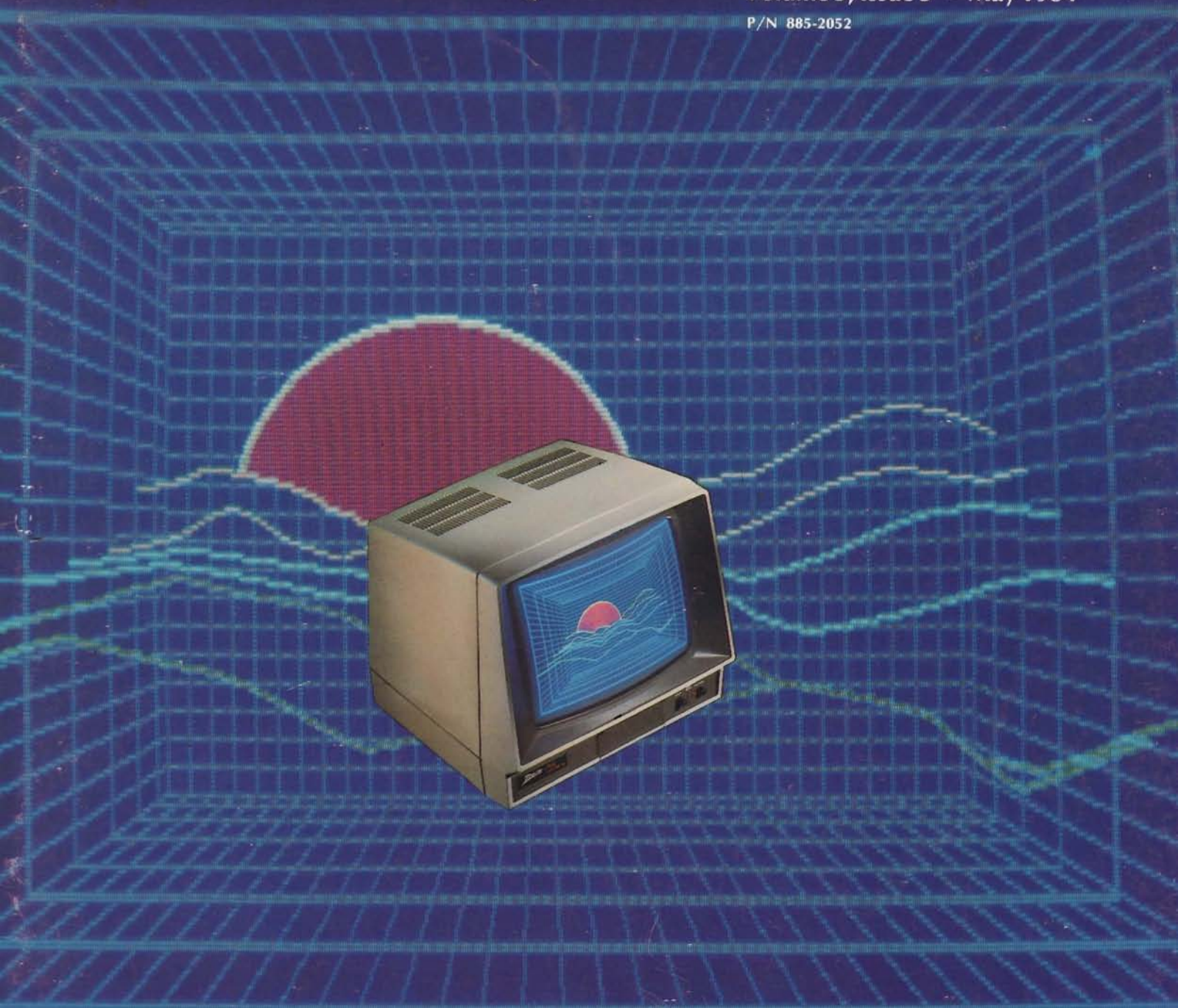
$2.50

REMark®

Volume 5, Issue 5 • May 1984
P/N 885-2052



Official magazine for users of HEATH ZENITH computer equipment.

Our Password™ modem's sophisticated new control panel

THE BUSIER YOU ARE, THE BETTER YOU LIKE IT.

PASSWORD™ is a modem so efficient and convenient there is little to do but turn it on to transmit at 300/1200 baud. Operating features include auto dial/answer, auto mode/speed select, full/half duplex. PASSWORD has all this, plus a two-year limited warranty, at a price of just $449.*

TELPAC™ telecommunications software (optional) programs PASSWORD to transfer files, in terminal or host mode, with multiple error checks. Phone directory gives choice of timed automatic or one-touch dial and logon. Command mode includes file display and update, menus and help, and much more. Write or call for complete specifications.

*Suggested list for PASSWORD complete with power, phone, RS232 Interface cables. TELPAC software optional extra, $79.

U.S. ROBOTICS INC.
1123 WEST WASHINGTON
CHICAGO, ILLINOIS 60607
(312) 733-0497

Heath/Zenith Users' Group

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

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

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

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

	U.S. Domestic	Canada & Mexico	International
Initial	$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 availble at $2.50 plus 10% handling and shipping. Check HUG Product List for availability of bound volumes of past issues. Requests for magazines mailed to foreign countries should specify mailing method and appropriate added cost.

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

616-982-3463

REMark®

Volume 5, Issue 5 • May 1984

on the stack

ON THE COVER: A Z-100 graphic presentation from Ray Massa of Studio Computers, 999 South Adams, Birmingham, MI 48011, representing some of the 3 dimensional effects that can be achieved with ZBASIC.

Outside Looking In

Bill Parrott
7010 Caenen
Shawnee, KS 66215

Since the announcement of the Z-150 personal computers by Zenith Data Systems a short time ago, there has been a great deal of discussion on the HUG bulletin board on CompuServe regarding the Z-100 and the impact the "new" computers will have on the future of that system. The majority of the comments have been negative and that bothers me. For one reason or another, people have gotten the idea that ZDS is planning on discontinuing support for the Z-100 in the immediate future and that that machine is rapidly drawing to the end of its life.

