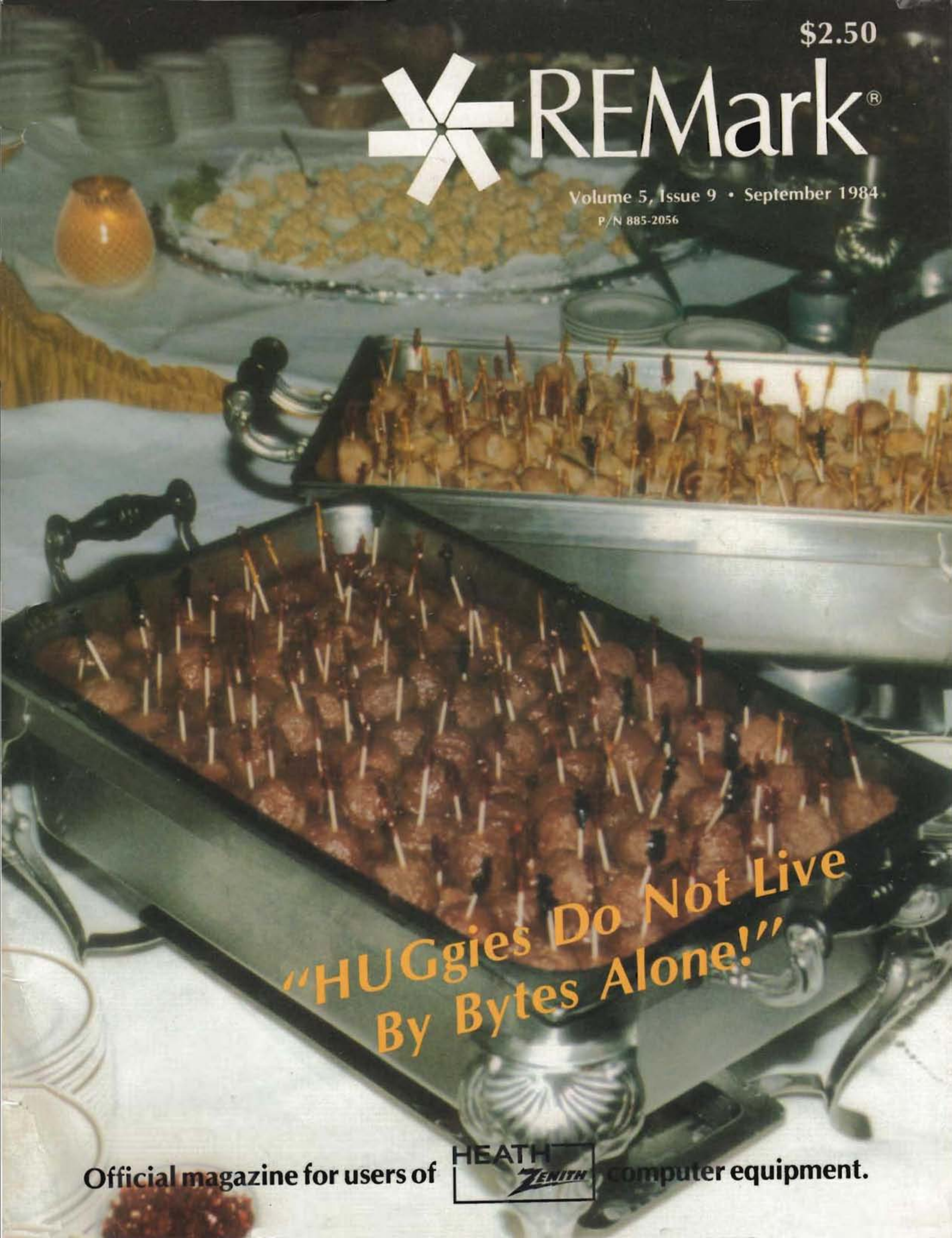
$2.50

REMark®

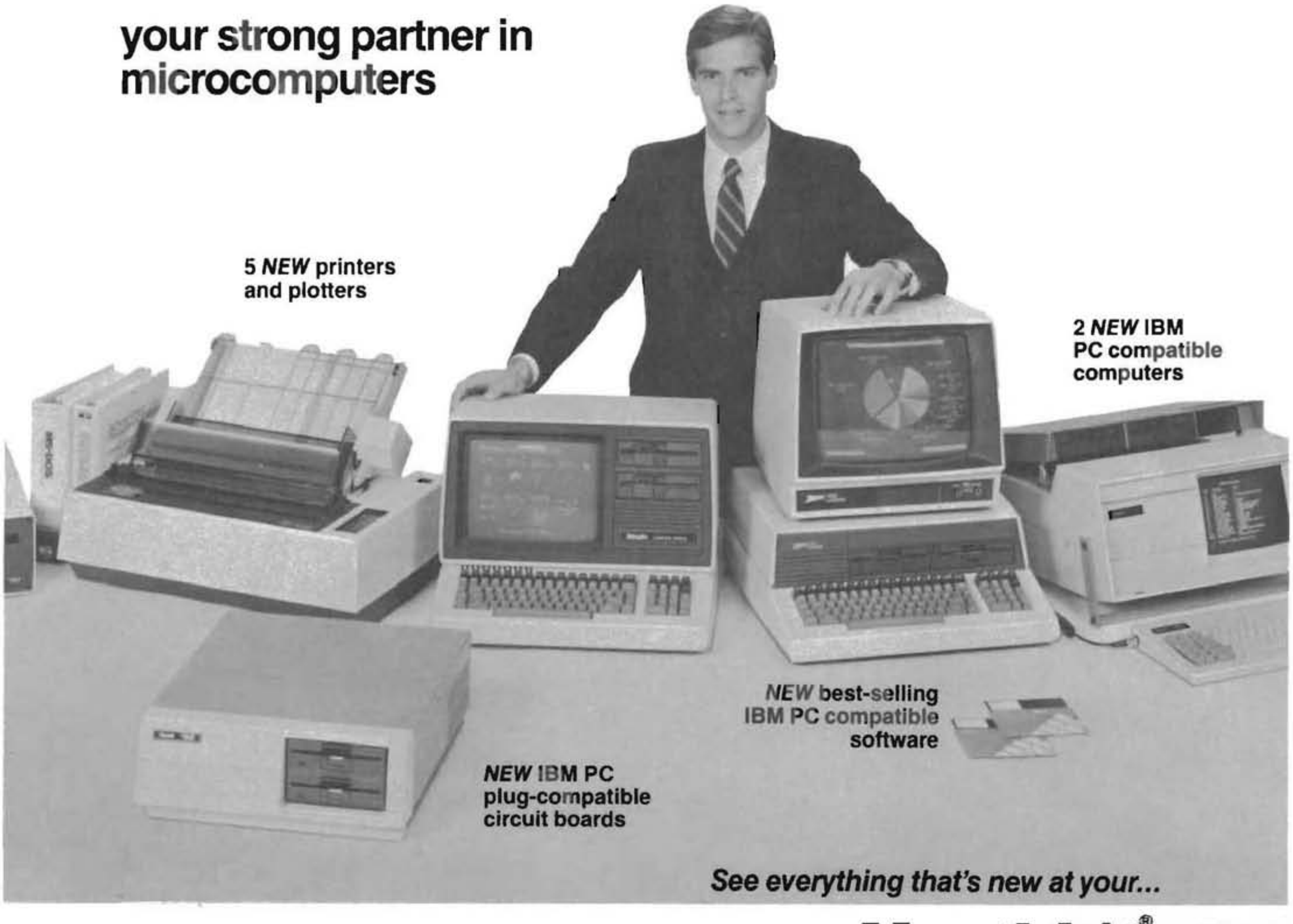
Volume 5, Issue 9 • September 1984
P/N 885-2056



Official magazine for users of HEATH ZENITH computer equipment.

The one-stop microcomputer shopping center

your strong partner in microcomputers



5 *NEW* printers and plotters
2 *NEW* IBM PC compatible computers
NEW best-selling IBM PC compatible software
NEW IBM PC plug-compatible circuit boards

See everything that's new at your...

Heathkit®
Electronic Center

Where you get more by doing

REMark®

Volume 5, Issue 9 • September 1984

Staff

Manager	Bob Ellerton (616) 982-3867
Software Engineer	Pat Swayne (616) 982-3463
Bulletin Board and Software Developer	Jim Buszkiewicz (616) 982-3463
Software Coordinator	Nancy Strunk (616) 982-3838
Secretary	Margaret Bacon (616) 982-3463

REMark

Editor	Walt Gillespie (616) 982-3789
Editorial and Advertising Assistant	Lori Latham (616) 982-3794
Graphics and Layout Assistant	Greg Martin (616) 982-3463
Printers	Imperial Printing St. Joseph, MI

	U.S. Domestic	Canada & Mexico	International
Initial	$20	$22*	$30*
Renewal	$17	$19*	$24*

*U.S. Funds.

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

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

Send Payment to: Heath/Zenith Users' Group
Hilltop Road
St. Joseph, MI 49085
(616) 982-3463

on the stack

ON THE COVER: Hot hors d'oeuvre and delectable prizes were served to HUGgies at the Saturday evening cocktail party. See page 39 for story.

MEMO:

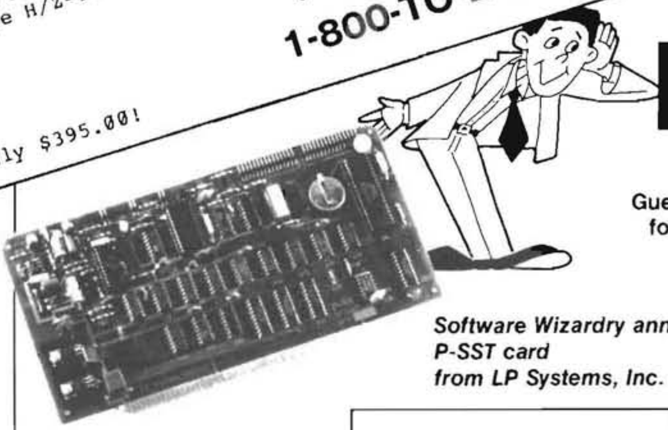
to: H/Z-100 users

```
Thank you for your overwhelming response to the P-SST card!  Our first
two editions are SOLD OUT!  However, new production is available as you
read this, so don't hesitate to order!

Don't forget that a disk-full of software comes with the P-SST,
including graphics/sound and speech demos, and three-voice musical
scores.  To encourage more software support, we are now selling the P-SST
Development Libraries, which make programming the P-SST amazingly easy!

The P-SST's functions are also creatively utilized by our applications
programs CHRONOLOGIC (CLOCK/CALENDAR), PALETTE GRAPHICS(JOYSTICK
SUPPORT), REACTOR-100 NUCLEAR REACTOR SIMULATION (SOUND/VOICE), AND
ZLYNK/II MODEM COMMUNICATIONS (CLOCK/CALENDAR).  We think you'll agree
the Programmable Sound/Speech/Time card is one of the finest support
products yet available for the H/Z-100!

Sincerely,
Software Wizardry, Inc.

P.S. The price is still only $395.00!
```

(314) 946-1968

1-800-TO-BUY-IT

PSST!



Guess what sings, hums, talks, follows your every move, and plugs into your Z-100?

Software Wizardry announces the P-SST card from LP Systems, Inc.

$395!

Phoneme Speech Synthesizer	Programmable features include a directory of 64 phonemes, with four pitch levels.
Joystick Interface	Two general purpose 8-bit parallel I/O ports. Software included to configure them for use with standard Atari joysticks.
IEEE-696 (S-100) bus	Designed for the Z-100 computer, the board complies completely with the bus standard and can be interfaced into any S-100 system.
Audio Power Amp	On-board power amplifier provides one watt of output power to drive standard 8 ohm audio speaker.
Audio preamp out	Provides low level audio signal out, suitable for connection to a standard preamplifier input.
Sound Synthesizer	Provides three channels of music/sound and one channel of white noise generation, independently programmable for simultaneous operation. Amplitude can be under direct program control or controlled by envelope generator. Control registers remain latched to most recent instruction to reduce CPU overhead. Frequency generator program listing provided, and sample sound and music programs.
Clock/calendar	On-board clock with 9-month lithium battery always knows the correct time and date. With the CHRONOLOGIC program, on boot-up ZDOS will always know the time. ALso can be programmed to provide elapsed time; Alarm interrupt at pre-set time and "heartbeat" repetitive interrupt at specified intervals to S-100 interrupt bus. "Standby" interrupt provides external TTL signal to control an external device, such as an relay to allow your Z-100 to turn itself on!

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!

ZENITH data systems

We admit it doesn't dance, but you might after you plug it in!

The Programmable-Speech/ sound/ time card is the latest result of Software Wizardry's crusade to fill the holes in your Z-100!

Not just a piece of hardware for programmers and system integrators, we're supporting it with SOFTWARE; we don't leave you out in the cold.

In addition to the standard utilities that come with the card, the clock read function is supported by our CHRONOLOGIC program; The joystick interface is used by our best selling (and still the best!) graphics drawing program PALETTE. (which fills another hole with its light-pen support), and appropriate sound effects issue from it when you use it with our REACTOR-100 nuclear reactor simulation.

And thats just the start; how about that for support! We know you all will be quick to make use of P-SST, because we love the Z-100, and we know you do too!

Try our card in your computer, and PSST!!!, pass it on!

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 greatest, shipping/handling plus sales tax if shipped to a Missouri address.

Software Wizardry

THE MAGIC TOUCH

1106 First Capitol
St. Charles, MO 63301
(314) 946-1968

BUGGIN' HUG

Sorted Directory Corrections...

Dear HUG;

In regard to Jeff Kalis' article in the April edition of REMark (ZD - A Sorted Directory, Vol. 5, Issue 4, Page 14), I could not get the program to run for either the 5-1/4" or 8" disks on my H/Z-100.

After quite a few hours of sifting through the code and looking at the directories on several disks, I found that the sizing of the pointer table in the scan subroutine and the source stack were too small. My DSDD 40TPI 5-1/4" disks have space for 112 directory entries and the DSDD 8" have room for 192.

After resizing these tables, the program runs beautifully.

Here are the patches for those who may be interested:

1) In the scan subroutine change the fourth line from-

```
        MOV   CX,120    ;Pointer table
   to read:
        MOV   CX,200    ;Pointer table.
```

2) At the end of the program, change the last line before 'CODE ENDS' to read:

```
        SRCE      EQU     PNTR+400
```

Hope these changes will be of use to other readers.

Sincerely,

Howard Rice
3360 Princeton Court
Santa Clara, CA 95051

Needed Root Canal

Dear HUG;

Over the last half dozen months, I have noticed the dramatic decrease of H/Z-89 and HDOS articles in REMark, and at the same time an increase of H/Z-100 articles. This is reminiscent of a few years ago when the H-8 suffered the same fate because of the introduction of the H-89.

It is interesting to see the amount of awesome power that IBM holds. The home computer industry prior to the active entry of IBM, had no clearly defined standards until IBM entered the market. Now any home uter manufacturer would be flirting with disaster if he did not make some attempt to produce, what might be called, an IBM compatible computer.

I am not writing this in the hopes of generating 10,000 followers to march up to the White House steps in protest, for I applaud the direction that IBM has brought to the industry.

I, for one, will miss seeing H/Z-89 and HDOS articles in REMark, for past articles have been rewarding by providing me with a vehicle for learning both hardware and software, that might not be available elsewhere.

As a final gesture, I salute REMark for their help and enlightenment. H/Z-100 users beware, you are next. RIP H-8, H/Z-89, H/Z-90 and HDOS.

I must go to the dentist for a needed root canal, perhaps that will cheer me up.

Hubert E. Roy
1 Holly Ridge Road
Severna Park, MD 21146

ED) Hubert, I beg to differ some with you. REMark is a user magazine. It's content reflects the interest of the HUG members through the manuscripts received. That is "We can't print it if nobody writes it". There has been a definite shift away from hardware modifications by the general membership and more emphasis on software. I checked back through the last 6 issues (2-7) and counted 49 articles that related to H/Z-89. Many of these also relate to the H-8. With the limits of space a major factor and approximately 50% of our membership owning H/Z-100s, we cannot print 5 versions of "How To Speed up your '89 CPU".

When HUG first started, it was only necessary to cover two computers and one operating system in REMark. Now, we have 5 computer systems, operating systems, a couple dozen languages, plus umteen various specialty software packages. To this add all the extra add-ons, such as printers, plotters, displays and boards, all handled by a magazine staff of 3 plus a back-up staff of 4....I think we do pretty good.

To paraphrase an ole saying: "You can satisfy all of the people some of the time, and you can satisfy some of the people all of the time, but you can't satisfy all of the people all of the time!"

Walt Gillespie, Editor

AM9511 Help Needed

Dear HUG;

As a professor of meteorology, my primary computing task is the analysis of large amounts of atmospheric data. When it became clear that my beloved H-89 was becoming too slow for this purpose, I purchased a Super-89 board from D-G with two ideas in mind. The increased clock speed would give me a factor of two, while the arithmetic processor (AM9511) should contribute another factor of three or four. Knowing the importance of software, I bought the new board through a company which would provide software to drive the 9511 from BASCOM; Ultimate Computer Systems of Hastings, MI.

After two trips back to the factory, the Super-89 board is working almost perfectly, but what about the software? Only about one-third of my MBASIC programs will run when compiled with the 9511 routines and some of those give strange results. In short, it seems that the UCS routines have serious bugs and - you have heard this before - the company has disappeared. D-G, which was so helpful with hardware problems, is sympathetic but that is all. This is fair, as the problem is not their fault, but what good is their expensive 9511 option without reliable software.

I would be happy to hear from HUG members about other sources of AM9511 software, or suggestions for fixing the bugs in the UCS routines.

Ronald B. Smith
Dept. of Geology and Geophysics
Yale University
New Haven, CT 06511

☞

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

PRICE BREAKTHROUGH

Super Sale on New Disk Drives

Starting at **$169.00!** **New Low Price**

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

40 or 80 Tracks — Single or Dual Head — New 3½" Drivette ™
Our Disk Drives are Capable of Single **and** Dual Density Operation
The **NEWEST** Technology Capable of Operating on Most Popular Computers
Drive a Hard Bargain!! ™ **For your TRS/80, Color Computer, IBM, Apple, Franklin, Max/80, 5 M.B.-20 M.B. Complete Systems......from $999.95**

Diskette Breakthrough — 10 Pack in Library Case — ~~$18.95~~ NEW LOW PRICE

PLEASE CALL FOR OUR MOST CURRENT PRICE REDUCTIONS.

TOLL FREE ORDERING 1-800-343-8841

GENERAL AND TECHNICAL 1-617-872-9090

Disk Drives (0123) TRS/80-IBM-Apple-TI-Franklin-Max/80-LNW.................◄
Model I/III/IV Upgrade (Disk Drives - Memory)................◄
Printers—Daisywheel/Dot Matrix................◄
Percom Double Density Controller (Model I)................◄
Color Computer Printer Interfaces................◄
Disk Drive Operating Systems................◄
Repair Services Now Offered—FAST Turn-a-Round................◄
Apple/Franklin Compatible Add-On Drives with Case & Cable................◄
Diskettes in Library Cases................◄
DISK DRIVE CASES AND POWER SUPPLIES................◄
Printer **Buffers 8K to 512K................starting at $143.95**
Holmes Model I/III Speed-up Mod................starting at $90.00
Cables—Printer/Disk Drive................starting at $16.00

CALL TOLL FREE FOR NEW PRICES

Warranty on Disk Drives — 1 Full Year Parts and Labor

SOFTWARE SUPPORT, INC.

271 Worcester Road, Framingham, MA 01701 (617) 872-9090 Telex-383425
Hours: Mon. thru Fri. 9:30 am to 5:30 (E.S.T.) Sat. 10 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.
Prices subject to change.

CANADA
MICRO R.G.S. INC.
751, CARRE VICTORIA, SUITE 403
MONTREAL, QUEBEC, CANADA, H2Y 2J3'
Regular **Tel. (514) 287-1563**
Canadian Toll Free 800-361-5155

Service! Service!
All in stock products are shipped within 24 hours of order. Repair/Warranty service is performed within 24 hours of receipt unless otherwise noted. We accept C.O.D., foreign and APO orders. School and D&B corporate P.O.s accepted.

TRS/80 Registered Trademark Tandy Corp. Apple Registered Trademark Apple Computer Corp
IBM-PC Registered IBM Corp. Franklin Registered Trademark Franklin Corp. Max/80 Registered Trademark Lobo Int.
LDOS Reg. Logical System Inc. Dosplus - Micro Systems Software Newdos/80 - Apparat Inc.

More CP/M DUMP Patches for '89

Dear HUG;

As others have said, there's hardly an issue of REMark that doesn't have something of interest in it, though with so much space given to the 16-bit machines, we '89 users get somewhat left behind. "Making the CP/M DUMP Program a Useful Utility" (Feb. 1984) was one of those things I wanted to sit right down and try. Further, the article informed me that I had the source code for the DUMP program. (I should get around to reading those manuals and directories more carefully!)

The patch works; but, as my nature is, I had to improve on it a little. In particular, with the patches given, the program fails to print the last line of ASCII -- which is not always a string of NULLs or Hex-1As. The remedy is simple. Make the PASCII sequence a callable subroutine, store it down with the ASCII subroutine, and insert "CALL PASCII" after both the JNZ NONUM instruction and the FINIS: label. Further, since 8-bit numbers are not treated as negative (except in Z80 relative jump and displacement instructions), the "ANI 80H" and "JNZ PERIOD" instructions are covered by the "CPI 'z'+1" and "JNC PERIOD" instructions a few lines below. Optionally, replacing the "CPI 'z'+1" with "CPI 7fh" includes the rest of the punctuation marks. The revised listing for ASCII is appended below.

The same article revealed that multiple ASM commands may be written on one line if separated by '!' -- Thanks, again.

Sincerely,

Lansing E. Tryon
29 Fairhaven Road
Rochester, NY 14610

```
ASCII:  PUSH H! PUSH D! PUSH PSW       ;Save environment
        LXI     H,BUF16        ;Our ASCII line buffer
        LDA     ASCCNT         ;Character Position in line
        MOV     E,A
        MVI     D,0
        DAD     D          ;Current position address in HL
        INR     A          ;Bump count
        STA     ASCCNT
        POP     PSW        ;Get current byte
        PUSH    PSW        ;Save again
        CPI     ' '        ;ASCII?
        JC      PERIOD     ;if not
        CPI     7FH        ;Top of ASCII
        JNC     PERIOD     ;if not
        CPI     '$'        ;Don't allow false PRINTF terminator
        JNZ     STORIT
PERIOD: MVI     A,'.'      ;Substitute for non-ASCII
STORIT: MOV     M,A        ;Save in Buffer
        POP PSW! POP D! POP H  ;Restore environment
```

A PASCAL Fan

Dear Walt;

A word of thanks to you and your colleagues for the fine work you are doing with REMark magazine. My scope of interest tends to be narrow, so it is not unusual for me to go through an entire computer magazine and not see anything I want to stop and read. REMark manages to keep coming up with goodies, thoug, and I usually find things to read in every issue.

I entered Pat Swayne's SCRNCLK and CLOCK assembly language programs (in the May issue) into my Z100 and found it to be great. In an effort to wean myself off BASIC, I did not get BASIC with my Z100. The only language I have provided so far is the Microsoft PASCAL compiler. Articles on PASCAL are less than plentiful, however, so learning some of the language's nuances is coming slowly.

For obvious reasons, then, I really found Karl Remmler's PASCAL program in the June issue most useful. I've been picking and learning goodies from that program for weeks. There's one thing I've been unable to track down anywhere, though, and I wonder if you can help me find it. How do you (from MS-PASCAL) call (run) a new program from the one you're in? I'd like to write a menu program for my PASCAL exe-files but I can't figure out how to do it. I asked the 'brains' at the local HUG unit but couldn't get an answer. Can you help?

Arnold R. Madeira
22 Locust Glen Drive
Cranston, RI 02920

Installing Your Business Computer

Dear HUG;

I enjoyed D. C. Shoemaker's article "Installing Your Business Computer". I must admit that when I first looked at it, I thought, "What's the big deal; you just find a spot to put it and plug it in!" Well, we can admit when we're wrong!

For those that are interested in setting up a really human- engineered office, I would recommend a book by Wilbert O. Galitz, titled "Human Factors in Office Automation". It was published in 1980 by:

Life Office Management Association
100 Colony Square
Atlanta, GA 30361

I have seen the book in the larger book stores, but I imagine it can be ordered from the publisher. The book spends a lot of time on the office in general, including space planning, lighting, noise and work station design.

Probably of more interest to the Heath/Zenith community, in general, would be the section on software system design. Here, the author covers such topics as screen formatting, the proper use of color and graphics, prompting, error control and error messages, and response time. The intent is to design a system that is not distracting and relatively pleasant to use for several hours at a time. It seems that many systems are designed more to show off the capabilities of the hardware rather than for ease of use. The book shows how to avoid this trap.

Thanks for another good issue!

Sincerely,

David A. Shaw
469 N. Howard
Elmhurst, IL 60126

The MPI 99G and Peachtext (Magic Wand)

Dear HUG;

Reference: March '84 REMark, Letters, Page 64, More On Peachtext.

I, too, have discovered a serious deficiency with Peachtext, and the compatibility, or lack of compatibility with the MPI 99 printer, which I had read the referenced article before I spent my money.

I was also disappointed to learn that the highly touted AP-PAK for the

MPI-99 would only run under CP/M.

It's great to have a Z-100, a nice printer and some really neat software, but it is sure frustrating to not get them to work together.

Has anyone figured out how to make Peachtext run properly with the MPI-99, that is, the specialty functions? If so, I would certainly appreciate a copy of your print driver for Z-DOS or CP/M.

Very truly yours,

Arthur I. Kriss
3029 Carnelian Street
Las Vegas, NV 89121

Dear Mr. Walberg;

I have an MPI 99G printer and Magic Wand and have had problems similar to those you describe. It was equally frustrating, until I found out that the Magic Wand command function 'OUT' can be used to send the required information to the printer. Thus to get underlining, I embed the command

(command slash)OUT 14(command slash)

and to stop underlining

(command slash)OUT 15(command slash)

Commands that require the escape character as a prefix are sent as OUT 27,14 for example, which would start elongated printing. I can now use the full capabilities of my printer.

One word of warning, when using elongated type, spaces are elongated as well, and Magic Wand does not count the double spaces used by the elongated form. As a result, the centering function does not work properly, and it is necessary to count the number of characters to a line or you can wind up with some wild results.

Similarly, if you change the pitch appreciably, remember to change the margin designations with an embedded command or things will go lopsided.

With a little practice however, you can get quite proficient and get the results you want quite easily.

Lawrence J. Durney
12 Sunset Drive
North Caldwell, NJ 07006

Dear HUG;

I noticed a letter in the "Buggin' HUG" column of the July issue of REMark concerning the problems with using Magic Wand and certain dot matrix printers. I am familiar with this problem, since I just purchased a TI-850 printer (similar in operation to the Epsons) and also have the Magic Wand word processing program.

After some playing around with the problem, I found a rather simple solution for printers, such as the Epson, which require certain escape codes to vary the type style, which requires no modification to the CP/M based Magic Wand.

1. Enter the "CHANGE" program of Magic Wand and select printer option #1. After the disc has been reconfigured by the CHANGE program, the file (CHANGE) can be deleted if desired.

2. After the desired text has been composed in the "EDIT" mode, ESCape to the COMMAND mode. Save the completed text to disc. Exit to the CP/M operating system by typing "END," followed by a carriage return.

3. After the A> prompt appears, hold down the CONTROL key and press an upper case "p", followed by a carriage return. This gives a screen dump to the printer. You may now transmit any sequence of escape codes to the printer for the type style desired. (ED: Be sure to use another "CONTROL P" to cancel the screen dump once your codes have been sent.)

4. Re-enter Magic Wand from CP/M by typing "EDIT filename" and print your document from the command mode.

Very truly yours,

Barry Atwood
209 Etta Ave.
Harrison, OH 45030

Interlace Anyone?

Dear Walt;

How about an article in REMark about the use of the interlace mode on the H/Z-100? Ie: 1) How to put the computer in the interlace mode, and 2) How to use this increased vertical resolution in graphics.

Any information you could send me on the above would be appreciated.

Sincerely,

Jerry A. Phelps
6013 Innes Trace Rd.
Louisville, KY 40222

ED:) Well Jerry, I'll put your question out there and let's see what comes back. As for me, I have trouble with the interlacing on my shoes.

A Personal View on Floppy Disks

Dear HUG;

Two particular aspects of microcomputing seem to create more controversy and confusion than warranted. The first is electromagnetic interference (EMI) produced by the various circuits within the computer, and the second is how best to protect diskettes when traveling. How serious a problem EMI is for the individual user depends largely on his or her system and where it's used. A 1977 vintage TRS-80 produces copious amounts of electromagnetic radiation, enough to drown out a nearby radio or television. Later microcomputers are better designed and built, and rarely cause EMI problems to radios and televisions two or three feet away from the computer.

The problem of electromagnetic interference has been well covered in numerous publications. Often, however, an excellent reference source is omitted. One of the most thorough practical discussions on the control of EMI is in the booklet, *How to Identify & Resolve Radio-TV Interference Problems*, from the Federal Communications Commission, available from the Government Printing Office for about $2.50. Most large cities have a branch; look in the Yellow Pages under US Government. This booklet covers in great detail how to determine what's causing the problem, what you can do about it, and if all else fails, where to turn for additional help.

The second problem, protecting diskettes while traveling, hasn't had so thorough a treatment. Most of what I've read has consisted of horror stories, theories and guesswork about the effect of X-rays on magnetic media. First, most of us will recall from our high-school physics that X-rays are high-energy electromagnetic radiation of much shorter wavelength than visible light. And it's certainly true

Vectored to 76 ☞

"My Favorite Subroutines"

Dear HUG;

Here is a subroutine that inputs numerals in calculator style. It checks for valid characters and allows backspacing to correct errors. Line 50 should be changed to "50 IF A$=CHR$(13) THEN RETURN" to return to the main program, here we just made the subroutine repeat after each entry. The PRINT USING statement in line 110 can be modified to accept entries in any fixed decimal format.

```
10 ' Money Input Formatter, written by Stephen Ruks,
20 ' of Diamond Data in ZBASIC
30 CLS:A1$="":DEFDBL A:'                    Start with zero
40 A$=INKEY$:IF A$="" THEN 40:'          Input a character
50 IF A$=CHR$(13) THEN 30'  Line to be changed per text
60 IF ASC(A$)<48 OR ASC(A$)>57
       THEN IF ASC(A$)<>8 GOTO 40:'                Valid?
70 IF A$=CHR$(8) AND LEN(A1$)<1
       THEN BEEP:GOTO 40:'                  Backspace ok?
80 IF A$=CHR$(8) THEN A1$=LEFT$(A1$,LEN(A1$)-1):
       GOTO 100:'                                 Back Up
90 A1$=A1$+A$'                 Concatenate new character
100 A=VAL(A1$)/100'                   Convert to numeric
110 LOCATE 10,32:PRINT USING "$##,###.##";A'    Print it
120 GOTO 40:'                     Get the next character
```

Stephen Ruks
7884 Highlander Dr.
Anchorage, AK 99502

Dear HUG;

Congratulations! You have finally added a segment to REMark that makes it all worth while for the beginner. I am referring, of course, to "My Favorite Subroutines." I just wish that something like it had been available years ago. I will try and contribute a gem now and again.

Re: Your July issue, Pg. 61; The entry from Mr. Kletter:

Try a neater, simpler check for a single letter input. Just replace line 60 with:

```
60 ON R GOTO 100,80,90
```

And replace line 1020 with:

```
1020 R=INT(1+INSTR(" yYnN",R$)/2)
```

Thereby checking for a response in either upper or lower case and returning with the corresponding number for a GOSUB directive. The change in line 60 was necessary, since a zero (0) will be returned if no match is made. Note the space in line 1020 " yYnN". Very important!

In a similar manner, the routine in David Warnick's article, 'Practical File Management', (June issue, Pg 18) as follows:

```
1170 M$=INPUT$(1)
1180 IF M$="A" GOTO 2000
1190 IF M$="C" GOTO 5000
1200 IF M$="D" GOTO 8000
1210 IF M$="L" GOTO 11000
1220 IF M$="N" GOTO 14000
1230 IF M$="P" GOTO 17000
1240 IF M$="S" GOTO 20000
1250 IF M$="X" GOTO 23000
```

may be simplified using the same procedure (as above), while also permitting the entry to be made in either upper or lower case:

```
1170 M$=INPUT$(1):
          R=1+INT(INSTR(" AaCcDdLlNnPpSsXx",M$)/2)
1180 ON R GOTO 1270,2000,5000,8000,11000,
       14000,17000,20000,23000
```

This procedure works exactly as the first, returning a match-up location for the ON/GOTO statement.

Perhaps you will like this rather unique procedure. Try it and see.

Raymond Dotson
214 S. Berkeley Blvd.
Goldsboro, NC 27530

Dear HUG;

This is a subroutine to move the cursor around the screen using the key pad. It was written in MBASIC (CP/M). It takes the number that corresponds to the key hit and compares it to the different possible keys. The results of all these comparisons are either added to or subtracted from the old cursor position (X,Y). The subroutine keeps track of the last key hit, so the cursor continues to move until you hit a new key.

```
10 WIDTH 255:Y=10:X=40
20 GOSUB 1000:GOTO 20
...
...
...
1000 O$=INKEY$:IF O$<> THEN I=VAL(O$)
1010 Y=Y-(I=1)-(I=2)-(I=3)+(I=7)+(I=8)+(I=9)
1020 X=X-(I=3)-(I=6)-(I=9)+(I=1)+(I=4)+(I=7)
1030 PRINT CHR$(27);"Y";CHR$(Y+31);CHR$(X+31);
1040 RETURN
```

Chris A. Toomey
11425 Lakeshore Dr. West
Carmel, IN 46032

Dear HUG;

This subroutine simulates the action of the BASIC function INKEY$. The routine assumes that the H and L registers contain the address of an 8-bit integer for the return of the console input character. If the calling program expects a 16-bit result, it must zero out the most significant byte at (HL)+1. A 'null', 00H, is returned if there is no character in the input buffer.

There are two conditional assembly variables. SAVE_STACK = 1 causes the routine to create its own stack and save all registers. The

Vectored to 82 ☞

ZENITH Users:

Catch These Bargains!

from
Husker Systems of Nebraska

GREAT PRINTER VALUES!		RETAIL PRICE	OUR PRICE
DOT MATRIX PRINTERS:			
MPI-99	80 col,100 cps,P,S	599	CALL
MPI-150	132 col,150 cps,P	995	CALL
PRISM	132 col,110+ cps,P,S	1499	1179
PRISM/graphic	132 col,110+ cps,P,S	1995	1549
OKIDATA	80 col, 80 cps,P	299	263
OKIDATA	80 col,160 cps,P	599	473
OKIDATA	132 col,160 cps,P	999	767
GEMINI	15 in,120 cps,P	549	433
DELTA	10 in,160 cps,P,S	649	499
RADIX	10 in,200 cps,P,S*	849	659
DAISY WHEEL PRINTERS:			
SILVERREED	10 in,14 cps,P	599	479
SILVERREED	13 in,16 cps,P	699	551
DIABLO	13 in,20 cps,S	1595	CALL
DIABLO	13 in,45 cps,P,S	2400	CALL
DAISYWRITER	16.5 in,16.5 cps,P,S	1445	1117
TRANSTAR	17 in,16 cps,P,S	895	689
TRANSTAR	16 in,40 cps,S	1695	1319

codes: col/in refer to carriage width, cps=char per sec (speed)
S=serial interface, P=parallel interface

Note: Most parallel interface printers have a serial option so call us for pricing. We will deal!

FINE HARDWARE ACCESSORIES

Z-204 Multi-Port S-100 Card. Adds 4 serial ports and 2 parallel ports to your Z100. Includes 1 DCE, 1 DTE, and 1 Parallel connector.

Call for our sale price!

RDM-256 MS-DOS RAM-DISK. Add the pleasure of instant file access from RAM with this new Zenith product.
Suggested List-$1025 Our Sale Price-$809!!

ROM Version 2.0 Upgrade. This new version provides built-in diagnostics, memory viewing, and other features to your Z100.

Call for our sale price!

HIGHLY RELIABLE 8 INCH DRIVES
Our 8 inch-disk drive include the cable and a *1 Year Warranty*. They are fan-cooled and are enclosed in an attractive white cabinet with a heavy-duty power supply. These drives provide 1.25 MB DSDD storage.
RETAIL PRICE - $959 OUR PRICE - $527!

PIICEON 20MB WINCHESTER DRIVES
Get the speed and capacity only hard disk can provide. These quality drives include the controller board, power supply, cabinet, all cables, and software to be supported under ZDOS. An unbeatable buy!
RETAIL PRICE-$2495 OUR PRICE-$1987

ZMAG: The Software Subscription Service-

A ZMAG Subscription gives you a magazine and a diskette every 6 weeks. That's 8 issues a year, each containing 10 to 15 good programs covering the scope of personal computing applications. And a membership also provides you with 8 free consulting calls, a 10% discount on our software, special discounts, interesting articles, and editorials. At ONLY $147 each year, your per program cost is less than $2, and you can select our Easy-Pay option--Pay $75 now and we'll bill you the rest later! Subscribe TODAY!!
[CP/M-80 hard sector; ZDOS soft sector]

VIDEO*PROFESSOR: Fine Educational Software-

The Video*Professor Educational Series uses the computer to teach you interactively with easy-to-follow text, examples, and menu-driven quizzes. Courses available are: MBasic(ZBasic) Programming I, Assembler Programming I, and Structured Programming. New courses in final development stages are ZBasic Graphics Programming, Pascal, Cobol, Random File Programming, and others. Normally, each course lists for $29.95, but with this offer get the 3 available courses for ONLY $22 each or all 3 for just $62--a 30% discount!
[HDOS,CP/M-80 hard sector; ZDOS,CP/M-85 soft sector]

COMM-PAK: A Double Value For Z100 Communications-

COMM-PAK is 2 of our best communications products in 1! It includes ZED-COMM, our popular, easy-to-use yet multi-function modem control program and REM-CON, the Remote Console Utility that allows you to use your Z100 from an ASCII terminal at a remote location. Via modem connections, you can run programs (including ZBasic) just as if you were sitting at your Z100 keyboard! Packaged together in an attractive linen binder, the regular price is $94.90. Your price is ONLY $69.95, a 25% discount! [ZDOS soft sector]

Z6000 FILE TRANSFER: Make Your Z100 Conversion Painless-

With Z6000 File Transfer System you can move files from your H8/Z89 to your Z100 and back again. It uses a direct cable connection which you can prepare yourself. Z6000 can transfer any size files and will accept wild cards for transfer of entire diskettes. You get 2 diskettes: 1 for your H8/Z89 and 1 for your Z100. Regular price is $34.95 but buy it now for JUST $24.95--or get BOTH the COMM-PAK and Z6000 for ONLY $91--That's 30% off!! [CP/M-80 hard sector AND CP/M-85 OR ZDOS soft sector]

SPECIFY PRODUCTS DESIRED AND FORMAT — Payment Method: ____Check ____Visa ____MC Send Catalog: ____

Product(s) ________ Format ________ Price ________ ____(x) ZMAG Easy-Pay Current Date: ____________

________ ________ ________ Name ____________ Address ____________

________ ________ ________ City ____________ State ________ Zip ________

________ ________ ________ *Total ________ *Nebraska Residents Add 5% Tax.

Call Toll-Free: (800) 835-7427 ext 277 Or Send To: Husker Systems of Nebraska, 6657 Redick, Omaha, NE, 68152

Z-FORTH: A BASIC Approach to FORTH

Ralph Nelson
205 Mercury Road
Newark, DE 19711

If you know nothing about FORTH, read a description of it before proceeding. The book "Starting Forth" (by Leo Brodie, $7, FORTH, Inc.) might be a good place to start. If you know something about FORTH, but have been frustrated in implementing it, this article should allow you to make rapid progress in writing error- free FORTH programs. You must be able able to write programs in BASIC (such as Benton Harbor BASIC) and to have access to a FORTH system (such as HFORTH79, by Pat Daugherty, $39, SoftShop).

As a programmer who is quite familiar with BASIC, I have been annoyed to find that many descriptions of FORTH are nearly impossible to follow. In 1958 I learned to program in FORTRAN II, and in 1965 I had no trouble learning BASIC. But even well written books and articles did not make programming in FORTH easy for me. I was quite eager to use FORTH for some programs which take too much time in BASIC, so I was forced to develop a solution to my problem. Success came through writing a translation table which gives the FORTH equivalents for the most common program elements in BASIC. I can now write a program in BASIC, debug it with the many convenient features of BASIC, then translate it to a very fast FORTH program by writing out the elements from my translation table. I call my technique "Z- FORTH".

Some might argue that novices ought to learn FORTH by itself, rather than through another computer language. But I think that the use of Polish notation and stacks is so foreign to people with no computing background that they will be overwhelmed by the new concepts. They may quit before they get a simple program to work properly. Consequently, I suggest that a new programmer become familiar with BASIC to see what computing is all about, then make the transition to FORTH through Z-FORTH. This may not give the most efficient code, but at least novices can get started on interesting programs with some confidence that the finished code will work. As the novices study books on FORTH and come to understand the concepts and the more advanced operations, they can upgrade the Z-FORTH programs.

My table gives the BASIC and FORTH code for twelve program elements. For clarity, I have included spaces in the BASIC elements, even though spaces are required in only a few places. Spaces are required between all words in FORTH, and I have added a few extra spaces for clarity. Note that the amount of typing required to write the code is about the same for FORTH as for BASIC.

To make comparisons with the FORTH version easier, I have generally listed only one BASIC statement per line. Most versions of BASIC permit several statements (separated by colons) on one line. FORTH requires new lines only for clarity, so the examples given here could be compressed to fewer lines with no special characters added.

Many versions of BASIC limit variable names to a single letter plus an optional single number, with $ added if the variable is a string. In HFORTH79 a name may contain up to 32 characters. The name must start with a letter, but doesn't need a $ or other character to indicate that the reserved memory will contain a string. To maintain parallelism I have used the same names in FORTH as in BASIC, but I could have used PLAYER-NAME instead of P$ in the HIROLLER program example.

Since FORTH allows the use of multi-letter names for subroutines (as well as for variables), a FORTH program is easy to understand without many remarks. If you wish to include comments, use REM xxxx in BASIC and {(xxxx)} in FORTH. Since FORTH used many punctuation marks as commands, I have enclosed FORTH statements in curly braces to set them off from the text. The unconditional jump to a specified line number (GOTO in BASIC) does not exist in FORTH. There are no line numbers, and all non-sequential execution paths are part of {IF...THEN...ELSE}, {BEGIN...UNTIL}, or {DO...LOOP} structures or subroutine calls and returns.

Some Comments on the Elements of BASIC vs FORTH:

#1. We want to assign values to variables within the program. In FORTH, we write the value, then the name of the variable to which it will be assigned, then {!} to store the value at the address allocated to the variable. In BASIC we need to allocate memory only if we plan to use an array. To do this, we write DIM A(50). In FORTH we must allocate memory for every variable used. It is most convenient to do this for all program variables at the start of the program, so that we won't have to search through the whole program to see which names have been used. I shall illustrate the reservation of space in only a few examples below.

