City-State ______________________________

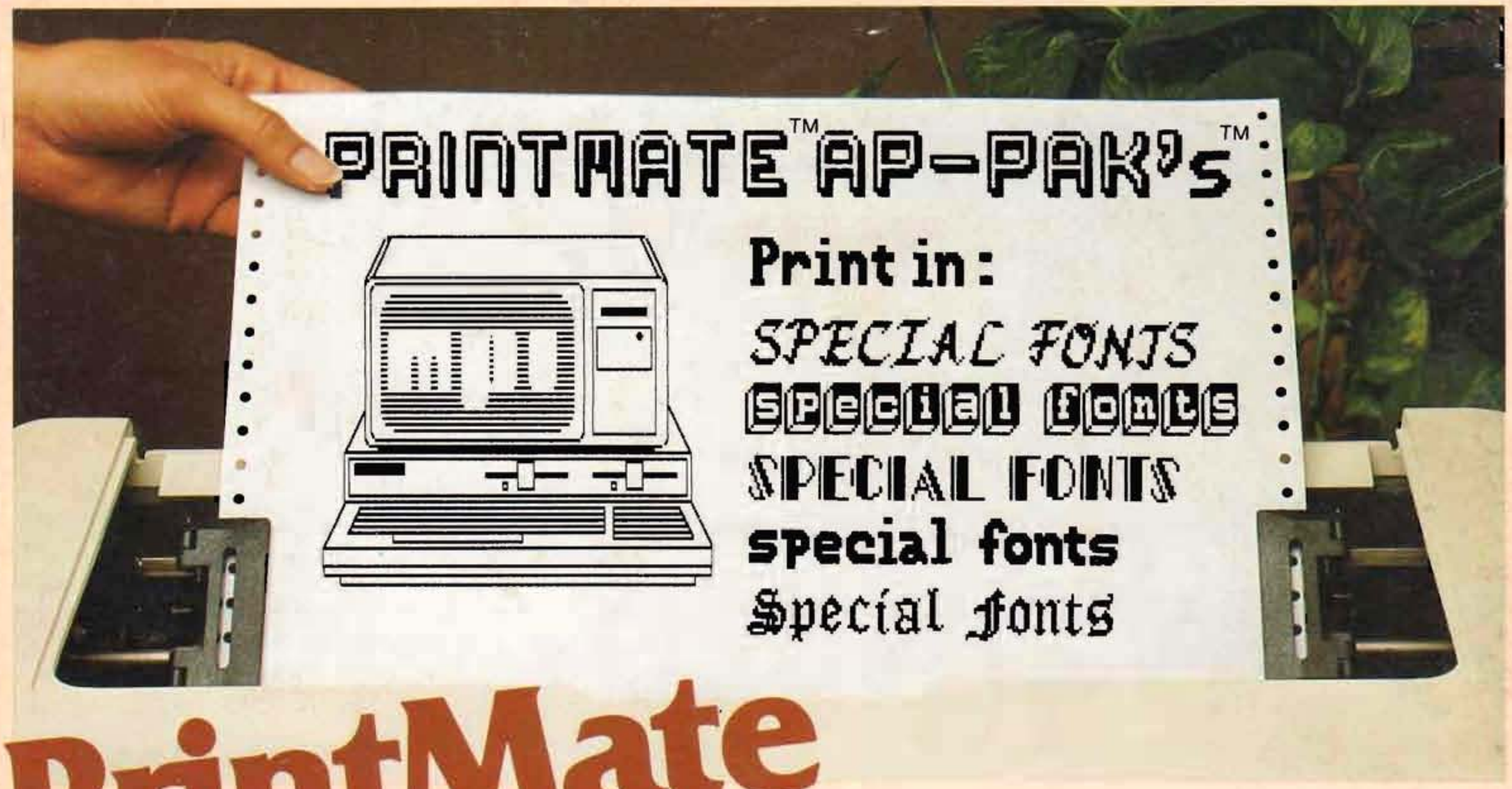
Zip ______________________________

REMEMBER - ENCLOSE CHECK OR MONEY ORDER

CHECK THE APPROPRIATE BOX AND RETURN TO HUG

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

* Membership in France and Belgium is acquired through the local distributor at the prevailing rate.



PrintMate
GOES GRAPHIC WITH H/Z100!

Picture this—your logos, graphs and fancy print fonts on paper with a PrintMate. Our exclusive AP-PAK lets you print what you picture. ▭ See and believe at your Heath and Zenith dealer today.

Micro Peripherals, Inc., 4426 South Century Drive, Salt Lake City, Utah 84107 **1-800-821-8848**

The Printer People ™ mpi

PrintMate 99

PrintMate 150

H/Z 100 is a registered trademark of the Heath Company.

Heath/Zenith Users' Group

Hilltop Road

Saint Joseph, Michigan 49085

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

Volume 4, Issue 10

POSTMASTER: If undeliverable, please do not return.

885-2045