To my mind the very idea that this might be true is ludicrous to say the least. Let me explain why I believe this. Consider the H-8 and H-89; both of those machines had a life span longer than that of the Z-100 assuming that it were discontinued immediately. (When I refer to a machine's life span, I am referring to the period during which it is or was actively supported by its manufacturer. In real terms, the useful life of a given system may last years beyond a manufacturer's support, but more on that shortly.) Now how many people REALLY believe that Zenith is planning on discontinuing the Z-100 today? Would those of you who raised your hands please see me about some "hot beachfront property" I have for sale afterwards?

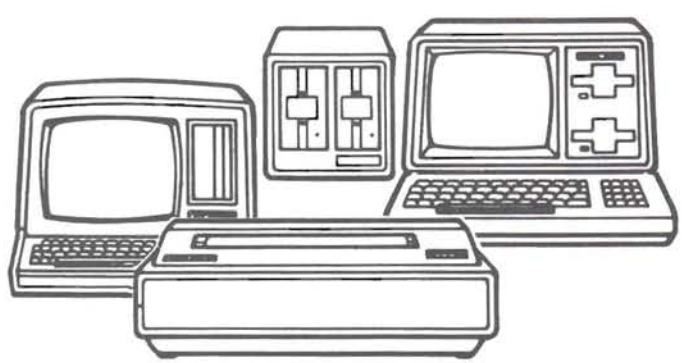
Consider among other things the current contracts Zenith has with the U.S. Air Force and Navy for the sale of Z-100's over the next few YEARS. "Well", the doomsayers might argue, "Zenith will just sell them Z-150's in place of the Z-100's." Right! If the Navy had wanted an IBM PC or a clone, it would have bought IBM PC's in the first place. Consider also, the cost to develop the Z-100. I have heard that it was well into the millions of dollars, and when I look at my Z-100 I have no trouble believing that. Then, there are the impending new products for the Z-100, including an 8087 arithmetic processor upgrade, a Local Area Network (LAN), MS-DOS (aka ZDOS) version 2.0, and Microsoft Windows to name a few. Then we have the rumored products including MP/M-86, Concurrent CP/M-86, and a new motherboard. Now I'll concede that there are people who feel that Zenith has made mistakes in the past, and that those people may be right in some cases, but would any company invest the time and resources necessary to bring these types of products to the market for a machine that's going out of production next month?

As for the life of the Z-100, if Zenith did choose to cease production for one reason or another, the machine would still not be dead. "But", I can hear them saying, "the Z-100 is obsolete." According to The American Heritage Dictionary of the English Language, the word "obsolete" is defined as "no longer used or useful...". My H-8 doesn't fall into THAT category. It may be true that the Z-100, like the H-8 and H-89, is not on what one might consider the "leading edge" of technology, but then, how much technology does it require to do effective bookkeeping, word processing, or just game playing? "Obsolete" is a word that computer salesmen and IC manufacturers use to sell new and "better" hardware. Using their definition of the word, the Motorola 68000 and the Intel iAPX 286 processors might be considered obsolete by some.

I currently own AND USE several computers, including an H-8, an H-89, and a Z-100. None of these machines are obsolete because I refuse to let them become obsolete. All see use on nearly a daily basis, and that is what it comes right down to. Any machine will remain useful so long as you have a use for it. There is almost nothing that I might want to do (barring graphics) with my Z-100 that I could not do just as well with my H-8, and I submit that this is true for the majority of us. I subscribe to PC magazine so that I can remain aware (if casually) of what sorts of things are being done with the IBM PC and its look-alikes. A stroll through the pages of a recent 750+ page issue of that publication (which is said to be the only magazine which must be delivered by common carrier) brings to light some interesting insights into the PC market, at least as far as software is concerned. A reader will find a large number of programs for sale, but nearly all of them fall into a few "standard" categories. For example, in the first HALF of this issue I counted advertisements for 17 data base, 15 word processing, 9 accounting, 9 modem, 9 instructional (how to use your computer), 5 spreadsheet, 5 tax preparation, and 5 integrated (of the 1-2-3 variety) programs. There was exactly one vertical market application represented. There was nothing listed that I could not do with my H-8 with any one of several commercially available packages. I'm not trying to dissuade anyone from buying the latest thing out, including IBM PC's. What I am trying to show is that just because a system is not at the leading edge doesn't mean that it is obsolete.

As an interested user of a Z-100, let me tell you where I expect Zenith to be going in the next couple of years. I have no inside information from which to formulate my decisions... only experience with Heath and Zenith, and faith that I won't be let down. If for no other reason than by virtue of the Government contracts, I expect the Z-100 to remain in production and to be updated for a period longer than either the H-8 or H-89. Further I will be looking for new hardware and software products to become available which will serve to enhance the usefulness of the Z-100 so as to extend its life well beyond its production. I don't see Zenith stopping development of new systems. To do so would be stupid given the advancing technology, but I'm not going to panic whenever they announce a "better mousetrap". As to what these hypothetical new systems might be, I haven't a clue but I don't expect to be disappointed. And finally, and most important, I expect to see Zenith lending an ear to see what the users of their systems think. If we all agree today that the fate of the Z-100 has already been decided and discourage everyone we know from buying one, then the death of that system will become a self-fulfilling prophecy. If we remain steadfast and enthusiastic in our support of the system, then it will likely remain in production longer than the IBM PC.

Finally, we have a most valuable resource and that is a very close knit user community. I have attended and participated in users' groups for various brands of hardware, including IBM, and nowhere have I seen a more enthusiastic, helpful, and caring group of users than in the Heath/Zenith community. If we tell Zenith that we no longer have interest in the Z-100 computer, then they will stop making them. If, on the other hand, we show strong and continued interest in the machine, then support will logically continue. We cannot expect Zenith to continue production of a product for which they can perceive no market. As for the Z-150, it can only be the result of a perceived market for a product and I wish Zenith success with the new system as that success can only be in all of our best interests.



Professional Graphics at Practical Prices

Add our Imaginator™ graphics upgrade card to your H/Z-19 terminal or H/Z-89 computer. It's quick and easy. You gain intelligent, highly efficient graphic display capabilities proven in countless Heath/Zenith terminal and computer updates.