#2. We want to prompt the user to type in values for a variable while the program is running. In BASIC we put the prompt inside quotation marks just after the INPUT statement. In FORTH, {."xxxx"} fulfills the same function as a BASIC PRINT "xxxx" statement, while the numeric input statement in FORTH is {#IN}. A carriage return-line feed is implicit in the BASIC statement. It must be explicitly stated using {CR} in FORTH.

#3. We want to display values, usually with some text to label what the value represents. The command {?} in FORTH displays the value for the variable name preceding it.

#4. We want to do algebraic operations. In FORTH we must contend with reverse Polish notation. This is really a very convenient and efficient system, but you have to get used to putting two values on the stack before doing a binary algebraic operation. The stack is like a pile of cards with numbers written on them. The numbers may represent values, addresses, or ASCII characters. FORTH uses several stacks. The term "stack" when used by itself in this article refers to the program stack.

The command {@} fetchs the value for the variable whose name just preceeds it and puts that value on top of the stack. We do this twice. The {-} operation is then performed, leaving the result on the stack in place of the two values previously on the stack. {C !} then stores this result in memory. Because FORTH stacks the result of a computation, it doesn't use the = symbol, And since FORTH executes code strictly from left to right, the name of the variable to which the result is assigned (C in our example) comes after (to the right of) the algebraic expression, rather than before as in BASIC.

#5. We want to assign a string of alphanumerics to a variable. A string is usually several characters long, so we must allocate adequate memory for assigned or typed-in strings. BASIC automatically allocates memory for strings. In FORTH we must reserve one more space than the number of characters we expect to enter. The {1 A$} puts on the stack the address of the first byte allocated to the A$ array. The statement {!" xxxx"} stores xxxx in memory starting at that address.

#6. We want to prompt the user to type in strings. Note that we may initialize an array by filling it with blanks (32 is the ASCII number for a blank). {max# EXPECT} takes in as many as max# characters until <CR> is pressed or max# of characters have been typed in, at which point it stores them starting at the address on the stack beneath max#.

#7. We want to display stored strings. In FORTH we must specify the starting point within the array and the maximum number of bytes to be typed. To eliminate the blanks at the end of a string (which may not fill the whole array), use the command {- TRAILING}.

#8. We want to repeatedly execute a section of program while incrementing an index. The usual structure for this is the loop, which in its simplest form has an integer index starting at 1 (one) and incrementing by one to a maximum (N9 in the example). When {DO} is encountered in FORTH, the two top values on the program stack are transferred to the loop stack. The loop index is the top value on the loop stack. It increments to a maximum that is one-less-than the next-to-top value on the loop stack. In the example, we setup this maximum using {N9 @ 1 +}.

The loop index may be copied onto the program stack by simply writing {I}. Since {I} is the name of a procedure (not a variable) in FORTH, it should not be used as a variable name. (Neither should J, K, or I1.) We need not set up a separate variable for the loop index comparable to the N in the BASIC example. You may store values at a particular index location in an array by putting the value, then the index, then the array starting address on the stack, then giving the store command, as in {6 I E !}.

#9. We want execution to continue at (branch to) different places in the program depending on whether the result of a specified comparison is true or false. In BASIC we may branch to a line which comes either later or earlier in the program. FORTH does not use line numbers, and the statements required to branch to an earlier part of the program (a backward branch) are different from those used to branch to a later part (a forward branch). I shall discuss backward branching in #10.

The branch statement in BASIC consists of IF, the condition to be tested, THEN (on the same line), the true-action (on the same line), and the false-action (on succeeding lines). The true- action may include a GOTO, with execution proceeding at a distant point in the program, either before or after the branch point. BASIC does not require that the two execution paths merge again after the IF. FORTH does require that they merge. The condition is true if after a comparison in FORTH, the value at the top of the stack is not zero. For a true condition, the operations between {IF} and {ELSE} are executed. The condition is false if the top-of-stack value is zero. For a false condition, the operations between {ELSE} and {THEN} are executed. In either case execution then continues with the operations following {THEN}.

#10. To setup a backward branch in FORTH, we use {BEGIN} to indicate where to branch back to. The condition test occurs at the word {UNTIL}. If the condition is false when the program encounters {UNTIL}, then execution continues at {BEGIN}. If the condition is true, execution continues with the word after {UNTIL}. {BEGIN...UNTIL}, {IF...ELSE...THEN}, and {DO...LOOP} pairs may be nested, just as FOR...NEXT loops are in BASIC.

#11. We want to set up a subroutine (a common set of operations which will be executed at several points in a calculation). In BASIC we write out a set of statements ending with RETURN. The subroutine may be placed either before or after the call statement (GOSUB 1000) in a BASIC program.

In FORTH we define a new word by starting the subroutine with {:} followed by the name of the subroutine, then the statements, and ending with {;}. The definition of a word must precede any use of that word in FORTH. This contributes to the odd appearance of many FORTH programs, since in our top-down thinking we expect the major structure of a program to be at the start, with the details handled below. It is possible to write FORTH programs so that blocks at the end are compiled before those appearing first in the listing. I have done this for the HIROLLER example to make it more parallel to the BASIC listing and because I believe that this is the most sensible way to present a FORTH listing. The statement {1005 LOAD} compiles block 1005 before proceeding with compilation of the current block.

#12. To use a subroutine in BASIC we use GOSUB and specify the line number at which it starts. In FORTH we simply use the names of the routines, which in our example might refer to writing a split octal address, moving a block of data, and scoring points.

A Worked Example of a Conversion

The listings of HIROLLER show what a simple BASIC program looks like when translated into FORTH using my table of elements. I start with a top-down program outline, which should be the first step in developing any program. The operations enclosed in square brackets refer to subroutines. These are written out in detail later on in the outline. When you have read and understood this example, you should be ready to convert one of your own BASIC programs to FORTH. Good luck, and may Z-FORTH be with you!

The Twelve Program Elements in BASIC versus FORTH

```
#1. TO ASSIGN A VALUE:

      100 A=3
  vs
      VARIABLE A   3 A !


#2. TO PROMPT FOR A VALUE:

      110 INPUT "Enter a number -> ";B
  vs
      VARIABLE B   ." Enter a number -> " #IN  B !   CR


#3. TO DISPLAY A VALUE:

      120 PRINT "Second Addend =";B
```

```
vs
    " Second Addend ="  B ?  CR
```

#4. TO SUBTRACT:

```
    130 C=A-B
vs
    A @ B @ - C !
```

#5. TO ASSIGN A STRING:

```
    140 A$="Earnings"
vs
    81 CARRAY A$  1 A$ !" Earnings"
```

#6. TO PROMPT FOR A STRING:

```
    150 LINE INPUT "Enter a string -> ";B$
vs
    81 CARRAY B$  1 B$ 80 32 FILL
     " Enter a string -> "  1 B$ 80 EXPECT
```

#7. TO DISPLAY A STRING:

```
    160 PRINT B$
vs
    1 B$ 80 -TRAILING TYPE  CR
```

#8. TO DO AN INTEGER LOOP:

```
    170 DIM E(32)
    175 N9=5
    180 FOR N=1 TO N9
    190 E(N)=2*N
    200 NEXT N
vs
    32 ARRAY E    VARIABLE N9
    5 N9 !
    N9 @ 1 +  1 DO
    I 2 *  I E !
    LOOP
```

#9. TO DO A FORWARD BRANCH:

```
    210 IF X>1 THEN X=1: Y=0: GOTO 230
    220 X=0: Y=1
    230 PRINT X,Y
vs
    X @ 1 > IF  1 X !  0 Y !
    ELSE 0 X !  1 Y !
    THEN X ?  Y ?  CR
```

#10. TO DO A BACKWARD BRANCH:

```
    320 X=0
    330 X=X+1
    340 INPUT "Enter 0 to sum, 1 to stop -> ";Q
    350 IF Q<>0 GOTO 330
    360 PRINT X
vs
    0 X !
    BEGIN  X @ 1 + X !
    ." Enter 0 to sum, 1 to stop -> "  #IN
    1 =  UNTIL
    X ?  CR
```

#11. TO SETUP A SUBROUTINE:

```
    500 REM ---SPLIT OCTAL ADDRESS---
    510 H=M/256: L=M-256*H
    520 PRINT "split octal:";H;".";L: RETURN
vs
    \ ---SPLIT OCTAL ADDRESS---
    : SPLIT  M @ 256 / H !  M @ H @ 256 * - L !
    ." split octal:" H ?  ."  ." L ?  CR  ;
```

#12. TO CALL SUBROUTINES:

```
    250 GOSUB 500: GOSUB 1530: GOSUB 2350
vs
    SPLIT  MOVE  SCORE
```

Top-down Programming Outline

Description: This is a dice game for 2-5 players, who start with 10-1000 dollars each and use 2-5 (standard) dice in 1-10 rounds of play. At the start of the game and the end of each round the order of play is reset so that the players with the most remaining cash go first. Players must bet something each round if they have anything left. The players who go last have an advantage in that they can see what dice earlier players have thrown, so they may bet only a little if there are high rolls on the table. (This is not a very sporting game.) At the end of a round the pot is given to the high roller and the players positions are summarized on the screen.

```
HIROLLER:
        Print title
        Print rules
        [INITIALIZE PLAY]
        For each round
                [PLAY A ROUND]
        Print ending message

INITIALIZE PLAY:
        Get number of players (2-5) [CHECK]
        For each player
                assign order = loop index
                get name
                get starting dollars (10-1000) [CHECK]
        [RESET ORDER OF PLAY]
        Get number of rounds (1-10) [CHECK]
        Get number of dice (2-5) [CHECK]

PLAY A ROUND:
        For each player
                print name and dollars left
                get bet (>0,<=dollars left) [CHECK]
                adjust dollars left for player
                add bet to pot
                [THROW DICE]
                keep index of highest roller to date
        Print winner's name
        [RESET ORDER OF PLAY]
        [SUMMARIZE SCORES]

RESET ORDER OF PLAY:
        For each player except last
                for players not yet compared
                        compare dollars
                        if not in hi-lo order, swap order

SUMMARIZE SCORES:
        For each player
                print name, dollars left

THROW DICE:
        Set sum = 0
        For each die
                generate random number (1-6)
                print it
                add to sum
        Print sum

CHECK:
        * prompt for input value with limits
        check input against hi-lo limits
        if not OK, print error message, go back to *
```

Variable Names:
I kept these the same in FORTH as in BASIC, but could have used longer and more descriptive names in FORTH.

```
Arrays
        P$ - player name
        D - player dollars
        0 - order of player indices

variables
        P9 - number of players
        R9 - number of rounds
        D9 - number of dice
        N - order of current player
        N2 - order of comparison player
        Q - index of high roller
        S - dollars in pot
        T - roll of current player
        T1 - roll of highest roller to date
        V - value of current die
        Z - input value
        Z1 - lower bound
        Z2 - upper bound

Variables used in BASIC only (stack used to hold these in FORTH)
        B - player's bet
        D - index of die throw
        P - index of current player
        P2 - index of comparison player
        R - index of round
```

BASIC Program Listing

```
00100 REM --- HIROLLER ---
00130 DIM P$(5),D(5),0(5)
00140 PRINT :PRINT :PRINT
00150 PRINT "84.06.06  *** HIROLLER *** by Ralph Nelson"
00152 PRINT :PRINT "This is a dice game with several rounds."
00154 PRINT "The high roller or first to roll a high tie wins."
00156 PRINT "The order is from highest dollars to lowest."
00158 PRINT "If dollars are the same, use previous order of play."
00210 GOSUB 300:REM initialize play
00230 FOR R=1TO R9:GOSUB 400:REM play a round
00240 NEXT R
00250 PRINT :PRINT "----- The game is now over -----":PRINT
00260 STOP
00300 REM --- SETPLAY ---
00310 PRINT "Number of players (2-5)";:Z1=2:Z2=5:GOSUB 700:P9=Z
00320 FOR P=1TO P9:0(P)=P:REM initialize order index
00325 PRINT :LINE INPUT "Name of player -> ";P$(P)
00340 PRINT "Starting dollars (10-1000)";:Z1=10:Z2=1000:GOSUB 700:D(P)=Z
00345 NEXT P:PRINT
00350 GOSUB 500:REM reset order of play
00360 PRINT "Number of rounds to play (1-10)";:Z1=1:Z2=10:GOSUB 700:R9=Z
00370 PRINT "Number of dice (2-5)";:Z1=2:Z2=5:GOSUB 700:D9=Z
00390 RETURN
00400 REM --- ROUND ---
00405 S=0:Q=0:T1=0
00410 FOR P=1TO P9:N=0(P)
00420 PRINT :PRINT P$(N);" has $";D(N);"."
00422 IF D(N)=0THEN PRINT "Sorry, you are out!":GOTO 470
00425 PRINT "Place your bet (>0)";:Z1=1:Z2=D(N):GOSUB 700:B=Z
00440 D(N)=D(N)-B:S=S+B
00450 GOSUB 600:REM throw dice
00460 IF T>T1THEN Q=N:T1=T
00470 NEXT P:D(Q)=D(Q)+S:PRINT :PRINT "The hi-roller this round is ";P$(Q)
00480 GOSUB 500:REM get new order of play
00485 GOSUB 650:REM summarize scores
00490 RETURN
00500 REM --- ORDER ---
00510 FOR P=1TO P9-1:FOR P2=P+1TO P9
00515 N=0(P):N2=0(P2):REM get indices for two players
00517 REM if dollars are not in hi-lo order, swap indices in the order
00520 IF D(N)<D(N2)THEN Z=0(P):0(P)=0(P2):0(P2)=Z
00530 NEXT P2:NEXT P
00540 RETURN
00600 REM --- DICE ---
00610 T=0:PRINT "Your dies are";
00620 FOR D=1TO D9:V=INT(RND(1)*6+1):PRINT V;
00630 T=T+V:NEXT D
00640 PRINT "  Your roll = ";T:RETURN
00650 REM --- SCORES ---
00660 PRINT :PRINT "Dollars left:"
00670 FOR P=1TO P9:N=0(P):PRINT P$(N),D(N):NEXT P
00680 PRINT :RETURN
00700 REM --- CHECK ---
00710 INPUT " -> ";Z:IF Z1<=Z AND Z<=Z2 THEN RETURN
00720 PRINT "*** bad, try again";:GOTO 710
```

FORTH Program Listing

The numbers at the top are the program block numbers, used for editing and for loading the program through the compiler.

```
---- 1001 ----------
4 0 DO 1005 I - LOAD LOOP ( This loads blocks in good FORTH order.)

: HIROLLER
  CR CR CR ." 84.06.01 *** HIROLLER *** by Ralph Nelson" CR
  CR ." This is a dice game with several rounds."  CR
  ." The high roller or first to roll a high tie wins."   CR
  ." The order of play is from highest dollars to lowest."  CR
  ." If dollars are the same, use previous order of play." CR
  SETPLAY   R9 @ 0 DO  ROUND  LOOP
  CR  ." --- The game is now over. ---" CR ;


---- 1002 -----------
: SETPLAY
    CR ." Number of players (2-5)"  2 Z1 ! 5 Z2 ! CHECK
    Z @ P9 ! P9 @ 1 + 1 DO  I I O !
    CR  ." Name of player -> "  I 1- 80 * P$  80 EXPECT  CR
    ." Starting Dollars (10-1000)"  10 Z1 ! 1000 Z2 !  CHECK
    Z @ I D !  LOOP  ORDER
    CR ." Number of rounds (1-10)"  1 Z1 ! 10 Z2 !  CHECK
    Z @ R9 !
    ." Number of dice (2-5)"  2 Z1 ! 5 Z2 !  CHECK
    Z @ D9 !  ;

 ---- 1003 ----------
: ROUND  ( play one round)
    0 S !  0 Q !  0 T1 !
    P9 @ 1 + 1 DO   I O @ N !
    CR  N @ 1 -  80 *  P$  80 -TRAILING  TYPE
    ."  has $"  N @ D ?  CR
    N @ D @ 0 =  IF ."   Sorry, you are out!" CR
    ELSE  ."   Place your bet"   1 Z1 !  N @ D @ Z2 !  CHECK
    N @ D @ Z @ - N @ D !   Z @ S @ + S !
    DICE   T @ T1 @ >  IF   T @ T1 !  N @ Q !   ELSE  THEN
    THEN  LOOP   S @ Q @ D @ + Q @ D !  CR ." The hi-roller is "
    Q @ 1 -  80 *  P$  80 -TRAILING  TYPE  CR
    ORDER  SCORES ;

: ORDER ( reset order of play)
    P9 @ 1 DO   P9 @ 1+ I 1+ DO   J O @ D @   I O @ D @   >
    IF  ELSE J O @ I O @   J O ! I O !  THEN LOOP LOOP ;

 ---- 1004 ----------
: DICE   ."  Your dice are "  0 T !  D9 @ 0 DO
    5 RANDOM 1 + V !  V ?  V @ T @ + T !  LOOP
    ."  Your throw = " T ?  CR  ;

: SCORES   ( list players and their dollars)
    CR ." Dollars left:"  CR
    P9 @ 1 + 1 DO
    I O @ 1 -  80 *  P$  80 -TRAILING  TYPE
    ."   $"  I O @ D ?  CR  LOOP  CR  ;

: CHECK ( get & check value within limits)
    BEGIN  ."  -> "  #IN  CR  Z !
    Z2 @ Z @ >=   Z1 @ Z @ <=  AND
    IF 1  ELSE ." *** bad, try again" 0  THEN  UNTIL  ;

 ---- 1005 ----------
 400 CARRAY P$   1 P$ 400 32 FILL
 5 ARRAY D    5 ARRAY O
 VARIABLE P9  VARIABLE R9   VARIABLE D9  VARIABLE T
 VARIABLE S   VARIABLE Q    VARIABLE N   VARIABLE T1
 VARIABLE Z   VARIABLE Z1   VARIABLE Z2  VARIABLE V
```

H/Z89 PERIPHERALS from SECURED COMPUTER SYSTEMS

PORT SERIAL
2/3 —CARD— I/O
PORT PARALLEL

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

Features:

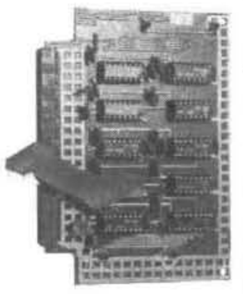
Chip independent design • Reduces computer data buss loading from 3 to 1 • Choice of Centronics or Epson parallel drivers for HDOS or CP/M • Complete documentation and installation instruction.

- 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

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

PRIMERS FOR THE BEGINNER

GETTING STARTED WITH CP/M AND MBASIC
WITH PARTICULAR REFERENCE TO RANDOM FILES

Featuring a complete "menu driven" ready-to-run disk mail list program, program explanations, and complete tutorials, this package forms the perfect introduction to MBASIC programming under CP/M, and includes useful information for those new to the CP/M operating system (ZBASIC also supported). Included is a 56 page manual and a disk containing sample programs. Specify Heath/Zenith Computer model number, disk size and format—hard- or soft-sectored, single- or double-density, 5¼" or 8", and CP/M or ZDOS. $25.00

GETTING STARTED WITH MS-DOS AND BASIC
(AS ABOVE, EXCEPT NO CP/M)

For Z-150/160, MS-DOS and GW-BASIC. 65 page manual and disk included. $25.00

GETTING STARTED WITH HDOS & ASSEMBLY LANGUAGE PROGRAMMING

A 36 page tutorial covering aspects of Assembly Language programming under HDOS. Provides significant information for HDOS which other manuals lack. $15.00

PLEASE SEND CHECK OR MONEY ORDER TO:

WILLIAM N. CAMPBELL, M.D.
855 Smithbridge Road
Glen Mills, PA 19342
(215) 459-3218

OPEN TO ALL HEATH/ZENITH USERS

Western Regional HUG Conference

November 10th & 11th, 1984

Disneyland Hotel
Anaheim, California

- SPEAKERS
- DISPLAYS
- PANELS
- PRIZES

Win a computer:
H/Z-100 with Winchester

Write for Registration Info to:
Conference Coordinator
Western Regional HUG Conference
1555 N. Orange Grove Ave.
Pomona, Calif. 91767

T3D3

Probably the most sophisticated 3-D Tic-Tac-Toe game you'll ever play!

- *Full Color Graphics*
- *Fast (Machine language)*
- *Rotate the playing cube in the X, Y or Z axis*

Requires ZDOS, color option, ROM Version 1.2 or above.

Full price just **$24.95***

by

WESTCOMP
517 N. Mountain Ave.
Suite 229
Upland, CA. 91786
(714) 982-1738

*Foreign orders add 20% for shipping/handling.
*CA residents add your sales tax!

Part 3

SPREADSHEET Corner

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

This month I will take the "PROFIT & LOSS" assignment through a step-by-step procedure using MULTIPLAN and PeachCalc/SuperCalc using the H/Z-100. The procedure will be similar to the one used last month for the LOTUS 1-2-3 software.

First, I will do Multiplan because it has a greater number of differences than PeachCalc/SuperCalc. The "Spreadsheet Preparation Form" will look a little different. The top row of letters--A,B,C,D,E,F--will be replaced with--1,2,3,4,5,6! Therefore, our "cell names" will be different. "A1" would be R1C1--Row 1 and Column 1. Lets try another, "E6" would be R6C5 --Row 6 and Column 5. Do you see how this works? Remember, the "cell" will be named for you in the lower, left hand corner of the "work area". I will let you prepare your Multiplan "Spreadsheet Preparation Form". YOU CAN DO IT! We will discuss the formulas a little later in this article.

Put your working disk "A" with the operating system and the Multiplan files into "boot" drive "A". Put a file disk "B", one that you "Format," but it does not require the operating system because it does not need to be "bootable", in drive "B". "Boot" your computer system and "Load" the Multiplan software by typing MP and Return. Your computer system should provide you with a blank "Worksheet" on the screen with a "Work Area" below. You should find the numbers--1 thru 7-- across the top and the numbers--1 thru 19--down the left side of the Multiplan "Window" and the "highlighted" cell pointer in the upper, left-hand corner of the blank "work screen". Do you find these? In the work area at the bottom area of the screen, you will find the word "COMMAND:" followed by two (2) lines of "possible" command choices. Below these lines, there will be a "message line". Below that line, you will find the "status line" with the cell name (R1C1) followed by the remaining storage space (100%) and then the worksheet name--Multiplan: TEMP. Are they there? Besides the highlighted worksheet cursor, you will find the "window number"--#1--in the upper, left-hand corner and the "edit cursor" on the command line, both highlighted. If you find all these, your system checks out like my system.

You can easily MOVE the "worksheet cursor" to any location in the "window," like we did in Part 2 for the LOTUS 1-2-3 software using the arrow keys and the home key. (Refer back to Part 2 and try the same things.) Multiplan uses a different method to KNOW if you are entering "TEXT" or "NUMBERS". To enter a "LABEL", we will press "A" which will select "ALPHA" command from the "Command Menu". The command line will now display ALPHA and the message line will inform you that the next step is to enter text. (NOTE:-"TEXT" must be enclosed in double-quotes (") when used in formulas. We will discuss this when we first use it.) To enter a "NUMBER", you can press V (VALUE Command) or if you type any of the digits (0 thru 9) or one of these characters--(=) equal sign, (+) plus sign, (–) monus sign, (.) period/decimal sign, (( ) left parenthesis, (") double-quote. To enter a "FORMULA" enter an = (equal sign) or + (plus sign) as the first character.

When using an H/Z-100 with Multiplan, pressing the "HELP" key or typing an ? will provide the user with the Help Screens. TRY THIS NOW! Type ?, you will see the first HELP screen. Type N (NEXT) and you will see the second HELP screen and so on. When you have finished with HELP, type R (RESUME). You can CANCEL the present command at any time by pressing the "ESC" key (Control-C, will also work.).

Now, while at the computer keyboard, with the blank worksheet on the screen, we will start to enter our "model" from the "Spreadsheet Preparation Form". Press the HOME key, so we will all start from the same cell on the "WINDOW". Here is the step-by-step procedure that I used this time with Multiplan:

1) Move the pointer to cell R1C3 by pressing the right arrow key twice.

2) Press A and type P, space, &, space, L and press Return.

3) Press G (GOTO), R (ROW), type 3, press TAB key, type 2, press Return.

NOTE: This is a big change from the way we moved to a different cell using 1-2-3 in Part 2. This might be a good time to type ? for the HELP screen for more information.

4) Press A, type JAN and press the right arrow.

5) Type FEB and press the right arrow.

6) Type MAR and press the right arrow.

7) Type YTD and press the Home key, press Return.

8) Press F (Format) and press Return.

9) Type :R3C5, press TAB key, press C and press Return.

NOTE: The colon (:) specifies a "range" and we specified a rectangle described by using a range with corners at R1, C5, R3, C1 which includes all the labels we have entered so far. The C specified the Label alignment to be "centered" in each of their cells. This is another big change from the 1-2-3 method. Did you notice? Be sure you understand this "range" method and why we use a rectangle! This is needed because we do not have a Global command.

10) Press the down arrow 3 times.

NOTE: We could have used G (GOTO) command as well! Your choice.

11) Press A, type SALES and press the down arrow.

12) Type COST and press the down arrow.

13) Type G.P. and press the down arrow.

14) Press the down arrow (Skips a line).

15) Type G & A and press the down arrow.

16) Type SELLING and press the down arrow.

17) Press the down arrow (Skips another line).

18) Type NET and press the right arrow and the ESC key.

NOTE: We have completed the P & L "template". The column of labels were left aligned by "default". Use ? for Help to get more information about alignment. NOW would be a good time to SAVE our work for safety-first!

19) Press T (transfer), press S (save), type B:P_LTEMP1 and press Return.

NOTE: We are saving our files on the "B" drive; thus, we need the B:. The filename must be typed out completely except for an extension!

20) Press V (VALUE), type =R6C2-R8C2-R9C2 and press the right arrow.

NOTE: We have now entered a formula. We need the V so Multiplan will not decide that the R is the start of a Label! I usually start formulas with the = (equal) or + (plus) signs. Do you understand why we tell Multiplan this? Also, note that cell R11C2 will have a zero "0" in it because Multiplan will calculate the formula we have just entered using the specified blank cell's data (blank cells are the same as 0 to Multiplan).

21) Type =R6C3-R8C3-R9C3 and press the right arrow.

22) Type =R6C4-R8C4-R9C3 and press the right arrow.

NOTE: Multiplan has built-in functions similar to those used by 1-2-3 as explained in Part 2. The SUM function looks like this (note NO "@" is required like with 1-2-3):

SUM(list)

A Multiplan "list" is a "range" of cells like: R1C1,R1C2,R1C3,R1C4.

We can tell Multiplan that we have a "range" for the "list" like this (The colon (:) means range):

(R1C1:R1C4) Therefore, for our example, the function will look like this:

```
SUM(R1C1:R1C4)
```

This SUM function replaces the need for us to type R1C4 for the formula. Which would you rather type? I usually put a **+** (plus sign) in front of SUM so that Multiplan will decide that the S is the start of a formula. Note, that you will find a 0 in the cell as we have explained above.

23) Type +SUM(R11C2:R11C4) and press Return.

24) Press G and R, type 6, press TAB key, type 2, press Return.

NOTE: Remember, it is up to the operator whether to use the GOTO command or the arrow keys to move the pointer. I will use both methods to keep you aware of this. Either one is okay at this entry.

25) Press V, type +R4C2-R5C2 and press the right arrow.

26) Type +R4C3-R5C3 and press the right arrow.

27) Type +R4C4-R5C4 and press the right arrow.

28) Type +SUM(R6C2:R6C4) and press the up arrow.

29) Type +SUM(R5C2:R6C4) and press the up arrow.

30) Type +SUM(R4C2:R6C4) and press Return.

NOTE: We have now entered everything from our Preparation Form. Lets SAVE our work again so we have it protected from our next changes. Do you remember how? Doing it over-and-over a few times will put it in your mind!

31) Press T and press S and Return, type y (overwrite existing file).

NOTE: Now, lets display the formulas so that we can compare them with the Preparation Form that we prepared. To do this, we must change the display to "TEXT".

32) Press F and O, press TAB key, type y and press Return.

NOTE: Multiplan, unlike 1-2-3, doubles the width of all the cells in the "WINDOW" so that the cells will be large enough to display the formulas without truncating them. However, the screen will not be large enough to display the complete "template" in one "WINDOW". Therefore, we MUST "scroll" the window using the right arrow key or we can use the following "actions":-

```
a—Page Up-----Press F8} Scroll to show the
b—Page Down---Press F7} next window-sized
c—Page Right--Press F6} portion of the
d—Page Left---Press F1} worksheet.
```

BE SURE to try these when you are performing your "check-out"! Use ? for Help. How does your "template" compare with the Preparation Form? If everything checks, we will go on. If it does not check, back-up to the steps where you went wrong and make your correction.

33) Press F and O, press TAB key, type N and press Return and HOME key.

NOTE: We will now enter the data. Refer to Part 2 for the data table. The only difference will be the cell names. You SHOULD be able to change the 1-2-3 cell names to the Multiplan cell names. Can you?

34) Press F and W, type 14, press TAB key, type 1 and press Return.

35) Press G and R, type 4, press the TAB key, type 2 and press Return.

36) Press V, type 12456 and press the right arrow.

37) Type 13567 and press the right arrow.

38) Type 14678 and press the down arrow.

39) Type 6787 and press the left arrow.

40) Type 6545 and press the left arrow.

41) Type 6234 and press the down arrow.

42) Press the down arrow three times.

43) Type 1122 and press the right arrow.

44) Type 1234 and press the right arrow.

45) Type 1356 and press the down arrow.

46) Type 678 and press the left arrow.

47) Type 567 and press the left arrow.

48) Type 456 and press Return.

NOTE: We have completed the P & L report. It does not look like a very good financial report, does it? Lets start to "dress" it up! What do we need? How about putting the data in "currency" format? Try the Help screens for more information about how this is done. Also, don't you think that our labels could be improved? How about telling what year this report is for and what company it is for? Before we start such changes, it would be prudent to SAVE our work! Before you read the next step, how about checking to see if you can do it without help. Can you?

49) Press T and S and press Return, type y to overwrite the file.

50) Press F and Return, type R4C2:R11C5, press TAB twice, type $, Return.

51) Press G and R, type 3, press TAB key, type 1 and press Return.

52) Press A, type 1984 and press the right arrow.

53) Type JANUARY and press the right arrow.

54) Type FEBRUARY and press the right arrow.

55) Type MARCH and press the right arrow.

56) Type YEAR TO DATE and press Return.

57) Press I and Return, type 2, press TAB key, type 1, press Return.

58) Press I and Return, press TAB key, type 6 and press Return.

59) Press I and Return, press TAB key, type 9 and press Return.

60) Press G and R, type 1, press TAB key, type 2 and press Return.

61) Press A, type space, space, S, space, P, space, R, space, E, space, A, space, D, space and press right arrow.

62) Type S, space, H, space, E, space, E, space, T, space, space, C, space, O and press right arrow.

63) Type space, R, space, N, space, E, space, R, space, space, C, space, O, . and press Return.

64) Press G and R, type 2, Press TAB key, type 2 and press Return.

65) Press A, type space,space,++++++++++++ (12 "+"), press right arrow.

66) Press A, type ++++++++++++++ (14 "+") and press Return.

67) Press C and Return, type 1 and press Return.

68) Press the down arrow once.

69) Press A, type PROFIT & LOSS and press Return.

70) Press G and R, type 6, press TAB key, type 1 and press Return.

71) Press A, type ************** (14 "*") and press Return.

72) Press C and Return, type 4 and press Return.

73) Press G and R, type 9, press TAB key, type 2 and press Return.

74) Press A, type -------------- (14 "-") and press Return.

75) Press C and Return, type 3 and press Return.

76) Press G and R, type 14, press Return.

77) Press C and F, type R10C2:R10C4, press TAB key, type R13C2:R13C4, press Return.

78) Press G and R, type 16, press Return.

79) Press A, type ============== (14 "=") and press Return.

80) Press C and Return, press Return again.

81) Press G and R, type 7, press TAB key, type 1 and press Return.

82) Press A, type TOTAL SALES and press the down arrow.

83) Type TOTAL COSTS and press the down arrow.

84) Press the down arrow once.

85) Type GROSS PROFIT and press the down arrow.

86) Press the down arrow once.

87) Type G&A EXPENSES and press the down arrow.

88) Type SELLING COSTS and press the down arrow.

89) Press the down arrow once.

90) Type NET PROFIT and press Return.

NOTE: Now, I think we have a good looking P & L report. Lets SAVE it. You should be able to do it now without reading the next step. Can you do it? Try it!

91) Press T and S and Return, type y to overwrite the file.

92) Press F and O, type y and press Return.

NOTE: Do you like the report better with the commas with the data? It would be your choice. Now, lets try a "WHAT IF" by changing January Sales data.

93) Press G and R, type 7, press TAB key, type 2 and press Return.

94) Press V, type 22456 and press Return.

NOTE: Did you follow or see Multiplan re-calculate the "model"? Don't you think that is fast? Try some changes of your own until you know what is happening! Try the Help screens for more information. Remember, whenever you want to QUIT Multiplan, ALWAYS SAVE your "model" FIRST!

95) Press T and S, type y to overwrite the file.

96) Press Q, type y.

NOTE: When you wish to restart, type MP at the A: and press Return. When you see the blank screen, do as follows:

97) Press T and Return, press the HOME key (any arrow key will work) and then use the arrow keys to move the "highlighted" cursor over your file name, then press Return.

Next, we will do the P & L assignment using PeachCalc/SuperCalc software. You will find that these spreadsheet programs have many similar features to the LOTUS 1-2-3. The blank "worksheet" will look like the following:

```
   A        B        C        D        E        F        G        H
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

The "cell" default width is nine (9) characters and it will be displayed in the "work panel" near the bottom of your computer screen along with the cell name, memory used, and an arrow showing the automatic cursor move direction. (Take note of this!) The cells will be named in a similar way as used by 1-2-3. You can obtain "HELP" nearly anytime by typing ? (question mark). The double-quote (") will tell the software you are entering "Text" or "Labels" and I like the use of the + (plus sign) to start formulas. You can repeat a character by typing an apostrophe (') followed by the character--'*--for example. There are two (2) easy ways to move the worksheet cursor. First, use the arrow keys like we have describe before. Second, type an = (equal sign) followed by the cell name where you want to go (=B5) and press Return, as an example. The biggest difference is the Command Menu. Press a / (slash) and you will see a "prompt line" of letters. A list of these letters and a short explanation follows:

```
B---BLANK COMMAND------SYNTAX: /B,[RANGE]
C---COPY COMMAND-------SYNTAX: /C,[OPTIONS]
D---DELETE COMMAND-----SYNTAX: /D,[R(row),C(column)]
E---EDIT COMMAND-------SYNTAX: /E [CELL NAME]
F---FORMAT COMMAND-----SYNTAX: /F [CHOICES]
G---GLOBAL COMMAND-----SYNTAX: /G [CHOICES]
I---INSERT COMMAND-----SYNTAX: /I [R,C]
L---LOAD COMMAND-------SYNTAX: /L (FILE NAME)
M---MOVE COMMAND-------SYNTAX: /M [LOCATION]
O---OUTPUT COMMAND-----SYNTAX: /O [OPTIONS]
P---PROTECT COMMAND----SYNTAX: /P [RANGE]
Q---QUIT COMMAND-------SYNTAX: /Q [ANSWER PROMPTS]
R---REPEAT COMMAND-----SYNTAX: /R [RANGE]
S---SAVE COMMAND-------SYNTAX: /S (FILE NAME)
T---TITLE COMMAND------SYNTAX: /T [PROMPT]
U---UNPROTECT COMMAND-SYNTAX: /U [RANGE]
W---WINDOW COMMAND-----SYNTAX: /W [PROMPT]
X---EXECUTE COMMAND----SYNTAX: /X (FILE NAME)
Z---ZAP COMMAND--------SYNTAX: /Z [PROMPT]
```

Use the "HELP" (?) to learn more about each of these commands. We will learn them as we use them. Using the commands over-and- over will help you to get to know them. This is the best way I know. One other item, Control-Z, will cancel a command that you may want to change. Try the ? for more information. Your "Spreadsheet Preparation Form" will look the same as the one for 1-2-3. If you did the form when you read SPREADSHEET Corner-Part 2, you are ready to start. If you DID NOT prepare the form, DO IT NOW! Refer back to Part 2 for information about how we did it.

I will be doing a step-by-step procedure using PeachCalc and the H/Z-100 computer. SuperCalc would be about the same. Put your working disk "A" with your operating system and the PeachCalc files into the "boot" drive "A". Put a file disk "B", one you "Format," but do not put the complete operating system on it because we will not be "booting" from this disk, into drive "B". "Boot" your computer system and "Load" your PeachCalc/SuperCalc software. You should now have a blank "worksheet," as we described it above. Now, practice moving the "worksheet cursor," as we describe in previous procedures, until you feel at ease with it. Also, try the "HELP" (?) screens at this time.

Lets start to enter our "model" from our preparation form. Press the HOME key so that we all start at the same cell. Here is my step-by-step procedure that I used this time with PeachCalc:

1) Press right arrow twice (pointer should move to C1).

2) Type "P, space, &, space, L and press Return.

3) Type =B3 and press Return.

4) Type "JAN and press Return.

NOTE: The arrow in the "work panel" in front of the cell name will point in the direction that the pointer will automatically Move. Do you see this? If not, watch for it to change as we proceed.

5) Type "FEB and press Return.

6) Type "MAR and press Return.

7) Type "YTD and press Return.

8) Type =A4 and press Return.

9) Press down arrow once.

NOTE: We did this extra step to "set" the automatic arrow pointer down. Did you see this and understand it?

10) Type "SALES and press Return.

NOTE: Do not forget the first letter must be a double-quote (") when entering "text" or "labels'" Did you remember? Use the Help (?) screen for more information when you need it.

11) Type "COSTS and press Return.

12) Type "G.P. and press Return.

13) Press the down arrow to skip a line.

14) Type "G & A and press Return.

15) Type "SELLING and press Return.

16) Press the down arrow to skip another line.

17) Type "NET and press Return.

NOTE: You have now completed the basic "template" from your P & L form. Now would be a good time to learn how to SAVE your work. Remember, SAVE often!

18) Press / and S, type B:P_LTEMP1, press Return, then type A (all).

NOTE: Did you remember your commands? (/) for the Menu and S for SAVE. Next we will enter the formulas to our "model" in the locations we have shown on our preparation form in PeachCalc format. I nearly always use a + (plus sign) as the first character of my formulas, that makes sure that PeachCalc will know I am starting a formula rather than Text or Labels if the first character of the formula is a letter.

19) Type =A12 and press Return.

20) Press the right arrow once.

NOTE: Remember, we must "set" the direction arrow for the automatic cursor move. Did you check the movement arrow (>)?

21) Type +B7-B9-B10 and press Return.

NOTE: We did not start with the double-quote this time because B is the first letter of our formula, NOT TEXT! We did use the + (plus sign) to tell the software that we are starting a formula. Is this clear? Do you know where the 0 (zero) in B12 came from? Check back to Part 2 for the explanation.

22) Type +C7-C9-C10 and press Return.

23) Type +D7-D9-D10 and press Return.

24) Type +SUM(B12:D12) and press Return.

NOTE: The SUM function works the same as LOTUS 1-2-3. Refer back to Part 2 for an explanation. You can also use "HELP" (?) for information. Notice that we DID NOT use the @ sign.

25) Type =A7 and press Return.

26) Press the right arrow once.

NOTE: You should know why we did this step by NOW! DO YOU?

27) Type +B5-B6 and press Return.

28) Type +C5-C6 and press Return.

29) Type +D5-D6 and press Return.

30) Type +SUM(B7:D7) and press Return.

31) Type =E4 and press Return.

32) Press the down arrow once.

33) Type +SUM(B5:D5) and press Return.

34) Type +SUM(B6:D6) and press Return.

35) Press / and G and F.

NOTE: Your "template" now displays your formulas in text form, but you will notice that some of them are truncated because our cell default width is nine (9) characters. We will change the cell width to 14 characters to solve this problem.

36) Press / and F and G, then type 14 and press Return.

NOTE: Now you can see all of your formulas. Compare your "worksheet" with your "Spreadsheet Preparation Form". Is it correct? If so, lets SAVE it! Do you know how? When you do it a few more times you will remember.

37) Press / and S, then type B:P_LTEMP1, press Return, type O (overwrite) and A (all).

NOTE: Would you like to stop now and take a rest? Lets QUIT! ALWAYS! SAVE! your work before you QUIT!

38) Press / and Q, type y (If you are using PeachText, type EN also).

NOTE: To restart where you left off, load the spreadsheet software as we did in the beginning and when you have the blank screen:

39) Press / and L (load), type B:P_LTEMP1, press Return, then type A.

40) Press / and G and F

NOTE: You are now back to your basic "template". Now lets enter the data. Refer back to Part 2 procedure for the table of data.

41) Type =A5 and press Return.

42) Press the right arrow.

43) Type 12456 and press Return.

44) Type 13567 and press Return.

45) Type 14678 and press Return.

46) Press down arrow once and the left arrow once.

47) Type 6787 and press Return.

48) Type 6545 and press Return.

49) Type 6234 and press Return.

50) Type =A9 and press Return.

51) Press right arrow once.

52) Type 1122 and press Return.

53) Type 1234 and press Return.

54) Type 1356 and press Return.

55) Press down arrow once and press the left arrow once.

56) Type 678 and press Return.

57) Type 567 and press Return.

58) Type 456 and press Return.

NOTE: We have now completed the entries for our P & L report. It does not look like a very good financial report, does it? I think it is now time to "dress" it up. We will first put it in "currency" format to make it look financial. I believe the labels need to be improved and that we need to tell what year this report is for and what company it is for. Do you agree? Before we start these extensive changes, it would be smart to SAVE our work again!

59) Press / and S, then type B:P_LTEMP1, press Return, type O and A.

60) Press / and F and G, then type $ and press Return.

61) Type =C1 and press Return.

62) Type "PROFIT & LOSS and press Return.

63) Type =A3 and press Return.

64) Press the right arrow once.

65) Type "JANUARY and press Return.

66) Type "FEBRUARY and press Return.

67) Type "MARCH and press Return.

68) Type "YEAR TO DATE and press Return.

69) Type =A4 and press Return.

70) Press the down arrow once.

71) Type "TOTAL SALES and press Return.

72) Type "TOTAL COSTS and press Return.

73) Type "GROSS PROFIT and press Return.

74) Press the down arrow once to skip another line.

75) Type "G&A EXPENSES and press Return.

76) Type "SELLING COSTS and press Return.

77) Press down arrow once to skip a line.

78) Type "NET PROFIT and press Return.

79) Type =A3 and press Return.

80) Type "1984 and press Return.

81) Press / and I and R, type 1 and press Return.

82) Type =A1 and press Return.

83) Press the right arrow once.

84) Type "space, space, S, space, P, space, R, space, E, space, A, space, D, space and press Return.

85) Type "S, space, H, space, E, space, E, space, T, space, space, C, space, O and press Return.

86) Type "space, R, space, N, space, E, space, R, space, space, C, space, O, . (period) and press Return.

87) Press / and I and R, type 2 and press Return.

88) Type =B2 and press Return.

89) Type "+ and press Return.

90) Press right arrow twice.

91) Type "space and press Return.

92) Type =A6 and press Return.

93) Type "* and press Return.

94) Press / and I and R, type 9 and press Return.

95) Type =B9 and press Return.

96) Type "- and press Return.

97) Type =B14 and press Return.

98) Type "- and press Return.

99) Type =B16 and press Return.

100) Type "= and press Return.

101) Press the HOME key.

NOTE: I like our report now. Do you? Lets SAVE it.

102) Press / and S, type B:P_LTEMP1, press Return, type O and A.

NOTE: Let me show you how to do a "WHAT IF" by changing data in a cell.

103) Type =B7 and press Return.

104) Type 22456 and press Return.

NOTE: Did you see what happened? That is what I call a fast re-calculation of our "model"! This is one reason that spreadsheet programs are so popular and worthwhile. How would you like to do even this simple "model" the old pencil and paper method? Now try some "WHAT IF" examples that you can think of. We have already SAVED our assignment so when you are finished, do a QUIT. I will let you do it on your own this time. We have done it together in the above steps!

Closing

For "homework", I would like you to do whichever spreadsheet "model" you are working with over-and-over until you can do the "model" using only your "Spreadsheet Preparation Form" and the "HELP" screens. This will save you a lot of time when working with the following assignments. I will not supply the detailed step-by-step instructions that I did for this assignment; but I wanted you to get off to a good start. I will be spending the time in future articles on the "NEW" items we will be learning with each assignment. Until next month, happy "SPREADSHEETING".

The Software Toolworks presents:
THE FUTURE

WELCOME TO THE FUTURE, FOLKS. THE COMPUTER AND THE SOFTWARE THAT MAKES IT TICK.
THE SOFTWARE TOOLWORKS 15233 VENTURA BLVD. SUITE 1118 SHERMAN OAKS CA 91403 PHONE (818) 986-4885
THE CHILDREN OF THE FUTURE. PLAYING ADVENTURE, WORD WIGGLE, AND MYCHESS. LEARNIN' COMPUTER OPERATION AND HAVIN' A GOOD TIME.
THAT'S NOT THE WAY YOU SPELL SILICON.
OH.
THE PARENTS OF THE FUTURE. USING MYCALC TO ADJUST THE FAMILY BUDGET AND COMPUTER CHEF TO PLAN TONIGHT'S DINNER.
WE'RE BROKE!
THAT'S OKAY HONEY. HOW DO YOU FEEL ABOUT 'SPAM AU JUS'?
MOM! DAD! "THE A-TEAM" IS ON TV!
WELL, ALMOST THE FUTURE.

Languages, games, productivity software, computer cookbooks and more are formatted for IBM PC, PCjr, most CP/M systems and Heath/Zenith HDOS. They range in price from $19.95 to $59.95. For a free 41 product catalog contact The Software Toolworks, 15233 Ventura Blvd., Sherman Oaks, CA 91403, (818) 986-4885.

CP/M is a registered trademark of Digital Research

Welcome to . . . *It's Great!*

DOODLER

Graphics Package

. . . for the Zenith Z-100

- **Full Featured Graphics Design Package.**
- **Pallette of 36 different color choices.**
- **Store Designs on Disc for later Reference.**
- **Playback Mode for Error Correction.**
- **Extended Text Capabilities including. . .**
  User Designed Character Fonts.
  Italic or Backslant Style and scaling.

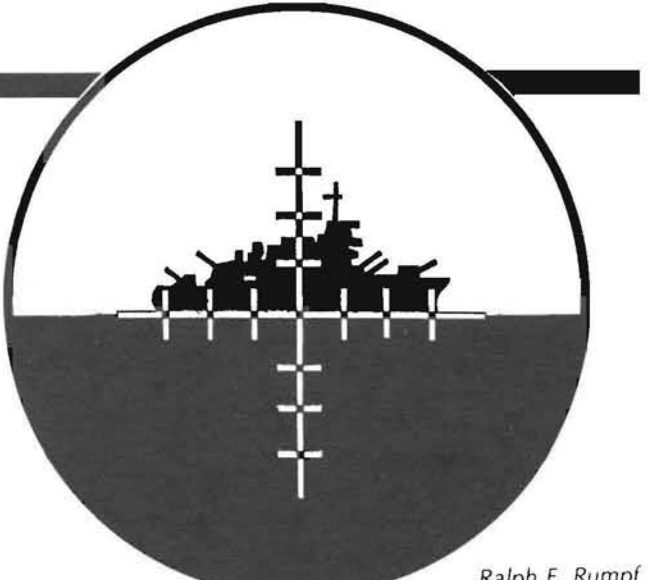
See DOODLER at your local HEATHKIT Store.

. . .or Send $ 79.95 directly to. . .

PAUL F. HERMAN
DATA SYSTEMS CONSULTANT
P. O. Box 535
St. James City, FL 33956
813-283-2227

Specification Sheet available on Request.

High Seas Adventure On The Z-100-PC

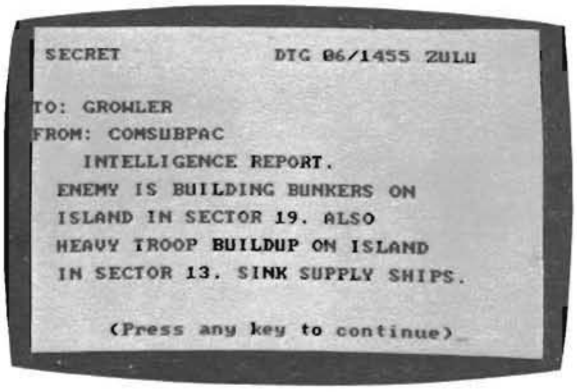


Ralph F. Rumpf
6036 Legion Rd.
Stevensville, MI 49127

Dawn in the South Pacific, quiet, serene, peaceful...... August, 1942.......

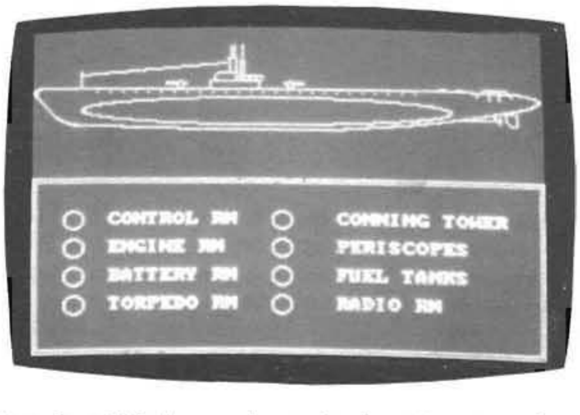
DIVE!...DIVE!...DIVE! AH-OOG-AH...AH-OOG-AH
*Take her down to periscope depth...**UP SCOPE!**...ahead two-thirds, come to bearing 270, OPEN TUBES ONE AND FOUR!.....Target bearing 260 range 10000, **MARK!...FIRE ONE!...FIRE TWO!...DOWN SCOPE!***

Two thin threads in the vastness of the Pacific find their way to the hull of the unsuspecting freighter....two thunderous explosions and a livid ball of flame shatter the morning silence...another day has begun...