Check Imaginator's special features:

- ☐ High resolution 504 x 247
- ☐ Accessible through any high level language FORTRAN, Pascal, BASIC, etc.
- ☐ Onboard microcomputer eliminates processing load on host
- ☐ Source code available
- ☐ Rich graphics instruction set
- ☐ Mix text and graphics
- ☐ Tektronix® 4010-4014 compatibility option with GIN mode
- ☐ Comprehensive documentation includes numerous examples
- ☐ Fully buffered for asynchronous operation
- ☐ All original H/Z features remain intact

Now check Imaginator's low cost:

- ☐ Assembled complete $445.
- ☐ Kits from $215.

Also ask us about our Imaginator 2 upgrade for H/Z-29 terminals.

Call or write us today for additional information.

CLEVELAND CODONICS, INC.

18001 Englewood • Cleveland, OH 44130 (216) 243-1198

Your H-19 Can Do This...

IMAGINATOR

Tektronix® Registered trademark of Tektronix, Inc.
Codonics and Imaginator are trademarks of Cleveland Codonics, Inc.

Software For the H-88

Dear HUG,

I submitted a program to REMark which was not accepted. It was written to run on an H-88 with 10.06.00 BASIC. Walt indicated that several pages of magazine space can't be allocated for articles which would not be informative, educational, or useful in some way to a fair percentage of REMark readers. Letters for Buggin' HUG and short cassette articles will be examined for their usefulness to other members. Thanks to the staff for taking the time to look at mine.

I have a couple of programs that might be useful to 'Tapors'.

BASED25: (Assembly Language)
Edits strings or program lines on the terminal 25th line. Requires an H-88 (some Z80 instructions) and 10.06.00 BASIC which is patched to allow an 82 character $INBUF.

CHEAPERCALC: (BASIC)
A spread sheet that runs fast enough to be useful. It uses function keys and shifted keypad on H/Z-19 type terminals.

If you are interested, send me a note. CHEAPERCALC reconfigures BASIC and PEEKs some addresses in BASIC so I'll need some information about your system, distribution software used, and if you have a BASIC source listing. As an example: I run an H- 88/48K and an H14 printer. I use XX.06.00 software and have source listings for 02.06.00 terminal debugger, 03.06.00 text editor, 04.06.00 assembler, and 10.06.00 BASIC.

After I get an idea about numbers to reproduce, I'll send more information about costs and hardware requirements to those who respond. My best guess at the moment would be $10-$15 for documentation and a tape.

Allen Zimmer
Rt. 1, Box 47B
Eagle, NE 68347

And Still More On Suppressing the Key Click

Dear Walt,

Only hours after mailing my letter to you of yesterday, I received the March issue of REMark. When I came across the short letter

INTERNATIONAL HUG CONFERENCE

Official Conference Registration Form

Pheasant Run Convention-Resort Hotel

July 27, 28 and 29

Name(s) _____

Address _____

Company _____

City _____ State _____ Zip _____

Enclosed is $22.00 per individual to attend the International HUG Conference to be held the weekend of July 27, 28 and 29, 1984. Please send tickets along with information regarding hotel reservations and transportation.

Amount Enclosed: _____ Number attending: _____

For our information:

Which Heath/Zenith computer do you now operate? _____

Are you a Non-User-Attendee? ☐Yes ☐No

Are you a Heath/Zenith related vendor? ☐Yes ☐No

If yes, do you want exhibit space during the Conference? ☐Yes ☐No

Special Notice to Vendors:

Vendor Information Packages will be made available to Heath/Zenith Related Vendors who are planning to exhibit their products while at the Conference. You must contact us prior to May 1, 1984.

For your information:

The $22.00 you are paying for your reservation to the International HUG Conference entitles you to all functions of the Conference. This includes Saturday breakfast, buffet lunch and hors d'oeuvres in the evening. The Prize Drawing will be held during the Saturday evening Cocktail Party. You must be present to win. Vendors and $22.00 ticket holders will be eligible for prizes. ALL prizes will be awarded at that time.

Visitor tickets, for those of you simply attending the seminars and looking at the exhibits, are available for $10.00. Visitor Tickets do not include meals or eligibility to the Prize Drawings.

Send your registration form or a suitable copy to:

Heath/Zenith Users' Group
Attention: International HUG Conference Registration
Hilltop Road
St. Joseph, Michigan 49085

Registration(s) must be postmarked no later than July 15, 1984. Cancellations will not be accepted after this date.

from Richard Hole of Big Rapids, MI on page 64 about another way to suppress the key click, I wasn't long in giving it a whirl. And it works perfectly.

I chose to implement the idea in a slightly different way than he did; rather than place the command line in an autoexec file, I simply used a batch file and then invoked it directly. At the same time, I made up a number of other batch files to enable and disable the reverse video, set the block and underline cursors, and set the blinking and non-blinking cursors. All work perfectly. And thus in a single swoop, a problem which has been perplexing me for months has been solved. I can't help but be impressed with the simplicity of this approach and wonder why someone hasn't pointed it out before considering how much has been written about the subject for a number of months in a number of publications.

At the same time I'm curious about why Mr. Shoemaker's assembly language program didn't work for me, although the matter has now become only a matter of curiosity in learning something about assembly language programming. Although I like the approach of learning assembly language by means of a few useful examples, I do think that an inexperienced user ought to be given more guidance about some of the pitfalls and problems that might and most likely would occur.

I do realize that an editor cannot always fill up his publication with perfect articles nor even screen with great care all of those that are published. And I really do feel that you are turning out a very good publication which I look forward eagerly to receiving each month.

Robert E. Heath
9 East Dunnrobin Bay
Sault Ste. Marie, Ontario
P6C 5T4

A Patch To The CAMERA Program

Dear HUG,

I have an H-8/H-17/H-19/H-8-5. I recently bought Hugman & Animation movie, p/n 885-1124, and quickly found out that one of the programs would not run on my system. The camera program will not run with the H-8-5 until you make the following change, 105341/350 to 105341/372. I found the address using Udump 885- 8004.