```
SECRET                 DTG 06/1455 ZULU

TO: GROWLER
FROM: COMSUBPAC
   INTELLIGENCE REPORT.
 ENEMY IS BUILDING BUNKERS ON
 ISLAND IN SECTOR 19. ALSO
 HEAVY TROOP BUILDUP ON ISLAND
 IN SECTOR 13. SINK SUPPLY SHIPS.

     (Press any key to continue)_
```

Mission Message Display.



```
O CONTROL RM    O CONNING TOWER
O ENGINE RM     O PERISCOPES
O BATTERY RM    O FUEL TANKS
O TORPEDO RM    O RADIO RM
```

Damage Report Display - no damage (we haven't gone anywhere yet).

When I was younger, like most kids my age, I played war and watched all the war movies on TV, pretending I was the hero and doing the impossible. Movies involving submarines, like Run Silent, Run Deep, were especially exciting because of the constantly changing situation and the danger involved when an enemy destroyer was about. It always seemed to me that it took a special kind of super-man to command a submarine in such a hostile environment.

Well, computer games have brought reality as close as the CRT on your monitor. In searching for some relaxing, entertaining form of game for the new Z-150-PC, I found a game called GATO. Generally, I become very bored with a game if it is too repetitive or easy to master. Not so with GATO.

GATO falls into a class of computer games that are known as simulations. A good simulation can keep you engrossed for hours and still provide plenty of challenge for another rainy afternoon. The popular MICROSOFT Flight Simulator is one such game and there are many more on the market today. GATO is what I would refer to as a historical simulation, because GATO is based on fact.

An Overview

During the Second World War in the Pacific Theatre, the backbone of the U.S. Navy Submarine Fleet was a class of submarines called the GATO. Equipped with ten torpedo tubes, diesel and electric engines, and radar, the GATO Class Submarine was well equipped for a variety of functions, from search and rescue missions, to mine laying, to sinking Japanese ships. Additional information about the GATO Class Submarine, including technical information and history, are included on the game disk.

The game of GATO centers on a small part of the Pacific, with several small islands sprinkled in your patrol area. A small corner of this area is more or less in Allied control and within is a single submarine tender that provides the necessary fuel, repairs and torpedos needed to keep your vessel operational. The rest of the patrol area is in enemy control and you venture forth at your own risk. There are several types of enemy vessels you may encounter; freighters, tankers, troop ships, patrol craft, and the ever pesky destroyer. The enemy ships are all based on actual Japanese ships which were sunk in the Pacific during the Second World War.

The implementation of the submarine is very well done. Standard controls include a compass, depth guage, and a speed guage, along with information detailing air and battery status, torpedo status, and torpedo door status. Additional displays provide damage report information, an active radar display, and a captain's log.

When the vessel is surfaced, as well as when you are using the periscope in the attack mode, you may view the area around you in 90 degree windows. The game provides a patrol area map as well as a patrol quadrant map. At lower levels of the game these are updated in real-time, so you know where everybody else is and maybe where you don't want to be.

If you decide to go beneath the waves, don't forget to switch to electric power. Submarines depend on ballast and dive planes for their ability to act like a fish. It can be a real nuisance if you are in the middle of an attack on a group of anchored patrol craft and you loose power. (I never knew submarines could do such strange things!) Likewise, when you come to the surface you should not forget to switch to the diesels (unless you can't). The ship can go much faster and it helps to re-charge the batteries and refresh the air. Commands are implemented by single key-strokes on various keys depending on the command you are requesting. These are described in the manual.

The Manual

The manual is nicely done. The first thing you encounter is a command list. This is very handy when things heat up. All aspects of the submarines' operation and capabilities are explained in the manual. The manual is divided into several sections, covering such things as the instrument panel displays, submarine operation, navigation, and hints on strategy and tactics. Considering the implementation of the game it might be a wise idea to read these before you attempt to play. The manual also includes an ID Sheet for enemy vessels.

The Disk

You will receive one diskette when you receive your software and it is copy protected. A backup disk is available from Spectrum Holobyte for a $5.00 charge, not unreasonable considering the cost of diskettes today. I was a bit upset with the copy protection, but in the software market today I can understand the reason for it.

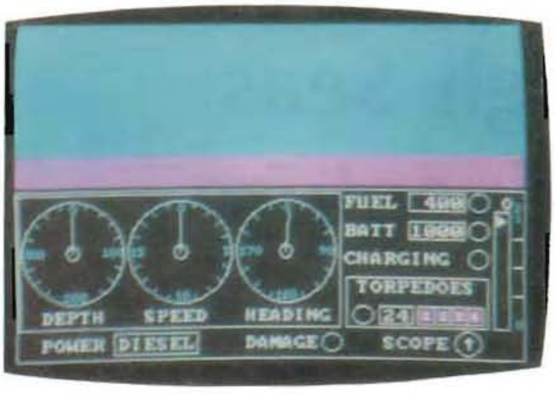
The diskette comes with the necessary programs to allow you to place the program in a demonstration mode, execute the program, or to read the history file that is included on the diskette. I might add here that although the graphics are not those of the Z-100, they are done quite well.

The Game

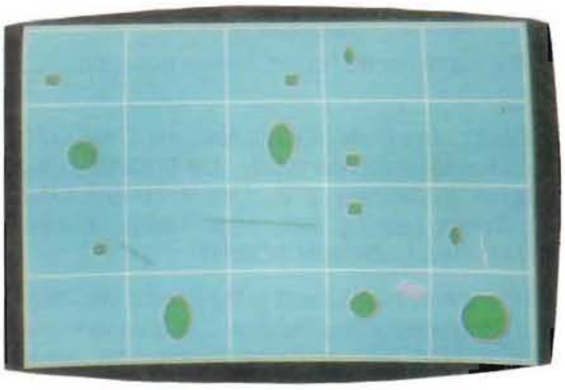
The game itself is quite sophisticated. It begins with a Morse Code message to your vessel from COMSUBPAC. For those of you who don't know Morse Code, don't worry, the message is also displayed on the screen. Be warned, however, at level 7 the printed message disappears and your entire mission depends on your ability to understand Morse Code. From now on, you are on your own.

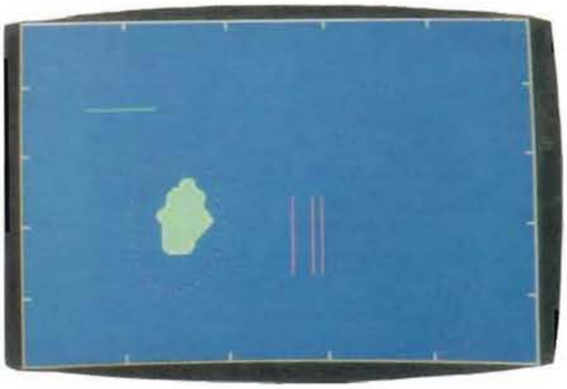
You begin in a friendly quadrant and it is up to you, as the vessel commander, to determine how best to accomplish your mission. You may even elect to ignore the mission and request another. This may prove to be a wise move, because rumor has it that the Japanese have broken some of the code and are transmitting bogus messages.

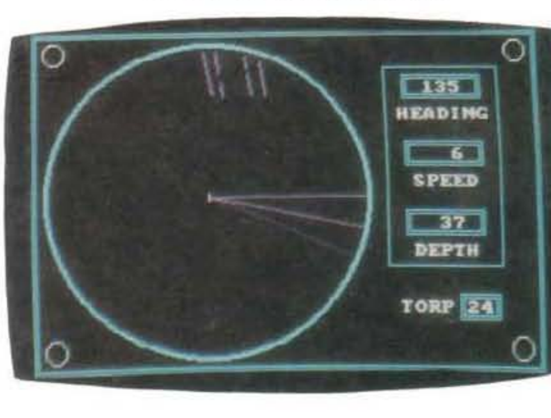
After you study the patrol area chart for a bit, you will be ready to start your engines and navigate a course to intercept your intended target. You will not always have an intended target, however. The game comes with a number of different missions, including rescue, resupply, and search and destroy missions. Once you reach your objective, there are several likely actions you will want to accomplish. Since you can't tell from the charts which ships are where, you need



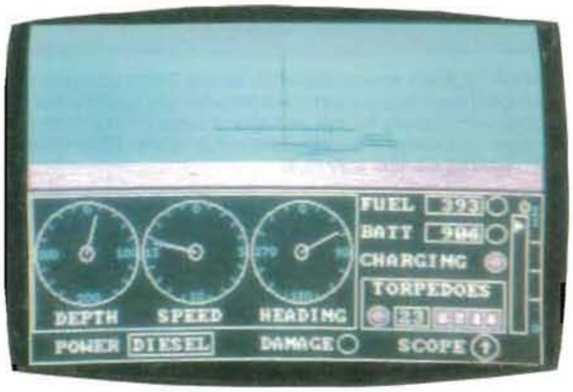
Ship's (sub) on the surface, full load of fuel and torpedoes, ready to go.


Area Chart Display. We're on our way to intercept the supply ships.

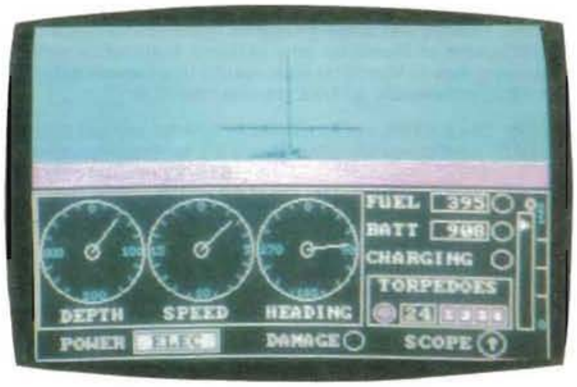

Quadrant Display - sneaking around an island to intercept the ships (red). Note the reef around the island.



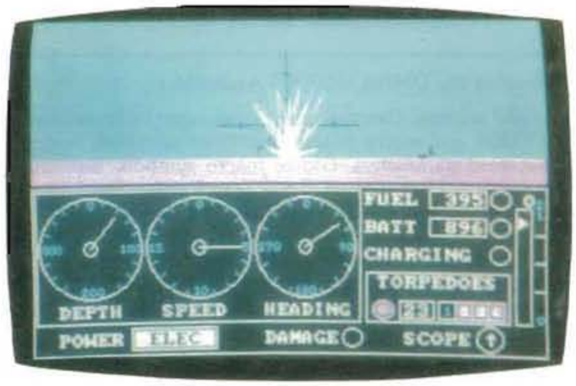
Radar Display. We have Target Acquisition.



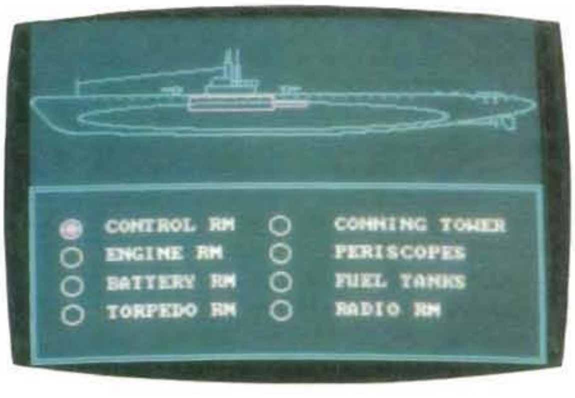
Moving in on the convoy. Still a ways off, running at 40' with torpedo doors open.



Attack Mode. Torpedo doors open and ready to fire.



Got a patrol boat!



Was a little slow getting the patrol boat. I now have damage to the sub.

to be careful approaching them. That most likely means you will have to submerge the ship, use your radar or periscope to locate your target, and move into an attack position.

Now things start to warm up! Patrol craft and destroyers are equipped with sonar, and they use it. You will know they are around when you hear the tell-tale beep of the sonar. Like the real thing, if they detect you, they will do their best to destroy you! They usually operate with a convoy, but are not opposed to luring unsuspecting subs to an early retirement.

If you are detected, you can do one of several things. You can run for it, you can stay and fight (be careful) or you can "run silent, run deep". If you choose the latter, remember the pressure hull is only good to 399 feet.

If you get a clear shot at your target, you will see some nifty graphics. The torpedoes even appear to have a run time to get to the target and the ships will try to avoid them. Each vessel you sink is added to your log. Of course, if you are sunk the log is erased and you start over. Once you complete your mission you can request another and try it again.

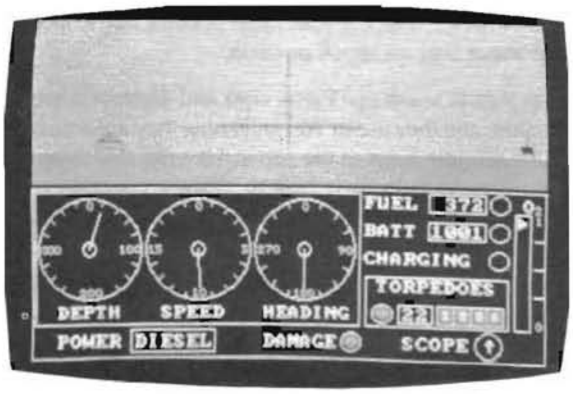
The game allows for several levels of play. Admittedly, on the basic level the game is fairly easy. As the levels go up, the game gets harder. For example, in the basic level (level 0) all charts are updated in real-time (you can always see what is going on, even miles away). At level three, only the quadrant chart is updated and you have to move into a quadrant in order to see your target. At level six, you will not even get an update on the quadrant chart. That means you have to rely on strategy, instinct, and radar to make contact with the enemy. Radar provides a slight advantage over visual, so you have to be fast. In the basic game, one torpedo will sink a ship. In the higher levels, the ships require more hits to sink, depending on the ship and where it is struck. The game provides for a lot of realism as your experience increases. More than once, you will most likely find yourself limping home with damaged engines, periscopes, and a bruised ego.

Comments

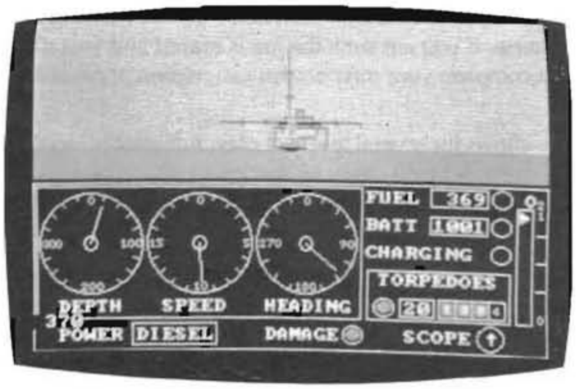
As an overall grade, I would give GATO a B+. The game provides a lot of excitement and it can become addictive. It is very difficult to quit when you've just been clobbered by a patrol boat, you have an overwhelming urge to visit him again. The game also has a hidden feature, as well. I guess you could call it something similar to Hyper-Space for submarines. The only clue I'll provide is a 'Z', but the rest are there if you look carefully enough.

The only major problem I encountered was an annoying habit on the part of the game to periodically freeze up. Since this somewhat limits playability, I contacted Spectrum Holobyte about this. During our conversation, I discovered that my copy was an early release and that the freeze-up problem was noted and corrected by Spectrum Holobyte. Since it did not occur that often, I never really got upset enough to worry about it (except when it happened), but it sure is nice to hear that it has been fixed.

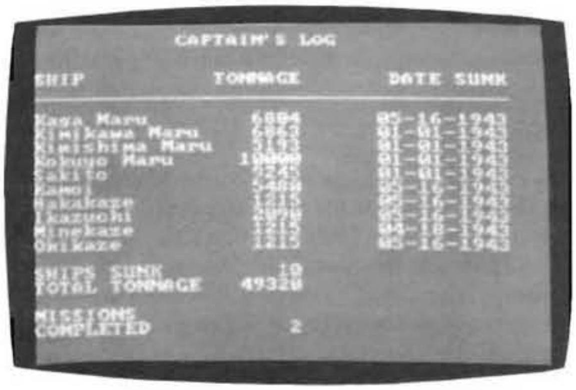
The graphics are well done, but I would love to see a version for the Z-100! Talk about realism! I had some minor objections to the way the screen is updated, but that is a function of the language the program is written in. GATO supports RGB Color and Black & White displays. Although color is more fun, not everyone can afford a good RGB monitor. My other objection was the choice of colors used in the color rendition. A purple sea? I understand that this is a result of the color palette available for the IBM. Perhaps if a Z-100 version should come along, we can have a green sea. My only other comment is that the number of missions appear to be somewhat limited.



Convoy is breaking up and running.



Pursuing the convoy, it's another hit on the stern of a tanker.



```
               CAPTAIN'S LOG

SHIP               TONNAGE        DATE SUNK
Kaga Maru             6884        05-16-1943
Kimikawa Maru         6863        01-01-1943
Kirishima Maru        5193        01-01-1943
Kokuyo Maru          10000        01-01-1943
Sakito                7245        01-01-1943
Kamoi                 5488        05-16-1943
Hakakaze              1215        05-16-1943
Ikazuchi              2070        05-16-1943
Minekaze              1215        04-18-1943
Okikaze               1215        05-16-1943

SHIPS SUNK              10
TOTAL TONNAGE        49328

MISSIONS
COMPLETED                2
```

Captain's log.

Although the skill level of play increases, you repeat the same missions. I would like to see some additional missions only available on certain play levels and some larger ships, like an Aircraft Carrier or a Battleship.

GATO is available from the following source:

Spectrum Holobyte Inc.
2006 Broadway, Suite 301
Boulder, Colorado 80302
(303) 443-0191

GATO costs $39.95 in single quantities and is available at a lower cost for group purchases. (Contact Spectrum Holobyte for details.) The game requires a minimum of 128KB RAM and a version of MS-DOS later than 1.0.

SMART MACHINES™

***SofTouch*™:** A key remapping utility for the Z-100 that redefines any keyboard keys to your specifications for any application program, assigning labels for the remapped keys on the 25th line. **Fast**—does not require a slow co-resident program. Price includes Z/STAR. $39/$3 manual only (3,4)

***SLEDGE*™:** The mean general ledger that does it the way a computer does it best. Includes format-controlled charts of accounts, customized statements, budget comparisons, help screens, and much, much more. $100/$15 demo and manual (1,2,3,4)

***FLIGHT SIMULATOR*:** Fly a 747/Cessna/glider anywhere. Many features and settable conditions. Requires 64K. $29/$3 manual only (1,2,3)

***GRAPH-IT*:** Plot up to 50 functions on any line printer. Create data files with SuperCalc, dBASE II, BASIC, WordStar, and other programs. Many options, thorough manual. $29/$3 manual only (1,2,3)

***Z/STAR*:** Defines and labels all special function and keypad keys for Z-100 users of WordStar (any version). Installation requires only copying files to WordStar disk, results in a **logical** extension of WordStar commands. $24/$2 manual only (3,4)

***Z/BASE*:** like Z-STAR, but for dBASE II. $24/$2 manual only (3,4)
***EP/STAR*:** configures WordStar thru v.3.2 for Epson printers $19/$2 manual only (1, 2)
***89/STAR*:** like Z-STAR, but for the H-89 and WordStar thru V.3.0. $29/$2 manual only (1,2)

*Add $2 per order for postage and handling
*Disk formats available indicated in parentheses

(1) HS-89-80 for H89 & CP/M-80	(3) SS-100-85 for Z100 & CP/M-85
(2) SS-89-80 for H89 & CP/M-80	(4) DS-100-ZD for Z100 & Z-DOS

INTERACTION DYNAMICS SYSTEMS™
KAPAAU, KOHALA, HAWAII 96755-0419

ASSEMBLER CAN BE FUN

Announcing the* OMNICODER™ *Assembler

Meaningful prompts. Descriptive error messages. Unlimited nesting of INCLUDE and macro. Macro definitions with both positional and keyword parameters. Global macro symbols. Subscripted macro symbols. User controlled disk search order. Symbols up to eight characters. List and display files can be directed to terminal, printer, or disk. Uses disk work files for expansions. Produces object modules to be linked by linkage editor. Object format designed to be compatible with multiple processors (up to 32-bit addressing) in order to support planned additions (initial release will only assemble Z80® code).

System Requirements
Z80®. 64K. CP/M®. One or two disk drives (depending on drive and storage capacity). Formats: 5-inch hard or soft-sectored (48 t.p.i.), 8-inch.

Future Plans
We plan to add multiple CPU and MNEMONIC capabilities. This assembler is intended to become part of our PANDORA OPERATING SYSTEM (watch for announcement).

Ordering Information
Price: $90 per site (USA only, no foreign orders). Ohio residents add 5.5% sales tax. MC®/Visa®: include name, number, expiration date, signature, and phone number. Allow six to twelve weeks for delivery. Send order to:

MOCKINGBIRD DATA SYSTEMS
2296 Hoover Rd.
Grove City, OH 43123

Trademarks: Zilog Z80®. Digital Research. CP/M® Heath Company: HDOS™ Mockingbird Data Systems: OMNICODER™. PANDORA™

MS-DOS 2.0 - What is it?

William M. Adney
P. O. Box 13186
Arlington, TX 76096-0186

MS-DOS 2.0 has finally been released. I've already had a number of questions from you about what it has, why it's so advanced, and what happened to Z-DOS. What are the differences? Since there is also considerable interest in the new Z-150, I'll also mention some interesting things that I've learned in the last month about that.

Some Questions and Answers

The most common question I've been asked so far is: "What happened to Z-DOS?" Z-DOS was available in at least 2 different versions, 1.10 and 1.25. With the release of Version 2, Zenith has changed the name to MS-DOS. Although I don't have any specific confirmation as to why, I suspect it has to do with the fact that the Z-DOS operating system was essentially unknown to people outside the Heath/Zenith community. Have you ever tried to order any software for Z-DOS and ended up spending a lot of time explaining that Z-DOS was simply the Zenith version of the MS- DOS? I've had that problem, and my best guess is that Zenith changed the name so that more people would recognize the name of the system. You may have also noticed that the operating system for the new Z-150 is also called MS-DOS, but would you believe that the Z-150 can also boot IBM PC-DOS version 2.1 from a distribution disk? More on that later...

Another interesting question is whether the H/Z-100 can use the MS-DOS version 2 system (OS-63-50) shown in the Heathkit spring catalog (No. 865R). Absolutely not! At least one reason is that the Basic Input/Output System (BIOS) has been customized for the Z-150 series which has a significantly different bus structure than the H/Z-100. Aside from that, the Z-150 was designed to be IBM compatible which means that a lot of the hardware design is different including the escape sequences which clear the screen, move the cursor, and perform other useful functions.

Perhaps the most interesting question is whether the Z-150 can boot the IBM PC-DOS! I booted IBM PC-DOS version 2.1 from the distribution disk on a Z-150 late last month, and the system came up with no problems. I exercised most of the IBM commands, including piping and I/O redirection. There seems to be no problem with any of the system commands, although IBM BASIC will not run on the Z-150. The "Divide Overflow" message is displayed when you try to run either BASIC or BASICA. That's really no surprise since the IBM BASIC uses the Read Only Memory (ROM) to perform some functions. See last month's issue of REMark for additional information on that. That is no real problem since GW BASIC (MS-5063-1) is available in the current catalog and at most Heathkit stores.

MS-DOS 2.0 - A Quick Overview

Let's take a quick look at the overall differences between version 1 of Z-DOS and MS-DOS version 2. This overview generally applies to the Z-150 as well as the H/Z-100. The major differences include command piping, input/output (I/O) redirection, a directory tree structure with new commands to support additional directories, and three filters. Since many of these features provide Unix-like capabilities, I'll discuss each of those topics in more detail in later paragraphs.

With the exception of the directory pathing, most of the commands perform as they did in version 1. That is, the command line syntax and the function remains the same if you ignore the pathing feature. Pathing adds some additional complexity to the commands which we'll look at in the discussion of directory trees.

New built-in commands include a clear screen function (CLS), and a command to display the MS-DOS version number (VER). A new command, VERIFY, is used to verify that disk writes were performed correctly. The old VERIFY command, which was used to locate bad sectors on a hard disk, has been changed to DETECT to avoid any command conflicts.

FORMAT allows you to add a disk label which becomes a hidden file, and the built-in command, VOL, allows you to display that label on the CRT.

Batch File Processing

Five built-in commands have been added to improve the batch file processing: ECHO, FOR, GOTO, IF, and SHIFT. I won't spend too much space discussing these commands since they are discussed in the documentation and my new FlipFast book. Suffice it to say that they provide conditional processing (IF, GOTO), iterative execution of commands (FOR), and suppress/display the commands as they are executed in the batch file (ECHO). The SHIFT command provides the capability to process more than 10 parameters in the file, although I can't imagine why anyone would create something that complicated. The command line would be a zoo with that many parameters.

Command Piping

Although command piping is new to MS-DOS, it's been around for a number of years in the Unix system. I like to separate the piping from the I/O redirection since it's a little easier to talk about the general concept of piping first. Piping is easy. All you have to do is separate commands with a pipe which happens to be the vertical bar (fi). The pipe takes the output from the program on the left side of the bar and "pipes" it as input to the program on the right side of the bar. The command, DIR fi SORT, takes the output of the directory command, sorts it into alphabetical order by filename, and displays it on the CRT. Additional explanations and more complicated examples of piping are included in my new FlipFast book.

Input/Output (I/O) Redirection

Redirection of input and output is another feature of the new MS-DOS. You can send input to a program from a file (instead of a CRT, for example) and/or send the output from a program to a file. Staying with the above example, let's say that we wanted to send our sorted directory to a file so that we could do some fancy batch processing without having to enter all of the file names. The command: DIR ¦ SORT > DIR.DAT, sends our sorted directory to the file DIR.DAT. All the file names are now in DIR.DAT, so all we have to do is a little editing to delete what we don't want. I've used a more complicated command to disassemble the MS-DOS programs to verify the error messages: DEBUG A:CHKDSK.COM <A:CHKDSK.DMP >B:CHKDSK.ERR. The CHKDSK.DMP file contains the appropriate DEBUG commands to disassemble and quit the debugger...it's the input. Note the "less than" (<) points toward the command, DEBUG. The output from the disassembly goes to the file, CHKDSK.ERR, which I then edited to delete all of the extra stuff that came from the disassembly. The "greater than" (>) symbol points away from the DEBUG which indicates that the output is redirected to the file. As you know, the input and output is normally done on the CRT. All we've done is specify a new source of input and a new destination for the output. Those of you familiar with the CP/M XSUB command will recognize that I/O redirection provides a way to provide input to buffered programs, like DEBUG and EDLIN, which was not previously available in Z-DOS. One of the most practical uses of this technique is to automate patches to any program with the DEBUG command. It's so powerful that I've used it in the new FlipFast book to show you how to recover erased files. The same process can be used to change the attribute bytes of files too.

Directory Trees and Pathing

A fair number of the 20+ new commands in version 2 are devoted to the handling of directories. In fact, a number of the old ones, like DIR, COPY, PRINT, DEL(ERASE), and TYPE, have the capability to include path names in the command. Some of the new commands, PATH, MKDIR, CHDIR, and RMDIR, deal exclusively with these new directories. Command details are presented in the documentation. Let's see what directory trees and pathing are all about.

Directory trees may be a new concept, but most people are familiar with an organization chart. The President of the company is at the top, the vice presidents are on the next level, and so on. And so it is with directories...the root is at the top followed by any number of subdirectories. In a simple case, let's look at a root directory with two subdirectories (vice presidents). We'll name the subdirectories PROGRAMS and DATA. The PROGRAMS subdirectory contains all of our programs on the disk which could be either MS-DOS programs (PRINT) or application programs like WordStar. The DATA subdirectory contains all of our text files, spreadsheets, and perhaps a few source listings for programs.

These subdirectories are created with the make directory (MKDIR) command. You can change your current working directory with the CHDIR command. Or you can delete a directory (only when it's empty) with the RMDIR command. The PATH command tells MS-DOS which directory to search for programs. So much for trees.

Filters

Three filters are included in version 2: FIND, MORE, and SORT. A filter is just a command that takes some input, does some processing on it, and then outputs it in a new form. We saw how the SORT program provided an alphabetical sort of our directory when we talked about command piping. The FIND filter searches for a specified string of text in a file. Have you ever used the DIR command and seen the directory scroll so fast that you couldn't read it? The MORE filter stops the scroll every 24 lines so you can read what is there.

MS-DOS 2.0 - A Summary

Version 2 is a significant improvement, and in my mind, it's worth it. A lot of new capabilities and commands allow you to do things that you couldn't do before under Z-DOS. One thing that I did learn is that there has been an extensive amount of customization to this new system. That's not just the BIOS either. I can tell that a lot of the MS-DOS programs have the Heath/Zenith touch, and it's an excellent job.

Z-DOS CP/EMulator

For those of you with a considerable investment and familiarity with 8-bit CP/M programs, this HUG software will definitely be of interest to you if you also use Z-DOS. You can run many CP/M programs under Z-DOS with the CP/EMulator. One of the nicest features is that it has two modes: the direct mode and the command mode. The direct mode allows you to run a CP/M program from a Z-DOS disk and then returns to the Z-DOS system prompt. It's particularly nice if you have utility programs, like a special print or dump, that you have written for CP/M. You can use the standard CP/M command line with the exception that all commands are preceded with CPM, which invokes the CP/EMulator. For example, you can use the LIST command from CP/M-85 to print the file -MYFILE.DAT- by using the command line: CPM LIST MYFILE.DAT.

The command mode is a little more flexible and provides its own CP/M-like utilities: DIR, ERA, REN, SAVE, and TYPE using the standard command formats. Invoking the command mode is done by entering the command: CPM *. This loads a Console Command Processor (CCP) simulator, called CMD.SYS, into Z-DOS which allows the CP/M emulation. To exit back to the Z-DOS prompt, all you have to do is enter BYE. Sounds something like Unix to me.

What do you have to do to use it? First, copy the appropriate COM files from a CP/M disk using the RDCPM command. Using the LIST program that I mentioned earlier, we would now have LIST.COM in the Z-DOS directory. Next, use the Z-DOS rename command to change the file type to CPM. The command is: REN LIST.COM LIST.CPM. Note that the format is different from the CP/M version of the rename command. And now you're ready to run the LIST command under Z-DOS in the direct or interactive mode.

Before I say anything else, you should know that I have a definite opinion on simulators and/or emulators. In the world of computers, they have been around for years, although it seems like everyone has just "discovered" them in microcomputers. The bottom line is that you always lose something, that is, an emulator can never EXACTLY duplicate the function that it is attempting to perform. Although Pat Swayne and Bob Metz have done an excellent job of getting them to work properly, there is simply no way to duplicate everything. If you need something like this, it's a great program to buy, but I'd look in the HUG catalog to find a program that actually runs under Z-DOS so you don't have to worry about whether the program will work or not. As a final note, they have thoughtfully included a patch to WordStar which suppresses the directory display as a result of some problems with that. I wasn't able to test that because my version of WordStar (3.3) is not the standard H/Z version. The December 1983 issue contains some additional details about CP/EMulator.

But if you need a spreadsheet program that runs under Z-DOS and don't want to spend another several hundred dollars for MultiPlan, read on for a nice twenty dollar spreadsheet.

Z-DOS CheapCalc

If you haven't looked at spreadsheet programs because you thought they were too expensive, you should take a look at CHEAPCALC. As a first experience with a spreadsheet, it's terrific! Although I specifically looked at the Z-DOS version, it also comes in CP/M and HDOS versions which I assume are essentially identical. In fact, if you don't have a spreadsheet program yet, I think that you can't afford not to have CheapCalc.

CheapCalc allows you to have a maximum of 15 columns and 40 rows which is generally adequate for most work. My experience is that the columns usually are labelled as months, and the rows contain the various categories for budget purposes. Column widths can be varied from 4 to 30 characters, and many of the H/Z-100 keys are used with or for the commands. For those of you who don't have a spreadsheet, it has on-line help screens which should help you work with it effectively without having to refer to the manual.

In case you're wondering what I use, well...that's a difficult question to answer. I have SuperCalc, MultiPlan, and Lotus 1-2-3. I guess I'd have to say that I prefer SuperCalc since that was the spreadsheet that I started with on the H-89. That preference is probably due to the fact that I'm most familiar with it more than anything else. Hindsight, which I've generally found to be the most correct and also the most useless, tells me that I would have appreciated a simpler spreadsheet, like CheapCalc, when I was learning about them. Using that as a base, I could have better chosen a final one. That's one of the best things about HUG software. You can learn something about different programs, like spreadsheets or word processors, for a very modest price. Then, when you're ready to step up to something more powerful, say Lotus 1-2-3 for example, you'll know what you're getting into. Enough about CheapCalc. It's recommended. The December 1983 issue contains more details about CheapCalc.

Dungeons & Dragons (DND)

I've mentioned before that I'm not much of a game player, but I found DND to be very absorbing aside from the fact that I was killed more times than I can count. What a dynamite game! In fact, I lost track of time while playing this game. The graphics are great, and even for an unsophisticated game player like me, I found it easy to learn. I managed to get the combination to the Heathkit vault, but I never found the vault during that session. If you haven't heard about DND, the March 1984 issue contains more details. But be careful...you may get caught up in the game like I did.

HUG Bulletin Board Handbook

I don't have a modem yet, but I will be getting one in the near future. The Bulletin Board Handbook was announced in the March 1983 issue as possibly being of use to non-users of CompuServe who would like to learn something about the HUG Special Interest Group (SIG). And I found it very interesting. I've ordered the modem, and with any kind of luck, I should have it in a couple of weeks. Then I'll have to learn how to use it, but I have what appears to be a dynamite modem program, called MTERM, from Micro- Systems Software (MSS). Ed Percy of MSS tells me it's "novice- friendly". I'll let you know. By the way, Terry Jensen has done a nice review on MTERM which appeared in the November 1983 issue. The difference is that I'm a novice on using modems, modem packages, and CompuServe. I'll let you know about the experiences of getting that going. Although I've worked on data communications for large computer systems, the microcomputer world is a whole different ball game. By the way, the Bulletin Board Handbook was announced in the March 1983 issue if you need any additional details.

SciCalc Scientific Calculator

What we need is another calculation program...right? Well, I was really impressed with this one. This program reminds me of my old HP-35 calculator which is exactly what it's intended to do. It runs under ZBASIC (Z-DOS), and I'm amazed at what people are able to do in BASIC these days. You may have noticed from earlier columns that I'm not much of a BASIC fan, but Brian Downs has done a nice job on this.

The screen display can appear to be formidable if you aren't used to the reverse Polish notation and the use of stacks in a calculator. I won't go into detail as to how that works except to say that I didn't have any trouble with it because some of the Hewlett Packard calculators used the same idea. Many scientists and engineers use those calculators, and this program can be a substitute. It's an absolute necessity if you're designing warp drives and transporters, not to mention photon torpedoes and phasers.

SciCalc has four menus: the main menu, a conversions menu, a constants menu, and an advanced math menu. The main menu provides trig functions, logs, and provides the capability to work with octal, decimal, and hexadecimal numbers. If you do any programming, it's handy to have for conversion from decimal to hexadecimal and vice-versa. I used it to figure hex file sizes to dump Z-DOS programs using DEBUG since the directory shows the decimal values which have to be converted to hex for DEBUG. Why did I do that? Well, I had to disassemble all of the Z-DOS/MS- DOS programs to obtain the error messages for the new FlipFast command reference for Z-DOS/MS-DOS. SciCalc was perfect for that.

The conversions menu provides a variety of formulas which can be activated by pressing a key. Conversions from kilometers to light years can be essential when you're working on warp drives.

Often used mathematical and physical constants are available in the next menu. A number of constants, such as pi, are available which are described in the March 1983 issue. Access to them is as easy as pressing a key. I wish I'd had this program when I was doing my electrical engineering homework at Purdue!

Although the next menu is called Advanced Math, it provides a number of easy calculations for the area and volume of solids, solving plane geometry problems, and calculating probabilities associated with common statistical distributions (e.g. Poisson, Binomial, Normal, etc.). Looks like a super tool for doing homework to me.

Another valuable feature is that you can save the current state of the calculator on those late nights when you just can't seem to figure out why the answer "doesn't look right". I would think that a lot of engineering, math, and science students would be especially interested in this program. There is a fairly heavy use of function keys, and I think that the use of graphics has helped simplify the screen. Although this program is not intended for everyone like a spreadsheet is, it's recommended for those who work with heavy math.

Offensive Ads

For those of you who haven't seen it yet, I'd like to call your attention to page 136 of the May issue of Popular Computing. An ad by Sundex Software clearly shows a Z-100 with what appears to be a NEC monitor. Someone is watering a plant on the top of the monitor, and the caption is: "Maybe it will make a nice planter". Really now! How ridiculous can you get? I, for one, am insulted by the implication that the Z-100 has neither the capability or the capacity to run their software. If you read the fine print, it's interesting to note that their software is not available in the Heath/Zenith format, and based on that ad, I doubt that they would sell much of it if it were. I don't mind side by side comparisons hardware to hardware or software to software, but comparing apples and oranges? When a software manufacturer starts taking shots at hardware, even by implication, I think it's time to draw the line! By the way, Sundex is marketing some kind of personal accounting software which doesn't run on the Z-100 anyway, but I've included their address at the end of the column for those of you who might like to write to them.

The HUG Convention

For those of you who were able to attend the HUG Convention, you probably know that I had a discussion group on the comparison between CP/M-80, CP/M-85, and CP/M-86 on Saturday afternoon. Since I have to write this column several months in advance of the publication date, I will include some of the key items that we discussed in the next column.

In The Mail

To those of you who have sent letters...my thanks and also my apologies for late replies. I've been living in Texas for several months now, and the mail has had some difficulty catching up with me. Aside from that, I've been doing a lot of traveling, and that hasn't helped my writing schedule either. Although I enjoy the writing, it has to fit into my schedule as a consultant on computer security and disaster recovery. And sometimes that's easier said than done! Aside from this column, the FlipFast books take an incredible amount of research since I make a concerted effort to insure the accuracy and intelligibility of the material. It's not just copied from the existing manuals as many of you have already noticed. I've done extensive testing on all of the commands in each book, and that helps me locate some of the previously undocumented features of the commands.

Some of you have noticed that my columns haven't been quite as regular as you might expect. I keep thinking that things will settle down, but we'll see.

I've also had a couple of questions about hobbies (other than computers!). You may have noticed that I occasionally refer to topics from Star Trek. I've been a science fiction fan for a number of years, even before it was popular with Star Trek and Star Wars. My favorite author is Isaac Asimov. I hope that the good Doctor, if he ever reads this column, notes that I spelled his name properly. As a matter of fact, it was largely due to my reading a number of his books and magazine articles that I started writing. I'm not much of a novelist, but I do have a considerable background in technical and business writing.

Next Month

Since I'll be talking at the HUG Convention in July about CP/M, it seems appropriate to share that information with you in this column. For those of you who attended the Convention, I'll have those comparison listings that we talked about.

Products Reviewed

FlipFast Command Guides	
CP/M-80/85	$12.95
Z-DOS/MS-DOS	
S-A Design Books	
515 W. Lambert, Bldg. E	
Brea, CA 92621-3991	
(714) 529-7999	
Z-DOS CP/EMulator	$20.00
Z-DOS CheapCalc	$20.00
Dungeons & Dragons	$20.00
HUG Bulletin Board Handbook	$ 5.00
SciCalc Scientific Calculator	$20.00
Heath Company Parts Department	
Hilltop Road	
St. Joseph, MI 49085	
(616) 982-3571	
MTERM	$79.95
Micro-Systems Software	
4301-18 Oak Circle	
Boca Raton, FL 33431	
(800) 327-8724	
Sundex Software Corporation	
4755 Walnut St.	
Boulder, CO 80301	
(303) 440-3600	

Heath/ZENITH Users' Group

C

Getting Started
Examples & Functions

New version for MS-DOS includes library with IBM-PC and Z100 graphics for CI-C86 v2 compiler. Structured, indexed collection of over 70 examples and functions for learning the C programming language. Data types, declarations, preprocessor, expressions, looping, relational tests, arrays, pointers, strings, functions, macro definitions, structures, bitwise, I/O, and customization to features of specific computer systems. Library of essential source codes is applicable to most standard C compilers with little or no modifications. Manual includes hints and comments for conversion between popular compilers such as C/80, BDS-C, Aztec, DRC and CI-C86. Available in two versions:

for CP/M (Z90,Z100 - SSDD 5" Disc)	$39.95
for MS-DOS (IBM-PC, Z100 - DSDD 5" Disc) . . .	$39.95

Send check or money order. 6.5% tax for California residents.

EIGENWARE TECHNOLOGIES
13090 La Vista Drive
Saratoga, CA 95070
(408) 867-1184

C and Pascal Compilers
Programming Editors

Write or call for low prices.

An Introduction to Assembly Language

and the 8088 Macro Assembler and ZDOS with the Heath/Zenith 100 Computer

D. C. Shoemaker
HQ USEUCOM BOX 897
APO NY 09128

Prologue

With the arrival of the new 8/16 and 16-bit computers, such as the Heath/Zenith 100 series and the IBM PC, the amount of power available to the small system user took a mighty leap forward. These systems use the Intel 8088 microprocessor, recently renamed the iAPX 88/10, to provide the computing power. No longer confined to the world of 64k of memory or the standard 4 MHZ processor speed, the new generation is fast and provides capabilities only dreamed of in personal computers two or three years ago. Unfortunately, when another microprocessor chip comes along, it brings with it another language to be learned, the assembly language that chip uses. In the real world of applied computer programming, there are two big considerations: do you need a quick solution to a one-time problem, or do you need a "permanent" software solution that will be used repeatedly. If the former is the case, then BASIC is often the best choice. If the latter, then assembly language is usually the answer.

Before you rule out the prospect of writing assembly language programs for your computer, consider that most of the programs you routinely run are written in that language and, like any other, assembly language is learnable, given time and application. The main requirement is the desire to learn, and the chance to get some practice. My theory of learning computer languages probably gives computer science professors nightmares, but I've found the hands-on approach works best. If you have an H/Z-100 computer, follow along and give assembly language a try. Most of what we cover will also apply to the IBM PC and it's software-compatible relatives. We'll look at everything you have to do and know to write your first program, and when you're done you'll have a program in your library that will do something useful and that you can modify to suit future needs.

For this article, I'll assume only that you have a general familiarity with the overall operation of your computer. You should know how to turn it on, insert and boot a disk, make working and back-up copies of disks and, in general, make the computer work. You'll need to know how to use a text editor, either the one that came with your system software or some other. I'll introduce you to everything else you'll need to know to get started.

Before we begin, remember that when you're writing a program, it's impossible to hurt the computer (unless you pick it up and throw it against the wall). The worst that can happen is that the system will crash or hang up somewhere, and you'll have to reboot and begin again. In rare cases, you can inadvertently give the operating system a software instruction that would cause it to overwrite something on the disk. This is one excellent reason for keeping back-up copies of your valuable software.

Throughout this article I'll be referring to sections of the Heath manuals that accompanied my H-120. If you're working with another computer, Heath's new 16-bit microprocessor trainer, the ET-100, for example, you will probably find something similar in your manuals. If you're not already familiar with the documentation that came with your system, now is a good time to learn where the assembly language related information is located. The best computer in the world (yours) is useless without good documentation. It isn't essential that you memorize everything in the manuals, just that you know where to find it when you need it.

Getting Started

Unlike the older Heath H8 computers, the H/Z-100 has no front panel the user can use to "talk" directly to the CPU. And, unlike the H/Z-89 and -90, there is no off-line key to allow the user to send something directly from the keyboard to the terminal. The H/Z-100 is a software-controlled machine, and the user communicates with the computer only by means of software. This is one reason for the great versatility and capability of the computer, especially in areas like redefining the keyboard from QWERTY to DVORAK or assigning new functions to special function keys or generating new character fonts. It can also create difficulties for the first-time user who may not be prepared to take advantage of these capabilities.

Every computer program is born of some problem. In my case, the problem I wanted to solve was to find a simple way to defeat the built-in key click feature of the H/Z-100 keyboard. The ZBASIC program in Listing 1 shows how this can be done in software, by sending the required escape sequence to the computer. This works, but has the drawback of taking several seconds to load the ZBASIC interpreter, load the program and run it, then return to the operating system. Clearly, not a procedure you'd want to go through very many times.

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

Listing 1. Turning off the H/Z-100 key click in ZBASIC.

ZDOS and the 8088

Many H/Z-100 owners may be familiar with 8085 (or 8080) assembly language programming and hopeful that their programming skills and techniques will transfer over to the 8088 system. I was in this category, and did a little research through the Heath manuals. The first Great Truth I found is that the 8088 will not run 8085 code, and

Microsoft's Macro Assembler (MASM) won't assemble 8085 source code. We'll need to write our new program in 8088 assembly language. A lot of unfavorable things have been said about how difficult 8086 and 8088 assembly language is to write, and given the great improvement in microprocessor capabilities, that's not hard to understand. Remember that this is intended as more of an "ice-breaker" than a complete tutorial, and there may be better ways to do what we're about to do.

First, a quick look at the manuals. When my H-120 arrived, it took me quite a while to work my way through the thousand or so pages of documentation that came with the system. When I finally reached Appendix P of Volume II of the ZDOS reference manual, I found something of great usefulness. Appendix P is devoted to "Notes on Writing Z-DOS Programs."

There are two basic approaches to writing 8088 code. One produces what Microsoft and Heath/Zenith call executable programs or code (.EXE files) and the more common .COM files. The main difference is that .EXE files require 512 bytes more memory, but can access more main memory directly, .COM files are shorter, but restricted to the first 64 K of main memory. For our short program this isn't important, so we'll work with the latter. All you need know for now is that there is a difference in programming techniques for the two file types. Heath's Appendix P helps make that clear.

If you look at the program listed on Page P.5 of the Heath manual, you'll see a program that will assemble into a .COM file. This is the one we will use as the basis of our program. For those of you who don't have a manual handy, or if your system isn't a Heath/Zenith, Listing 2 shows the program as I've modified it.

```
        TITLE   KEYOFF.COM          ;A program to set the
                                    ;H/Z-100 key click
                                    ;feature to off.
        PAGE    ,132                ;Set .LST width to 132
                                    ;characters.

        .XLIST                      ;Don't list INCLUDE
                                    ;files.
INCLUDE DEFASCII.ASM                ;Useful definitions that
INCLUDE DEFMS.ASM                   ;we can use.
        .LIST                       ;Turn list on again.

PGMSEG  SEGMENT                     ;What memory segments
                                    ;to be used.

        ASSUME  CS:PGMSEG,SS:PGMSEG,DS:PGMSEG,ES:NOTHING

        ORG     100H                ;Starting memory
                                    ;location of program.

START:
        MOV     DX,OFFSET MESG      ;Get address of control
                                    ;string.
        MOV     AH,DOSF_OUTSTR      ;Get function to output
                                    ;text.
        INT     DOSI_FUNC           ;Print the control
                                    ;string.

        INT     DOSI_TERM           ;Terminate program.

MESG    DB      CC_ESC,'x2','$'     ;Control string to kill
                                    ;key click.

PGMSEG  ENDS
        END     START
```

Listing 2. The basic 8088 assembly language program to turn off the key click.

Look at Listing 2 and note the PAGE ,132 statement. This is called a pseudo instruction. It's not a real 8088 statement (hence the term "pseudo"), but it causes MASM to create a listing of the program that's 132 columns wide. This statement is helpful in tailoring the listing to your printer's characteristics. If you have an H14, an MX-80, or some other printer that prints 80 columns wide, you could change this to PAGE ,80 or leave it out altogether (the default width is 80 characters). The comma after the word PAGE means that we have left out the length of page specification. The default page length is 50 lines.

Next, notice the INCLUDE statement. This is how we refer to other files that contain useful pieces of code we can use without having to type them into our program each time. Aside from the time wasted in retyping commonly used code, there's the risk of introducing fat-finger errors. Also, by using common blocks of code, we can shorten the printed listing of our programs. It's not necessary to list all the INCLUDEd code here, since it's already on your distribution disk. In our example, we have two INCLUDE files, DEFASCII.ASM and DEFMS.ASM. DEF is a name convention that tells you the file is a set of definitions, rather than a block of code that performs some action. The file DEFASCII.ASM contains the code EQUates or equivalences of the ASCII characters we'll use in any program. This saves setting up EQU statements in each program we write. DEFMS.ASM contains the most commonly used "hooks" or system calls to the operating system, ZDOS, itself. These system calls are standard for all computers using ZDOS, and as such, they should be used. A wise programmer refrains from writing non-standard input-output subroutines, for example, since the program may not work with later versions of the operating system. Thus, we don't have to insert code calling for an output routine to communicate with the CRT. If you're familiar with CP/M, it's a little like the SYSCALLs or function calls of that operating system.

The next part of the program reveals the greatest difference between the 8085 and the 8088. The 8088 can directly address vastly more memory than the 8085 can, and it needs a way to manage all that memory. Program segments (PGMSEG) are one way to do this. A short digression to examine the architecture or structure of the 8088 might be useful. Look at Figure 1. It shows how the segment registers are organized.

CS – Code segment
DS – Data segment
SS – Stack segment
ES – Extra segment

Figure 1. The 8088 segment registers.

The ASSUME assembler directive in our program tells the assembler that it can plan to use the code segment, the data segment, and the stack segment, but that we don't plan to use the extra segment. This is a memory directive, and it's used to organize the computer memory. For more on assembler directives, see Page 10.90 in the ZDOS reference manual. For now, just type it in as shown.

The ORG statement does for the 8088 assembler just what it did for the 8085 assembler. It tells the assembler where to locate the start point for the generated code when it's ready to be executed. In this case, it's to start (ORiGinated) at memory location 100 hex, or 256 decimal. This is the normal starting location for user programs like ours, but special-purpose programs, such as debuggers, may start elsewhere in memory.

START: is another assembler directive. It tells the assembler that the program source code begins here. Our program STARTs with two MOV instructions which tell the computer to MOVe something into one of the CPU registers. Here it's telling the computer first to MOVe into register DX, the address of the escape sequence we want printed, and then MOVe one of ZDOS's function calls into register AH. Figure 2 may help to clarify this.

AH	AL
BH	BL
CH	CL
DH	DL

Figure 2. The 8088 general registers.

Figure 2 shows the layout of the general registers the 8088 uses to keep track of data, instructions, and addresses. AH and AL stand for the high and low byte of the A register. The first MOV statement loads, into the DX register, the two-byte address of the memory location where the program stored the escape sequence we want to send to the computer. The escape sequence itself doesn't move anywhere. The next MOV calls the ZDOS routine DOD—OUTSTR that prints data at the terminal. This is very much like a CP/M system call (SYSCALL) or function call in that it represents a frequently used subroutine that we don't have to invent and write out each time we want to send something to the terminal. Next, the INTerrupt command tells the computer to execute the DOS—OUTSTR function. Finally, the last INTerrupt command ends the program by returning to ZDOS.

Next comes the actual message we want to send to the computer. This is the character string ESCape x2 that turns off the terminal key click. The CC—ESC is the assembly language expression for the ESCape character, and the 'x2' is the actual command. The '$' marks the end of the character string for the assembler. The actual contents of this string will depend on the commands your CRT or terminal respond to.

At the end of the program comes two lines to tell the assembler that this is the end of the ProGraM SEGment and the end of the program.

If you're familiar with CP/M-85 or CP/M 2.2, this example used the equivalent of the CP/M function 9, the character string output to the console.

Assembling the Program

To begin, type in the program as it appears in Listing 2. You can use the editor that comes with ZDOS, or some other. After typing the program, save it as a disk file called KEYOFF.ASM.

With the program saved to disk, here are the steps to take to assemble it. First, call the assembler by typing the following:

```
MASM KEYOFF <return>
```

where <return> means press the RETURN key.

After some disk activity, you should see a message on your screen that looks like Figure 3.

```
The Microsoft MACRO Assembler
Version 1.07, Copyright (C) Microsoft Inc. 1981,82

Object filename [KEYOFF.OBJ]:    <return>
Source listing  [NUL.LST]:       <return>
Cross reference [NUL.CRF]:       <return>

Warning Severe
Errors  Errors
0       0
```

Figure 3. Screen output for MASM. Press the RETURN key in response to each of the assembler's questions. If we had given other names to the .LST and .CRF files, we would have created those files on the disk.

If you get an error message, it means you typed something wrong. Examine the error message; sometimes you can tell from it what the trouble is. Often it's a case of rechecking to see that you typed what you meant to. Call the editor, review the program, locate the problem, correct it, resave the program on the disk and repeat the assembly.

Now we'll link the object code produced by the assembler. Since the object code produced by the assembler is relocatable, the linking process resolves all the memory location references and "fixes" the program to run in a certain location. Here's where those segment registers are important. To link the object file, type the following command:

```
LINK KEYOFF <return>
```

Look at the output from LINK, in Figure 4. The error message is the result of the fact that the object code module doesn't contain a statement allocating stack space in memory, but we used the stack definition in the ASSUME directive. For some programs this could be serious, but in this one we can ignore it.

```
Microsoft Object Linker V1.10
(C) Copyright 1981 by Microsoft, Inc.

Run File [KEYOFF.EXE]:   <return>
List File [NUL.MAP]:     <return>
Libraries [.LIB]:        <return>
Warning: No STACK segment

There was 1 error detected.
```

Figure 4. Screen output for LINK. Press RETURN in answer to each of the options offered. We could have created the optional files by giving them other names than the ones in brackets. We would identify any libraries we wanted to link to our program at this point. See the text for the explanation of the error.

Now we have a linked code file in an .EXE format, but we still have to convert the .EXE file to a .COM file. Like CP/M, ZDOS recognizes .COM files as immediately executable programs that reside in, what CP/M calls, the transient program area, or TPA. To convert our program, type this command line:

```
EXE2BIN KEYOFF.EXE.COM <return>
```

Once this is completed, you will have the following four new files on your disk:

KEYOFF.ASM - the original source code file.
KEYOFF.OBJ - the object code file produced by MASM.
KEYOFF.EXE - the result of the LINK operation.
KEYOFF.COM - the executable file produced by running EXE2BIN.

We can test the results by typing KEYOFF. There will be some disk activity and the ZDOS prompt should drop down a line. When you press a key, the key click will be silenced. Success!

After testing the program, the .OBJ and the .EXE files can be deleted from your disk. If you leave them on your disk and you have to reassemble the program to correct an error, MASM will overwrite these two files with the up-dated versions. You no longer need them once you have the .COM file, and you may want the disk space later. Save the KEYOFF.ASM file, however, as there are other things you can do with it by substituting other escape sequences in the line to be sent out to the computer. For instance, by changing the output address, printer set-up control sequences can be sent prior to your printing a document.

We could have generated some other files as a result of the assembly. You have the option of producing a listing of the assembled program (a .LST file) that can be useful for debugging and for retaining a record of the program. This listing contains the hexidecimal equivalents of the machine-language code produced by the assembler, as well as a copy of the original source code statements. This lets you compare the assembler output with the source code input. You can also generate a cross-reference file (a .CRF file) that can be used with the ZDOS CREF utility to produce a sorted cross-reference listing of all the variables you used and where they appeared in the program. It also shows all the system calls used. This is useful for debugging, too, since it gives you all the references to the other files used by the INCLUDE statement. The contents of these files aren't listed here because they're fairly long, but I encourage you to re-run the assembly procedure and generate all the optional files so you can see what they look like.

AUTOEXEC

There is a ZDOS utility that will let you identify a program, like KEYOFF, and run it whenever you first boot the system, without your having to type anything. It can process almost any instruction that it finds in a special file called AUTOEXEC.BAT. This is a "batch file" that can contain any command that you could type in from the keyboard. This means that if we created an AUTOEXEC.BAT file that contained the command KEYOFF, ZDOS would run the KEYOFF program every time you boot the system. To do this, use your editor to create a one-line text file containing just the word KEYOFF. Save this file under the name AUTOEXEC.BAT. Now reboot your computer. When you press any key after the boot sequence is complete, the key click will be off.

This has been a simple example of how to create an assembly language program that does something useful that would be cumbersome or difficult to do any other way. This program could be made to do many other things by sending different escape sequences to the computer. After going through this procedure, assembly language programming on the H/Z-100 computers, under ZDOS, shouldn't seem such a formidable task.

References

Those of you who are interested in further reading on this topic may want to refer to the following books. These aren't in any special order, and do not reflect my judgement of their usefulness, but I felt that all were worth the cost.

"8086/8088 16-Bit Microprocessor Primer"
Christopher L. Morgan and Mitchell Waite.
McGraw-Hill, 1982. About $16.00

"The 8086 Book (Includes the 8088)"
Russell Rector and George Alexy.
Osborne/McGraw-Hill, 1980. About $18.00

"Advanced Microprocessors, an Individual Learning Program"
Model EE-8088, the Heath Company, 1983. About $100.00.
(Note: While intended as a programmed instruction course to accompany Heath's 16-bit trainer kit, the ET-100, which is a "big brother" to the ET-3400, the course itself is an excellent introduction to 16-bit programming, in general, and the 8088, in particular.)

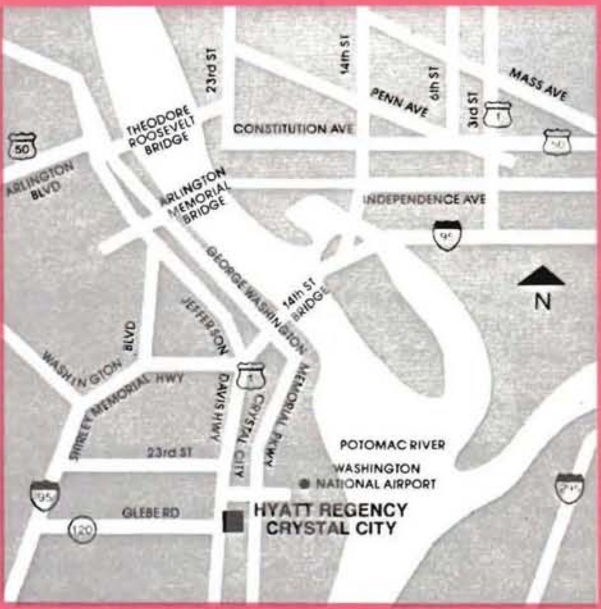
About the Author:

***D. C. Shoemaker** has worked with computers since 1977, when the first Heath H8 kits narrowly edged out the TRS-80 as first on the market in the Seattle area. Specializing in microcomputer applications, he teaches computer science for the City Colleges of Chicago in the Stuttgart, Germany area.*

CHUGCON 84

THIRD ANNUAL
CAPITAL
HEATH USER'S GROUP
REGIONAL CONFERENCE

NOVEMBER 3, 1984



ALL DAY CONFERENCE FOR

ZENITH COMPATIBLE PEOPLE *

Crystal City Hyatt Regency, Arlington, VA 22202

This conference is free and open to the public. Registration begins at 8:00 AM on the second concourse level of the Hyatt Regency.

SOFTWARE AND HARDWARE VENDORS

More than 5800 sq ft of carpeted display area for vendors! There is no display fee! Space allocated on a first come, first serve basis.

TUTORIALS AND DEMONSTRATIONS

Sessions on a wide variety of topics will be offered on all levels — novice to advanced.

FANTASTIC SWAP MEET

Trade or auction your unwanted hardware, peripherals & supplies (Table space must be reserved). Need parts? Bring a list of what you need.

FOR MORE INFORMATION CONTACT

CHUGCON 84	Mike Supley	Bruce Gewirz
JANR Technical Services	>CHUG President	Swap Meet Chairman
P. O. Box 10515	(703) 435 1189	6029 Sligo Parkway
Alexandria, VA 22310	7PM - 9PM EDST	Hyattsville, MD 20782

or check the >CHUG BBS (703) 360 3812

* We support H8, H289, S100 buss and IBM PC buss systems.

ZBREF and BRNCHREF Patches

Pat Swayne
Software Engineer

In the first 200 or so copies of the MS-DOS Utilities II disk (885-3014-37) that were sold, there are problems in the ZBREF and BRNCHREF programs. The programs can be corrected with patches by following the instructions below.

ZBREF Patch

The ZBREF program will hang up if you try to send its output to a printer. To correct the problem, use DEBUG to install a patch as shown here (what you type is shown in bold print).

```
-NB:ZBREF.COM
-L
-E6CC
xxxx:06CC 80.90
-W
Writing 0F55 bytes
-Q
```

The above example assumes that you have DEBUG on drive A:, and ZBREF.COM on drive B:. The xxxx designates a number that will be different for different systems. The prompt displayed by DEBUG will be a '>' character if you are using the MS-DOS/Z-DOS 1.x version. If the current value at the location 6CC is not 80, do not make the patch. You have a corrected version. If you want to patch the source code, find the line

HDR3 DB '.BAS',EOL

and change it to

HDR3 DB '.BAS'

BRNCHREF Patch

The BRHCHREF program has an instruction that was left in by mistake when some changes were made. That instruction causes some Z-150 codes to be executed when the program is run on a Z-100, resulting in some WILD INTERRUPT messages. To fix BRNCHREF, use DEBUG to make this patch.