If you have a Siemens FDD-100-5 disk drive, you can make it step faster, like 8ms or so, by changing R48 on the drive to 20K. It is presently 33.3K. It is easier just to jump about a 40K resistor in parallel with R48.

Jeff Dovel
Rt. 3, Box 2010 #21
Ellensburg, WA 98926

A ZBASIC Program

Dear HUG,

I have really enjoyed the program listings in REMark and decided it was my turn to contribute. The following ZBASIC program features a mobil tank that shoots at a moving target. If the red spot on the target is hit, then the target explodes. It is a simple program, but shows the use of DIM, PUT, GET, and XOR.

```
10 REM *** SHOOTER.BAS by Edward A. Byrnes ***
20 REM *** "4"= TANK TO LEFT ******************
30 REM *** "5"= FIRE PROJECTILE ***************
40 REM *** "6"= TANK TO RIGHT ****************
50 REM *** "Z"= END PROGRAM ******************
60 REM *** MAKE AND GET EXPLOSION *************
70 CLS:FOR X=1 TO 30:F=RND*70:G=RND*35:PSET(F,G):
   COLOR RND*7
80 NEXT X:DIM E#(125):GET(0,0)-(70,35),E#:CLS:COLOR 7:
   DIM B#(8)
90 REM *** MAKE AND GET PROJECTILE ************
100 LINE(0,0)-(0,6):GET(0,0)-(2,6),B#:CLS
110 REM *** DRAW AND GET TARGET ***************
120 COLOR 7:PSET(0,10):DRAW "U3R20D6L20U3":
    PAINT(10,9),7,7
140 CIRCLE(10,10),3,4:PAINT(10,10),4,4:DIM C#(50):
    GET(0,0)-(30,15),C#:CLS
150 COLOR 2:LOCATE 24,64:PRINT "TARGETS HIT=";
152 LOCATE 24,2:PRINT "TARGETS LOST=";:CL=0
160 REM ***DRAW TANK *************************
170 LINE(300,200)-(318,205),2,BF:
    LINE(304,197)-(314,200),2,BF
180 LINE(308,190)-(311,197),2,BF:DIM A#(20):
    GET(300,190)-(318,205),A#:RM=300
190 N=0
200 LINE(0,0)-(639,224),4,B:LINE(1,1)-(638,223),4,B
210 REM *** MOVE TARGET GET KEYBOARD INPUT ***
220 PUT(0+N,20),C#,XOR
230 IF N>600 THEN GOSUB 480
240 ZZ$=INKEY$: IF ZZ$="6" THEN GOSUB 300
250 IF ZZ$="4" THEN GOSUB 340
260 IF ZZ$="5" THEN D=0:GOSUB 380
270 IF ZZ$="Z" THEN COLOR 6:CLS:END
280 PUT(0+N,20),C#,XOR:N=N+7:GOTO 220
290 REM ***MOVE TANK TO THE RIGHT ************
300 PUT(RM,190),A#,XOR:RM=RM+8:PUT(RM,190),A#,XOR
310 IF RM>610 THEN PUT(RM,190),A#,XOR:RM=20:
    PUT(RM,190),A#,XOR
320 RETURN
330 REM *** MOVE TANK TO THE LEFT ************
340 PUT(RM,190),A#,XOR:RM=RM-8:PUT(RM,190),A#,XOR
350 IF RM<20 THEN PUT(RM,190),A#,XOR:RM=604:
    PUT(RM,190),A#,XOR
360 RETURN
370 REM *** MOVE PROJECTILE UP ***************
380 FOR T=1 TO 18:PUT(RM+9,189-D),B#,XOR:
    PUT(RM+9,189-D),B#,XOR
390 REM *** CHECK FOR HIT ON RED PIXEL *******
400 IF POINT(RM+9,189-D)=4 THEN BEEP:GOSUB 440
410 D=D+10:NEXT T
420 RETURN
430 REM *** BLOWUP TARGET *********************
440 PUT(0+N,20),C#,XOR:PUT(RM-10,15),E#,XOR
450 PUT(RM-10,15),E#,XOR
460 GA=GA+1:LOCATE 24,76:PRINT GA;:N=0:
    PUT(0+N,20),C#,XOR:RETURN
470 REM *** PLACE TARGET IN START POSITION ***
480 PUT(0+N,20),C#,XOR:N=0:PUT(0+N,20),C#,XOR:CL=CL+1:
    LOCATE 24,15:PRINT CL;
490 RETURN
```

Edward A. Byrnes
Intuitive Logic
412 Taylor Street
Rochester, MI 48063

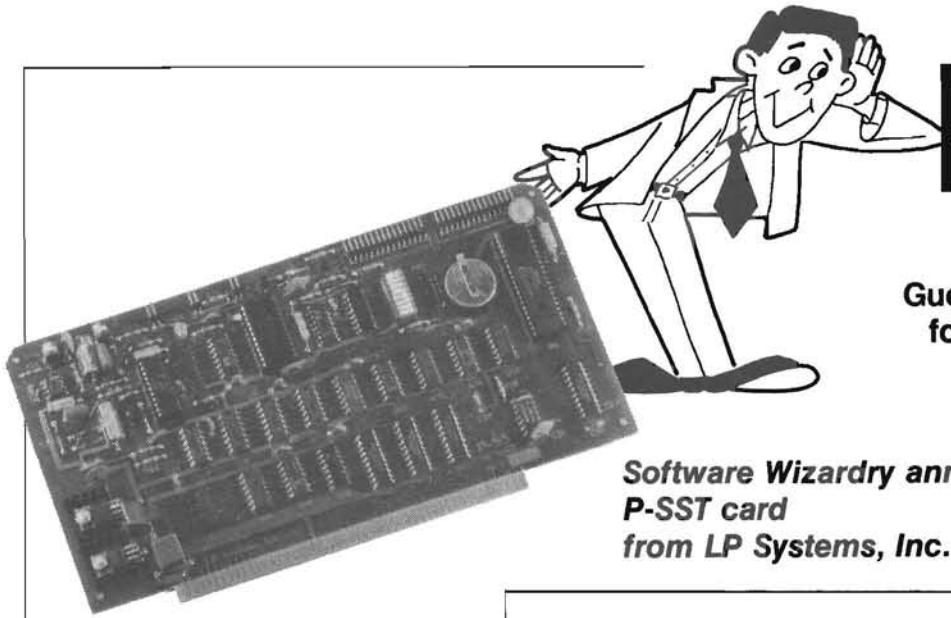
Correction To The Article "I/O Baud Rate Programmer"

Dear HUG,

I have been a member of HUG for a couple of years now, and therefore a subscriber to your excellent REMark magazine. I find useful information in every issue, without exception. One of the things that has always impressed me is the high degree of accuracy and freedom from bugs evident in your program listings. While no publication is perfect in that regard, it certainly seems that you take extra care to ensure that the programs you publish actually work.

Despite your superior track record, though, I did manage to find a

Vectored to 62 ☞



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.

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!

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 greater) shipping/handling plus sales tax if shipped to a Missouri address.

Software Wizardry, Inc. also has over 450 other items of interest to Heath/Zenith users as well. Call or write for our FREE price list. Dealer inquiries and HUG group purchases welcome!

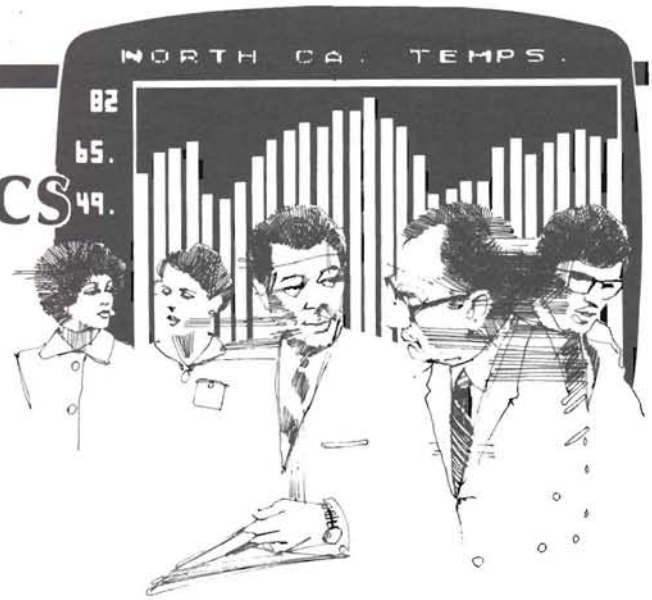
MasterCard VISA

ZENITH data systems
AUTHORIZED SALES & SERVICE

Software Wizardry
THE MAGIC TOUCH

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

Computer Graphics On The H/Z-100



Randy Meyers
2200 N. W. Highland Dr.
Corvallis, Or. 97330

When I first purchased my H-100 system I had several projects in mind. One of the things that generated more than passing interest was the system's excellent graphics. The 640 x 225 resolution and eight colors made it one of the best microcomputer graphics systems available.

I was at least a little disappointed, when, after searching through all the system documentation, I discovered that the only interface to ZDOS's graphics was through the ZBASIC interpreter. I'm not a language purist who looks down his nose at someone who programs in BASIC, but it's an interpreted language, and relatively slow. Some of my more complicated programs took an hour and longer to run.

Using the Latice C compiler (I am very happy with it) I wrote some utility routines to turn pixels on and off, draw lines, etc. I later converted the logic to assembly language for a further increase in speed.

When examining something complicated or outside my normal range of experience, I've found it simplifies things if I can break the problem into small pieces, then attempt to understand each of the pieces separately. If my approach is too slow for you, I apologize. Here, then, are some of the things I learned while pouring through the technical manuals which came with the system.

I noticed over and over in the H-100's internal organization that many of the software problems inherent in a multi-colored, memory mapped display system have been given a helping hand with hardware. Because of this, the machine performs much better than if it were entirely software driven, but is also more complicated. I, for one, would rather put up with complexity and have the additional performance.

Before I get too deep into this, I would like to define some of the terms I will be using. Computer graphics, like everything else, builds complicated ideas from simple things. One of the simplest things in computer graphics is the 'Pixel'. A pixel is one dot on the display. It may have only one color, or eight colors, or eight million colors, but it is still just one dot, and it's the smallest item the system can handle. Another term I will use a lot is 'Scan Line'. A scan line is all the dots in one row. The H/Z-100's display is basically a TV, which is nothing more than an electron gun scanning across the inside of the picture tube, one row at a time, turning some dots on, and ignoring others.

There is a separate 64K memory segment that is allocated to each primary color, blue at C0000H, red at D0000H, and green at E0000H. Each of these memory segments is also called a color plane. Notice that the primary colors used in computer graphics are not the same as those used by the rest of the world. This is because red, blue, and green phosphors are the easiest (most economic) to put in a picture tube, and can be combined to generate any color.

With only the standard green segment installed you can do monochrome (one color) graphics. If the other two sets of memory chips are installed you have eight colors available. The color planes can be turned on and off in various combinations to give the colors black, blue, red, magenta, green, cyan (blue green), yellow, and white. If you have the extra memory installed, but don't have a color monitor, the colors will show up as different levels of brightness, or 'Gray Scale'.

Video Control Register

One of the key elements in the Z-Machine's color graphics is the Video Control Register. This is an I/O port at 0D8H. It is a bi-directional port, which means, what has been written can be read back. The Video Control Register serves two separate functions. The upper four bits control the CPU's access to video memory and are referred to as the 'Video Access Control Bits'. The lower four bits control what is displayed on the screen and are called the 'Video Display Control Bits'. For the type of computer graphics dealt with in this article, we're only interested in the Video Access Control Bits. As the name implies, these define how the video memory is accessed.

Bit 7 is called VRAM ENABLE, it is the main control switch for the system's video memory. When the VRAM ENABLE bit is set to one, the system's video ram is turned off. The CPU can't read or write data in the video memory segments. If the VRAM ENABLE bit is reset to zero, the CPU has access to the graphics memory and is able to read and write them. For normal operations the VRAM ENABLE bit should be reset to zero.