```
-NB:BRNCHREF.COM
-L
-E0A4B
xxxx:0A4B E8.90— DD.90— 04.90
-W
Writing 06D0 bytes
-Q
```

Where you see the underline symbol in the example above (—), you should type a space. If the original values at the patch addresses are not E8, DD, and 04 as shown above, do not make the patch. You have a corrected version. To patch the source code, locate the comment PRINT INTRODUCTION and remove the first instruction below it (CALL TYPTX).

A Printer For All Reasons

Are you looking for an alternative to that gleaming Diablo 630 in your dealer's showroom?

This one prints and types, and it won't give you sticker shock.

George Istvan
3187 Reva Drive
Concord, CA 94519

Introduction

The Brother HR-15 is typical of the new generation of excellent, letter quality printers offering cost-effective, proportionally spaced copy, although at a relatively slow rate of print. The one factor, however, which distinguishes the HR-15 from other printers in its class is its optional, detachable keyboard that allows it to be used as a typewriter. "So what," you may say. "Typewriters with interfaces have been around for years." Yes, but that is only part of the story. A typewriter has a design speed appropriate for typists. When you install an interface and connect it to a computer, it is now moving twice as fast, and mechanical problems have occurred in some models as a result. The HR-15 is a printer, not a typewriter. It was designed from the ground up for the heavier demands of computer printing. More about the keyboard later.

Specification Summary

Throughput:
11-13 Characters/second.

Price:
$540 serial, $510 parallel; keyboard $175;
Orange-Micro, San Francisco, (415)673-0170.

Characters/line:
110 @ 10 pitch, 132 @ 12 pitch, 165 @ 15 pitch;
proportional spacing is about 12 char./inch.

Buffer:
3K standard, 5k optional.

Printwheel:
96 characters (10 million character life span).

Dimensions:
18-1/4" wide, 13-3/8" deep, 6-1/2" high.

Repairs:
Ship unit to regional repair facility; 90 day warranty.

Comments:
Pitch and spacing are switch or software selectable.
Print head is bidirectional and logic seeking.
Maximum paper width is 13-1/2" although the print head spans 11" (110 char. @ 10 char./inch).
Copy mode will make an original plus four copies.

Interfacing and Getting Started

My H-89 has the HA-88-3 serial interface board, and I had to specify the RS-232C version of the printer at the time of purchase. Inmac of Sunnyvale, Ca., (408) 737-7777, provided me with 10 conductor, shielded cable and two DB-25 connectors (male) with the solder-type pins. I wired pins 1, 2, 3, 4, 5, 6, 7, 8, 11, and 20 straight through to the identical pins of the connector at the other end. The remaining 15 pins are not used. You should solder the shield lead to pin 1, the chassis ground, at both ends of your cable. Incidentally, I first tried a 1,3,4,7-pin cable, which works on my Epson printer and some other models, but it didn't work here.

The inevitable DIP switches are located on the rear panel, exposed for easy access. I have never had to unloosen any screws or undo panel covers to set up or use the printer. Most DIP settings are obvious from the manual, but I shall mention that I have data length set to "8 bits" and parity set to "odd". The DIP switches also determine baud rate. I tried 4800 baud and the machine produced a printout that might have resembled one of Poe's ciphers for buried treasure. I run mine at 300 simply because it works there, and this was also the factory setting. At 600 baud a few mistakes occurred, and I suspect that 600 may be an error threshold for this particular system.

CP/M users must invoke the Configure utility to complete the interfacing. Call up submenu A and set the handshaking (item M) to "high" and the signal polarity (item N) to "RTS". Then use item C, also of submenu A, to choose your I/O port and your I/O baud rate. I operate my HR-15 from I/O port 0d0H (320Q) and my Epson printer from 0e0H (340Q), a setup which requires a separate disk configured for each printer. With the two printer system, remember to do a reset and a cold boot if you decide to go from one printer to the other. If you switch disks and just do a warm boot, you'll find that the disk which worked perfectly yesterday will only produce silence, interrupted occasionally by the sound of someone cursing. The data in RAM will reconfigure the new disk so that it is no longer compatible with the other printer; you'll then have to run CONFIGUR.COM all over again. I'll have more to say about the economics of the two-printer system in the summary.

Using the HR-15

Well, just how fast does it print? On a two page test printout of a Magic Wand file, which was double-spaced, right-justified, and proportionally-spaced, I clocked mine at 9.8 characters/second. Not much danger of the paper smoking, but then you have to ask yourself just what combination of features you want, and at what price. Brother claims a maximum throughput of 13 CPS, but I believe my test is more accurate from a practical point of view. Proportional spacing, right justification, and double spacing will slow down any printer, but these are features which a person would normally use on this type of printer.

Line spacing and pitch are software or switch selectable. These control switches are readily accessible on the front of the console, and one may select line spacing in single, 1-1/2, and double spacing and pitch in 10, 12, and 15 characters/inch. You are also able to choose proportional spacing from these panel switches whether or not your word processor is equipped with this feature; the printout looks best, however, coming from the word processor. The standard printwheels will do proportional spacing, but for best results you ought to purchase the special "Anelia P.S." printwheel. You should customize your word processor to the appropriate protocol, and the HR-15 follows either the Diablo 630 or 1610. Magic Wand users must choose item #2 (Diablo 630) of the CHANGE.COM menu since the PRNT1610.HEX file is not included on the distribution disk.

The HR-15 will underline, boldface, auto strike-out, and shadow print, and it will even print in red if you get tired of looking at black. These functions are actuated by ASCII ESC codes. For instance, ESC E initiates underlining while ESC R terminates it. Continuing our Magic Wand-CP/M example, we would imbed OUT commands in the text to accomplish these operations, with \OUT 27,69\ activating the underlining and \OUT 27,82\ stopping it. The numbers are the decimal equivalents, respectively, of these ASCII codes. The printer will not do italics from the word processor, although I shall describe how this may be accomplished using the keyboard shortly.

Brother offers 143 different printwheels in 13 languages, including one in Greek which could be used for technical and mathematical work. The bad news is that each costs about $24. I checked very carefully to see if these daisy wheels were made of gold, but they're space-age plastic. Four types of cassette ribbon are also available, and the multi-strike carbon at $6 is the best buy. Daisy wheels and ribbons can be changed in seconds. Express Computer Supplies, San Francisco, (415) 864-4949, is an excellent source for these accessories, including correction tape for typing from the keyboard.

The printer has an optional single-sheet feeder at $260 and a tractor feed at $140. The standard HR-15 is friction feed only, and I have found that you can use continuous sheet without the tractor if you are careful when you first insert the paper. Another technique I use for long manuscripts is to employ the FORMS command of Magic Wand; I readjust the paper as necessary when the machine pauses after completing each page.

The HR-15 has a clone: the DX-15 by Dynax. The units are virtually identical, and Brother is the parent company. If you have to print 132 columns of data you might consider the HR-25, the HR-15's "Big Brother". It takes the wide paper and prints faster than the HR-15, although it costs about $900.

Using the Optional Keyboard

The keyboard instruction manual, like that for the printer, is adequate and concise. For $175 the addition of the keyboard creates a full featured typewriter. The printer does not, however, become a terminal. The design of the keyboard is well thought out and shows attention to ergonomics. It may be used flat and at a 10 or 15 degree tilt. The face of the keyboard is concave, with each row of keys at a slightly higher angle than the row below it; this property effectively decreases the distance your fingers have to move to find keys on different rows. The keys have a pleasant touch also. My H-89's keys always feel "broken" to me, as if they aren't connected to anything, but these keys have just the right amount of resistance. You may select 10, 12, or 15 pitch and single, 1-1/2, or double line spacing. Alas, you cannot type in proportional spacing mode. You may also set tabs and margins, move forward or backward by full or half spaces, and move up or down by full or half spaces. A repeat key is present as well as an auto-repeat for the various space keys.

Several of the keys have international symbols engraved on them in addition to the English, ASCII notation. With the appropriate printwheel installed and a quick change in DIP switch configuration, one may create umlauts, accents, and other foreign language figures. A single switch on the keyboard will select ASCII, International, or a third position called "Symbol", which refers to the Greek lettering that scientists or fraternity types would use.

What if you make a mistake while you're typing? The print head assembly takes a standard size roll of correction tape, which operates under keyboard control just like the real typewriters. A full-line capacity memory makes it easy: just type out the error and retype the correct letter. The correction tape will work only with specific ribbons. I use the single-strike carbon (correctable) ribbon when I am typing, and the multi-strike ribbon at all other times. The printer comes from the factory with a roll of red ribbon installed where the roll of correction tape would go; only one or the other can be in place at one time. Thus you may print in black or red under software control, but you may type only in black as well as use the correction mechanism.

How about italics? First, you must insert blank spaces in your edited file corresponding to the word or phrase to be italicized. If you are using proportional spacing with Magic Wand, you would use the HS command to create full size blanks. Then, after the machine has printed out the entire file, reinsert each sheet singly, substitute the special italics printwheel, fire up the keyboard, and type away. The paper guide has a red line which permits you to position the page precisely in the printer. You can dub in superscripts and subscripts similarly, using the half- space up or down keys to type over imbedded blanks; some word processors like Magic Wand have a special command for this purpose which works nicely.

I thought there were a few minor inconveniences: first, with the keyboard in place, the distance from the keys to the print head is longer than for a regular typewriter. I am told that, like a pianist, one is not supposed to peek at the keys, but those of us who are not professional typists may find ourselves making long head movements from the keys to the paper to see what's going on up there. If you expect to do a lot typing with your system, you might want to spend some time with your dealer's machine before buying. My other objection was the single, platen control knob on the left; perhaps one on each side or a single knob on the right would have been better.

Summary

The HR-15 is an excellent, cost-effective, letter quality printer, which offers many desirable word processing features including proportional spacing. The optional keyboard enhances the printout and provides a complete typewriter capability.

If you already have a good typewriter, the HR-15 alone would make a fine printer, assuming you can find something to do while you wait patiently for it to finish printing. Here's your chance to read War and Peace like you always promised yourself you would.

I would like to thank Gayla Newsome of Orange-Micro, San Francisco, for her courteous assistance with specific aspects of HR-15 operation. For questions you can reach me at (415) 687- 8913, Concord, CA.

(LISP) for H89

Artificial Intelligence Language
UO-LISP Programming Environment
The Powerful Implementation of LISP for MICRO COMPUTERS
Excellent for developing A.I., ROBOTIC, EXPERT and INTELLIGENT SYSTEMS

The usual **LISP** Interpreter Functions Data Types, Structure Editor, Screen Editor with Mouse Support, Optimizing **LISP** Source Code Compiler & Assembler, Assembly & **LISP** Code Intermixing, Compiled Code Library Loader, Numerous Utility Packages, Hardware and Operating System Access, Session Freeze and Restart, Comprehensive 350 page Manual with Usage Examples, and much more is available with the ADVANCED INTERPRETER, your environment may cause constraints, review catalog.

Other **UO-LISP** products include **META** a translator writing system, **RLISP** a high level language, and **LISPTEX** a text formatter.

!!NEW!! LEARN **LISP** FOR $39.95

The "LEARN LISP" Interpreter includes a subset of the above description, manual includes tutorial.

The **UO-LISP** Programming Environment runs on the **H89** with CPM.

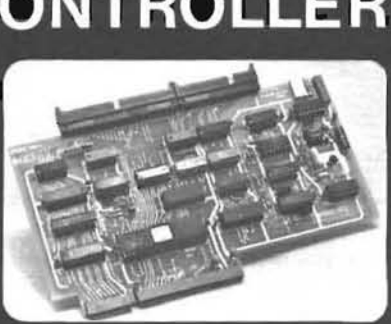
Package Prices range from $39.95 to $300.00.
Manuals may be purchased separately.

TO ORDER: First send for your FREE catalog which lists technical details, distribution formats, and ORDER FORMS.

VISA and Mastercard accepted.

Northwest Computer Algorithms
P.O. Box 90995, Long Beach, CA 90809 (213) 426-1893

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.

PRICED AT $395

Includes controller board CP/M boot prom, I/O decoder prom, hardware/software manuals BIOS source listing.
HDOS driver now available for $50.00.

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

ANNOUNCING THE MODIFIER

A disk utility that modifies the CP/M BIOS to be able to read and write to a number of 5.25" CP/M disk types.

There is a growing need for the everyday user of computer systems to be able to take data files home from the office to continue to work on them. The computers at home and at work may both run a version of CP/M, but the disk structures may be incompatible. This is especially a problem in the 5.25" world. MODIFY 89 was designed to address this problem. MODIFY 89 makes the CP/M operating system access a specified 5.25" drive as one of the below disk types.
Disks placed in that drive that are of the specified type can be used as if they were one of the standard disk types accepted by the H8, H/Z89 or H/Z90 computers. Thus PIP, STAT, DIR and others will work for that disk also. The price for MODIFY 89 is $49.95.

MODIFY 89 is set for the following disk types:

- Access S.S. D.D.
- Cromemco S.S. D.D.
- DEC VT180 S.S. D.D.
- IBM PC/Zenith 100 (CP/M) S.S. D.D.
- IBM PC/Zenith 100 (CP/M) D.S. D.D.*
- Kaypro II S.S. D.D.
- Morrow Micro Decisions S.S. D.D.
- NEC PC-8001A S.S. D.D.
- Osborne S.S. S.D.
- Osborne S.S. D.D.
- Otrona D.S. D.D.*
- Superbrain Jr. S.S. D.D.
- TI Professional S.S. D.D.
- TRS-80 Model I (Omnikron CP/M)
- TRS-80 Model III (MM CP/M)
- Xerox 820 S.S. S.D.
- Xerox 820-II S.S. D.D.
- Standard (Tests for H/Z37 and C.D.R. Disk types)

* = Double sided 5.25" drive required

S.S. = single sided, D.S. = double sided, S.D. = single density, D.D. = double density

Limitations: MODIFY 89 is not a disk duplicate program. It is currently available for use with an H/Z89 or H/Z90 computer that has an FDC-880H double density 8" and 5.25" controller, using C.D.R.'s BIOS V.2.9 or with an H8 computer using the FDC-H8 by C.D.R. Systems, Inc.

MODIFY 100 will soon be released for the Z100 line of computers at a price of $75.00.

Contact:
C.D.R. Systems, Inc.
7210 Clairemont Mesa Blvd., San Diego CA 92111
Telephone: (619) 560-1272
Or a C.D.R. Systems, Inc. dealer near you.

Heath Users' Group 3rd Annual International Conference

Tom Huber
Related Products Editor

Introduction

Take a midwestern town, surrounded by corn fields, add one luxurious resort complex of 550 rooms, two swimming pools, an 18- hole championship golf course, small restaurants and shops that have the flavor of New Orleans, and over 1300 Huggies, and what do you get? Slightly inadequate facilities for the 1984 International HUG conference.

Needless-to-say, this year's conference was another in a series of successful conferences, topped by a cocktail party and awards, the likes of which haven't been previously seen with six big prizes from Heath Company and over 75 prizes from the attending vendors.

In addition, John Guenther from the Air Force, presented the reasons behind the selection of the Z-100 in last fall's contract award to Zenith Data Systems. Vendors and manufacturers were again very much present with over 45 present and accounted for. Software most present were word processors and communications packages for the Heath/Zenith line of computers.

Sessions

The sessions were well attended, to the point of having to turn away many at some of the more popular sessions. Most notably were those sessions Saturday morning before the exhibit area opened for the day. Out of respect for those attending the crowded sessions, I chatted briefly with some of the speakers afterward and did not attend nearly as many sessions as I would have liked. Incidentally, Bob and Walt promised me we'd have more room next year.

One of the most exciting sessions was put on by Bill Parrott in the spacious Governor's Hall (the only room really big enough) who, along with two others, is under contract with Heath Company (not ZDS), to produce H-DOS 3.0. He was allowed this year to present the details of this long awaited operating system for the H-8, H/Z-89/90, and the H/Z-100.

It is an ORG 0-based system, meaning that H-8 and '89/90 owners will have to have the ORG-0 modification. Mounting and dismounting disks is no longer mandatory, although if you mount a disk as before, you will have to remember to dismount it. Nearly 100% of existing H-DOS software is compatible, except for those packages that go in and mess around with the operating system itself. Among the new goodies are a RAM disk device driver allowing owners of H/Z-100 and systems that go beyond the normal 64K limitation in the H-8 and '89's to utilize that extra memory. There are universal device drivers, including a parallel one, where you can set your own parameters. All Heath/Zenith devices are supported, including the Winchester systems (Z-67 and Z-217). Benton Harbor BASIC will be included with the system, but it is not known at this time if the Z-100 version will include graphics statements, ala Z-BASIC.

No release date was given, nor speculated upon, as the software is not yet finished. However, the documentation is being developed as the software is being written; it is not being written by Heath or Zenith. Since Heath did not want to spend the time rewriting the existing documentation, the documentation for the new version will be an addendum, which should help speed the package through Heath, once the tested code has been released. In my view, this is probably a wise move, since software documentation from ZDS has not always been timely.

Tom Dornback, who followed Bill Parrott in the Governor's hall, for those of you who are not familiar with the ZDS organization, is a key individual in the development and support of new and existing products. He is a Vice President of ZDS and is in charge of software development. While his talk was obviously of the canned "corporate" variety, he provided a number of interesting revelations, including what I consider to be the foremost comment, those that indicated continued support for the Z-100 in terms of new software. Part of this continued support has to be credited to the purchase of this computer by the Air Force, Navy, and Marines under a three-year contract that is now in its eighth month. Toward the end of his talk, Mr. Dornback made some predictions: higher processor speeds, greater floppy disk storage, more internal RAM capacity, increased size requirements in future software packages, more multifunction, multiuser, and multitasking equipment, and an increased importance placed on ergonomics, the appearance of the hardware and software. Tom also introduced the Zenith Data Systems Software Submissions Program.

Following Mr. Dornback was a surprise speaker, John Guenther from the Air Force. He spoke on why the Air Force chose the Z-100 as the small computer in their contract. Here are the highlights of his address from his transcript, which he graciously provided to this reporter.

"The Air Force standard microcomputer contract started back in 1982, when it became obvious that separate Air Force major air commands and individual bases were using microcomputers. There was a lot of development being done throughout the Air Force on a multitude of different micros... The acquisition (purchase) of small computers at that time was not producing the best possible equipment, the best configurations, the best prices for the government, the best support, nor standard software. Although this work was beneficial in a narrow scope, it only benefitted small segments of the Air Force, and in each case, required full compliance with government procurement procedures.

"...A requirements contract is an open contract for a specified number of systems, a specified period of time, and a limited amount of money. Each system purchased must be justified through normal ADPE justification processes... This requirements contract is a joint project between the Air Force and the Navy with the basic intent being to lessen acquisition time and produce the best prices and performance for the government. Another major benefit is that we now have available a standard system throughout the Air Force and Navy, with standard operating systems, CP/M and MS-DOS, along with the much needed software tools...

"The process used in selecting the Air Force Standard Micro was more than a normal low bid process. The first step was a survey of all potential Air Force/Navy users, to determine the needs of the end users. After the survey was returned, and all the responses analyzed, the most desirable items were placed in a request for information, which was sent out to industry. The request for information was used to determine if the needs...were available as standard off-the-shelf commercial products. At the close of the request for information, all of the information was evaluated for compliance with Air Force/Navy needs. The request for proposal was then generated and sent out to industry for proposal submission.

"The basic system requirements for the government were: an 8-bit processor running standard CP/M software with at least 64K of memory; floppy disk storage of 320K, and a video display. We also wanted a 16-bit upgrade capability with the MS-DOS operating system and a minimum of 128K of memory. There was also a requirement for dot matrix printers with graphics capability, a letter quality printer, a plotter with at least six pens, both acoustic and direct-connect modems, memory upgrades, and a Winchester drive with a minimum of 5 megabytes of storage. All of this had to be compatible with both operating systems and all software provided.

"When all of the proposals were received, we used a technical evaluation method to determine the best machine for the money. Each item in the request for proposal was assigned a points value based on the overall importance of the item (to the needs of the Air Force and Navy functional users). After the close of the request for proposal, all vendor proposals were evaluated by a joint Air Force/Navy team of technical experts along with procurement officers from the Air Force Computer Acquisition Center. Each proposal was evaluated on the vendor's ability to meet or exceed each requirement in the request for proposal. Points earned during this technical evaluation were then divided by the total contract life cycle cost for each vendor, to produce the apparent winner based on highest technical points for the lowest cost figure. This does not necessarily produce the lowest cost for the government, but does insure that what is purchased best meets the needs of the government at the most advantageous price.

"After an apparent winner has been determined, that vendor is required to provide every item that would be on the contract for functional test demonstration to the Air Force and Navy... The testing performed during the functional test demonstration is actually conducted by government personnel with the assistance of the vendor's personnel. Each item of hardware and software was tested in all possible configurations...to insure that it performs up to the capabilities offered by the vendor and required by the government..."

"Why was Zenith chosen? That is simple to answer: they had the highest technical points for the lowest dollars per point. Zenith also met, or exceeded, all of the requirements of the request for proposal and in many cases offered technically superior hardware and software. During the functional test demonstration, the Zenith Z-120 performed almost flawlessly with only minor software problems, which were corrected on the spot (or within a couple of days)."

Before he addressed the group, I chatted with John and he mentioned several other interesting points. Zenith was not the number one choice coming out of the request for proposal; they were number two. What put the ZDS team ahead of the number one contender was the technical support rendered during the functional test demonstration. The number one contender sent hand packed equipment with five marketing personnel (salespersons) and one technician. Zenith sent one marketing person and five technicians (engineers). Our equipment was provided virtually off-the-shelf and in multiple units, whereas the other guys provided one unit for each item of the request for proposal.

Future products, according to John Guenther, include additional requirements for computers used in security work, "a portable computer, and a family of multiuser small computers with upward compatibility throughout the family. All of these small computers must be at least data compatible and preferrably run much of the same software as we currently have with the Z-100."

The one thing I noticed throughout Mr. Guenther's address was the absolute attention paid by those in attendance. Certainly, much of this attention has to do with the continuing success of the Z- 100 family of computers. I also believe that Zenith and Heath are certainly in a good position to receive a good portion, if not all, of the future contracts to be awarded. Additional support has come from Southwestern Michigan's congressman, Mark Siljander.

Other addresses given during the conference included two by Barry Watzman for advanced programmers on Multitasking operating systems and 16-bit assembly language for 8-bit programmers. Barry is the former product line manager for Heath and ZDS computer products and was the moving force behind the H/Z-100 computer series. Since leaving Heath/ZDS, Barry has formed his own software development and consultation company.

Pat Swayne, well-known for his programming efforts in the HUG organization, presented three sessions on operating system extensions. Walt Bilofsky, president of Software Toolworks, and recognized at this conference for being the first outside vendor to support Heath computers, presented two sessions on the C Language. Brian Barnes, a systems and language programmer for ZDS, presented a session on MS-DOS. Susan Hayes, also of Software Toolworks, presented two sessions for the gals (and guys so inclined) on the computer chef. Bruce Denton, president and designer for D-G electronics, presented a session on H-8 and H-89 hardware. Ron Hackney, a marketing support engineer for ZDS, presented three sessions for beginners in introduction to computers. Dale Wilson, creator of Palette, reviewed in the August issue of REMark, presented three sessions on Z-100 graphics. Bob Todd, disk distribution coordinator for the SIG/M and PC/BLUE public domain software libraries, presented a session (packed to the limits of the room, unfortunately, because I couldn't attend) on public domain software; he also presented a session on hardware servicing. John Hubbard, a senior designer for Heath's Educational group, presented a session on robotics, another session that was limited by the room size. Dysan, one of the big names in diskettes had one of their representatives present a discussion on disk care. In addition, Dysan had a booth exhibiting their Interrogator diagnostic disk package for the H/Z-100 computers, introduced at this conference.

Bill Adney, author of numerous articles for REMark and several books, discussed the differences between CP/M for 8-bit computers and CP/M-86, the 16-bit operating system from Digital Research. Mark Foster, senior systems engineer for ZDS, in addition to appearing on the Software/Hardware panel, presented a session on the Z-150 hardware. Finally, Mike Cogswell, from the Capitol Heath User's Group, presented a paper on the H-100 hardware. Two other sessions, one panel, chaired by Skip Gwyer and four of his software consultants from ZDS, fielded numerous questions on the software sold by Heath and Zenith. The other panel, already mentioned in introducing Mark Foster, dealt with both the software and hardware of the Heath and Zenith computer products.

In the Exhibition Hall

Actually, there were two halls set up next to each other for the exhibitors this year. As usual, the vendors were friendly, courteous, and willing to show, explain, and in many cases, sell their wares. As was the case last year, there were too many products to be able to cover all of them, so this section will be limited to some of those vendors that were offering new, improved, or different products. Of course, you can bet that I missed somebody in this review, so please forgive any oversight I may have given you or your products.

CDR Systems -- Want to speed up your Z-100 without having to do any soldering? Now you can do it. Or at least, that was the message from CDR systems, who were showing off the ZS100 speed module, a plug-in board for the Z-100 series. With the exception of some 8088's in a limited number of Z-100's, the clock speed of the Z-100 is increased by 50% (switchable), so that non- input/output functions speed up considerably. For most folks, the increase in speed is particularly noticeable in Multiplan, Lotus, WordStar and other operations where internal calculations take up some real time. The unit plugs into U236 on the main board of the Z-100, so no soldering of the four-layer main board is needed. Furthermore, should your 8088 (or some of your other IC's) be one of those that cannot cope with the faster response times, the switch that installs in any spare DB-25 slot on the back panel can be set to S (for slow) until you can get a new, faster 8088. The price is reasonable ($49.95) and a real bargain for the increased performance.

I mentioned Dysan as conducting one of the sessions. During a lull in the proceedings, I stopped by the Dysan booth and asked them what brought them to this conference. I was told that they felt that they would receive better response at this type of conference than at a more general show. And response they were getting! While the H/Z-100 version of the Interrogator Drive Diagnostic Program will just be getting to some of the regional offices, you should be able to see the program in operation within a month or so of receipt of this issue. By calling the toll-free number (800) 551-9000, you will either be able to order the package directly or obtain the location of the nearest Dysan branch for a demonstration. It is interesting to note that Dysan has also selected the Z-100 as a prime development tool for their systems. In addition to the Interrogator at $139.00, the digital diagnostic disks are available for $40 (double-sided) or $30 (single-sided) in either the 8- or 5.25-inch formats (either 48 or 96 tpi). The interrogator and diagnostic disks are designed to offer two-stages of service: determination of the disk-related problem (where it is at), and the signals necessary for servicing and repairing the drives. In addition, the program is designed to be used as a preventive maintenance tool, allowing diagnostics of disk drives that are not within specifications and through timely adjustment, preventing major problems in magnetic media storage before it happens.

Of the word or text processing packages offered at the show, two caught my eye. The first, from S & K Technology is WatchWord 2. This package will eventually serve as the heart of a typesetting system and is well worth exploring. The other, which allows almost any code to be placed into the text area (including escape codes) is TXTPRO from SoftShop.

Cleveland Codonics, those folks who have brought you high resolution graphics to the H/Z-19 and H/Z-29 have improved on their work. The new H/Z-29 upgrade features a 1K by 1K memory plane with a window of 672 by 500 pixels. In addition, but not being shown at the show, the company has added color to the monitor (16 colors user selected from a palette of 32,000 colors).

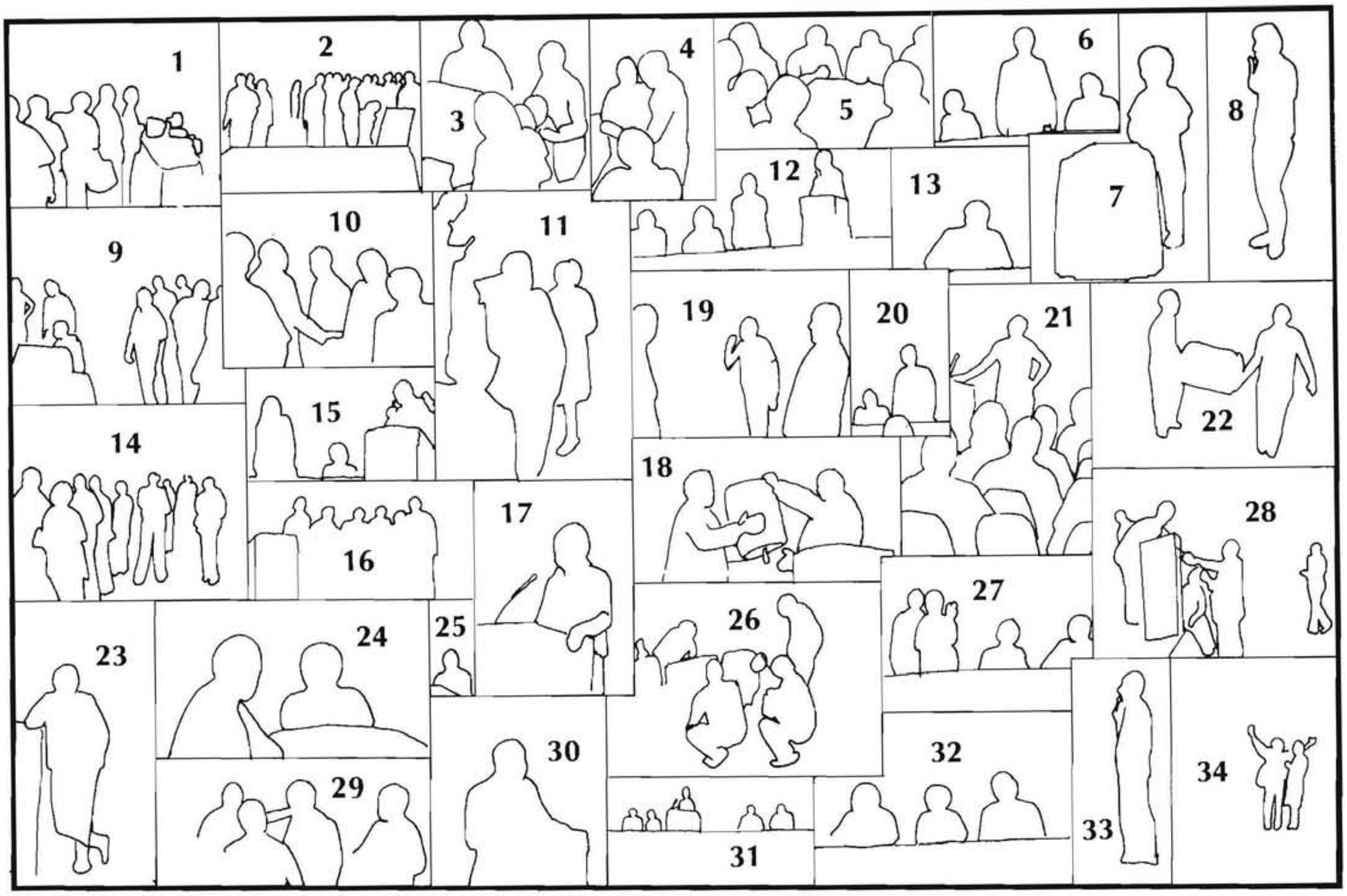
PIICEON was showing their 256K upgrade board for the Z-100 and Z-150 computers. These products are available through many local Heath/Zenith dealers and Veritechnology stores. In addition, the company is now producing the HZD-10 and HZD-20, 10 and 20 megabyte external Winchester systems. Their representatives told me that interested parties should contact their local store for details or ask them to contact this manufacturer.

Among the communication packages were a number that might interest various readers. Most offered plenty of flexibility, but some offered more: ACCESS from Hilgraeve featured command scripts, which allows the computer command file to call other computers, gather and save the data, process it with a separate software program, recontact the remote computers, and transmit the results of the program. It is available in all CP/M and MS- DOS formats. Mite is a non-hobbyist oriented communication package, designed to be used by end-users who are not hardware or software oriented. It is considered one of the best around and was recently reviewed in InfoWorld.

If you are tired of waiting for the Heath/Zenith 8087 coprocessor board for the H/Z-100, D.E.L. Professional Systems from Richmond, British Columbia, was showing their board and selling it through the Veritechnology store at the convention. Again, this is an OEM manufacturer, who needed the 8087 and so designed and built their own. When they were learned about by the local stores, they were contacted to build and supply the 8087 boards.

Other vendors at the show included Computer Consultants to Busi-

HUG
HUG
Pheasant Run
Pheasant Run
MPI
Pheasant Run



1. The Line Up ... Heavy activity during Conference registration.

2. Two HUGs are better than one.

3. Henry Fale (H-Scoop) gets a little help with breakfast.

4. Hmmmm? ... I put that bit here somewhere.

5. Ok .... Who took my juice?

6. Will the real Greg Martin please stand.

7. Dad, can I have one?

8. Sing another one Dale!

9. Get a Shuttle Lander of your very own from Hoyle & Hoyle.

10. Barry Watzman was really there!

11. I got HERO JR. now! Let's see if she can get it back!

12. Lori, you can sit down now!

13. What's so funny Blackmax?

14. I wonder if this is a one-way street?

15. Look into my eyes ...

16. Winners of special awards include Bill Parrott, Larry Ruh (CHUG), Bill Adney and Walt Bilofsky. Susan Hayes and Bob were just along for the ride.

17. Would somebody press the start button again?

18. Quick! ... Get me a glass.

19. Jeeeez ... What ya gonna do with all these empty boxes?

20. Somebody press the start button!

21. A C.E.T. is a ....

22. Roll out the barrel!

23. NO! ... I always stand this way!

24. Relaxation was hard to come by.

25. Is that you, Walt?

26. ... and, if you pray long enough ...

27. Margaret receives special attention from the vendors.

28. Joe Schulte presents one of the H-150's given away Saturday night.

29. We've got him now! You get him if he trys to go your way!

30. Let's give away another 'K'.

31. The Software Panel in action.

32. I know that one!

33. And, for my next trick ...

34. A one anda two anda ...

ness (showing StandardNet, a new "standard in Winchester Disk Systems and Local Area Networks", D-G Electronics, First Capitol Computer, Groffics and Hoyle & Hoyle (showing the Shuttle Lander program), Husker Systems of Nebraska, Kres Engineering (showing their portable conversions for the H/Z-100 and H/Z-89 computers), MPI (showing the popular printer series), Microservices (showing Z-PALETTE and Z-ANIMATE), Newline Software, Quikdata Computer Service, Redwood Development (showing a Z-100 graphics and plotter package), Sextant Publishing, Studio Computers, Sunflower Software (where the Huggies could guess at the number of sunflower seeds in a bottle), Technical Advisors, TMSI, Zeducomp, and ZPAY Payroll Systems. Of course, Veritechnology was present and selling lots of good Heathkit products at fabulous discounts (I picked up a couple of 5-1/4- inch 50-disk DiskBanks for $14.95 each).

Of particular notice this year, is the addition of the Western Regional HUG Conference, to be held at the Disneyland Hotel in Anaheim, California, on November 10th and 11th, later this year. In addition to a visit from our own Bob Ellerton, will be an address from the always controversial Adam Osborne, speaking about looking to the future. Gee, Bob, Michigan starts getting awfully cold in November -- hint, hint.

And of course, the East coast has, once again, the third annual CHUG conference on November the third, at the Crystal City Hyatt Regency in Arlington, Virginia, just outside Washington, D.C.

The Ceremonies

Bob Ellerton, in his usual modest self, started off the Grand Opening Breakfast by quoting the Bible for the benefit of the IBM-PC users in the crowd. "In the beginning," he said, "there were Huggies."

As has been the tradition at past opening ceremonies, local user's groups were asked to stand and introduce themselves. Groups were represented from as far away as Washington state, California, Florida, and New York. Reflecting the International flavor of the conference, Ottawa represented Canada. Of course, the midwestern United States were well represented with groups from Illinois, Wisconsin, Michigan, Indiana, and Ohio.

"Yes Virginia, there will be H-DOS 3.0," Bob said to the delight of many in the audience. Attendance approached 1,000 at this function. Reference was made here, of course, to Heath's contract with Bill Parrott and two others for version 3.0 of this popular operating system. See the section on the sessions for the highlights of Bill's address.

After the staff was introduced, Margaret Bacon, who later became known as "Ma HUG" (and don't forget to send her a mother's day card or she'll never forgive you), was given a special thank you award for her outstanding efforts in getting the convention organized.

The Special Awards were presented:

Lifetime Membership:	H. W. Bauman
Outstanding User:	Bill Adney
Outstanding Club:	Capitol Heath User's Group (for the 2nd year in a row -- they are putting on another conference this year -- see my report under sessions.
Outstanding Vendor:	Walt Bilofsky -- Walt has been there since the beginning -- the first to support Heath Computers with software.
From User to User:	A special recognition award. Because of the support from HUG's members, Thomas L. Rogers donated funds that were put toward a special award -- from user to user -- and this year was presented to Bill Parrott for the original conception of the National HUG Conference.

Phil Cole, Director of Product Planning at Heath Company and the keynote speaker at the breakfast session, introduced the new catalog, including an array of IBM-PC (and Z-150/Z-160) compatible products. He emphasized that what placed Heath above the rest of the mail order vendors for these products were a number of factors: Heath's famous after-sale service and support, the testing and analyzing of the products to pick the "best of the breed" in the

product, aggressive pricing, and a 90-day, money-back guarantee.

Some of the products introduced in this catalog included the AST 6-pack plus, the Novation Modem board, and the D-G magic RAM card. For the Z-100, the US Robotics Modem, was introduced. A Smart cable, stand-by power supply, Kowala pad, and a host of printers were also illustrated.

For the future, according to Mr. Cole, Heath is planning an expanded computer accessories section in the catalog and would welcome product suggestions from the users.

At the cocktail party, a large number of prizes from various vendors were awarded to the Huggies attending that session. Included were:

John Elashkar, Holt, MI	WordStar	Studio Computers
Richard Novak, Chicago, IL	P-SST Card	Software Wizardry
Gary Rasmussen, San Bernadino, CA	Universal Parallel Board	SigmaSoft and Systems
Steve Howard, Cottage Grove, MN	Palette	Software Wizardry
Capt. Glenn D. Watt, Jr., USAF Academy, Colorado Springs, CO	1 Free Software Package	Sunflower Software
Gary Ashley, Green Bay, WI	Dezign	Zeducomp
Neal Van Eck, Pt. Washington, WI	ZPay	ZPay Payroll Systems
Dennis Cunningham, Vicksburg, MI	DataStar	Studio Computers
Gloria Dalton, Carson, CA	ZPay	ZPAY Payroll Systems

Pheasant Run

DYSAN CORPORATION
CE DIVISION
Dysan
Z100
H8
Colorworks
Z100
SIGMASOFT AND SYSTEMS
The Software Toolw
4 GREAT PROGRAMS EACH LESS THAN
Generic Software
HUG
HUG

Ann Schuppert, Chicago, IL	1 Free Software Package	Sunflower Software
Joanne Lennstrom, Rochester, MI	ESP	Software Wizardry
Robert Young, Vermont, IL	ZLYNK III	Software Wizardry
Myron Edgerton, Constantine, MI	$100 Gift Certificate	Software Toolworks
Maria Moore, Chicago, IL	Condor	Studio Computers
Heidi Walker, Ada, MI	1 year subscription to BUSS	Sextant Publishing
Bonnie Budnick, St. Charles, MO	WatchWord 2	S & K Technology
William Lake, Jr., Niceville, FL	$50 Gift Certificate	Quikdata/H-Scoop
Monte Rubenstein, St. Charles, MO	"Your Fortune in the Micro Business"	Quikdata/H-Scoop
Patrick Manis, Las Vegas, NV	"Flip Facts Guide for CP/M"	Quikdata/H-Scoop
Clarence Wahner, Milwaukee, WI	"How to Use the H/Z- 100"	Quikdata/H-Scoop
Perry Straw, Chicago, IL	Spike Protector	Quikdata/H- Scoop
Dennis Danielson, Fairborn, OH	Graph-Pac II	Micro-Doc
Scott Sundell, Chicago, IL	Graph-Pac II	Micro-Doc
Jack Minelli, Olyphant, PA	COMPAT Disk Compatibility	Mycroft Labs, Inc.
Roger Fields, Rock Island, IL	MITE Data Communications	Mycroft Labs, Inc.
C. Schmid, Beaver Creek, OH	Graphics for the Z- 100	Microservices
David Ozarowicz, Wrightstown, WI	Set of graphics software	Microservices
Bruce Carskadon, Moorestown, NJ	$100 Gift Certificate	Husker Systems of Nebraska
Lee Schumacher, Woodbridge, VA	5.25-inch disk cleaner kit	Husker Systems of Nebraska
John Malone, LaGrange, IN	Diskette Flippy File	Husker Systems of Nebraska
David Mongold, Xenia, OH	1 Software Package	Husker Systems of Nebraska
Milton Krauthoff, Menomonee Falls, WI	$25 Gift Certificate	Husker Systems of Nebraska
Howard Leighton, Walpole, MA	1 box of 5.25-inch disks	Husker Systems of Nebraska
Sharyn Deeringer, Bolingbrook, IL	Investment Ma$ter	Generic Software
Marcus Burke, Starkville, MS	I-Bert	Generic Software
Cinda Kesler, Orient OH	Z-DOS Shuttle Lander	Hoyle & Hoyle
Lois McGovern, Arlington Hts, IL	SSI WordPerfect	Generic Software
Ed Jones, East Point, GA	Space Warrior	Generic Software
Steven Nelms, Tupelo, MS	Loan Ma$ter	Generic Software
Robert Brunner, Crystal Lake, IL	Game-Pac I	Generic Software
Gary Wojcik, Chicago, IL	Fund Ma$ter	Generic Software
Judy Elwood, Beaver Creek, OH	Fund Ma$ter	Generic Software
George Elwood, Beaver Creek, OH	Game Pack I	Generic Software
Douglas Miller, Columbus, OH	Fund Ma$ter	Generic Software
Martha Schooley, Evanston, IL	F-Trans	Generic Software
Charles Martin, Albuquerque, NM	Football Picks	Generic Software
Nancy Branyan, LaPorte, IN	Football	Generic Software
Richard Gulbrandsen, Addison, IL	Football	Generic Software
Jeff Teter, Hayes, KS	Football	Generic Software
Bill Radcliffe, Des Plaines, IL	Catalog Master	Generic Software
James LaFrentz, Leavenworth, KS	C.A.K.E.	Generic Software
Shari Trower, Valley Center, KS	Castle-Gor	Generic Software
Wanda Vodek, Oran, NY	Check Ma$ter	Generic Software
Allura Jones, Cooperstown, ND	C.A.K.E.	Generic Software
Bea King, Iola, KS	Archive-80	Generic Software
John Heidler, McHenry, IL	BackSaver Chair	Fusaro Associates, Inc.
Steven Fensler, Ft. Wayne, IN	Interrogator	Dysan
John Stetson, Silver Springs, MD	Interrogator	Dysan
Kelly Russell, Kenner, LA	Interrogator	Dysan
John Baxter, North Prairie, WI	Interrogator	Dysan
Michael A. Wolf, Xenia, OH	PAT-2+ tester	Dysan
Kathy Pervin, LaCrosse, WI	D-G Super 89	D-G Electronics
James Guggemos, San Pablo, CA	Digidraw for Z-100	Computer Consultants to Business
Mike Couch, W. Des Moines, IA	RMS-90 for Z-90	
Eric Bruder, St. Clair Shores, MI	Choice of ZMS-90 or ZMS- 100	
Barbara Strand, Bryan, TX	Choice of Z-Form or I-Form	
Joyce Dubbs, Pittsburgh, PA	Z3D Graphics Editor	Colorworks
Alfred French, Jefferson, LA	Plotware -- Z-Graphics	
Richard Kelly, Hales Corners, WI	C/80 C/Library	
Tom Lane, Naperville, IL	ZS-100 Speed Module	CDR
Clifford Lundberg, Elk River, MN	ZS-100 Speed Module	CDR
Clay Montgomery, Dallas, TX	Querty2 and ReDesign	

Pheasant Run
Pheasant Run
Pheasant Run

Gregory Garnier, Hales Corners, WI	SuperRAM 50A/ZD RAM	
Mike Lougee, Ann Arbor, MI	2 year subscription to Sextant	Sextant Publishing
JoAnna Jones, Bolingbrook, IL	HERO Jr	VEC
John D. Guenther, Montgomery, AL	ACCESS with Autopilot	Hilgraeve, Inc.
David Scrobel, Greenfield, WI	Castle-Gor	Generic Software
William Adney, Arlington, TX	Disk-Patch	Generic Software
Danny Jones, Cooperstown, ND	C.A.K.E.	Generic Software

For excitement, Phil Cole, Director of Product Planning, of Heath Company presented Hero Jr. to JoAnna Jones. Her husband was more excited than she was. You could sure see who was going to have fun with that prize.

Joe Schulte, President of Veritechnology, presented an HF-151 kit to Garrett Cantwell of Wilmington, Delaware.

Bill Johnson, President of Heath Company, presented the second HF-151 kit awarded that night to David Zimdars of Billings, Montana.

Finally, Jay Jarrett, Vice President of Product Development for Heath Company, and on behalf of Zenith Data Systems, presented the Grand Prize, a ZW-151 Winchester system, to Ernst Duesterhoeft of Helenville, Wisconsin.

Bill Johnson then presented, not one, but **two** $1,000 Gift Certificates from the Heathkit catalog. The first went to Karen A. Bakos of Merrillville, Indiana, and the second was awarded to Michael C. Frieders of Fairfax, Virginia.

Bob Ellerton thanked everyone for attending and ran down the statistics for the conference. Truly international with members attending from Australia, South Africa, Israel, West Germany, Mexico, and Canada.

The following are statistics on user types:

H/Z-100	573
H/Z-100 PC	80
H/Z-89/90	510
H-8	160
H-11	4
Robot	3
N-U-A	13
Others	217

Comments on the Conference

Note: The comments that follow are those of the reviewer and speakers, and do not necessarily reflect the viewpoints or policies of the Heath User's Group, Heath Company, Zenith Data Systems Corporation, or Zenith Electronics Corporation.

Generally, the attitude of most of those that were in attendance at this conference was no different than previous HUG conventions: One big family -- willing to trade ideas with one another and an overall friendliness not always found elsewhere. Several things did strike me, however, that deserve some editorial comments from this reporter.

First and foremost, was the continued rumors that ZDS is dropping support for the H/Z-100. Some people, evidently in a position to be believed, have it in their mind that if a product is superseded by another product, that support for the original will go away, no matter how big or small the company that makes this product.

I, for one, don't adhere to this narrow-minded viewpoint. My past experience has indicated otherwise. Whether the company is Heath, Zenith Data Systems, Apple Computer, Tandy, or even IBM, there has almost always been continued support for existing products long after a newer (not necessarily, more advanced) product has been announced. This is true of the H/Z-150 and H/Z-160 when compared to the H/Z-100. (I'm using the general model numbers, rather than the family to avoid confusion here.)

The H/Z-100 is, and continues to be, state-of-the-art technology. It represents the finest of two worlds with the addition of a very powerful graphics system that doesn't take a back seat to many, more expensive computer systems. The H/Z-150 and H/Z-160 don't come close to the power and overall flexibility of this system. For instance, it supports three different disk media at the same time (5-1/4", 8", and Winchester), it supports both 8- bit and 16-bit software, it has a vast (and growing) library of *company-produced* software packages, and it continues to sell well, with each month bringing better sales figures than the month preceding. No Virginia, the H/Z-100 is not dead, nor even ailing. It is growing stronger with each passing month.

As further evidence of this, look at the comments from Mr. Tom Dornback, Vice President of Software for ZDS: new Microsoft packages are coming for the H/Z-100, including windows, GW-BASIC II, MS-DOS 2.0 (now shipping); Lotus Symphony; and denials that other packages (including dBASE III) were not going to be offered in the future. Read what you will into those comments, but I see a clear indication of future support and additional software for the H/Z-100. And how about Mr. Dornback's predictions for the future? Higher speeds, greater floppy disk storage, more internal RAM capacity, software size requirements will increase, greater emphasis on graphics, windows, and mouse pointing devices. Software will become more complex and have more features, but will be easier to learn. More multifunction, multiuser, and limited multitasking. Appearance will play a more important part in the microcomputer industry. What better machine to fill this need, than the H/Z-100? Have you ever seen or used a more *imposing* computer? Have you really explored the ramifications of the graphics power in the H/Z-100, something that only just started showing itself at this conference? Do you realize that the computer technology is there to support a full 1K by 1K of pixel-oriented, 8-color graphics with the existing video logic board hardware (only the monitors are not up to supporting that kind of bandwidth)? How about 7.5 MHz clock speed? Its here now, not from ZDS, but from a support vendor. How long do you think ZDS will sit back and not offer something similar? H-DOS for the Z-100. We now know that that operating system is coming from Heath sometime down the road.

Is the H/Z-100 dead? I really don't think so. Who do you believe?

Educational software -- the key to the future. Talking with Henry Fale of Quikdata, we touched on a subject that has long been one of the ZDS's shortcomings: the educational marketplace, both at the grade school and university levels. The lack of a Logo package, yet being tied to CDC's Plato network (by CDC) and the entrance into over 200 university and college locations tells me that some people are looking beyond the software and willing to place their university or college on the line with an excellent piece of hardware. The choice of the Z-100 by Clarkson college was no fluke. Likewise, the regents at the University of Kansas didn't just draw names out of the hat. So where are you ZDS? Your educational package (Teacher/Student) is a poor second choice to the more flexible Pilot language and Logo. Yet nowhere do these two valuable computer aided instruction packages exist in your line. With the St. Joseph, Michigan, school district rated 4th in the state in quality of education, one would think that this school district would be an ideal beta test site for quality educational products. Heath is continuing to gain a reputation for excellence in its educational products. Yet where is the support in this field for the schools? Why haven't you capitalized in this very lucrative area, with 95% of the schools yet to decide on what computers they will purchase in the next decade, why aren't you, Heath and ZDS, at the forefront of creating and producing excellent computer aided instruction software, both of the remedial nature and the computer awareness nature?

Welcome to the micro world. One of the Huggies mentioned the lack of personalized training, such as that offered by Honeywell, Burroughs, and others. I've grown with the microcomputer industry and having cut my teeth with NCR corporation twenty years ago, I can well sympathize with people graduating from the limited flexibility of the mainframes (its just too expensive to experiment with these dinosaurs of the past) to the new, exciting world of the microcomputer, particularly the H/Z-100. In one package, selling for well below $10,000 (even $5,000), you can have more power on your desk than was possible even ten years ago. Yet, with the cost of doing business, that power came to the forefront for only two reasons, space-age technology and mass marketing. IBM became a leader in the microcomputer industry because it decided to sell its products through non-traditional (for IBM) channels: the retail dealer. When IBM tried to market its very competitive 5100 and 5110 computers back in the late 1970's and early 1980's (a machine very similar to the H/Z-89/90 in architecture and power), it did it through its normal outlets, the factory- and branch-based sales force. It failed to make a dent on the likes of Tandy, Apple, and Heath. It did not make use of the advertising power of Byte magazine, or others. Yet you, occasionally, still find these computers in some locations. Why? because they were bought by office managers that never "were fired for buying IBM." So what does this lead to? Can IBM, Tandy, or for that matter, anyone else selling into this price bracket, afford to sell through distributors (even Tandy has its profit-oriented distribution network), and still provide classes of the nature of a Honeywell or Burroughs? No, they can't. The money just isn't there and the technology moves too fast. Twenty years ago, new products were Beta tested for two or more years before being sold to the public. Today, you're lucky to be able to beta test new equipment for two weeks. No, I'm sorry for those of you from the mainframe environment, this industry (regardless of the vendor) just isn't able to sell these products at these prices and offer you the kind of training you would like.

However, there is a bright point on the horizon: The Heath Educational products offered through Heathkit (Veritechnology) stores and the Heath catalog. The educational products will continue to grow in the computer era. Just as earlier ventures produced a whole series of educational products for the automobile, you can expect a whole series of educational products for applications in the computer area. There is your training -- personalized, inexpensive, and self-pacing, along with the Heath tradition of "we won't let you fail." If that all sounds like advertising hype, perhaps it is. But it does serve to fill the needs of at least one conference attender. By the way, why didn't the Heath store at the convention have any educational products? I bet they would have sold very, very well.

Well, that rounds out the conference coverage from this reporter. If you have never attended anything like this (and their aren't too many to equal the excitement in a conference like this), why don't you plan on joining us in 1985?

See you next year!

WATCHWORD Version 2

...the ultimate in word processing...

For the H/Z 100 under ZDOS

New Features:

***** Files of Arbitrary Length *****
***** Extensive Configuration Facility *****
***** Multiple Fonts Loaded From ALTCHAR Files *****
***** Support For ZDOS Version 2.0 Subdirectories *****

Plus all of the features of Version 1 including: superscripts, subscripts, underlining, boldface, centering, formatting, microjustification, arbitrary line length with horizontal scrolling, word wrap, auto save, split screen, macros, user settable defaults and definable keys, global search and replace with settable zone.

Demo disk: $3
WATCHWORD with Manual: $100
HUGCON Special: $60

S&K Technology, Inc.
4610 Spotted Oak Woods
San Antonio, TX 78249
512-492-3384
ATTN: Steve Robbins

Visit our booth and pick up your complimentary ZDOS programming card.

Add Command Line Processing To Tiny Pascal Programs

Christopher Hall
554 Harper Avenue
Auburn, AL 36830

If You Knew Tiny...

The Tiny Pascal package offered by HUG as P/N 885-1086 provides the hobbyist-on-a-budget with an excellent opportunity to enter the fascinating world of structured programming for the small admission fee of $20. While this compiler lacks many of the features of the complete Pascal, it provides enough capabilities to do some useful programming. Browsing through the HUG Software Catalog, we find that there are several products that were written in Tiny. I've used Tiny to write a small DBMS-type home finance package, some graphic games, and a few utilities, all of which I'd probably still be working on had I attempted to write them in assembly language.

The utilities mentioned above are all of the type that operate on some file or device, and most are of the type usually known as "filters". In general, a filter takes the data from an input file, modifies it in some useful way, and writes the modified data to the output file. The nature of this type of program suggests the use of a command line invocation of the form:

```
        >command infile outfile
```

There is, alas, no provision in Tiny to allow user programs to access command line arguments specified in this way, Tiny requires that we prompt the user for any required parameters.

So Let's Fix It

When I first addressed this problem, I was hoping that a simple Tiny Pascal routine could be written to recover the purloined command line, but, a little hand-disassembling of parts of a few compiled Tiny programs revealed some bad news: the initialization part of Tiny's run-time code resets the stack pointer, thereby destroying the command line information in subsequent stack operations. (For a good explanation of how HDOS passes command lines to user programs, see Pat Swayne's 'Getting Started With Assembly Language - Part VI', in REMark 45.)

I was left then with a choice between attempting to modify the Tiny system (without source code), or devising an assembly language fix to be applied to individual, already compiled Tiny programs. Faint of heart, I chose the latter course, which, as it turns out, was not so difficult, almost fun in fact.

There is good news, though; all Tiny programs have the same value for the Load Address (ABS.LDA=07000A); likewise, they all have the same value for the Entry Address (ABS.ENT=103064A). We'll see shortly how this consistency is helpful. Of course, these values are guaranteed only for the version of Tiny Pascal that I'm using, namely Version 4.1.

The Program & How It Works

The program I came up with is called ACLP.ASM, and is presented here in Listing One. Primarily, this program will append the code between the labels APN.COD and APN.LEN to a compiled Tiny Pascal program. The appended code simply moves the command line information, if any, out of danger. It also strips white space from the command line, follows each argument with an ASCII NUL (00 byte), and counts the number of arguments and characters in the command line. These counts precede the arguments in the destination buffer, and are preceded themselves by a pair of NUL's. Together the counts and all the NUL's serve as a "signature", so that Tiny routines may be reasonably sure that there is valid data where expected. (See function arg_count, below.)

In doing its job, ACLP.ASM does a fair amount of verification and patchwork. To begin with, it verifies that the file to be modified is most likely a compiled Tiny Pascal program. This is where the consistency in Load and Entry Addresses comes in handy. Without that knowledge, it would be a tedious job indeed (in fact, I'd rather try to modify the Tiny system). With it, it's simply a matter of comparing the values of ABS.LDA and ABS.ENT to our previously determined "constant" values. Note that these and other possibly version-dependent values are defined near the beginning of the program, and will probably need modification for use with other versions.

Next, since some Jump instructions must be used in the appended code, and we have no way of knowing the addresses at assembly-time, a relocation scheme must be used (an alternative to this would be to use Z80 Jump Relative instructions). Restricting myself to 8080 code, I simply placed an EQUate after each Jump instruction with "*-2" in the operand field, so that the label actually "points" to the Jump instruction's operand. Then at run-time, the required offsets are calculated and patched to the "*-2" labels. Two values in the .ABS file header must also be modified: ABS.LEN and ABS.ENT. The first is simply increased by the value APN.LEN, which is the length of the appended code; the second is replaced by the relocated value of APN.COD, so that the appended code is executed before Tiny's initialization routine. The Jump at the end of the appended code is likewise patched so that execution resumes with Tiny's init code, i.e., the original ABS.ENT.

An Illustrative Example - Counting Words

A Tiny program that makes use of the command line is included in Listing Two. The program, WC.TPS, is based on the C program by the same name which is included in Kernighan & Ritchie's book about C. WC provides a rudimentary wordcount utility, but should not be depended upon for serious work; its short-comings, if not obvious at first, will become clear once you use it on a text file with embedded text-processing commands.

The reason for presenting WC.TPS is to illustrate the use of three Tiny functions which are included. The functions arg_count, arg_pointer, and arg_fopen were coded to allow WC and other utilities to use command line arguments, in lieu of prompting the user for the necessary information. The operation of these functions is described in the initial comment section of each routine. Normally, these routines will be kept in an include file, possibly named CLPLIB.TPI, and included in Tiny programs with the pragmat $CLPLIB; however, for clarity, they are shown here as part of the source file.

Putting It All To Work

To use these programs, first assemble ACLP.ASM with the command ASM ACLP=ACLP. (Let's assume everything's on SY0: for simplicity; this means some disk RESETting will be necessary.) Next, compile WC.TPS with the Tiny Pascal compiler, and save it as WC.ABS. Then, with ACLP.ABS and WC.ABS both on the disk in SY0:, type ACLP WC. The result of this command will be a modification of WC.ABS so that it may be invoked with the command WC textfile.ext.

Some Ideas & Suggestions

To put ACLP's verificatiQn abilities to the test, try it out on some non-Tiny files such as PIP.ABS, ACLP.ASM, or LP.DVD. Or try to run it on the same copy of a compiled Tiny program twice. Or try something else; when you make it bomb, I'd like to hear about it.

Of course, more sophisticated and useful applications than the word-count utility given here may make use of these routines. Note, however, that LOADing a file into the 4K buffer will destroy any unused command line information.

Another useful Tiny routine would be one to evaluate Unix style switches such as:

```
>command [-sw1sw2sw3...] infile outfile
```

An interesting modification to ACLP.ASM might be to change the appended code to handle the command lines the way CP/M does, i.e., assume the first argument is a filename, and put it in the READ filename area, likewise putting the second argument in the WRITE filename area.

Anyway...

Hopefully, this new capability will inspire more HUGgies to use Tiny and add some new software to the pool. Now if someone will just come up with a way to add structures and floating point....

In the meantime, WAR EAGLE!

About the Author:

Christopher Hall is a Staff Sergeant in the USAF, presently attending Auburn University where he graduated in March 1984 with a Bachelor's in Aerospace Engineering, after which he will attend USAF Officer Training School. Of course, his No. 1 hobby is his Z100/H89/H14 system, but he is also active in amateur astronomy, and likes to travel abroad.

Listing One

```
        TITLE   'Add Command Line Processor to Tiny Pascal'
        STL     'by Chris Hall'

*       ACLP.ASM
*       Version 1.0     20 Sep 83

        XTEXT   HOSDEF          These are .ACM files and are
        XTEXT   HOSEQU          included with HDOS 2.0
        XTEXT   ABSDEF
        XTEXT   ASCII
        XTEXT   ECDEF
        XTEXT   FILDEF
        XTEXT   ROMSUBS         H17 ROM Subroutines

MBUFLEN EQU     100             HDOS's maximum type-ahead buffer length
*                               command-line length cannot exceed this value

*** The following values must be changed if you're using
* a different version of TPascal.
*
TP.LDA  EQU     070000A         Tiny Pascal's usual load address (Ver 4.1)
TP.ENT  EQU     103064A         Tiny Pascal's usual entry point (Ver 4.1)
TP.VER  EQU     '4'             version #
TP.SUB  EQU     '1'             subversion #
TP.BUF  EQU     042200A         where to put command line info

        ORG     USERFWA
        STL     'Mainline Code'
        EJECT

ACLP    EQU     *               HDOS puts command line on stack
        LXI     H,0
        DAD     SP              find stack pointer
        MOV     A,L
        CPI     #STACK          has it moved from default value
        MVI     A,EC.FNR        file name required
        JZ      ERROR           tell user
SKIPSP  MOV     A,M             get next char
        CPI     SP              space?
        INX     H               point to next char
        JZ      SKIPSP          advance HL past it
        CPI     TAB             tab?
        JZ      SKIPSP          advance HL past it
        DCX     H               back up to 1st non-white char
        LXI     D,DEFAULT       now HL => fname to operate on
        XRA     A               channel 0
        SCALL   .OPENU          want to use read & write
        JC      ERROR           file doesn't exist or is write-prot.
        LXI     SP,USERFWA      reset stack

*  request 3-sector buffer for 1st & last sectors
*  ( 3rd sector is for a possible overflow of last sector )

        LXI     H,F.SECT+768    3 sectors starting at F.SECT
        SCALL   .SETTOP
        JC      ERROR           not enough memory
```

```
*  read 1st sector

        XRA     A               channel 0
        LXI     B,256           1 sector to read
        LXI     D,F.SECT        where to put it
        SCALL   .READ
        JC      ERROR           can't read file

*  determine if file is FT.ABS

        LXI     H,F.SECT        HL => 1st sector
        MOV     A,M             A = ABS.ID
        INR     A               binary flag is -1
        MVI     A,EC.IFC        illegal file contents
        JNZ     ERROR           file is not binary
        INX     H               HL => File Type byte
        MOV     A,M             A = FT.???
        ORA     A               FT.ABS should be zero
        ERRNZ   FT.ABS
        MVI     A,EC.IFC        illegal file contents
        JNZ     ERROR           file is not ABS

*  make sure load & entry addresses correct for version of Tiny

        LXI     D,TP.LDA        compare Tiny's usual LDA
        LHLD    F.SECT+ABS.LDA  to present program's LDA
        CALL    $CDEHL          with H17 ROM routine  DE=HL?
        LXI     H,LDA.ERR       invalid load address message
        JNZ     TPS.ERR         tell user

        LXI     D,TP.ENT        compare Tiny's usual entry
        LHLD    F.SECT+ABS.ENT  to present program's
        CALL    $CDEHL
        LXI     H,ENT.ERR       invalid entry point message
        JNZ     TPS.ERR         tell user

*  update the file header & patch relocated JMP vectors

*       patch jmp from ACLP code to original entry address

        LHLD    F.SECT+ABS.ENT  HL = original entry point
        SHLD    VEC.TPS         store it in the appended code

*       calculate new entry address & put it in file header

        LHLD    F.SECT+ABS.LEN  HL = ABS.LEN
        PUSH    H               save it for later
        XCHG                    DE = ABS.LEN
        LHLD    F.SECT+ABS.LDA  HL = ABS.LDA
        DAD     D               HL = address of last byte of original
        INX     H               HL = new entry point
        SHLD    F.SECT+ABS.ENT  so store it there

*       patch jmp to NONE

        PUSH    D               save ABS.LEN
        LXI     D,NONE-APN.COD  offset to NONE
        DAD     D               HL = relocated NONE
        SHLD    RLNONE          patch the JMP
```

```
*       patch jmps to DONE

        LXI     D,DONE-NONE     offset to DONE
        DAD     D               HL = relocated DONE
        SHLD    RLD1
        SHLD    RLD2            patch the JMPs to DONE

*       patch jmps to MOVARG

        LXI     D,MOVARG-DONE   offset to MOVARG
        DAD     D               HL = relocated MOVARG
        SHLD    RLMG1
        SHLD    RLMG2
        SHLD    RLMG3           patch the JMPs to MOVARG

*       patch jmps to MOV2

        LXI     D,MOV2-MOVARG   offset to MOV2
        DAD     D               HL = relocated MOV2
        SHLD    RLMOV2          patch the JMP

*       patch jmp to NEXTARG

        LXI     D,NEXTARG-MOV2  offset to NEXTARG
        DAD     D               HL = relocated NEXTARG
        SHLD    RLNEXT          patch it

*       calculate new ABS.LEN code length & put it in file header

        POP     D               DE = original ABS.LEN
        LXI     H,APN.LEN       HL = length of appended code
        DAD     D               HL = new ABS.LEN
        SHLD    F.SECT+ABS.LEN

*  find the last sector of the file

        POP     H               restore HL = ABS.LEN
        LXI     D,ABS.COD       add header length
        DAD     D               HL = ABS.LEN+ABS.COD = actual file length
        PUSH    H               save it
        MOV     C,H             C = # of last sector
        XRA     A               use channel 0
        MOV     B,A             BC = # of last sector
        SCALL   .POSIT          position channel cursor there
        JC      ERROR           what could go wrong here?