The next three bits (bits 6 to 4) are called Blue Enable, Green Enable, and Red Enable, respectively. These bits determine how the three video memory segments are handled when the CPU attempts to write data to them. All three bits perform the same function for their associated color segments. If the Blue Enable bit is cleared to zero, the blue segment is enabled, and responds to a write to any of the video segments. That is, if you reset Blue Enable to zero and write to the green memory plane, it will show up in the blue plane also. The data goes to both places. Bit 5 performs the same service for the green memory segment and bit 4 for the red plane. If you want to write to each memory segment separately, you should set the Video Control Register to 0111xxxxB. If, on the other hand, you want to control all three video planes with one access, you should set the control latch

to 0000xxxxB. In this case a write to any of the color segments will cause the corresponding byte or word to be written in the other two segments as well, and white will appear on the screen. Either case will allow the CPU access to graphics memory because bit 7 (VRAM ENABLE) is zero.

One word of caution. You should make every effort not to modify the lower four bits of the Video Control Register. If bit 3, called FLASH, is set to one, the display will be one solid color. Which color it is depends on the values in the three lowest bits. I've accomplished this a number of times. Believe me, it's almost impossible to run the system when you can't see what you're typing. The best method of modifying the Video Control Register is to read it, modify the proper bits, then write it back. If you do this you won't hand any surprises to another program using the other half of the register.

Pixel Address Calculations

Using the green segment as an example, I'll show you how to calculate the memory address required to turn on or off specific pixels. Remember, the red and blue color segments are organized exactly like the green, except they are in different 64K memory segments.

From the programmer's perspective, the easiest way to specify a pixel is with its X and Y coordinates. Because of the way the computer addresses the video memory, I decided to put my origin in the upper left corner of the display. The pixel in that corner is addressed as 0,0. Since there are 640 dots across and I started at zero, the last pixel in the first row is at 639,0. Similarly the lower left pixel is at 0,224 and the lower right is at 639,224.

To manipulate the video memory a pixel at a time you must know which byte, and which bit within that byte controls the pixel. Also, you must know what color the pixel will be. A subroutine to display pixels must be able to convert the X and Y coordinates to a byte address, a bit number, and evaluate the color value to determine if the bit is to be turned on or off in the different color segments.

Zenith decided to give the character display software a hardware assist. Any complex system is usually a compromise between many considerations. In the H/Z-100, the Design Engineers decided to optimize the graphics system for character display. I believe this is a good design decision, but it makes the pixel graphics a little slower and a little more complicated than they could be.

Specifically, they decided to make each character display line start at an address that is a multiple of 2K (2048). A displayed character is eight pixels across and nine pixels tall. The character's location in memory can be calculated by multiplying the row number by 2048 and adding the character's column number. The character can be displayed by poking the nine bytes which define the character into the memory locations at the characters origin address, the address + 128, the address + 256, etc. Because all the calculations are powers of two, they can be done with shifts, no multiplication is required.

Scrolling is also handled with hardware. When the display scrolls, memory data is not moved. Shuffling all those bytes is too slow - the screen would show a visible 'ripple' which would be objectionable. Instead, the video controller chip has a register which tells it which memory address to use as the 'origin' for display data. This register is updated to point 2048 bytes higher in the display memory. When this happens the top line disappears, the entire display moves up one character line, and a new bottom line is displayed. The display memory for the new bottom line is cleared to zeros, causing it to be blank.

From the CPU's perspective the contents of all the video memory have been moved. If you poke the value 128 into memory address 4096, segment E0000H, a dot will be displayed on the screen in the upper left corner. If the display is scrolled, the dot will move up nine graphics lines. Poking the same value into the same memory location will produce a second dot on the display, nine graphics lines below the first.

Since each byte in the display memory controls eight pixels and since there are 640 pixels in each row, the system requires 80 bytes to display each row. The next row does not start with the 81st byte, however. Because the display has been optimized for the character display, each row starts 128 bytes after the start of the previous row. Also, since each character row displayed starts at an even 2048 byte boundary, there is an 896 (7 x 128) byte gap between the bottom row of one character and the top row of the next.

A pixel's row address may be calculated by dividing its Y coordinate by 9. The quotient defines the character row (0 to 24) the pixel lies in, and the remainder defines which scan line (0 to 8) within that character row. If the quotient is multiplied by 16 and the remainder added to it, it defines a pseudo scan line number. It is a pseudo line number because it assumes there are 16 scan lines for each character instead of 9. This is OK because it compensates for the seven non-existent scan lines at the end of each character row. If this result is multiplied by 128, it points to the memory location of the first dot on that graphics display line.

A pixels X coordinate or column address is much easier to calculate. Since there are eight pixels in a byte and eighty bytes to a row, we can calculate the pixel's column address by dividing the pixel's X coordinate by eight. The result is the byte offset in the row, and the remainder is the pixel's bit in the byte. The only confusing thing about this is the pixel's bit number. It assumes we are numbering from high bit to low, which is the reverse of the normal numbering conventions.

The specific byte to manipulate is the sum of the pixel's X and Y coordinates. We can use the pixel's bit number to construct a bit mask which will allow us to turn on or off a specific pixel.

This long-winded explanation can be condensed into a couple formulas which, I suspect, show the concepts much clearer than I have.

```
row := ( ( x_coordinate / 9 ) * 16 ) +
       ( x_coordinate mod 9 )

char_byte     := y_coordinate / 8

bit_address   := y_coordinate mod 8

byte_address  := ( row * 128 ) + char_byte
```

An Example: SETPOINT

Using these concepts I developed the SETPOINT routine. Perhaps an explanation of it will make the whole thing clearer. If not, at least you will have a useful subroutine to add to your library.

The first few lines of the program are comments which serve to document it's purpose and calling conventions. This particular routine was designed to be called from Latice C procedures. It's parameter passage conventions, group and segment names reflect that. In addition, I've added a few comments to define the assembly language interface.

The next section defines some equates which specify the location of the various color segments, and which bits in the value parameter control which color planes. The bits assigned to the various color planes were selected by experimentation. The blue color plane contributes the least to a monochrome display's intensity and the green the most. By assigning the bits in this manner, the display

intensity increases as the magnitude of the value parameter increases.

Next, I defined a data structure called SPSTR. A structure doesn't allocate any space or create any variables, it just defines how the data is organized in memory. That is, which variables come first, if they are byte or word variables, etc. In this case, the SPSTR structure defines the state of the stack when the procedure is entered. Using this data structure it is possible to do 'stack relative' addressing. We can retrieve the X and Y coordinates of the point and its color value.

The PUBLIC pseudo opcode defines SETPOINT as a procedure entry point which can be accessed by other programs. When the final program is 'linked', the linker uses PUBLIC declarations to hook together routines which were assembled or compiled separately.

We finally get to the place where the program begins. I save the BP register, then load it with the current value of the Stack Pointer. The BP register will be used as a pointer into the stack's memory space. If I defined the SPSTR structure properly, it contains a snap-shot of the section of the stack containing SETPOINT's parameters.

Next, I calculated the address for the start of the row the pixel is in. The row coordinate is divided by nine and the quotient and the remainder are both saved. The quotient is multiplied by sixteen by shifting it left four times. The remainder is added in and the sum is multiplied by 128. The multiplication is a little tricky. To multiply by 128, I could have shifted left seven times. The same result can be accomplished by exchanging the upper and lower bytes and rotating the result right. AX now points to the start of the pixel's row.

The bit within the byte is calculated by masking the X coordinate to the low three bits. If I number the bits backwards, with the left bit as bit 0 and the right bit as bit 7, CL defines the bit number we want to turn on or off. The bit number is saved in the CL register.

The only thing left to do is calculate the number of bytes the X coordinate is offset from the start of the row. This can be done by dividing the X coordinate by eight, which is the same as shifting BX right three times.

The sum of the row offset (in BX) is added to the row's start address (in AX). The result is moved to the index register SI.

Next, I construct a bit mask. This will be used to turn the pixels on or off. I load the binary value 1000000B (128 decimal) into the BL register. By shifting BL right CL times I move the one bit to the proper column in the byte. The complement of this value is saved in BH.

To make sure I can get access to the color segments, I read the Video Control Register and put 0111 in the upper nibble. This allows me to access each of the three color segments independently. Notice, I am careful not to modify the lower four bits.

If I logically 'OR' the value in BL with the value in memory, I will turn the pixel on. If I 'AND' the BH value with memory I can turn the pixel off. This is the essence of what I do in the last third of the routine. I examine the blue_bit in the color value. If the bit is set, I turn the blue segment's pixel on. If it is clear, I turn it off. In a like manner I turn on or off the pixels in the red and green segments.

Finis!! I pop the saved registers off the stack and return to whatever called me.

Summary

That's all I have to say about the H/Z-100's video organization. I have covered a lot of material. I hope I've clarified some of the ideas behind the graphics display. Once you understand the basic organization of the video display, it becomes relatively easy to write utilities which access it. I have used these same ideas to write a graphics dump program which is callable from ZBASIC, or can be invoked with the Shift-F12 key.

The routine presented here should be useable with any language system. The only changes which should be necessary would be in the parameter passage. This routine assumes they will be passed by value; that means the actual value of the parameter is loaded onto the stack before the routine is called. Another popular mechanism is 'pass by reference'. Instead of pushing the value on the stack, a pointer to the value is pushed. This means the SETPOINT routine would have to load an index register with the pointer, then, using the index register, read the value.

If you don't wish to type in and debug this routine, I will provide it, along with the source code of a graphics dump routine, on a single sided ZDOS disk for $7.00. If you supply the disk and a stamped, self addressed mailer, it will be $2.00. I'm not in the software business, so these routines are offered on an 'as is' basis.