*  read the last sector

        XRA     A               channel 0
        LXI     B,256           1 sector
        LXI     D,L.SECT        where to put it
        SCALL   .READ
        JC      ERROR           can't read last sector

*  append the command line code

        LXI     B,APN.LEN       length of code to move
        POP     H               restore HL = actual file length
        PUSH    H               and save it
        MVI     H,0             just want offset to EOF
        LXI     D,L.SECT        point to beginning of sector
        DAD     D               HL => last byte of file
        INX     H               advance to next free byte
        LXI     D,APN.COD       where the code starts
        CALL    $MOVE           routine in H17 ROM
        PUSH    H               save address of next TO byte

*  position 'cursor' to first sector

        XRA     A               channel 0
        MOV     B,A
        MOV     C,A             BC = sector 0
        SCALL   .POSIT
        JC      ERROR           can't locate sector 0?

*  rewrite first sector

        XRA     A               channel 0
        LXI     D,F.SECT        write 1st sector
        LXI     B,256           just one
        SCALL   .WRITE
        JC      ERROR           can't write sector 0

*  see how many 'last' sectors there are (1 or 2)

        POP     H               HL = address of next TO byte
        LXI     D,-L.SECT       subtract FWA of Last SECTor
        DAD     D               HL = # of bytes in 'last' sector
        XRA     A
        CMP     H               less than 1 sector?
        LXI     B,256           assume only 1
        JZ      WRITE           then write one only
        CMP     L               1 sector exactly?
        JZ      WRITE           then write one only
        LXI     B,512           if >1 then write 2 sectors

WRITE   POP     H               HL = actual length of original file
        PUSH    B               save # bytes to write
        MOV     C,H             C = # of last sector
        XRA     A               channel 0
        MOV     B,A             BC = # of last sector
        SCALL   .POSIT
        JC      ERROR           can't locate last sector?
        POP     B               BC = # bytes to write
        LXI     D,L.SECT        where to get 'em
        SCALL   .WRITE
        JC      ERROR           can't write last sector(s)

*  close file & back to HDOS

CLOSE   XRA     A               channel 0
        SCALL   .CLOSE
        JC      ERROR           can't close
        XRA     A               normal exit
        SCALL   .EXIT

TPS.ERR CALL    $TYPTX
```

```
        INX     D               TO pointer
        DCR     B               count down a char
        JZ      DONE            patch at runtime; should never happen
RLD2    EQU     *-2             ReLocated DONE

        MOV     A,M             get a char
        CPI     SP              white space separates arguments
        JZ      NEXTARG
RLNEXT  EQU     *-2             ReLocated NEXTARG

        CPI     TAB
        JNZ     MOV2            if not white space, then continue
RLMOV2  EQU     *-2             ReLocated MOV2

NEXTARG INR     C               count an argument
        XRA     A               separate arguments with 00 byte
        STAX    D               store it
        INX     D               TO pointer
        DCR     B               count another byte
        JNZ     MOVARG          get next argument, if any
RLMG3   EQU     *-2             ReLocated MOVARG

*  note that the code should NEVER fall through here
*  since B=MBUFLEN to begin with and there can't be
*  more chars in the command line than that

NONE    DCR     C               zero arguments
DONE    INR     C               count last argument
        XRA     A
        STAX    D               end arguments with 00 byte
        LXI     H,TP.BUF+2      where to put 3rd byte of sig.
        MVI     A,MBUFLEN
        SUB     B               A = # chars in argument
        MOV     M,A             3rd byte of the signature
        INX     H               where to put 4th byte of sig.
        MOV     M,C             # of arguments
        JMP     0               patched at runtime to vector back
VEC.TPS EQU     *-2             to the Tiny program's original entry addr

APN.LEN EQU     *-APN.COD       length of the appended code

        STL     'Storage & Messages'
        SPACE   5,15

LDA.ERR DB      'loa','d'+NULL          rest of messages are in calls to
ENT.ERR DB      'entr','y'+NULL         $TYPTX in the TPS.ERR routine

DEFAULT DB      'SYOABS',0
F.SECT  EQU     *               First SECTor storage
L.SECT  EQU     F.SECT+256      Last  SECTor storage

        END     ACLP
```

Listing Two

```
{ Program WordCount }
```

```
        DB      'Incorrect',SP+NULL
        SCALL   .PRINT          HL => either LDA.ERR or ENT.ERR messages
        CALL    $TYPTX
        DB      ' address for Version '
        DB      TP.VER,'.',TP.SUB,' of Tiny Pascal.',ENL
        JMP     CLOSE

ERROR   MVI     H,NL            follow errmsg with NewLine
        SCALL   .ERROR          let HDOS print the msg
        XRA     A               normal exit
        SCALL   .EXIT

        STL     'Code to be Appended to TPascal Programs'
        EJECT

*  This is the code to be appended to the Tiny program
*  The routine determines if there are any command-line
* arguments & if there are, it moves them into Tiny's
* 4K buffer.  It also leaves a 'signature' with the data.
*
*       Register usage
*               HL => original location of arg's
*               DE => where to move them
*               B counts down chars in the list
*               C counts up args in the list
*
APN.COD EQU     *               Start of code to be appended
        LXI     D,TP.BUF        where to move the arguments
        MVI     B,MBUFLEN       max # chars in type-ahead buffer
        LXI     H,0             to find stack pointer
        MOV     C,H             use C to count arguments
        SHLD    TP.BUF          1st word of ACLP's signature 00
        INX     D               advance past the 1st word of sig.
        INX     D
        INX     D                past the argument list length
        INX     D                and past the number of arguments
        DAD     SP              still finding stack pointer
        MOV     A,L             LSB of SP changes if there's args
        CPI     #STACK          has it moved from standard value?
        JZ      NONE            patch relocated NONE at runtime
RLNONE  EQU     *-2             ReLocated NONE

        DCX     H               ready our FROM pointer for the loop
MOVARG  INX     H
        MOV     A,M             get a char
        CPI     SP              skip white space
        JZ      MOVARG
RLMG1   EQU     *-2             ReLocated MOVARG

        CPI     TAB             skip tabs, too
        JZ      MOVARG
RLMG2   EQU     *-2             ReLocated MOVARG

MOV2    ORA     A               end of entry?
        JZ      DONE            patch relocated DONE at runtime
RLD1    EQU     *-2             ReLocated DONE

        STAX    D               store the char at TP.BUF+
        INX     H               FROM pointer
```

```
{
  usage:
  >wc textfile

  counts 'words', characters, and lines in textfile
  adapted from word count function in Kernighan & Ritchie's
  The C Programming Language, by Chris Hall.
}

CONST   NL=10;  TAB=9;  SP=32;  ESC=27;
        NO=0;   YES=1;  PAD=0;
        IOREAD=1;

VAR     open,           { used in calling arg_fopen }
        ch,
        nchars,
        nwords,
        nlines  : INTEGER;

        inword  : INTEGER; { Boolean }


$PROCLIB        { HDOS & H19 Procedures included with HUG's TPascal }

FUNCTION arg_count;

{
        checks to see if ACLP has been used
        returns -1 if it has not
        else returns # of arguments with which program was invoked
}

CONST   ARG_LOC = %042200;      { Address of arguments }

BEGIN { arg_count }

        IF  ( (MEM[ARG_LOC] <> 0)
          OR (MEM[ARG_LOC+1] <> 0)
          OR (MEM[ARG_LOC+3+MEM[ARG_LOC+2]+1] <> 0) )
         THEN
                arg_count := -1 { ACLP hasn't been here }
         ELSE
                arg_count := MEM[ARG_LOC+3]
END;



FUNCTION arg_pointer(arg_num);

{
        returns -1 if arg_count = -1
        returns  0 if arg_num > arg_count
        else returns pointer to arg_numth argument
}

CONST   ARG_LOC = %042200;
```

```
VAR     pointer,
        count   : INTEGER;

BEGIN
        count := arg_count;

        IF (count = -1)         { ACLP hasn't been used }
            THEN
                BEGIN
                  arg_pointer := count;
                  EXIT
                END;

        IF (arg_num > count)    { too few arguments }
            THEN
                BEGIN
                  arg_pointer := 0;
                  EXIT
                END;

        IF (arg_num = 1)        { 1st argument is special case }
            THEN
                BEGIN
                  arg_pointer := ARG_LOC+4;
                  EXIT
                END;

        count := 1;                     { use to count arguments }
        pointer := ARG_LOC+4;           { actual beginning of 1st arg }
        REPEAT
                REPEAT          { leave pointer pointing to arg separator }
                        pointer := pointer + 1
                UNTIL (MEM[pointer] = 0);
                count := count + 1
        UNTIL (count = arg_num);
        arg_pointer := pointer + 1
END; { arg_pointer }



FUNCTION arg_fopen(arg_num,iotype);

{
        returns -1 if arg_pointer = -1
        returns  0 if arg_num > arg_count
        returns  1 otherwise
        the actual fileopen attempt may cause an IORESULT error
}

CONST   ARG_LOC    = %042200;   { address of 4K Buffer in Tiny  }
        IOERR_ADDR = %070000;   { address of IORESULT           }
        EC_ICN     = %016;      { Illegal Channel Number        }
        EC_IFN     = %007;      { Illegal File Name (too long)  }
        NAME_LEN = 16;  { maximum filename length in HDOS 2.0   }

        IOREAD  = 1;            RNAME   = %063336;  { addresses of the }
        IOWRITE = 2;            WNAME   = %063315;  { filename storage }
        IOLOAD  = 4;            LNAME   = %063274;  { areas }
        IOSEEK  = 5;            SNAME   = %063357;

        ERR = -1;       { Command Line Argument error }

VAR     offset,         { Index offset into the filename area           }
        pointer,        { Points to argument                            }
        name_addr       { Address of name area for specified iotype     }
            : INTEGER;


BEGIN { arg_fopen }

        pointer := arg_pointer(arg_num);        { get pointer to argument }

        CASE pointer OF
                ERR     :       arg_fopen := ERR;    { ACLP hasn't been used }
                0       :       arg_fopen := pointer { not enough arguments }
                ELSE
                        arg_fopen := 1
        END; { Case }

        CASE iotype OF
                IOREAD  :       name_addr := RNAME;
                IOWRITE :       name_addr := WNAME;
                IOLOAD  :       name_addr := LNAME;
                IOSEEK  :       name_addr := SNAME
                ELSE
                        BEGIN   { Illegal iotype (channel #) specified }
                          MEM[IOERR_ADDR] := EC_ICN;
                          EXIT
                        END
        END; { Case }

        offset := 0;    { into name area }
        MEM[name_addr] := MEM[pointer];

        REPEAT          { move the requested argument }
            offset := offset + 1;
            pointer := pointer + 1;
            IF (offset > NAME_LEN)
                THEN
                  BEGIN         { Name Supplied is tooo long }
                        MEM[IOERR_ADDR] := EC_IFN;
                        EXIT
                  END;
            MEM[name_addr+offset] := MEM[pointer]
        UNTIL (MEM[pointer] = 0);

        CASE iotype OF          { Do the actual OPEN }
                IOREAD  :       RESET(':');
                IOWRITE :       REWRITE(':');
                IOLOAD  :       LOAD(':');
                IOSEEK  :       UPDATE(':')
                ELSE   WRITE('Crash!',NL)
        END

END; { arg_fopen }
```

```
BEGIN { WordCount }

        open := arg_fopen(1,IOREAD);    { 1st argument, open to read }

        IF ( open = -1 )
                THEN
                    BEGIN
                        WRITE('This program has not been modified ');
                       WRITE('to use command line processing.',NL);
                        EXIT     { to HDOS }
                    END;

        IF ( open = 0 )
                THEN
                    BEGIN
                        WRITE('Usage: WC textfile<cr>',NL);
                        EXIT
                    END;

        IF ( IORESULT <> 0 )
                THEN
                  BEGIN
                    HDOSERRORCODE(IORESULT);
                    WRITE('Can''t open file!');
                    EXIT
                  END;

        nchars := 0;
        nwords := 0;
        nlines := 0;
        inword := NO;

        GET(ch);
        WHILE ( ch <> PAD )  DO
                BEGIN
                  nchars := nchars + 1;
                  IF ( ch = NL )
                        THEN
                          nlines := nlines + 1;
                  IF ( ( ch = SP ) OR ( ch = NL ) OR ( ch = TAB ) )
                        THEN
                          inword := NO
                        ELSE
                          IF ( inword = NO )
                                THEN
                                  BEGIN
                                    inword := YES;
                                    nwords := nwords + 1
                                  END;
                  GET(ch)
                END;
        WRITE('Characters',TAB,nchars#,NL);
        WRITE('Words',TAB,TAB,nwords#,NL);
        WRITE('Lines',TAB,TAB,nlines#,NL)
END.
```

Controlled Data Recording Systems Inc.

ANNOUNCING THE FDC-H8

DOUBLE DENSITY 8" AND 5.25" CONTROLLER FOR THE H8 COMPUTER

Has all of the capabilities of our popular FDC-880H controller, with the added features of;

- Direct memory access (DMA) data transfer.
- Hard sectored controller (H17) incorporated on the board.
- Runs with the standard 8080 CPU card and with Z80 CPU upgrades.
- Accesses both hard sectored disk formats and soft sectored disk formats through the same drives attatched to the FDC-H8 without hardware additions. **Price $495.00**

NEW PRODUCTS FOR THE FDC-880H

DM-1 DUAL BOARD MODIFICATION KIT **$29.95**
Allows for both the FDC-880H and the H88-4 controller cards to interface with the same 5.25" drives. Drives will run as both hard sectored format and soft sectored format depending upon the logical drive letter.

CDR BIOS by Livingston Logic Labs **$60.00**
Enhanced version of Heath/Zenith CP/M 2.203 BIOS with ZCPR. Supports all Heath/Zenith disk formats through the FDC-880H and the H17 controllers.

CDR DVD by Livingston Logic Labs. **$40.00**
HDOS driver for running double density HDOS through the FDC-880H

Shugart Slimline 5.25" 40 track double sided drives **$275.00**

Shugart Slimline 8" double sided drives **$525.00**

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

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

HUG NEW PRODUCTS

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

P/N 885-5004-37
CP/M-86
TERM86 & DSKED $20.00

Introduction: This disk contains two utility programs. One allows the user to communicate with an external device or computer using a modem. The other program allows you to directly edit data on any size disk drive, including a winchester.

Requirements: DSKED will work on ANY computer running the CP/M-86 operating system. TERM86 also requires the CP/M-86 operating system, as well as, the following hardware. Some sort of modem, including interconnect cable, an H/Z-100 or H/Z-100 with any Godbout serial interface, or TMSI's H1000 with a standard Heath H89 serial interface, or any S-100 computer system using a Godbout serial interface and H19 or H29.

The following files are included on the HUG P/N 885-5004-37 CP/M- 86 TERM86 & DSKED disk.

HZTERM86	.CMD	DSKED	.A86
TMTERM86	.CMD	ASCII2	.LIB
TERM86	.A86	JMPVEC	.LIB
TERM86	.DOC	BDOSCALL	.LIB
DSKED	.CMD	README	.DOC

Authors:
TERM86 -- Jim Buszkiewicz
DSKED -- Frank O'Neal

TERM86: TERM86 is a modem communications package, written in assembly language under the CP/M-86 operating system. In addition to being able to communicate with a host computer via modem and phone line, TERM86 is capable of transferring files either to or from itself, using two types of protocol. The two types of protocol used are: simple capture buffer (ALL available memory is used, up to 1Mb) with XON-XOFF, and XMODEM protocol with checksum verification. XMODEM protocol was chosen because of its popularity with microcomputer users, bulletin boards, RCPM systems, and CompuServe. This protocol was thoroughly defined by Ward Christensen, the author of CBBS in Chicago, and has become the hobbiests' pseudo-standard.

Another feature is the ability of TERM86 to automatically log onto CompuServe with two initial keystrokes. Permanantly stored messages can also be sent to the host computer (or modem) with two keystrokes. These messages can be dialing information for smart modems or can be single line log-ons for different time- share systems. These "canned" messages can be changed from within TERM86 and these changes can be temporary or permanant. The baud rate can be modified from the keyboard when running TERM86, and also, can be a temporary or permanant change. Disks can be changed under TERM86 with a single "Reset Disk System" keystroke. Full and half duplex mode is also supported. The following is the command list available under TERM86.

TERM86, Version 1.0Z
Storage Buffer = 125365 Bytes
TERM86 Commands

HELP	ESC-~	Print This List
F0	ESC-J	Print This List
	ESC-E	Clear Screen
F1	ESC-S	Miscellaneous Functions
F2	ESC-T	Reset Disk System
F3	ESC-U	Store Mode Toggle
F4	ESC-V	Save Data To A Disk File
F5	ESC-W	Transmit A Disk File
F6	ESC-P	Duplex Mode Toggle
F7	ESC-Q	Clear Storage Buffer
F8	ESC-R	Return To CP/M-86

Even though the commands may look like the CP/M version of TERM, TERM86 is a complete re-write and not a translation. A brief version of the command list is maintained on the 25th line, as well as, the duplex status and capture buffer status. The source code is very well commented, making modifications or additions easy.

DSKED: DSKED is a disk editor that allows you to directly modify any track, sector, or byte, on any disk drive you may have. This program is written entirely in assembly language and includes all commented source code, as well as, required library files. DSKED is recommended for the advanced user, for it is possible to actually edit your disk to 'death' with this program. DSKED has the following commands available:

D(rive)	- allows the changing of default drives
T(rack)	- set the track number
S(ector)	- set the sector number
B(lock)	- set block number
R(ead)	- read selected sector from disk
W(rite)	- write selected sector to disk
A(scii)	- use ascii characters in editing sector data
H(ex)	- use hex numbers in editing sector data
+	- plus one sector (advance)
–	- minus one sector

F(ile)	- enter file name and display it (no track or sector info)
Q(uit)	- return to CP/M-86

DSKED was written using all proper system calls and is not hardware or version dependent.

Comments: Aside from a directory program, I have found these two programs the most useful in my library of utilities. TERM86 manages to get the job done with the least amount of effort and learning. Being entirely menu driven, one could use this program even if the documentation was lost. DSKED has already twice help me find obscure bugs in the operating system, BIOS, which I wrote. It only lacks the track and sector data when displaying a file.

TABLE C Rating: (10), (5), (3), (1), (0)

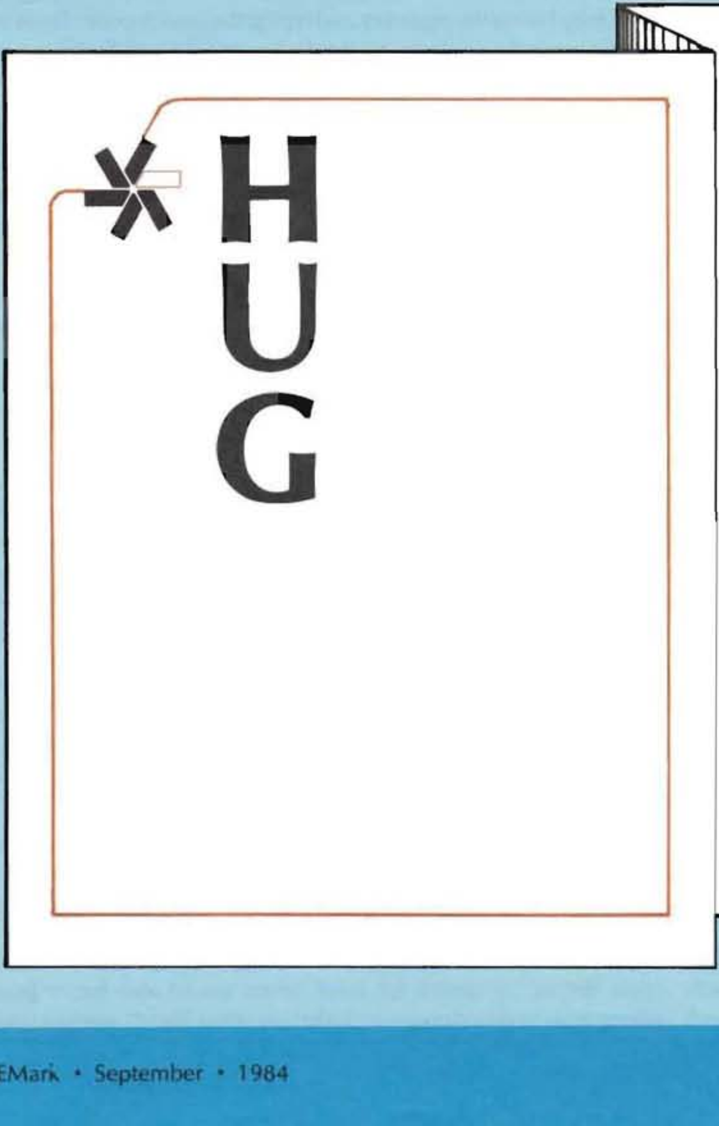
Our Mistake !

Recently, the Heath Users' Group released a product known as RDZDOS on HUG disk 885-1235-37. Since this introduction, it has come to our attention that a similar product, by the same name, was previously made available from Computer Consultants to Business. Therefore, with apologies to Computer Consultants, we have changed the name of this HUG product to COPYDOS. COPYDOS will be available on the same HUG disk 885-1235-37. A description of this program can be found in Volume 5, Issue 7 of REMark.

Ordering Information

For Visa and MasterCard phone orders; telephone Heath Company Parts Department at (616) 982-3571. Have the part number(s), descriptions, and quantity ready for quick processing. By mail; send order, plus 10% postage and handling ($1.00 minimum charge, up to a maximum of $5.00. UPS is $1.75 minimum -- no maximum on UPS. UPS Blue Label is $4.00 minimum.), to Heath Company Parts Department, Hilltop Road, St. Joseph, MI 49085. Visa and MasterCard require minimum $10.00 order.

Any questions or problems regarding HUG software or REMark magazine should be directed to HUG at (616) 982-3463. REMEMBER - Heath Company Parts Department is NOT capable of answering questions regarding software or REMark.



We want to

Thank

Dr. Herb Friedman
Chris Gillespie
Connie Huber
Jim Jones
Jane Kabelman
Liz Martin
Kris Young

For their
Help **U**nselfishly **G**iven

at the
1984 International HUG Conference

Heath Users' Group

Inventory and Information Management As Simple As PIE

Jerry E. Shepherd
3964 South 3600 West
West Valley City, UT 84119

As a member of HUG and a reader of REMark, you obviously have an enthusiastic interest in the home computer. It is likely that you also have a collection of books and magazines and a supply of parts, tools and equipment that is used for the pursuit of this exciting hobby. In addition to all of this, there is little doubt that you have an assortment of items that have been accumulating over the years for one purpose or another.

Have you ever tried to locate one of these items and not been able to find it? Of course you have. Have you ever experienced the difficulty of locating an interesting magazine article that you had read earlier or had to skim through a pile of magazines to find articles that relate to a particular subject? I would be willing to bet that you have.

Keeping track of where we keep things is a universal problem, but there is an inexpensive method of using your computer to solve it and the use of this method is 'as easy as PIE'. As a matter of fact, this method is based on an unusual use of the full-screen editor provided for the H-8 and H/Z-89, which SoftStuff calls 'PIE'. Other editors, such as the one supplied with HDOS can also be used for the same purpose, but this article will focus on the use of PIE as your management system.

To use PIE in an information or inventory management system, a file is created which contains a list of the items that you want to keep track of with the system. Each entry in the list will include the name of the item, where it is located, and any other information that you feel will be useful about the item. Each entry will be preceeded by a unique string of characters which we will call a search code. This search code will be used by PIE to locate the particular entry and its related information.

The search code will be some string of characters that describe some characteristics about the associated item. Since many items may fall within several general categories, a search code can be provided for each of these. When a search is initiated for a particular code, each entry will be located in sequence by performing successive searches for the code.

With PIE, a search can be made in either the forward or backward directions through the file. To set up a search, hit the enter key and enter the search code that you want to look for. Then, a forward search for this code is initiated by hitting the '0' key on the small keypad of the H/Z-19, which is required when PIE is used. A backward search is initiated by hitting '.' on the same keypad. By repeatedly hitting either of these keys, you can step through the file, stopping at each entry which is flagged by the search code PIE was set up for. When all of the entries which are marked with this search code have been located, PIE will automatically notify you of this by giving you the message 'search key not found'.

This can be illustrated by the following example. Suppose your system is set up to keep track of the books in your personal library and you would like to use it to search for a copy of the book which deals with the 8080 microprocessor. If your management system was previously set up so that each entry dealing with the 8080 is preceeded by the search code '*8080', you can set PIE to the start of the file by hitting the special function key, F3, and hit the home key to be sure the cursor is in the proper position. Then you can set up the search by hitting the enter key and typing the search code. Once this simple procedure is done, hit the '0' key and PIE will locate the first entry marked by the code. If the displayed description of this book indicates this isn't the book you are looking for, continue the search until the desired one is located. If you want to go back and take another look at an entry that you have previously looked at, just search backward by hitting the '.' key until you are back to the desired entry. Once the desired book is located, you can use the information in the file to determine the physical location of the book.

Since the search codes are placed at the start of each entry, the description for the entry is automatically placed at the top of the screen. If the description occupies more than 24 lines on the screen, you can use the special function keys F1 and F2 to scroll forward either a page or a line, respectively. F3 and F4 are used in a similar manner to scroll backward a page or line, respectively.

It is usually helpful to provide an index at the first of your management files to help you remember which search codes are required for the various categories. By placing it at the start of the file, the index will be displayed immediately after typing 'PIE filename', where filename is the name of the file used to store your management system. At times, there may be a large number of categories and the index will fill several screens and require an excessive amount of memory and disk space. If this is the case, you may wish to adopt a coding scheme which will let you determine the search code needed to search through a particular category of items so that you can eliminate the need for the index. Alternatively, you may wish to keep the index on a sheet of paper near your computer.

One possible coding scheme is to use the names of the items themselves as the search codes. In doing this, PIE will locate the desired items that you aren't interested in. For example, if the management system which was keeping track of your library books was set up so that each book dealing with BASIC programming used the search code 'BASIC', a search for these books would also locate books whose titles or descriptions included the string 'Basic', such as books

on 'Basic Algebra', 'Basic Chemistry', etc., since PIE will locate all occurrences of the string it was set up to search for.
One method of avoiding this is to preceed each search code with a special character so the code is a string of characters less likely to be used in the title or description of any books described in the file. By changing the above search code to '*BASIC', PIE would skip past those entries which include the string 'Basic' without the asterisk and only the desired entries will be located.

Another useful device is to use lower case characters for your search codes and upper case characters for your descriptions. This will eliminate the possibility of your search codes being used inadvertently in the descriptions, since the search function included with PIE differentiates the lower and upper case characters.

A 'REMark Information Control System'

In order to clarify the use of PIE as an Information Control System, let's take a look at a system that you can set up to keep track of the articles in 'REMark' that you might want to refer to from time to time. Once you understand this, you can modify the system to keep track of your other magazines and books, the inventory in your home workshop, or whatever.

Lets call the system we are about to set up the 'REMark Information Control System' and name it's file 'REMark'. In this system, let's provide an index so that it is easy to determine which codes are required to locate articles belonging to articles from several categories.

The entry for each article in the file will provide the title and author of the article, a brief description of it's contents and a cross-reference to any corrections and interesting comments made about the article in future issues of 'REMark'. A lot of other features could be added to make the system even more useful, but these are enough to demonstrate how you can set up a system of your own.

To get started, place a disk which contains PIE in SY0: and a blank disk in SY1:, (you can set up this system with one drive but you will be more limited in the amount of information that can be provided with each entry). Type 'PIE REMark' and execution of PIE will begin. You will receive the message "File not found: will create it" to indicate that PIE will enter the text you are about to enter into the designated file. As you enter data into PIE, you can transfer it to the file on the disk at any time by hitting the control and 'V' keys simultaneously. To perform the same function and exit to HDOS, hit the control and 'E' keys simultaneously. As you are creating the file, it is a good idea to save your data periodically so that unexpected problems, like a power failure, do not wipe out your efforts.

Start by typing the search code index, which should look something like that shown in Figure 1, where 22 different categories and their corresponding search codes are listed. As indicated by the brief instructions provided at the bottom of the index, all of the articles in the file which belong to a particular category can be located by moving the cursor to the bottom of the screen and searching for the special code placed at the left of the desired category. For example, a search for '*g' would cause each article in the file, which deals with the use of HDOS, to be selected for your review.

Moving the cursor to the bottom of the screen is suggested, because failing to do this may cause the first search to stop at the search code located in the index. This isn't serious since subsequent searches will allow PIE to locate the desired entries in the file, but it is a bit sloppy and nice to avoid.

Once the index is created, hit the control and 'V' keys simultaneously to save your work in the file and you are ready to enter the data for the articles into the file. Group the articles into sections which correspond to the various issues of 'REMark' and arrange these sections in sequence, starting with the first issue. A typical example of how this might appear is shown in Figure 2.

Each section will begin with a line that identifies the issue number, the publication date, and optional comments used to identify special features of the article. Following this, one or more lines which are labeled 'Highlights' are used to provide a summary of the articles contained in the issue. After the Highlights, a subsection is used to describe each article or other features that may be of interest to the user in the future.

These subsections provide detailed information about the contents of articles and/or other features in the issue so that the user can get a good idea of it's contents. Each subsection starts with a line which identifies the issue and page number of the feature, a list of the search codes which can be used to locate the description,and the name of the contributor of the feature which will be the name of an author, advertiser, etc. This first line is followed by one or more lines which are used to provide the more detailed comments about the contents of the article. Following this, optional lines will be used to indicate the location of any comments and corrections about the feature which appear in future issues of REMark.

The entries in Figure 2 describe the contents of Issue #1 and some of those of Issue #2. A simple description for subsequent issues would be entered into the file in the same manner. To illustrate how one of these entries may be located, suppose a search is initiated for the search code, '*j'. This would cause PIE to locate the editorial located just below the Highlights of the first issue since it's description is marked by this code. Any other editorials in the file that are marked by this code would also be located by subsequent searches for this code. The first editorial could also be located by a search for '*h', 'Editorial' and 'Robert Furtaw' since these strings are all included in the description for this editorial.

In Figure 2, you will see that any comments and/or corrections about the issue are indicated after the Highlights of the issue. Similarly, any comments and corrections about an article within the issue are indicated after the subsection which describes the article. A comment about Issue #1 was found in Issue 3, Page 26. If the comment covers something you may wish to refer to at a later time, it would be placed after the Highlights for Issue #1 as shown. Issue #2 contained a list of modifications for some H-11 interface boards. A subsection, which describes this list, is shown at the bottom of Figure 2. As shown by the cross-reference at the bottom of this subsection, a correction of these modifications appears in Issue #3, Page 27. The user would add this cross-reference to the file after receiving Issue #3 and finding the correction. By doing this, each time the user locates the description for this article, the correction is also immediately apparent so it can be considered when desired.

You may disagree with what I included as Highlights for interesting articles in Figure 2. This data was organized to reflect my own interests and requirements and should not be interpreted as a reflection on the fine work performed by the authors of material omitted here. It isn't practical to include everything which occurs in each issue of the magazine, since your computer's memory and disks will fill up quickly enough, even when you limit your entries to the most interesting articles.

Once your memory or disk is full, it will become necessary to use new files and disks to cover the remaining issues. This should be minimized as much as possible, since there are some difficulties in searching through multiple files and disks because of the necessity to leave and re-enter PIE in order to change files and disks.

Since the section and subsections are all preceeded with the issue number, it is possible to locate the entries for this issue by using this number as a search code. Notice how the issue number is specified as ##1 at the start of each section and as #1 at the start of each subsection. This was done to increase the searching possibilities. If the number was specified as #1 in both cases, successive searches for '#1' would cause PIE to locate the first line of the section for Issue number 1 and the first line of the subsection for each article, in turn.

Specifying these numbers as shown allows you to perform the same search as described above. In addition, by searching for '##', successive searches will cause PIE to jump from the first line of the first article to the first line of subsequent articles, in turn. As you do this, you can read the Highlights of each article to determine if it contains any subject matter you would like to read through. This provides a simple method for you to browse through a whole set of the issues to find something interesting to read that doesn't fit into any particular category.

By organizing the file as described before, there are some other searching modes that are available if you know something about the article you are looking for. For example, if you are only interested in the articles from a particular author and you know his or her name, a search for the author's name will cause each of these to be displayed. For example, to locate any contributions from REMark's current editor, 'Walt Gillespie', a search for 'Walt', 'Gillespie', etc, will cause PIE to locate any of his entries that are represented in the file. (Note that a search for the text editor, PIE, would also locate each of Walt's contributions since this string is included in the last part of his name.

If you wanted to limit your search to the editorials, use of the search code, 'Editorial', will locate the description of any editorials represented in your file, as long as these descriptions include the string 'Editorial'. A little thought in the selection and placement of your search codes will allow a variety of searching modes to make your searching tasks easier.

After you have your file set up, you may realize that you omitted some articles or features from your file that you have become interested in. You may also realize that some of your entries are no longer of interest to you. Each time you receive a new issue of REMark, you should look for corrections and comments about any articles of interest. If you find any of these, you will want to add their location to the subsection in the file which describes these.

With PIE's editing features, it is a simple matter to add and/or delete data. To do this, study the manual that was provided with your editor and you will find this a simple task. Be sure to save any permanent changes by hitting the appropriate keys before leaving the management system or they will be lost.

Conclusion

The 'REMark Information Control System' described in this article illustrates the methods required to use PIE for keeping track of information. By changing the text in your file and modifying the format to fit your needs, you can use the same procedure to keep track of the tools in your workshop, your inventory of parts and supplies, or any other set of materials that can be reduced to a list of items organized into a set of pre-determined categories. You can also create management systems to help you keep a diary, keep track of important addresses and telephone numbers, or a variety of other things which you deal with every day. The list is endless.

For the modest price of $50 in your Heath catalog, all of this may seem like 'PIE in the sky' (a pun was intended - sorry). But if you spend a little time thinking about it, you can find many practical uses for PIE. You can even use it as a text editor!

Figure 1. An index that can be used in the 'REMark Information Control System' described in the text. These entries reflect the authors interests and can be modified to fit your own requirements.

```
...............................................................................
                                        :
Search Code    Category                 : Search Code    Category
___________    ________________         : ___________    __________________
                                        :
#a             Issue Highlights         : #l             Hardware Hints
#b             High Interest            : #m             Programming Hints
#c             Owned Products           : #n             Assembly Language
#d             H-8 Hardware             : #o             'BASIC'
#e             H-8 Software             : #p             Misc. Languages
#f             H-8 Advertisements       : #q             Computer Applications
#g             HDOS                     : #r             Educational Programs
#h             HUG/REMark               : #s             Recreational Programs
#i             Heath/Zenith             : #t             Utility Programs
#j             Editorials               : #u             Product Information
#k             Miscellaneous            : #v             Interesting Facts
...............................................................................
Move the cursor to the bottom of the screen by hitting 'shift' and 'home'
to locate articles belonging to a particular category, search for code given at
left of desired category.  You can also search by title, author's name etc.
Use '0' & '.' on keypad to advance forward or backwards, respectively
hit Control Z twice to exit to HDOS.
...............................................................................
```

Figure 2. Entries used in the 'REMark Management System' to describe the articles found most interesting by the author. In your own system, these would be modified to suit your own requirements.

```
...............................................................................

REMark             Issue ##1              1978                  Premier Issue

Highlights:  Introduction to HUG, DEC Connection, BASIC Programming Course,
             Questions About H-8, ET-3400 & Microprocessor Course, Computing
             As A Hobby, Configured vs. Distribution Tapes, Floppy Disk System,
             B.H. BASIC Patches, Misc. Info, Readers letters, Coin-collector
             Program, Color Guessing Game, Mini-Nim (game), Sorting, New
             Software Announcement, and Employment Offer From Heath.

             A comment about this issue appears in "BUGGIN HUG" Issue #3, p26

#1 p3        Search Codes: #h  #j                           Robert Furtaw

             Editorial

             Philosophy of REMark, programming pep talk, future prospects.

             (location of any future references to this editorial appear here)

#1 p4        Search Codes: #b                               Lou Frenzel

             HUG: What's in it for you?

             An introduction to the HUG/REMark philosophy.

             (location of any future references to this article appear here)

#1 p8        Search Codes: #d

             Most Frequently Asked Questions About the Heathkit H8 Computer,
             Reason For Using the 8080, H8 Bus, Front Panel, Octal Notation

             (location of any future references to this article appear here)

#1 p16       Search Codes: #o

             Installing EX. B.H. BASIC Patches

             Patches to turn on command completion in remark statements
             and eliminate skip intermittent skip during pause.
```

```
          (location of any future references to this article appear here)

.     .     .     .     .     .     .     .     .     .     .     .     .

REMark              Issue #02              1978

Highlights:  Product Introduction, ASCII/BAUDOT Driver For H8, H11 Printout
             Test, Computer Hobbyist, H8 Console Driver, H9 Lower to Upper
             Case Modification, H17 Disk System, H8 Disassembler

#2 p3        Search Codes: *i  *u                          Lou Frenzel

             New Product Introduction - The Facts of Life

             Heath's philosophy about product information.

             (location of any future references to this article appear here)
                                      .
                                      .
                                      .
#2 p27       Search Codes: *l

             H11 owners

             A list of modifications required to improve the performance of
             the H11-5 serial card and the H11-2 parallel card.

             A correction to this list found in "BUGGIN HUG", p27 of Issue #3
                                      .
                                      .
                                      .
                                      .
```

LEARN A NEW WORD FOR YOUR VOCABULARY

ZPAY\ zē pā\ n. 1: A computer payroll system for the Zenith and Heath computer systems 2: The act or fact of paying or being paid 3: The status of being paid by an employer

ZPAY PAYROLL SYSTEMS Presents

ZPAY available for ZDOS, IBM-PC Compatable or CPM

Some of the features available in the **ZPAY** Payroll System

- User Friendly — Menu Driven
- Pays by Hourly, Salaried & Commissioned
- No interpreters necessary
- Super FAST operation
- Attractive 3 ring binder manual
- User changeable State & Federal tax tables
- User selected pay periods
- Maintains payroll records required by law
- Prints out easy to use and read reports
- Choice of two different check formats
- *Plus* many features of systems costing much more

May be seen at many Heathkit Electronic Centers or order yours today from **ZPAY** Payroll Systems — **Only $100.00.**

Please send me **ZPAY** for ☐ ZDOS ☐ IBM-PC ☐ CPM

Name ____________________

Street ____________________

City ____________ State ________ Zip ________

Phone ____________ Visa/MC Card # ________ expires ________

Orders shipped by UPS. Enclose $100.00 plus $4.00 per order for shipping and handling. Payment by check, money order, or Visa/Master Card. *Allow 2—4 weeks for delivery.*

ZPAY PAYROLL SYSTEMS

3516 Ruby Street • Franklin Park, IL 60131 • (312) 671-3364

You don't have to be a WESTERNER to attend the

Western Regional HUG Conference

Speakers • Panels • Displays • Prizes

Grand Prize: H/Z-100 Computer with Winchester Drive*

November 10th & 11th, 1984

Disneyland Hotel • Anaheim, California

The Official Hotel At The Magic Kingdom

Write for Registration Information to:

Conference Coordinator
1555 North Orange Grove Avenue
Pomona, California 91767

* You must be present to win.

Patching A .REL Routine

William Brandoni
37926 Wright Street
Willoughby, OH 44094

The following article describes a technique for patching a program or subroutine that is in Microsoft .REL format. The procedure assumes that you are very familiar with assembly language programming.

This technique has been used to fix three errors in the FORTRAN runtime library (FORLIB.REL) distributed with the CP/M version of Microsoft FORTRAN-80 (Version 3.40) sold by Heath/Zenith. The errors are:

1) Extended (4-byte) integer multiplication will sometimes produce non-zero results even though one of the variables equals zero.

2) Extended integer exponentiation produces wrong answers when a non-zero value is raised to the zero power.

3) Extended integer values in the range -65535 through -32769 and in the range +32768 through +65535 do not print correctly in formatted output.

These errors are quite serious. If you use extended integers with FORTRAN-80, you should be cautioned.

The first two errors occur in the module INT4 and the last one occurs in the module FORMIO. These are both modules in the FORLIB.REL library.

This article will NOT present the patches required to fix the library. As you will see, the patching procedure is too involved for that. The patches are available from the Heath Users' Group. To receive a copy of them, send a disk and $10 to the Heath Users' Group, Attn: Nancy Strunk, Hilltop Road, St. Joseph, MI 49085. The patches are in the form of two relocatable files, each corresponding to one of the modules in error. The files are useless unless you own the original FORTRAN-80 package, since each module depends on others in the library to work properly. The HUG contribution also contains installation documentation. Installation requires the use of Microsoft utilities that again are part of the FORTRAN-80 package. So unless you are a licensed user of the CP/M release of Microsoft FORTRAN-80 version 3.40, the specific patches given to HUG will be useless.

What I want to present here is the TECHNIQUE used to patch the library. This same technique can be used to patch other routines in FORLIB or in any other file in Microsoft relocatable format.

The procedure is quite time consuming, and should only be considered when it is absolutely necessary. If you are like me, you can plan on spending several weeks working evenings to patch a single routine.

Throughout this article, I will be referring to various programs from Microsoft. You should have the reference manuals to these programs available, especially if you are unfamiliar with some of the features and commands. The entire article assumes that you are a licensed user of a Microsoft high-level language package such as FORTRAN-80, and that you have a real need to patch a system library routine.

I will be using the INT4 module in FORLIB.REL in all of the examples. The same techniques should apply to other routines in other libraries.

I will also be referring to a disassembler called RESOURCE which is in the public domain and is available from the CP/M Users' Group. The following quote from the documentation for RESOURCE summarizes a philosophy that I share.

*"**Note:** Pardon the editorial, but I feel hardware without good software is useless to 99% of us. Most good software has to be paid for. I strongly support the legitimate purchase of licensed software...*

Please use (RESOURCE) in the spirit in which it was contributed: to enlarge your understanding of the micro-computer world around you, and to allow you to customize programs which you legitimately own, for your own use."

Microsoft Relocatable Code

Microsoft has a family of compilers and assemblers that all produce "relocatable" object code. Heath/Zenith supplies MACRO-80 (assembly), FORTRAN-80, COBOL-80, and the Microsoft BASIC Compiler BASCOM. If you have ever used any of these products, you know that the result of an assembly or compilation is a .REL file -- a file containing the object code in Microsoft relocatable format. Standard libraries for the compiler products are distributed in this format. Microsoft provides the LINK-80 linker to convert your main program and the required portions of the library from .REL format into a CP/M compatible .COM file for execution.

There are many advantages to this system. First, you can create your own libraries of often-used routines in relocatable format and use them repeatedly in your application programs by just linking them when needed. Second, it is conceivable that you could write some sections of code in one language and other sections in another language. The resulting .REL files could all be linked together to create a final program. Finally, standard function libraries for each language are distributed by Microsoft in .REL format; after linking, your final program will only contain those routines that it needs. This means that larger programs can be developed than would be possible if the entire support package had to be included.

There is, of course, a penalty to be paid for all of this convenience. Relocatable object code is written in a unique bit-stream format that results in files that cannot be listed or executed. In fact, individual machine instructions in the relocatable files are not even aligned on byte boundaries. Therefore there is almost no way to decipher the contents of a .REL file or library. If an error is discovered in one of your custom libraries, you will have to return to the original source for the routine, fix the error, recompile the routine, and then rebuild your library. If an error is found in one of the system libraries, you will not have the source, so a patch seems to be out of the question.

When I recently discovered three significant bugs in the FORTRAN-80 library, I wanted to fix the errors. That's when I decided there HAD to be a way to get at the code for parts of FORLIB.REL.

After many weeks, I was finally successful in extracting the code for the offending routines, correcting the errors, and rebuilding the library with the patched routines. I want to pass on these techniques in case you have a need to patch a relocatable routine when the source for that routine is not available.

How Relocation Works

To understand the technique that I used, you first need to understand how relocation works. At least, you need to know what "hooks" are available to tell LINK-80 which modules to link together. Let's use an example.

A FORTRAN program might contain the following statement:

```
A = SIN ( B )
```

The compiler would recognize SIN as a reserved word for a function that computes the sine of an angle. It would generate a call to the appropriate library routine. This would be flagged as an EXTERNAL symbol in the .REL file produced by the compiler. When the program is linked using LINK-80, the linker will encounter the EXTERNAL symbol reference and will search the FORTRAN system library for a module that contains an ENTRY with the same symbol name. That module will then be included in the program image being created. Only at that time will the actual memory address of the SIN function be defined. If the SIN routine needs to use other routines from the FORTRAN library, it would contain EXTERNAL references to them, and LINK-80 would include those routines as well.

Thus, the "hooks" required by the linker are EXTERNAL references (references to routines or data in another module) and ENTRY locations (addresses that are made available for use by another module). The linker's job is to make sure that every EXTERNAL reference is matched by an ENTRY into another routine.

There is a lot more to the linking process, since addresses may be assigned to CODE, DATA, and COMMON areas of the program. But let's leave well enough alone for now.

Steps Required To Alter a .REL Module

Let's define the steps needed to modify a module from a library in relocatable format:

1) Determine which module is in error.

2) Isolate the module from the library.

3) Convert the module into normal program (.COM) format.

4) Convert the program into an assembly language source file.

5) Edit the assembly language source file to correct the errors.

6) Assemble and test the corrected module.

7) Insert the revised module into the library, replacing the old version.

The Microsoft packages sold by Heath/Zenith, along with any text editor, provide all of the functions necessary to perform these tasks except for step 4, which is a program disassembly. An excellent disassembler is available in the public domain from the CP/M Users Group. Other disassemblers are available on the market and from HUG.

Let's take the steps one at a time and look at the details involved.

1) Determine which module is in error.

Assuming that you want to fix an error in one of the system libraries, you first need to find out where the error is occurring. The first step, then, is to write a short program that will reproduce the error. Compile the program, specifying a listing file that includes the assembly code generated by the compiler, along with the source code. If the test program is called TEST.FOR, the following commands will generate enough information to find the offending routine:

```
A>F80 TEST,TEST=TEST        (compile with FORTRAN-80)
A>L80 TEST,TEST/N/E         (link with LINK-80)
A>TEST                      (execute)
```

The first command compiles the FORTRAN-80 program, producing both a .REL file and a .PRN file. The file TEST.PRN is the listing, and will contain the source and pseudo-assembly listings. (This is the default for listing files with Microsoft compilers.)

The second command links the program, producing an executable image. The third command runs the program to make sure that the error is present.

Once you have verified that you can reproduce the error, you should examine the listing in an attempt to find assembly level call statements to library routines. (Note that the assembly code appears in the listing about one or two source lines below the statement that generated it.)

Figure 1 shows a portion of a compiler listing for a program that performs extended integer multiplication. The library routine $MY' performs the actual multiplication. (Most Microsoft library routine entry points have a dollar sign as the first character.) Thus, you know that somewhere in the library, there is a module with the symbol $MY defined as an ENTRY. You need to know which one.

The Microsoft LIB-80 library manager utility can be used to generate a cross-reference listing of the library, but you must be careful here or you can destroy the library. The best technique is as follows:

```
A>↑P                (control-P toggles the printer on)
A>LIB               (use LIB-80 utility)
*FORLIB/L           (request listing with /L switch)
*↑C                 (exit LIB-80 with a control-C)
```

The Control-P will send to the printer a copy of all output that goes to the console screen. This will be a long printout, but you will need to refer to it as we proceed. Be sure to exit the LIB utility with a Control-C, since the normal exit procedure will destroy FORLIB in this example.

The last part of the listing is a table of all of the symbols in the library, along with the name of the module that defines the symbol (ENTRY) and every module that references the symbol (EXTERNAL). Look through the listing for the symbol $MY. When you find it, note the module that defines that symbol. This is the one you need. Hang on to the library listing. You will need it later on.

Figure 2 shows part of the listing for FORLIB, where the symbol $MY is shown as defined in module INT4.

2) Isolate the module from the library.

You want to isolate the module from the library so that, if you make any errors in later steps, the linker will not extract more than this one routine. You can use the library manager again. This time you will use it to create a new library.

Since the symbol $MY is located in the module INT4, you need to extract INT4 from FORLIB. The following commands will do that:

```
A>LIB                   (use LIB-80 utility)
*INTLIB=FORLIB<INT4>    (create INTLIB.REL)
*/E                     (exit with /E to save INTLIB)
```

Here, LIB is used to create a new file, INTLIB.REL, by extracting the single module INT4 from the existing FORLIB.REL file. File extensions should not be given to LIB, as they are all defaulted. The /E switch is required here to properly exit LIB by writing out the new file.

3) Convert the module into normal program format.

The INTLIB.REL file is still in Microsoft relocatable format, and you cannot do very much with it. You need to convert it into standard machine code. To do this, you will have to write two very short assembly language programs. The first program will contain dummy calls to every ENTRY point in INT4. The second will contain dummy routines to define every EXTERNAL reference made by INT4. You need to do this so that you can identify all of these symbols later. After all, when you are done, you want the new version of INT4 to contain all of the same "hooks" to the linker that the original had.

If you return to the cross-reference listing made in step 1, you can identify all of the references that you need. Look through the first part of that listing for a section with the heading "Module INT4 of FORLIB REL". That section will show every ENTRY point and every EXTERNAL reference made by the module. See Figure 3.

You are now ready to write the two assembly routines.

Let's call the first one INT4XTRN, since it will contain EXTERNAL references for every ENTRY symbol in INT4. This program should have three lines of code for every INT4 ENTRY. One section of code would be:

```
EXTERNAL      $MY
DB      'Call $MY'
CALL    $MY
```

Use any editor (CP/M's ED is fine) to create this program as a file called INT4XTRN.MAC. The short DB statements will identify each label during program disassembly later on. After the final call, finish the program with these two lines:

```
DB      'End of INT4XTRN'
END
```

The second program will be similar to the first, except that you now want to define ENTRY points for every EXTERNAL symbol in INT4. This routine could be called INT4NTRY. This program requires two lines of code for every EXTERNAL symbol in INT4. One section of code would be:

```
        ENTRY   $AB
$AB:    DB      'This is $AB ........'
```

It is a good idea to make each DB in this program at least 20 characters long, and maybe even longer, since external references can include offsets. For example, the original code could have contained a statement like:

```
        LHLD    $AB+16
```

and if the DB isn't long enough, you will misinterpret the label during disassembly. After the final entry, finish the program with these two lines:

```
        DB      'End of INT4NTRY'
        END
```

When you have finished writing these two routines, you are ready to create a machine code program that can be disassembled. The following steps are all that is required:

```
A>M80 =INT4XTRN
A>M80 =INT4NTRY
A>L80 INT4XTRN,INTLIB,INT4NTRY,INT4NEW/N/E
```

The first two statements assemble the two short routines using the MACRO-80 assembler, producing the files INT4XTRN.REL and INT4NTRY.REL. The third command tells the linker to load INT4XTRN into memory, then INTLIB, and finally INT4NTRY. The linker then resolves all references and creates the file INT4NEW.COM in machine executable format. The link order isn't really critical, but I like this method because it forces the routine you want to disassemble into the middle of the .COM file, so that its actual beginning and end are very easy to determine.

If L80 displays any unresolved "globals" at the end of the link step, then you did not create the proper references in your two assembly routines. Go back and fix the errors and try again. Incidentally, this is why we isolated the module in step 2. If we hadn't isolated INT4, but had just linked into FORLIB with a search switch, the linker could have included other modules from the library.

4) Convert the program into assembly language format.

This step, commonly called disassembly, is the most time consuming and exacting part of the entire process. The underlying principles are, however, quite simple.

A disassembler reads the machine instructions of a program and converts them into standard assembly mnemonics. The disassembler that I am most familiar with is called RESOURCE. It is available from the CP/M Users Group. I recommend this disassembler to anyone interested in 8080 assembly language program development. The CP/M Users Group disk contains a fairly complete documentation file which should help you get started. However, there is no easy way to learn program disassembly. You just have to do it and learn by experience. If you are a good assembly language programmer, then the disassembly should go fairly smooth.

If you decide to use RESOURCE, take advantage of the "A" and "B" commands. "A" is an "attempt" to define DB statements in the program, and "B" will "build" default labels. Neither command is foolproof, but they will do most of the work for you.

Once you have used the "A" and "B" commands, locate the DB string:

```
        nnnn    DB      'End of INT4NTRY'
```

and insert an end-of-disassembly code into the control table at this point. Any information beyond that DB is not part of the program, with the possible exception of a short data area. Then you should define all of the ENTRY and EXTERNAL labels. The DB statements that you entered in your dummy programs will show up in the disassembly to aid you. For example, when you encounter:

```
        aaaa    DB      'Call $MY'
        bbbb    CALL    xxxx
```

you will want to enter the label $MY for the address xxxx. Similarly, when you encounter:

```
        nnnn    DB      'This is $AB ........'
```

you will want to enter the label $AB for the address nnnn. Unfortunately, RESOURCE will not accept a dollar sign as part of a label. My way around this is to substitute the two-character string ZZ for every dollar sign. You can later tell an editor to substitute a dollar sign for every occurrence of ZZ and you will have the proper labels in the source file.

Once you have entered the labels for all the EXTERNAL and ENTRY points, you should try to make the rest of the program seem reasonable. If the code just doesn't make sense, try different techniques. And watch out for some pitfalls.

* Should a section be disassembled as instructions, DB's or DW's?

* Are all labels defined, or are some "hidden" in the middle of instructions?

* Did you generate too many labels?
(Example: most LXI instructions will refer to labels, but some may have used actual data.)

* Did the original programmer play tricks to gain a few bytes, or maybe to confuse someone trying to disassemble the code?

There is one other point you should keep in mind. Microsoft's languages are designed to generate ROMable code. This means that there should be no DS instructions for defining scratch areas of memory within the body of a routine. And there should be no self-modifying code, either. Data areas should always be in separate data sections. They can appear at the beginning or end of your disassembly.

Stick with the disassembly and remember to save your intermediate results often. There is nothing more frustrating than working for hours on a problem and then losing it all due to a disk error or power failure.

5) Correct the errors.

Once you have successfully disassembled the program, you should save it as a text file.

You may be tempted to start right in trying to find the error in the routine. (Remember that we started all of this to fix a bug.) Don't do anything yet, until you have verified that the disassembly really represents the original code for the routine in question. The only editing that you should do at this point is the minimum required to convert the routine to Microsoft MACRO-80 format.

First, delete all program lines from the two dummy routines that you wrote. These will be all lines from the beginning through the line that reads

```
        DB      'End of INT4XTRN'
```

and all lines from the first

```
ZZxxx:
        DB      'This is $xxx .........'
```

through the end of the file. The only exceptions should be data areas that are referred to from the main body of the program.

Second, add a NAME statement at the beginning of the source, using the same name as the module you are disassembling (INT4 in this case.) Then identify the body of the program with a CSEG statement and the data area (if any) with a DSEG statement.

Third, use the editor's search and replace facility to change all "ZZ" strings into "$" symbols.

Next, add EXTERNAL and ENTRY statements for all of the labels requiring them. These should be the same as the list generated from LIB way back in step 1 (See Figure 3).

At this point, it is a good idea to assemble the new file using MACRO-80 to check for any obvious syntax errors. If the assembly is OK, check the resulting .REL file with LIB.COM to get a cross reference list of it. If the disassembly and reassembly were error-free, you should be able to reproduce an exact cross reference match between the original routine and the new one. If not, then something is wrong. Don't attempt any code changes until you are sure you have a good starting point. You may have to go back to the disassembler and try something different.

One more test before you go after the error in the routine. You should make sure that the reassembled version behaves just like the original. Try the following:

```
A>L80 TEST,INT4NEW,FORLIB/S,TEST/N/E
A>TEST
```

The first command will link the original TEST program and the new version of INT4, along with other required routines from FORLIB, into an executable program. The second command runs the program. At this point, the output should match the original exactly. Only after you have passed this test, should you begin your search for the error.

6) Assemble and test the corrected module.

List the assembly code on your printer and begin your analysis of how the routine works. As you gain new understanding of the code, you may want to rename some of the default labels from the disassembly. You will also begin to zero in on the section of code responsible for the error. Ultimately, you will be able to fix the routine to work properly.

Assemble the patched routine and test it, using the same linking step used above. When you have successfully demonstrated that the patch works correctly, you are ready for the last step.

7) Insert the revised module into the library.

Finally, you will want to insert the patched module into the system library. The recommended method is as follows:

```
A>REN OLDLIB.REL=FORLIB.REL
A>LIB
*FORLIB=OLDLIB<..INT4-1>
*INT4NEW
*OLDLIB<INT4+1..>
*/E
```

The first step renames the standard library so that you won't accidentally destroy it. The remaining steps create a new library from the old one by extracting all modules from the first up to but not including INT4, then inserting the new INT4, and finally extracting all modules from the one following INT4 to the end. The "/E" exits LIB by writing out the new library.

And that's it. If you made it this far, you have the basics for patching any routine in any library that is in Microsoft relocatable format. I wish you luck, and hope you won't need to use this procedure too often. It sure can kill a lot of time. But its good to know that, should the need arise, there really is a way after all.

```
26      C
27      C COMPUTE ZERO * CONSTANT
28      C
29              IVALX   =       IVAL * 10
*****   003F'   LXI     B,0025"                         ; this code
*****   0042'   LXI     D,IVAL                          ; was generated
*****   0045'   LXI     H,[01 00 00 00]                 ; by previous
*****   0048'   MVI     A,05                            ; instructions
*****   004A'   CALL    $I4                             ; in the
*****   004D'   LXI     D,TEN2                          ; source
*****   0050'   LXI     H,[01 00 00 00]                 ; program
*****   0053'   MVI     A,02                            ;
*****   0055'   CALL    $I0                             ;
*****   0058'   CALL    $ND                             ;
30              JVALX   =       JVAL * TEN4
31              KVALX   =       KVAL * TEN2
32              DIFF1   =       JVALX - IVALX
33              DIFF2   =       KVALX - IVALX
34              WRITE(TERM,  60)
*****   005B'   LXI     H,IVAL                          ; this code
*****   005E'   CALL    $L4                             ; is from
*****   0061'   LXI     H,000A                          ; source
*****   0064'   CALL    $MY                             ; line 29
*****   0067'   LXI     H,IVALX
*****   006A'   CALL    $T4
```

Figure 1
Part of Compiler Listing
With Assembly Language Included

```
Symbol  Value   Defined Referenced

$MU     0076'   DBLUTL  INT4     DSQRT   DATAN   DLOG10  DSIN
                        DBLEXP   DEXP    DLOG    DBLPLY
$MV     0237'   INT4
$MY     005A'   INT4
$N1     0031'   INT4
$S1     0019'   INT4
$SB     0068'   FADD    INT4     COSIN   ATAN2   ATAN
```

Figure 2
Part of LIB Listing for FORLIB.REL
Showing Module That Defines $MY

CP/M is a registered trade mark of Digital Research, Inc.

The CP/M Users' Group is not affiliated with Digital Research. CP/MUG only provides software on 8" SSSD and 5" Northstar disk formats. See REMark Issue 29 (pg 15) for details on how to get CP/MUG software in Heath 5" formats. RESOURCE is available on CP/MUG volume 42.

FORTRAN-80, COBOL-80, MACRO-80, BASCOM, LINK-80 and LIB-80 are trademarks and/or products of Microsoft, Inc.

```
Module INT4      of FORLIB  REL

Length of Program        856
Length of Data area      17


Entry point(s)

$A1     0003'   $AE     0249'   $AV     022B'   $AY     0000'
$C0     0280'   $C4     0306'   $C5     030A'   $C6     0310'
$C7     0314'   $CD     0340'   $CL     027D'   $D1     00CA'
$DE     025B'   $DV     023D'   $DY     00C7'   $E1     013B'
$EE     0261'   $EV     0243'   $EY     0138'   $L4     0043'
$M1     005D'   $ME     0255'   $MV     0237'   $MY     005A'
$N1     0031'   $S1     0019'   $SE     024F'   $SV     0231'
$SY     0016'   $T4     004E'

External reference(s)

$AB     024D'   $AC     034A'   $AC4    031A'   $ARG    0131'
$AU     022F'   $CK     027E'   $CY     030E'   $D90    01DE'
$DAC    033B'   $DAC0   0336'   $DB     025F'   $DNRML  033E'
$DU     0241'   $EB     0265'   $ERR    02B8'   $EU     0247'
$L1     0219'   $MAF    0215'   $MB     0259'   $MFA    0225'
$MU     023B'   $SB     0253'   $SU     0235'   $T1     0222'
$T3     020A'
```

Figure 3
Part of LIB Listing for FORLIB.REL
Showing References for Module INT4

About the Author:

William R. Brandoni *works for Diamond Shamrock Chemicals Company as the Process Simulation Manager in the Engineering Department. His group is responsible for the computer simulation of large chemical processes, as well as other engineering and technical computer applications. Most of their engineering programming is done in FORTRAN. Over the past 20 years, Bill has worked with all sizes of computers, from mainframes (IBM, Univac) through super-minis (DEC) to small micros (Heath, Atari). Since purchasing his Heath H-88, he's experimented with various versions of BASIC as well as ALGOL, Pascal, Assembler, and most recently C.*

The Sprinter By MPI

Jim Buszkiewicz
HUG Software Developer

My first impression of this new generation printer was "why portability?". After thinking a few moments, the answer was clear; what good is a transportable (or portable) computer like the H/Z-160 without a printer with the same capability. If you're going to take your computer with you, you're going to need a printer if some heavy programming is anticipated. MPI's Sprinter is ideal for this purpose. Weighing in at a mere 16 pounds, the Sprinter, with its cover, looks more like a portable typewriter than a high speed printer.

Micro Peripherals Inc. (MPI), located in Salt Lake City, Utah has been manufacturing the ever popular models 99 and 150 dot matrix printers for some years now. Only recently have they introduced their new "Sprinter".

General Overview:

The Sprinter is a true portable printer, not just a new case around an old unit. It comes with shock mounted components to insure that it performs properly after traveling. The case is molded from high impact resistant materials to assure ruggedness and to maintain its attractive appearance. It comes standard with 4k print buffer that can easily be expanded up to 68k using MPI's memory expansion board. As with all of MPI's printers, a parallel port is standard, but can be changed to serial with another add-on circuit board.

The Sprinter provides a close-to correspondence quality print with its single pass hi-density character set without having to over print. Being microprocessor controlled, the 160 CPS print rate is realized by logic seeking head control and high speed paper advance which is concurrent with head movement while the large print buffer continues to receive data.

The Sprinter has a user-friendly keypad for operator control. This feature allows control of forms length, print density, horizontal and vertical tabs, baud rate (if the serial interface is installed), character sets and other built in features. One of the keyboard entries provides a status printout indicating how some of the controllable options are set.

One of the included features in the Sprinter is the built-in front paper feed system. The front feed allows the user to feed single sheets of paper from the front and will automatically feed paper to the first print line. Having both friction and tractor feed methods available, paper can be fed from the front, rear, or bottom to suit the requirements of the work area. I did find it a bit difficult to feed light weight paper into the mechanism from the front. With only one set of tractors some sort of drag had to be provided to keep the paper taught between the top and bottom of the print head. Feeding the paper past this drag point presents a frustrating problem. Again with only one set of tractors, there is no way to reverse index the paper either from the keyboard or the host computer. This is a serious deficiency and can lead to wasted paper when trying to position the top of the form.

High resolution dot addressable graphics capability comes as a standard feature. This capability can be used for plotting, printing of screen graphics, special fonts, and character sets. Up to 6120 dots per square inch can be addressed and horizontal dot spacing can be set to one of six dot densities.

Operation:

The first problem I had was with removing the cover. After reading the manual, it became obvious that the cover was to be lifted and pushed back off its rear catches for removal. As mentioned earlier, paper feeding was also difficult. The first sheet had to be torn into a point to get it past the ribbon area. Since the H/Z-100 Applications pack was included, interconnection was easy. The cable supplied with that pack matched the printer on one end, and the H-100's parallel connector on the other. After configuring CP/M-86 and Z-DOS for parallel printer operation, the Sprinter appeared to respond as one would expect any printer to from either operating system.

I also received the 64k Memory Mate™ print buffer expansion board. Installation went according to the manual, however, the board I received had its share of problems. Several hours of trouble shooting and one or two calls to MPI revealed bad solder joints and defective memory chips. I must say, however, their customer service people were very helpful. Even at this point, I can not over emphasize the desirability of this print buffer option. Having a 68k (64k Memory Mate™ + 4k in the Sprinter) print buffer is more than sufficient for most applications. Once a large print file is sent to the printer, full control is returned to the computer even though the printer is still printing. It's almost like having a multi-tasking operating system or a printer spooler without, of course, the poor keyboard response associated with spoolers.

The ribbon is easily changed by releasing the spring catch holding it in place. Once the two tractors are unlocked and moved, the ribbon cartridge can be removed and a new one inserted. Realigning the tractors is a fairly simple task.

I found the latching mechanism for the paper friction feed device to be very difficult to disengage. A small lever has to be pulled back with one finger (lack of space to use two) using the right hand while locking it in place with another lever with the left hand.

Being classified as a fairly high speed printer, I was interested in its true print speed. True print speed is measured by printing 80 columns per line, one line after another. The Sprinter rated a true 106 characters per second compared to the Zenith Z-125 at 79 characters per second.

Software:

If you're going to be using this printer with the H/Z-100 type systems, I would highly recommend that the applications pack be obtained for it also. This pack contains MPI's Artisan software as well as a parallel interconnect cable.

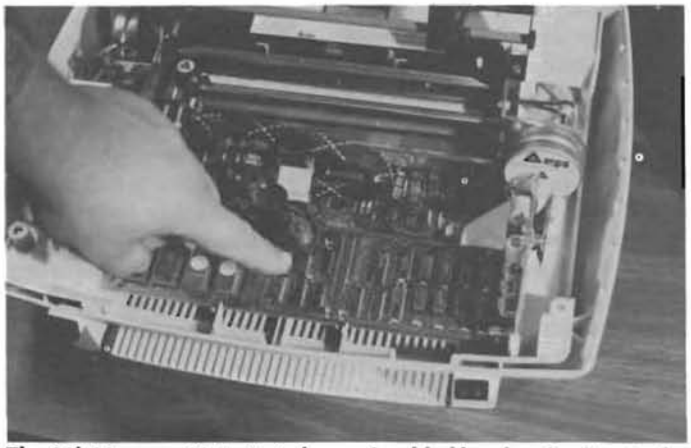
Once the Artisan software is installed, the entire H/Z-100 screen can be printed, no matter what's being displayed. Artisan also has the ability to print a wide variety of type fonts which are included on the supplied diskette. Thirty-three additional different fonts are also available on a separate diskette from MPI dealers, as well as Heath Electronic Centers. Included with Artisan is a program called CALLIGRAPHER. This program allows you to create characters for your own personalized font file. The nice part about using Artisan is that it can be used with WordStar. By utilizing the different fonts and font sizes, various page heading formats can be created, as well as emphasized portions of the document being created.

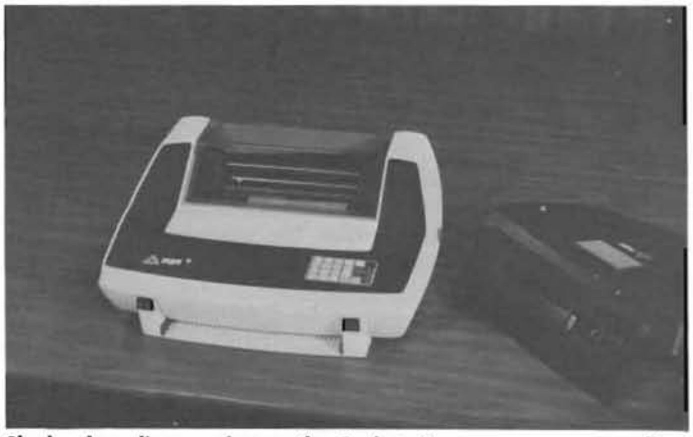
Documentation:

The manual I received for the printer was not satisfactory. It fell short of explaining, in detail, the operation of several functions of the "User Friendly Keyboard," as well as information on downloading and accessing different character sets. Typographical errors, as well as contradictory information was also found. Technical information, including schematics would also be a desirable addition. It appeared that the manual was "rushed" so the printer could ship on time. My observation was confirmed by a representative of MPI's customer service department.

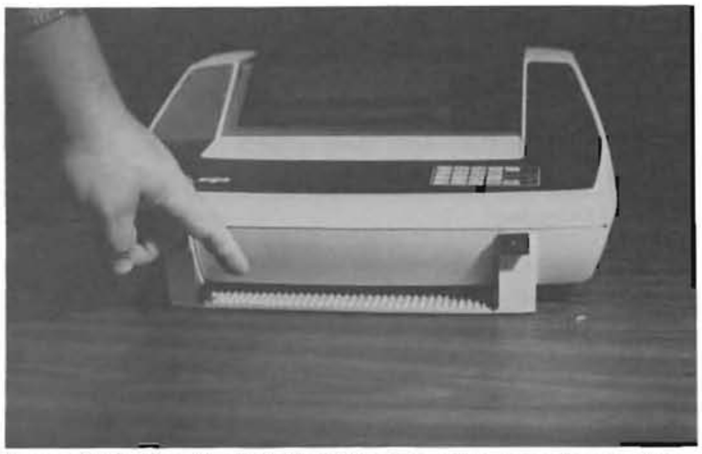
Summary:

Even with some of its shortcomings, the Sprinter is a welcomed addition to the H/Z-100 line of computers. If the serial interface is added to the Sprinter, it will also complement an H/Z-89/90 or H-8 system. For about the same price as the original H-14 sold for, the Sprinter provides an order of magnitude of additional features, as well as, speed and high quality printing.


The 64k Memory-Mate (tm) shown, is added by plugging it onto the mother board.


Sleek, clean lines, enhance the Sprinter's appearance, as well as keep noise to a minimum during printing.

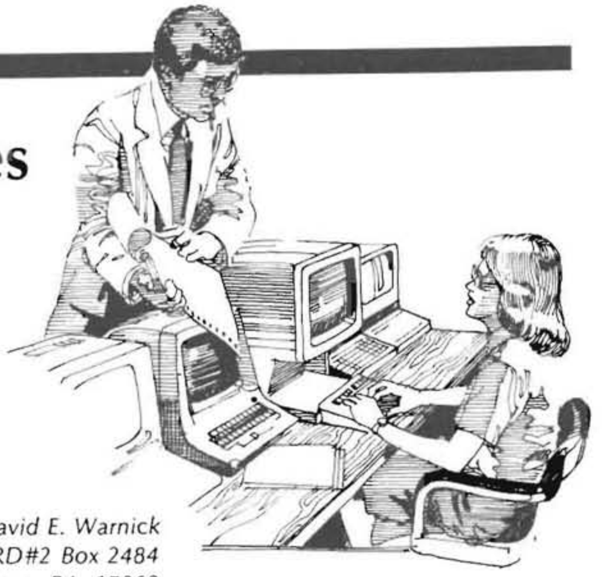
The user-friendly keypad allows the setting of options with minimum hassle. Musical feedback is provided with each key stroke.


Front, single sheet feed, is desirable for word processing applications. Tractor fed paper can also be fed here, through the bottom, or from the back side.

With the cover in place, the Sprinter looks like a portable typewriter and weighs only 16 lbs.

Part 4 - Adding More Features

Practical File Management



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

This month we'll start by providing a complete print module for our file handling program. The module we provided last month always started at the beginning of our data file and continued to its last entry. The complete program will let us select a starting and ending date for the chart. If a starting date which does not exist in the file is selected, the program will pick the last entry date prior to the one requested. If there is none, the chart will start at the beginning of the file. The same philosophy applies to the ending date. If a nonexisting date is chosen, the chart ends on the next file entry after that date. This way the chart will never contain less information than requested unless that information doesn't exist in the file.

Figure 1 shows the steps required to print the chart. When the letter P is selected from the menu, program execution is directed to line 17000. That's where we'll put our print module. With the screen set up in the by now familiar Month, Date, Year layout, the operator is requested to input a month for the beginning of the chart. The option is given to press the return key if we want to begin with the first entry in the file. Line 17060 takes a single character using INPUT$(1). The test for the return key is made on line 17070. If the return key (ASCII character no. 13) was pressed, the info is kept as a "flag" (FR$="Z") and program execution is directed to the ending part of the routine. If return was not struck, the first character struck is saved. The test for the length of M$ on line 17100 forces 2 characters to be taken.

Lines 17120-17160 and 17170-17210 take the date and the year. As the return option is no longer available, they can use INPUT$(2). The entire process is then repeated for a chart ending date on lines 17240-17430. Each time a starting or ending date is completed, the information is placed on the screen by line 17220 or 17420.

Figure 1

```
17000 PRINT FNCA$(2,1);EE$
17010 PRINT FNCA$(3,21);"YOU HAVE SELECTED THE PRINT OPTION"
17020 PRINT FNCA$(10,21);
  "INPUT MONTH TO START THE CHART AS 2 DIGITS"
17030 PRINT FNCA$(11,21);
  "(01 - 12) OR RETURN FOR BEGINNING OF FILE"
17040 MS$=""
17050 PRINT FNCA$(14,21);"MONTH  ";
17060 M$=INPUT$(1)
17070 IF M$=CHR$(13) THEN PS$="AT THE BEGINNING OF THE FILE"
  :FR$="Z":GOTO 17220
17080 MS$=MS$+M$
17090 PRINT M$;
17100 IF LEN(MS$)<2 GOTO 17060
17110 PRINT FNCA$(10,1);EL$;FNCA$(11,1);EL$
17120 PRINT FNCA$(10,21);
  "INPUT DATE TO START THE CHART AS 2 DIGITS (01 - 31)"
17130 PRINT FNCA$(16,21);"DATE  ";
17140 DS$=INPUT$(2)
17150 PRINT DS$
17160 PRINT FNCA$(10,1);EL$
17170 PRINT FNCA$(10,21);
  "INPUT THE LAST 2 DIGITS OF THE YEAR TO START THE CHART"
17180 PRINT FNCA$(18,21);"YEAR  ";
17190 YS$=INPUT$(2):PRINT YS$
17200 FR$=YS$+MS$+DS$
17210 PS$=MS$+"-"+DS$+"-"+YS$
17220 PRINT FNCA$(5,21);"CHART WILL START ";PS$
17230 PRINT FNCA$(10,1);EE$
17240 PRINT FNCA$(10,21);
  "INPUT MONTH TO END THE CHART AS 2 DIGITS"
17250 MF$=""
17260 PRINT FNCA$(11,21);
  "(01 - 12) OR RETURN FOR END OF FILE"
17270 PRINT FNCA$(14,21);"MONTH  ";
17280 M$=INPUT$(1)
17290 IF M$=CHR$(13) THEN PE$="AT THE END OF THE FILE":
  LR$="Z":GOTO 17420
17300 MF$=MF$+M$:PRINT M$;
17310 IF LEN(MF$)<2 GOTO 17280
17320 PRINT FNCA$(10,1);EL$;FNCA$(11,1);EL$
17330 PRINT FNCA$(10,21);
  "INPUT DATE TO END THE CHART AS 2 DIGITS (01 - 31)"
17340 PRINT FNCA$(16,21);"DATE  ";
17350 DF$=INPUT$(2):PRINT DF$
17360 PRINT FNCA$(10,1);EL$
17370 PRINT FNCA$(10,21);
  "INPUT LAST 2 DIGITS OF YEAR TO END THE CHART"
17380 PRINT FNCA$(18,21);"YEAR  ";
17390 YF$=INPUT$(2):PRINT YF$
17400 LR$=YF$+MF$+DF$
17410 PE$=MF$+"-"+DF$+"-"+YF$
17420 PRINT FNCA$(7,21);"CHART WILL END ";PE$
17430 PRINT FNCA$(10,1);EE$
```

We then continue with Figure 2. This is where we translate the information from the steps above into something the computer will understand, random file record numbers. For the start date, if the flag FR$="Z" was set, we assign the first record number, FR%, as 1. Otherwise we must use the binary search routine on lines 17450-17520 to find the record number for starting. If a mismatch is not found on line 17510 or 17520, the data requested exists and the record number is set on line 17530. If a mismatch is always found, the upper limit of the search will pass the lower limit. When this happens, line 17470 sets the first record number to the lower of the values. This gives the last entry prior to the requested date.

The whole process is repeated on lines 17540-17610 for the last record to be included in the chart. Finally, lines 17620 and 17630 assure that we won't get a BAD RECORD NUMBER error by preventing a record number less than 1 or greater than the highest numbered record in the file from being specified.

Figure 2

```
17440 IF FR$="Z" THEN FR%=1:GOTO 17530
17450 A%=1
17460 B%=NR%
17470 IF A%>B% THEN FR%=B%:GOTO 17530
17480 C%=INT((A%+B%)/2)
17490 GET #1,B%(C%)
17500 IF A$<FR$ THEN A%=C%+1:GOTO 17470
17510 IF A$>FR$ THEN B%=C%-1:GOTO 17470
17520 FR%=C%
17530 IF LR$="Z" THEN LR%=NR%:GOTO 17620
17540 A%=1
17550 B%=NR%
17560 IF A%>B% THEN LR%=A%:GOTO 17620
17570 C%=INT((A%+B%)/2)
17580 GET #1,B%(C%)
17590 IF A$<LR$ THEN A%=C%+1:GOTO 17560
17600 IF A$>LR$ THEN B%=C%-1:GOTO 17560
17610 LR%=C%
17620 IF FR%<1 THEN FR%=1
17630 IF LR%>NR% THEN LR%=NR%
```

Figure 3 completes the steps required for the chart print function. Because we want to spread our curve over 50 spaces (I assume the use of an 80-character wide printout here), we must know what its upper and lower limits are. To find this, we first set impossible limits. We set the lower value, LV, to 999 and the high value, HV, to zero. On lines 17660-17680 we dimension arrays to hold each date, A$( ), and each value, V( ), to be printed. Next on lines 17690-17760 we read the data from our data file for the range of dates to be included on the graph. Data is placed in the arrays and the value for each date is compared to the present high and low values. If it is greater than the high or less than the low, the appropriate variable, HV or LV, is changed. When the last record has been read, LV will contain the lowest value and HV will contain the highest value to be included in the output to the printer.

Lines 17770-17790 are another operator error check. How often have you sent something to the printer when it was turned off. These lines are written in an attempt to prevent such an error. Lines 17800-17890 provide a header for the chart. (Once again HDOS users, LPRINT is for CP/M and tells the operating system to send print data to the line printer. For HDOS you'll have to include OPEN #3,LP: early in the program. Line 300 would be fine. Every time you see LPRINT change it to PRINT #3,. Don't forget to LOAD LP: before you load MBASIC.)

Lines 17900-17930 go through a loop for every set of data we've placed in the arrays. Line 17910 puts the date on the left margin of the paper. Line 17920 places an asterisk (*) at the appropriate place on the chart. V(X%)-LV is the offset from the base of the chart. It is divided by the value spread of the whole chart (HV-LV) to determine the percent of the whole chart. This is converted to spaces away from the base of the chart by multiplying the Percentage-Of-Chart by the Spaces-In-Chart, 50. We add 15 because the base of the chart is 15 spaces from the left margin of the paper. Finally line 17920 prints the actual value beginning on column 70. When the graph is completed, line 17940 sends a form-feed to the printer. Refer to your computer manual for the chart of ASCII characters and you'll see that character 12 is the form-feed. Verify this in your printer manual. The purpose of this line is to advance the paper to the next page when the plotting is done. Line 17950 returns execution to the menu.

Figure 3

```
17640 LV=999
17650 HV=0
17660 Z%=LR%-FR%+1
17670 ERASE A$,V
17680 DIM A$(Z%),V(Z%)
17690 FOR X%=FR% TO LR%
17700 W%=X%-FR%+1
17710 GET #1,B%(X%)
17720 A$(W%)=A$
17730 V(W%)=CVS(B$)
17740 IF V(W%)<LV THEN LV=V(W%)
17750 IF V(W%)>HV THEN HV=V(W%)
17760 NEXT X%
17770 PRINT FNCA$(11,21);
  "MAKE SURE THE PRINTER IS READY AND TURNED ON"
17780 PRINT FNCA$(13,21);
  "THEN PRESS ANY KEY TO PRINT THE CHART"
17790 P$=INPUT$(1)
17800 LPRINT "PERFORMANCE OF ";F$
17810 LPRINT "CHART STARTS ";PS$
17820 LPRINT "CHART ENDS ";PE$
17830 LPRINT "THE LOWER LIMIT OF THE CURVE IS ";LV
17840 LPRINT "THE UPPER LIMIT OF THE CURVE IS ";HV
17850 LPRINT
17860 LPRINT
17870 LPRINT TAB(10)"LOWER LIMIT";TAB(37)"MIDDLE";
  TAB(60)"UPPER LIMIT"
17880 LPRINT TAB(10) LV;TAB(37) (LV+HV)/2;TAB(60) HV
17890 LPRINT TAB(15)"V";TAB(40)"V";TAB(65)"V"
17900 FOR X%=1 TO Z%
17910 LPRINT MID$(A$(X%),3,2);"-";RIGHT$(A$(X%),2);"-";
  LEFT$(A$(X%),2);
17920 LPRINT TAB(INT(((V(X%)-LV)/(HV-LV))*50)+15)"*";
  TAB(70) V(X%)
17930 NEXT X%
17940 LPRINT CHR$(12)
17950 GOTO 1000
```

This module will probably be the one you modify most as each application of your file management system requires a different form of output from charts to tables to address labels. Add all the programming from lines 17000-17950 to your program. Don't forget the backups.

Run the program, specify a file name, and drives containing data. Select the print option and perform the following tests.

1) Enter RETURN for start and stop dates. The whole file should print.

2) Using the graph from test 1, enter existing dates for the start and stop date. The chart should begin and end where requested.

3) Enter dates between existing dates. The chart should begin on the last entry before the requested beginning date and end on the first entry following the requested ending date.

4) Enter dates before and after any in the file. The entire file should print just as it did in test 1.

The remaining enhancements to this program will allow us to change, delete, or look up information which has been placed in the file. Figure 4 is the beginning of our change module. When completed it will permit us to change the value for any date existing in the file. Dates will not be changed by this routine as they are key items. Because key items are not affected by the change option, it will not be necessary to perform a sort when it is complete. As with any of the options which require finding information existing in the file, however, it is necessary to have a sorted file prior to selecting the change option. The primary purpose of the change routine is to correct errors in values entered previously. While this may have little use in this application, it is important to see how it works, as other applications (name and address files, inventory files, etc.) rely heavily on the ability to go into the file and modify information existing there.

Beginning on line 5000 we set up the month-date-year screen and request a 2-character input. Line 5100 tells us to press the red key if we're finished making changes and want to go back to the menu. The operator is instructed to input the date as 2 digits on line 5110. Again I'll stress the importance of leading the program user through his work step-by-step. It doesn't take much. Just dedicate a line or two of the screen to instructions as we've done here. When an operator action is required, put the instruction there. As soon as the program verifies that the instructions have been followed, erase those lines.

The input operations are repeated for the date and the year. The red key option is not permitted at this point, so the test is not included. When all the inputs have been received, they are combined in the order of year, month, and date to agree with the date field of each record. This information is contained in RC$ (Record to Change).

Figure 4

```
5000 PRINT FNCA$(2,1);EE$
5010 PRINT FNCA$(3,21);"CHANGE OPTION SELECTED"
5020 PRINT FNCA$(9,21);"MONTH"
5030 PRINT FNCA$(11,21);"DAY"
5040 PRINT FNCA$(13,21);"YEAR"
5050 PRINT GM$
5060 PRINT FNCA$(10,27);"zz"
5070 PRINT FNCA$(12,27);"zz"
5080 PRINT FNCA$(14,27);"zz"
5090 PRINT GO$
5100 PRINT FNCA$(24,21);
  "PRESS THE RED KEY TO RETURN TO THE MENU";FNCA$(1,1)
5110 PRINT FNCA$(18,21);
  "ENTER THE MONTH TO CHANGE AS 2 DIGITS (01-12)"
5120 PRINT FNCA$(9,27);
5130 M$=INPUT$(2)
5140 IF M$=CHR$(27)+"Q" GOTO 1000
5150 PRINT M$
5160 PRINT FNCA$(18,1);EE$
5170 PRINT FNCA$(18,21);
  "ENTER THE DAY TO CHANGE AS 2 DIGITS (01-31)"
5180 PRINT FNCA$(11,27);
5190 D$=INPUT$(2)
5200 PRINT D$
5210 PRINT FNCA$(18,1);EL$
5220 PRINT FNCA$(18,21);
  "ENTER THE LAST 2 DIGITS OF THE YEAR TO CHANGE"
5230 PRINT FNCA$(13,27);
5240 Y$=INPUT$(2)
5250 PRINT Y$
5260 RC$=Y$+M$+D$
```

Figure 5 completes our change procedure. We begin with a binary search to find the record for the date requested. If that date doesn't exist, B% will become greater than A%. If that occurs, line 5300 directs program execution to line 6000 where a message is printed and execution stops until any key is pressed. This gives us time to think about it before continuing. When the key is struck, program execution goes back to the request for an input.

If the record does exist it will be assigned to RN% on line 5350. The information which existed in the file is placed on the screen by lines 5360 and 5370, and a new value is requested with all the necessary information by lines 5380-5430. Once again there is a provision to bypass the data changing operation by pressing the return key. There's no need to key in data if the existing data is correct. Line 5440 tests the value of the input variable, V. As RETURN is less than zero on the ASCII table, new information would not be placed in B$ and the data taken from the file would remain there. If numeric data had been keyed in, it would be greater than zero and would be placed in B$. Line 5450 writes the data back to the same record number from which the GET command took it during the binary search. The screen is then reset and execution is returned to line 5100 to make the next change.

Figure 5

```
5270 PRINT FNCA$(18,1);EL$
5280 A%=1
5290 B%=NR%
5300 IF A%>B% GOTO 6000
5310 C%=INT((A%+B%)/2)
5320 GET #1,B%(C%)
5330 IF A$<RC$ THEN A%=C%+1:GOTO 5300
5340 IF A$>RC$ THEN B%=C%-1:GOTO 5300
5350 RN%=C%:V=0
5360 PRINT FNCA$(18,21);"THE PRESENT ENTRY FOR ";
  M$;"-";D$;"-";Y$
5370 PRINT FNCA$(19,21)"IS ";CVS(B$)
5380 PRINT FNCA$(21,21);"NEW VALUE"
5390 PRINT FNCA$(23,16);
  "1/8 = .125 1/4 = .25 3/8 = .075 1/2 = .5"
5400 PRINT FNCA$(24,21);"5/8 = .625 3/4 = .75 7/8 = .875";
  FNCA$(1,1)
5410 PRINT FNCA$(20,10)
  "ENTER THE NEW VALUE IN DECIMAL FORM, OR RETURN ";
5420 PRINT "FOR NO CHANGE";FNCA$(21,31);
5430 INPUT "",V
5440 IF V>0 THEN LSET B$=MKS$(V)
5450 PUT #1,RN%
%½@? FOR X=1 TO 500:NEXT X
5470 PRINT FNCA$(18,1);EE$;FNCA$(9,26);LE$;FNCA$(11,26);
  LE$;FNCA$(13,26);LE$
5480 GOTO 5100
6000 PRINT FNCA$(18,21);"NO ENTRY EXISTS IN THIS FILE FOR ";
  M$;"-";D$;"-";Y$
6010 PRINT FNCA$(20,21);"PRESS ANY KEY TO CONTINUE"
6020 W$=INPUT$(1)
6030 PRINT FNCA$(18,1);EE$
6040 GOTO 5470
```

Add program lines 5000-6040 to your file handler and save it. Run the program, make a printout of your file for reference, then change some of the values found there. When you make a change, note it on the printout. Request dates which do exist and others which aren't in the file to make a thorough test of its operation. After you've completed some changes, press the red key when told to and print the chart again. Make sure the changes you made are on the new chart.

We're really getting somewhere now. The program, as you have it now, is capable of handling stock values and plotting their performance on a graph. Or, how about storing and plotting the growth (height or weight) of your child. Enter that value rather than a price. It will work just fine. The program will keep track of any number which changes with dates. If you're going to go beyond the limits of zero and 999 with that number you'll have to change lines 17640 and 17650.

The only features left to be included in our program are Lookup and Delete. We'll take care of those next month. After that we'll shrink the size of the program by reducing duplication of programming lines to a minimum. This will be appreciated by owners of 48K systems as there's not too much room on top when the program is loaded, and next month's additions would run out of memory without the shrinking. You'll be surprised at how much RAM we can save and how easy it is to do. For now, have fun with this new program and apply it to as many things as you like. The real power of your computer is at your fingertips.

See you next month.

☜ Vectored from 8
that if X-radiation is strong enough, it will affect magnetic storage media such as tapes and diskettes. But how strong is strong enough? The main concern is over the luggage inspection devices installed in airports for security checks of carry-on items. And since we all know what can happen to checked baggage, we tend to want to carry our valuable data tapes and diskettes with us. The question is, how can we carry magnetic materials through airport security checks without undue risk. Must we always carry diskettes and tapes separately; or are there alternatives? Naturally, we can choose to hand-carry magnetic media past the X-ray machines, but this is slow and cumbersome, and often causes problems, especially in foreign airports. Ever try to explain to a Turkish guard why you should not pass a box of disks through the X-ray machine?

This is not the place to dwell on the mathematics or the physics of the problem, but I'll share some practical experience instead. Over the past seven years I have either carried, or directed to be carried, hundreds of full size computer tapes, cassettes, 8" disks and 5 1/4" disks. All have passed the dreaded X-rays with no loss of data at all. So whence the horror stories of people who have carried disks in briefcases, only to find at the final destination that the data was damaged?

First, keeping in mind that the airport X-ray machines are generally fairly low powered devices, there appear to be three main reasons why a disk might loose it's data. All have to do with defective equipment or materials. The most likely explanation is that the disk wasn't in satisfactory condition at the start of the trip, after its last recording. It may have had a weak area or magnetic domain that was incapable of retaining the magnetic record written to it, and lost the recorded pattern spontaneously, or the weak area may have been sufficiently susceptible to the influence of the X-rays to lose the pattern. Such a disk might well have only a small "bad spot," making it possible to recover almost all of the material. Naturally, this clearly would be practical only in the case of ASCII text. If the disk were of sufficiently poor quality, the defective areas could be quite extensive, making recovery impossible. This needn't be a factory defect; wear on the disk surface can have the same results in terms of loss of data.

A more likely source of damaging radiation is the motor system that drives the conveyor belt that carries your bag through the X-ray machine. The motor is likely to produce a more powerful electromagnetic force than the X-ray machine, and while the latter is usually a foot or more away from the disk, the motor may be only an inch or so. Even the simple act of setting your briefcase on the end of the conveyor belt table to open it and remove the disk may be sufficient to zap the disk. It depends on where the motor is located. If you want to have the disks hand checked and not run under the X-ray machine, remove them from your luggage before you get to the end of the belt.

Finally, there's the possibility that the disk drive used to make the disk was defective in that a weak pattern was written. The resulting disk could have a great many "bad spots" instead of just a localized area.

Of increasing interest to users of portable computers is the effect of X-rays on hard disks. Personal experience fails me; I don't have a portable with a hard disk. Actually, I have neither a portable not a computer with a hard disk. However, much the same reasoning applies. Be especially cautious about avoiding magnetic fields. Most hard disk systems installed in the portables I've seen are inside a shielded box to protect the magnetic storage, but some have been in plastic, sufficient only for dust protection. Keep in mind that the much more densely packed data will be more vulnerable to the direct effects of the X-ray machine, and that the metal disk on which the magnetic media is deposited will also play a part. This base will respond to X-rays with more scattering than the mylar base of a floppy disk, and induced magnetic fields are a possibility. On the whole, in the absence of experimental results, it's best to carry your portable through the inspection area, keeping away from the X-ray machine altogether. X-rays should have no effect on portables like the TRS-80 Model 100, which store data in CMOS (low-power, always-on RAM.)

What can the normal computer user do about any of this? The first and easiest measure, is to always use quality disks for valuable data and programs. It stands to reason that "no-name" bulk sales disks are riskier than "name brands," simply because the maker has no reputation to lose. Dysan, Maxell, Wabash and the other major makers do have a reputation, and test their disks more carefully. Who knows, some of the "no-name" disks may be rejects from big-name makers. Disks are cheap, but the information they contain can be very expensive. Look at the brands used by the major software vendors.

Many people advocate the use of metal boxes for carrying disks. Often the use of lead foil envelopes is recommended. If you suspect the quality of your disks or the quality of your drive's recording capability, the use of the foil bags may help. If the container is of sufficient quality for high-speed photographic film, it should be effective for magnetic media. I've never found it necessary to use this approach, so I can't comment on its effectiveness. Personally, I'd prefer the use of a magnetically shielded box, for reasons I explained above. They're not easy to find, but the peace of mind they may bring you might be worth the effort and cost.

If your drives routinely produce disk copies that cannot pass through the X-ray inspection without a failure, you have good reason to suspect a drive problem. Often there will be other symptoms, such as the inability to create a disk that another, similar computer cannot read. Accumulated dirt on the head may be a possibility, as may head misalignment. Both are more likely in older, more worn drives. Drives which use stepper motors to position the head are subject to having the internal magnets shift slightly over time and with heavy use. Voice coil actuators can weaken. Have them cleaned and checked by someone who knows what they are doing. A poor cleaning job is worse than no cleaning at all, and the alignment procedures take special equipment and a fair degree of skill. I rarely clean my drive heads, but when I do, I use grain alcohol (NOT rubbing alcohol) and a cotton swab. I usually find very little dirt on the heads. My personal preference is to avoid cleaning disks, as much as possible.

When you check out your drives, be sure that you do it in accordance with the manufacturer's guidelines. Consult the drive maker's maintenance manual. If you didn't get one with your computer, someone short-changed you. That's as important a book as the operator's manual. Ask your dealer for one, or write the maker. They're often $20.00 to $30.00, which is why not all computer vendors provide them. IBM is a good example. They have one available, but they'll charge you for it.

On another topic. The notion that you can safely use both sides of your single-sided disks by punching the appropriate holes and flipping the disk over, should be laid to permanent rest. I still see this dangerous procedure being recommended from time to time, as a procedure to gain extra storage space when using single-sided drives. Disks are designed to rotate in only one direction. Iron oxide, dust and other contaminants that accumulate on the disk are cleaned off and trapped by the fiber matrix inside the plastic disk jacket.

Vectored to 83 ☞

Interrupts and CP/M
Advanced Assembly Language Techniques

Robert W. Lewis
P.O. Box 273
Chester, MD 21619

During the development of dCOM, an all-mode communications monitor package capable of copying CW, Baudot and ASCII codes, it was necessary to implement hardware interrupts while using CP/M. The Heath version of CP/M (unlike HDOS) makes no provision for use of the H-8 or H/Z-89 user interrupt system. When I called Heath Systems Software, I was told that it is not possible to use the interrupt system with CP/M. After a detailed study of the Heath CP/M BIOS listing and a good deal of experimentation, I was able to devise a method of implementing the interrupts. The information presented here will be of value to the assembly language programmer who wishes to utilize real-time hardware interrupts with an H-8 or H/Z-89 running CP/M. A complete understanding of 8080 assembly language and the Heath interrupt system is assumed throughout this presentation. Extreme caution should be used when debugging the interrupt portion of application software, since a bug can cause system crashes capable of corrupting any or all data on a disk (including the CP/M tracks). Be sure that you have backup copies of everything. Use a COLD boot (shift-reset) to recover from any crash. This will ensure that the complete operating system is re-loaded from the disk.

During the CP/M cold boot operation, the operating system is loaded from the disk into RAM. It is possible for your assembly language program to modify the interrupt vectors which are located in CP/M page zero. They will remain as set until you either change them again or perform a cold boot to reload from the disk. It is important to note here that your program MUST provide an orderly exit to CP/M, changing the vectors back to their original condition prior to performing a warm boot. A warm boot itself will not re-load page zero from the disk. Failure to reset the vectors will cause all sorts of problems when you load and run some other program. Interrupt level 5 is not used by CP/M and is presently disabled by the following code located in page zero:

```
        EI                  ; Re-enable the interrupt
        RET                 ; Return to whatever was happening
```

The following code in your program will modify page zero so that it vectors to your level 5 interrupt service routine (which we will label INT5) whenever a level 5 interrupt occurs. You end your routine with EI and RET in the normal manner.

```
JMPINS  EQU     0C3H        ; OPCODE for a JMP instruction
INT5VEC EQU     0029H       ; Interrupt 5 vector address in page
                            ; zero.

        DI                  ; Disable all interrupts
        MVI     A,JMPINS    ; Get JMP code
        STA     INT5VEC-1   ; Write it to the vector
        LXI     H,INT5      ; Get service routine address
        SHLD    INT5VEC     ; Write it to the vector
        EI                  ; Enable all interrupts
```

In addition to interrupt level 5, dCOM was needed to update several timing functions every two milliseconds. This was accomplished by altering the 2 Ms interrupt vector in CP/M page zero, causing the processing to pass through our routine (CLKINT) on the way to the CP/M clock service routine. The following code will accomplish this:

```
INTMS   EQU     0009H       ; Clock interrupt vector address in
                            ; page zero.

        DI                  ; Disable all interrupts
        LHLD    INTMS       ; Get CP/M clock service address
        SHLD    CLKVEC+1    ; Tell our routine where it is
        LXI     H,CLKINT    ; Get our clock routine address
        SHLD    INTMS       ; Tell CP/M where it is
        EI                  ; Enable all interrupts
```

The last line of our clock service routine provides the jump to CP/M's service routine. Note that the interrupts are NOT enabled by our routine. This is left to the CP/M service routine. Our routine looks like this:

```
CLKINT  (Service routine
         code as required)

CLKVEC  JMP     0       ; Bytes 2 & 3 of this instruction are
                        ; written to by the preceding routine
                        ; in order to provide a valid address.
```

The following exit code can be used to return the modified vectors to their original condition and then return to CP/M.

```
EIINS   EQU     0FBH        ; OPCODE for an EI instruction
RETINS  EQU     0C9H        ; OPCODE for a RET instruction

EXIT    DI                  ; Disable all interrupts
        LHLD    CLKVEC+1    ; Restore CP/M
        SHLD    INTMS       ;   clock vector

        MVI     A,EIINS     ; Disable
        STA     INT5VEC-1   ;  CP/M
        MVI     A,RETINS    ;   level 5
        STA     INT5VEC     ;     interrupt vector.

        EI                  ; Enable all interrupts
        JMP     0           ; Warm boot
```

Now for the real trick. CP/M service call subroutines utilize their own small stack area. When you initialize a call to BDOS (0005H), the stack pointer is moved from your program stack to one of these small areas. If an interrupt occurs during this time and your service routine begins pushing registers on the stack, it will overflow, writing over CP/M code. This results in a system crash (during one experiment the disk even had to be re- sysgened before it would boot again). The trick is to make your interrupt service routines reset the stack pointer to their own special stack area, thereby preserving the BDOS stack, as well as your system stack. This is accomplished with the following code:

```
; This sets up a new stack and saves the old.

INT5     SHLD  HSAV        ; Save HL
         POP   H           ; Save
         SHLD  RETSAV      ;  return address.
         PUSH  PSW         ; Save processor status word
         LXI   H,0         ; Move SP
         DAD   SP          ;  to HL.
         LXI   SP,LCLSTK   ; Move SP to local stack area
         PUSH  H           ; Save old SP
         PUSH  D           ; Save DE
         PUSH  B           ; Save BC

; (Your service routine code goes here)

; This restores status and the old stack

         POP  B            ; Restore BC
         POP  D            ; Restore DE
         POP  H            ; Reset SP to
         SPHL              ;  to old stack.
         POP  PSW          ; Restore processor status word
         LHLD RETSAV       ; Restore
         PUSH H            ;  return address.
         LHLD HSAV         ; Restore HL
         EI                ; Re-enable interrupts
         RET               ; Return to whatever was happening

; This is your local stack area

         DS      20        ; 20 Bytes (or whatever you need)
                           ; for your stack.
LCLSTK   DS      1         ; Label the top of your stack
RETSAV   DS      2         ; 2 Bytes to save return address
HSAV     DS      2         ; 2 Bytes to save HL
```

One local stack area may be used for all of your interrupt service routines, as long as you do not re-enable interrupts until the end of your routine. This ensures that a service routine will not be interrupted causing the same stack area to be utilized twice.

I hope that this information will encourage the development of real-time application software for the H-8 and H/Z-89 using CP/M.

Note: dCOM is available from the author for $24.95 plus $2.00 shipping and handling.

Heath/Zenith Users' Group

NEW Z-DOS WORDKEY™ SIMPLIFIES WORDSTAR™

Introducing WordKey. WordKey replaces **ALL** of WordStar's control-key commands with much simpler and faster keypad and function-key commands. Each function and keypad of your H/Z-100 is used two or three times over to define more than 90 key combinations. You'll never need to hold down the control key again! Look at these features:

- Keys are logically grouped so that the most often-used commands are single keystrokes. Move around the screen quickly and easily. Lesser used commands are two separate keystrokes that can be entered with one hand.
- Full onscreen help is always instantly available, including two different key diagrams and a separate explanation of each key's function. Just press the HELP key.
- Blank ruler line...Block cursor and key-click on/off...Detailed User's Manual and printed keyboard diagrams...And much more.
- Use with either WordStar version 3.21 or 3.30.

Please send me:
Z-DOS WordKey: _____ copies @ $49.95 $ ________
Special: C.Itoh Prowriter graphics screen
dump program _____ copies @ $19.95 $ ________
Calif. residents please add 6% tax $ ________
Total Enclosed (includes mailing) $ ________

Name ________

Address ________

____

DelSoft
Gary Deley, 564 Calle Anzuelo, Santa Barbara, CA 93111
(805) 967-9566 eves and weekends REM

H/Z-100 AND 150 PC GRAPHICS SOFTWARE

MICROSERVICES continues to expand its support of Heath/Zenith computers.

For H/Z-100s:

ZANIMATE. Make color animation sequences using ZBASIC **$64.95**

ZPALETTE. Draw images in 92 colors using ZBASIC. **Improved** with three painting modes and graphics generator **$59.75**

ZPATTERN. Test your color monitor **$24.95**

Z3D from **COLORWORKS.** Make three-dimensional objects in color **$75.00**

For H/Z-100s and 150 PCs:

SOFTKITS from **KERN INTERNATIONAL.** We are now carrying a full line of this excellent series of books and disks for graphics and business/scientific applications.

SNAPSHOT. Print screen images on your black/white or color printer **$65.00**

DROP BY OUR BOOTH FOR SPECIAL HUGCON DISCOUNTS!

MasterCard

MICROSERVICES
P.O. Box 7093
Menlo Park, CA 94026
Phone: (415) 851-3414

VISA

Include 3% p/h ($3.00 min.) California add 6½% tax.

Improving Graphics on MPI Printers

Pat Swayne
Software Engineer

I recently acquired a new addition to my crowded office -- an MPI-150 printer. The MPI family of printers, now featured in Heathkit catalogs, offers some excellent choices for anyone who needs a printer capable of high resolution graphics. They are fast, easy to program, and the AP-PAK software provided by MPI makes them very versatile. However, the MPI company seems to have made a poor choice in selecting the tractor assemblies they use on their printers. The tractor assemblies are the mechanisms that pull the paper upward via the punched holes on either side. Figure 1 illustrates the problem they cause. This figure was made by having my MPI-150 printer draw 20 bars using 5 wires in the printhead (5 dots high) and then advancing the paper the distance of 6 dots. The horizontal resolution was set to 50 dots per inch. The bars should have been equally spaced on the paper, but as you can see, the paper was not pulled upward the same amount each time it was advanced.

I also have a Z-25 printer in my office, and I noticed that the tractor assemblies on it would probably fit on the MPI, and that they appeared to be made better. I got a couple from the Heath Parts department, found that they did fit on the MPI-150, so I printed the graphic bars again. Figure 2 shows the result. As you can see, the bars are much more evenly spaced. I had originally discovered the problem while trying to draw the HUG logo with the MPI, and found that it looked different after every attempt. With the Z-25 tractors installed, it turns out pretty good, as you can see in Figure 3.

MPI uses the same kind of tractor assemblies on its MPI-99, Sprinter, and MPI-150 model printers. If you have one of those, and your graphics don't seem quite right, you may want to replace your tractor assemblies with the Z-25 type. They are available from the Heath Parts department as part no. 266-1076 for the left tractor and part no. 266-1077 for the right tractor. To change the assemblies on an MPI-150, you will have to remove the printer cover, the paper advance motor (two screws), and the screws at each end of the round tractor bar. You will also have to remove the press-fit retainer at the right end of the square tractor bar. You may be able to do this by pressing on the end of the bar with your thumb, or by gently prying it off with a scribe. You can then remove the two bars and replace the tractor assemblies. There are index marks on the square holes in the assemblies (that the square bar goes through) that must be aligned (on the same side of the square bar).

On an MPI Sprinter, the entire tractor bar assembly can be removed via a couple of screws, and the bars can be removed to replace the tractor assemblies. As of this writing, the replacement has not been tried on an MPI-99, so I do not know what would be involved in the replacement on that printer.

Comments on the MPI-150

In the rest of this article, I will present some of my likes and dislikes concerning the MPI-150 printer. One of the things I like about it is that when you do graphic drawings on it, you do not have to send any control characters (except ESCape) or other special characters to the printer. There has been some discussion in the "Buggin' HUG" section of REMark on the "Dreaded HT Bug", which affects graphic drawing with some printers, but you will never have to worry about it with an MPI.

Another thing that I like about the MPI is the AP-PAK software, but I do not like the inconsistencies in the two versions available. There are several more downloadable fonts provided with the CP/M version than with the Z-DOS version, and the fonts are created differently so that you cannot just copy a CP/M font over to Z-DOS and load it.

There are two classes of downloadable fonts supported by the Z-DOS software, called graphic fonts and printer fonts. Graphic fonts must be used in conjunction with the AP-PAK software, but printer fonts can be loaded into the printer and then used without any special software. With the CP/M software, several graphic fonts are provided, but no printer fonts. With the Z-DOS software, two printer fonts are provided in addition to the graphic fonts, but fewer fonts in

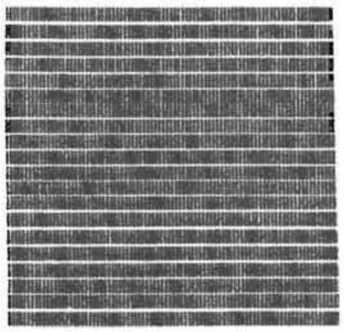


Figure 1. Graphic bars made on an MPI-150 using the original tractor assemblies.

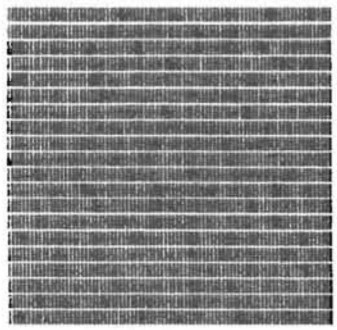


Figure 2. Graphic bars made on an MPI-150 with Z-25 tractor assemblies installed.

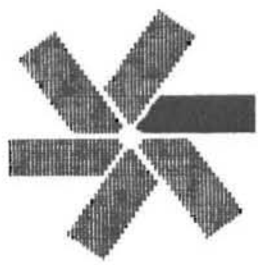


Figure 3. HUG Logo, done on the MPI-150.

all than with CP/M. Since the printer fonts do not require special software, you can copy them over to CP/M and load them using the following BASIC program, which will run in either MBASIC or ZBASIC.

```
10 REM  FONT LOADER PROGRAM
20 E$=CHR$(27):DIM C$(128):ON ERROR GOTO 230
30 PRINT "MPI PRINTER FONT LOADER":PRINT
40 LINE INPUT "ENTER NAME OF FONT TO LOAD: ";F$
50 OPEN "I",1,F$:CLOSE #1
60 OPEN "R",1,F$,128
70 FOR I=0 TO 127:FIELD 1,I AS D$,1 AS C$(I):NEXT I
80 PRINT:PRINT "LOADING FONT...";
90 GET 1,1:S=ASC(C$(6))*256+ASC(C$(5))
100 IF S<>1316 AND S<>2068 THEN PRINT:
    PRINT "BAD FONT":END
110 IF S=2068 THEN C=16:C1=27:S$="F":
    ELSE C=10:C1=43:S$="H"
120 WIDTH LPRINT 255:LPRINT E$;S$;
130 FOR I=7 TO 127:LPRINT C$(I);:NEXT I
140 FOR I=2 TO C:GET 1,I
150 IF EOF(1) THEN PRINT:PRINT "BAD FONT":END
160 FOR J=0 TO 127:LPRINT C$(J);:NEXT J
170 NEXT I:GET 1,C+1
180 FOR I=0 TO C1-1:LPRINT C$(I);:NEXT I
190 PRINT:PRINT "DONE!":PRINT
200 PRINT "SWITCH TO NEW FONT? (Y/N) ";:
    A$=INPUT$(1):PRINT A$
210 IF A$="Y" THEN LPRINT E$;"B";
220 LPRINT CHR$(13);:END
230 IF ERL=50 THEN RESUME 40 ELSE RESUME 240
240 PRINT "ERROR NO.";ERR;"ON LINE";ERL
```

Any printer font (7 by 9 or 11 by 9) that is created with the Z-DOS version of the Calligraphy program can be loaded with the above program.

A couple of things to consider if you are thinking of getting either an MPI-150 or an H/Z-125 (formerly H/Z-25) printer are the facts that the MPI does not support reverse indexing, and that you cannot adjust the printhead to platen clearance in it. That means that if you will be printing multipart forms, you would be better off with the H/Z-125, because you can adjust for the thickness of them, and you can easily align them with the ability to move the paper forward or backward using the front panel. The only way you can move the paper backward in an MPI-150 is to turn the power off and turn the gear on the tractor bar by hand, while pulling the paper out of the bottom of the printer, and you cannot go passed a fold in fan fold paper. In spite of these shortcomings, however, I would recommend the MPI-150 if you will not be doing forms, because it is smaller, quieter, and faster (due to a larger buffer) than the H/Z-125, and has dot graphic capability.

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

LEARN TO PROGRAM H89/H19 GRAPHICS IN MBASIC

BASIC GRAPHICS PRIMER is a computer based course that will teach you fundamental MBASIC programming techniques to control the H19 terminal or H89 computer graphics. Seven lessons will teach you many forms of graphic plots, data plotting, multiple sequential and random access files used for graphics, basic figure animation techniques, and simple computer graphics game design. Each lesson consists of simple illustrative program segments with complete explanations of the programming techniques involved, questions to reinforce the lesson material, and one or more example programs. Most H19/H89 terminal commands are covered in the lessons, including use of the special function keys. **BASIC GRAPHICS PRIMER** is a "HOW TO" course with plenty of examples. If you want to learn how to take full advantage of the H19/H89 graphics features in your programs, **BASIC GRAPHICS PRIMER** is a must. For CP/M only. Requires a printer.

ONLY $49.95!

CP/M version for H89, H8, H19 requires CP/M and MBASIC

NewLine SofTWARE

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

NAME ______________ Send me ____ BASIC GRAPHICS PRIMER
STREET ______________ program(s) for CP/M
CITY ______________ at $49.95 each.
STATE ________ ZIP ________

Check one: ☐ payment enclosed ☐ send COD (add $3.00)
Send order to:
NEWLINE SOFTWARE, P.O. BOX 402, LITTLETON, MA 01460
Foreign orders: add $3.00 Airmail, $10.00 for non-U.S. checks

CP/M is a trademark of Digital Research, Inc.
MBASIC is a trademark of Microsoft, Inc.

HOW MUCH FREE SOFTWARE COULD YOU USE?

FIND OUT WITH OUR GIANT PUBLIC DOMAIN DIRECTORY

- SUPPLIED ON DISK FOR EASY COMPUTER ACCESS
- MORE THAN 4,500 ENTRIES
- SUBJECT AREAS INCLUDE:

ASTRONOMY, AVIATION, BUSINESS, EDUCATION, ENGINEERING, GAMES, GRAPHICS, HAM RADIO, MUSIC, PROGRAMMING, TEXT EDITING, VOICE SYNTHESIS, UTILITIES AND MUCH MORE.

Yes! I need to know what free software is available. Send me the public domain directory on the Heath CP/M* 5¼ format checked ☐ ON 3 HARD SECTORED SD $17.00 ☐ ON 2 SOFT SECTORED DD $12.00 ☐ ON 1 DOUBLE SIDED DD $7.50

HEADWARE
2865 AKRON STREET
EAST POINT, GA. 30344

TERMS: NO RISK, MONEY back guarantee. Add $2 domestic $4 foreign per order for S&H. Enclose your check or M.O. with your order. Sorry, no charge or phone orders.

Name ______________

Address ______________

City, State, Zip ______________

*CP/M Reg. TM Digital Research Corp.

SCORES.BAS

Bob Moskus
2511 Alpine Trail
Huron, OH 44839

SCORES.BAS is a disk file subroutine which can be merged with a game program written in BASIC. It emulates the score-keeping features of arcade video games.

This is accomplished by reading in sequential data from a file, XXXX.DAT, and comparing each OLDSCORE with the NEWSCORE from the game just played. If the NEWSCORE is equal to or greater than the OLDSCORE, then the NEWSCORE is inserted and pushes the remaining scores down one step in the list. Finally, the new data is written to the disk file.

The program will run, as written, under Microsoft MBASIC, ver. 5.2, which allows for variables of up to 40 characters in length. I wrote the program using the long variable names, so that it would be self-explanatory. For other BASICs, change the variables to whatever you like.

To customize for a particular game, change the equates in line 10020 as follows:

N= where X is the number of scores to be kept on file.
GAMES$= where XXXX is the file name of the game.
NEWSCORE= where XX is the variable containing the final score within the game.

If you intend to track 10 or more items, add this line:

```
10025 DIM SCORE(N+1),NME$(N+1)
```

Before using this subroutine, run this short program to create the file XXXX.DAT.

```
10 CLEAR
20 OPEN "O",1,"XXXX.DAT"
30 FOR I=1 TO 5
40 WRITE #1,A,A$
50 NEXT I
60 CLOSE #1
```

Merge this subroutine with your BASIC game program and you are up and running.

The program presented below is "bare-bones" to save space. Enhancements can be added such as clearing the screen, reverse video to highlight the new entry, etc.

```
10010 REM   SCORES.BAS      Disk-file, score-keeping
      subroutine
10020 FLAG=0:N=X:GAME$="XXXX.DAT":NEWSCORE=XX
10030 OPEN "I",1,GAME$
10040 IF EOF(1) THEN 10090 ELSE A=A+1
10050 INPUT #1,OLDSCORE,OLDNAME$
10060 IF FLAG=0 AND NEWSCORE=>OLDSCORE
      THEN SCORE(A)=NEWSCORE:FLAG=A:A=A+1
10070 SCORE(A)=OLDSCORE:NME$(A)=OLDNAME$
10080 GOTO 10040
10090 IF FLAG>0 THEN INPUT "Enter your Name ";NEWNAME$
10100 PRINT:PRINT "TOP SCORES :":PRINT
10110 FOR B=1 TO A
10120 IF B=FLAG THEN NME$(B)=NEWNAME$
10130 PRINT SCORE(B);:PRINT " - "NME$(B)
10140 NEXT B
10150 CLOSE:OPEN "O",1,GAME$
10160 FOR C=1 TO N:WRITE #1,SCORE(C),NME$(C):NEXT C
10170 CLOSE
```

BASICally Speaking

Del Tapparo

(Note: The following material was taken from SMHUG, the Southwest Michigan Heath Users' Group Newsletter, 1054 Blanchard S.W., Wyoming, MI 49509.)

This month's REMark magazine (#47) has a program for generating "automatic titles" in your BASIC programs. This article led me to a short program for automatically "date stamping" BASIC programs within the program itself. Drawing on an old TRS-80 graphics programming technique called "string packing", and using the ZDOS/ZBASIC DATE$, I was able to do it.

Here is a line by line description of what happens. Line 40 defines a string variable that contains a 10 character date compatible with the DATE$ format. Line 70 finds the address of the variable pointer, i.e. the address will contain information about the string, not the string itself. POINTER contains the length of the string, POINTER+1 contains the MSB of the string address, and POINTER+2 contains the LSB of the string address. Line 80 calculates the actual address of the string. Lines 90 through 110 POKE in the new string, DATE$, one character at a time.

After you RUN this program, list it. You will see that the line 40 string definition has changed to the new date, today's date (assuming of course that DATE$ was initialized on power-up). If you merge this short program to the beginning of your ZBASIC program, the date will be automatically updated every time you run the program, but will only be saved when you actually do a SAVE, which is usually after you have just made a change to the program and tested it. I will leave it up to you to enhance the program a bit. How about an "automatic program name" insertion?

```
10  'This self modifying program records the date it was
     last saved
20  '
30  '==========================================================
40  'LASTSVEDON$="12-21-1983"          By Del Tapparo
50  '==========================================================
60  '
70  POINTER =VARPTR(LASTSAVEDON$)
80  ADDRESS=PEEK(POINTER+2)*256+PEEK(POINTER+1)
90  FOR CHARCOUNT = 1 TO 10
100 POKE ADDRESS+CHARCOUNT-1,ASC(MID$(DATE$,CHARCOUNT,1))
110 NEXT CHARCOUNT
```

Vectored from 9

original stack and registers are restored before returning. CLEAR_BUFFER = 1 causes the input buffer to be cleared before returning.

** WARNING ** When CP/M does output to the CONsole or List device (Punch ?), it checks the console input buffer for various control characters. If you input a character during this output, the operating system may grab the character before INKEY$ gets to it. Once the operating system grabs a character other than a legal control character, it will not take any more characters and the CP/M control functions will no longer work.

Sample Fortran usage (to stop a program with an ESC):

```
          integer=0
    100   CALL INKEY$(integer)
          IF(integer.EQ.27) STOP
          <process>
          GOTO 100

    PUBLIC  INKEY$          ; make INKEY$ available

        BDOS          EQU  0005h ; BDOS entry point
        DCIO          EQU  06h   ; Direct Console
                                 ;   I/O function
        F6IC          EQU  0FFh  ; Function 6 input code
        SAVE_STACK    EQU  1     ; 0 if stack and registers
                                 ;    don't need saving
        CLEAR_BUFFER  EQU  1     ; 0 if input buffer
                                 ;   doesn't need clearing

        CSEG             ; Relocatable Code Segment

INKEY$:                  ; *** ENTRY POINT ***

if SAVE_STACK
        SHLD   HLTEMP    ; save (HL), for the result
        LXI    H,0       ; get old stack pointer
        DAD    SP
        LXI    SP,STACK; create a new stack
        PUSH   H         ; save old stack pointer
        PUSH   D         ; save the other registers
        PUSH   B
        PUSH   PSW
else
        PUSH   H         ; save result address
endif

        MVI    C,DCIO    ; function 6 will not echo
        MVI    E,F6IC    ;         the character
        CALL   BDOS      ; call BDOS

if SAVE_STACK
        LHLD   HLTEMP    ; get result address
else
        POP    H         ; get result address
endif

        MOV    M,A       ; store it

if CLEAR_BUFFER
        ORA    A         ; test returned value
        JZ     EXIT      ; 0 means buffer is clear
CLRBUF:                  ; clear the buffer
        MVI    C,DCIO
        MVI    E,F6IC
        CALL   BDOS
        ORA    A
        JNZ    CLRBUF    ; <>0 means more in buffer
EXIT:
endif

if SAVE_STACK
        POP    PSW       ; restore the registers
        POP    B
        POP    D
        POP    H         ; get old stack pointer
        SPHL             ; restore it
        LHLD   HLTEMP    ; restore (HL)
endif

        RET              ; and return

if SAVE_STACK
        DSEG             ; Data Segment

HLTEMP:
        DS     2         ; temporary storage for (HL)
        DS     10        ; new stack (5 PUSH's or CALL's)
$$$$$!                   ; new stack pointer
endif

        END
```

James T. Malone
461 Shore Acres Road
Arnold, MD 21012

Dear HUG;

I have a subroutine for your "My Favorite Subroutines" Column. This routine converts a BASIC string of all capitals, to a string of capitals and lower case.

for example: A$="MATT JONES"

After the subroutine: A$='Matt Jones"

I have used this in several of my CAI programs, where the name is printed, to improve the screen image.

```
10 REM DEMO PROGRAM
20 INPUT A$
30 GOSUB 100
40 END
...
...
100 FOR I=1 TO LEN(A$)
110 TP$=MID$(A$,I,1)
120 IF TP$=" " THEN I=I+1:GOTO 160:REM if space, skip it
     and the next character
130 IF I=1 THEN 160:REM do not concert first character
140 IF ASC(TP$) = 97 THEN 160:REM if it is a capital,
    no need to convert it
150 MID$(A$,I,1)=CHR$(ASC(TP$)+32:REM do the conversion
160 NEXT I:REM do next character
170 RETURN
```

Sincerely,

Matt Jones
117 Freeman Dr.
Warner Robins, GA 31093

Dear HUG;

David Harvey's BUGGIN' HUG letter, in the June REMark, reminded me of another of "My Favorite Subroutines" that may be of interest to other Z-BASIC programmers.

This little gem, when inserted in the beginning of a program, will automatically provide the RANDOMIZE function with 65454 different seed numbers each day as a function of time. Thus, each time the program is run, a new seed number is available.

```
10 RANDOMIZE TIME / 1.32 - 32700
```

The TIME function is described on page 10.170 of the Z-BASIC manual. The constants shown will provide, very nearly, the optimum number of seeds allowable. 24*3600 / 132 - 32700 = 32754. When Time = zero, at midnight, the seed number is -32700, and one second before the following midnight the seed number will be, 32754, which for practical purposes is close enough to the allowable limits for seed numbers. See page 10.143 of the Z-BASIC manual.

Arthur Calhoun
16 Cedar Valley Lane
Huntington, NY 11743

Vectored from 76

When the disk is "flipped" to use the other side, the direction of rotation is reversed, the trapped material comes loose, and guess where it's going to come to rest? Right under the drive head, in the ideal location to abraid the disk surface at a faster rate than normal.

Even worse than that is what's happening to the other (first) side of the disk. There's a felt pad that keeps the disk's writing surface pressed closely to the drive head, and the pad also accumulates it's share of dust and dirt. In fact, it may get even more than its share because most single-sided drives have the recording head on the under-side, in order to use the cleaner surface. The pad traps all the dust that settles on the disk from above, aided by the static electricity sometimes produced by the disk rotation. So please, let's hear no more about the chance to "double your disk storage." You run a much greater risk of scouring information off the first side while you try to use the second.

Along these same lines, some drive makers, such as Siemens make drives that allow you to do the same thing as the "flippy" kits without modifying the disk. They do this by equipping the drive with a second set of disk position sensing light-emitting diodes (LEDs) and light detecting photo-transistors on the opposite side from the normal ones. In other words, there's a second LED and photo-transistor pair for the write-protect cut-out, and a second pair for index hole position sensing. This allows you to simply flip the un-modified disk over and insert it into the drive. As before, it's now turning in the opposite direction from the normal rotation, and the same dirt and abrasion problems occur. These seem attractive, at first glance, for the same reasons that the modification kits do. The results will be the same. I know some folks that use this approach for archiving software, and their success rate hasn't been bad, since they rarely use the disk for anything except re-creating working disks. It still seems risky, though, especially when you stop to think why someone would be archiving software to begin with. The usual intent is to preserve it for future use, so why jeopardize that by risking damage from reversed rotation?

One other not-so-obvious drawback to the "flippy" drives, is that the second side cannot be read without a similar drive. Merely cutting new holes in the appropriate places won't solve the problem. A few moments spent thinking about the geometry of the disks and the drives will reveal the problem. The second side, being timed with the index holes in the same relative position as the first, will create a "skew" to the recording that prevents it from being read by any, but a similar drive. In this respect, cutting new holes in the disk makes for greater compatibility, if you really must use that second side.

Remember that disks are relatively cheap compared to the data stored on them. This is especially true with information you carefully typed in over a long period of time. Companies that rely on the integrity of their computer data for their business take great pains to protect working copies, backup copies and archive copies. That care represents a certain amount of effort (money), but they feel the investment is worth it. You should probably look at it that way, too.

I hope some of this may be useful, especially to those new to the field. Someone once commented that one of the sadder aspects of micro-computing is that we're all re-learning the techniques first learned by the early computer users 35 or 40 years ago. If that's true, then lets avoid their great failing in not teaching others what they learned.

Sincerely,

LTC D. C. Shoemaker
HQ U.S. European Command
APO NY 09128

Index of Advertisers

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

HUG ID Number: ____________________

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	$20 ☐	$17 ☐
CANADA	$22 ☐	$19 ☐ US FUNDS
INTERNAT'L*	$30 ☐	$24 ☐ US FUNDS

* Membership in France and Belgium is acquired through the local distributor at the prevailing rate.

Heath/Zenith Users' Group

Hilltop Road
Saint Joseph, Michigan 49085

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

Volume 5, Issue 9

POSTMASTER: If undeliverable, please do not return.

P/N 885-2056