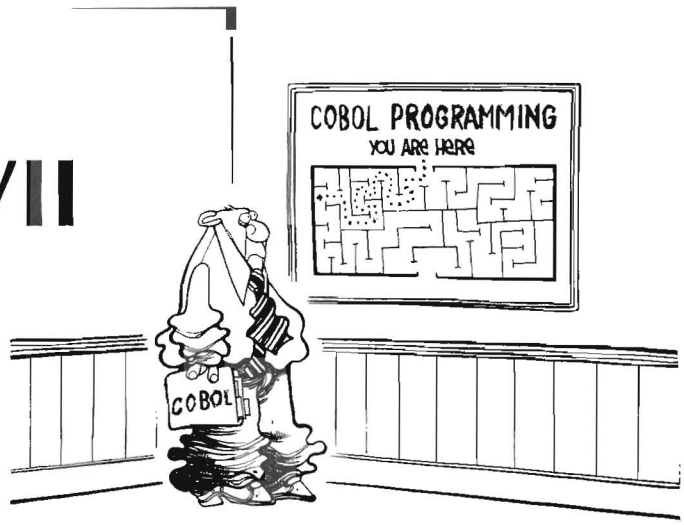
```
        page    60,132
        title   SETPOINT.ASM - Set a point on the video
                               display.
;
;  Called with:
;       setpoint( x, y, value )
;       Where X and Y are the column and row to turn on
;       and VALUE is the color value to use [0..7]
;
;  or from assembler call with...
;
;       Push    Value
;       Push    Y
;       Push    X
;       call near ptr setpoint
;
; Uses:
;   AX, BX, CX, SI
;
pgroup  group   prog
prog    segment byte public 'PROG'
        assume  cs:pgroup
;
video_latch equ 0d8h            ; Video access mode
;                                 latch.
;
green           equ     0e000h  ; Location of green
;                                 color plane.
red             equ     0d000h  ; Location of red color
;                                 plane.
blue            equ     0c000h  ; Location of blue
;                                 color plane.
;
blue_bit        equ     01h     ; Value for blue plane
;                                 enable.
red_bit         equ     02h     ; Value for red plane
;                                 enable.
green_bit       equ     04h     ; Value for green plane
;                                 enable.
;
; This structure defines the state of the stack
; when the routine is entered.
;
spstr   struc
        dw      ?       ; BP register
        dw      ?       ; Return address.
xcoord  dw      ?       ; X coordinate.
ycoord  dw      ?       ; Y coordinate.
value   db      ?       ; Color value for point.
        db      ?
spstr   ends
;
        public  setpoint
;
setpoint proc near
```

```
        push    bp                      ; Do some setup.
        mov     bp,sp
        push    ds
;
;  First calculate the address of the first byte of
;  the row.
;
        mov     ax,[bp].ycoord          ; move Y coord
;                                         to AX.
        mov     bl,9
        div     bl                      ; AL := AL div 9;
        mov     bl,ah                   ; BL := AL mod 9;
        xor     ah,ah                   ; AH := 0;
        shl     ax,1                    ; AX := AX * 16;
        shl     ax,1
        shl     ax,1
        shl     ax,1
        or      al,bl                   ; Add in
;                                         remainder.
        xchg    ah,al                   ; AX := AX * 128;
        ror     ax,1                    ; AX := Address
;                                         of pixel's scan row.
        mov     bx,[bp].xcoord          ; BX := X
;                                         Coordinate
        mov     cl,bl                   ; CL := bit
;                                         number [0..7].
        and     cl,07H
        shr     bx,1                    ; BX := Pixel's
;                                         column address.
        shr     bx,1
        shr     bx,1
        add     ax,bx                   ; AX := Pixels
;                                         byte address.
        mov     si,ax
        mov     bl,128                  ; BL := Pixel
;                                         mask.
        shr     bl,cl
        mov     bh,bl                   ; BH := NOT
;                                         Pixel mask.
        not     bh
;
; Set color planes to allow direct writes only.
;
        in      al,video_latch
        push    ax
        or      al,070h                 ; Set color
;                                         access control bits.
        and     al,07fh                 ; Clear video
;                                         access bit.
        out     video_latch,al
        mov     cl,[bp].value           ; CL := Color
;                                         value to use.
;
; Do the Blue plane.
;
do_blue:
        mov     ax,blue                 ; Load blue
;                                         pointer
        mov     ds,ax
        test    cl,blue_bit             ; Blue bit set?
        jz      blue_off                ; No, skip on
;                                         and do off.
        or      ds:[si],bl              ; Turn the
;                                         proper bit on.
        jmp short do_red
blue_off:
        and     ds:[si],bh              ; Turn the bit
;                                         off.
;
; Do the RED plane.
;
do_red:
        mov     ax,red                  ; Load red
;                                         pointer.
        mov     ds,ax
        test    cl,red_bit              ; Red bit set?
        jz      red_off                 ; No, turn video
;                                         bit off.
        or      ds:[si],bl              ; Turn it on.
        jmp short do_green
red_off:
        and     ds:[si],bh              ; Turn the bit
;                                         off.
;
; Do the green plane.
;
do_green:
        mov     ax,green                ; Load Green
;..                                        pointer.
        mov     ds,ax
        test    cl,red_bit              ; Green bit set?
        jz      green_off
        or      ds:[si],bl
        jmp short done
green_off:
        and     ds:[si],bh              ; Turn off green
;                                         bit.
done:
        pop     ax                      ; Restore video
;                                         latch
        out     video_latch,al

        pop     ds                      ; restore ds
        pop     bp
        ret                             ; all done
;
setpoint endp
;
prog    ends
;
        end
```

COBOL Corner VII

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



Introduction

Are you ready to start YOUR design of Sample Program #2? If you were able to complete Sample Program #1 and you understood the Phases of program development, you are now ready! If not and you need additional help, be sure to write me about your problems in detail (with SASE, business size). It is a must that you understand how to work with the development PHASES and the COBOL SOFTWARE that we have covered up to this point.

System Analyst vs Programmer

Here at COBOL Corner, I will be the System Analyst in the Data Processing Department. The System Analyst supplies you, the Programmer, with the necessary instructions so that you can design and write the Program.

Program #2

The Program will be an Employee Address/Telephone List that will be prepared from an Employee Transaction File.

In general, to prepare a business problem for solution, the first step is to thoroughly describe the functions to be performed and the objective to be accomplished. The most important part of this analysis is the description of the Format of the Output Report. Once the Output Report has been described, the Input Record can be described. The second step is to design the program, and the third step is to write the program. The COBOL language was created specifically to facilitate the processing of the data generated by business and industry.

Program Specification

PROGRAM NAME: EMPLOYEE LIST — PROGRAM ID: PRGM02

Program Description:

This program reads the Employee Records File containing the Employee's Social Security Number; Last, First and Middle Initial Name; Address; City; and Telephone Number. The program will print out the Employee Address/Telephone List.

Input File:

Disk FILEL2.DAT contains the Transaction File Data. It contains each Employee's Social Security Number; Last, First and Middle Initial; Address; City; and Telephone Number.

Output File:

Each Employee's Last, First and Middle Name; Social Security Number; Address; City; and Telephone Number (printed with "HYPHENS" inserted in the appropriate positions in the Social Security and the Telephone Numbers).

List of Program Operations:

1. Read each Employee's Record from the Disk Data File -- FILEL2.DAT.
2. For each Record, print the following fields on the Employee List Line:
   - **a.** Employee Name -- last, first and middle initial.
   - **b.** Employee Social Security Number with hyphens.
   - **c.** Employee Address -- street and city.
   - **d.** Employee Phone Number with a hyphen.
3. Double-space each Employee Record.
4. COBOL will be the programming language.

OUTPUT REPORT LINE FORMAT

PRINT POSITIONS	FIELD NAME	COMMENTS
1-5	FILLER	PROVIDES LEFT MARGIN
6-17	EMPLOYEE LAST NAME	
18-19	FILLER	PROVIDES SPACE BETWEEN NAMES
20-30	EMPLOYEE FIRST NAME	
31	FILLER	
32	EMPLOYEE MIDDLE INITIAL	
33-34	FILLER	
35-45	EMPLOYEE SOCIAL SECURITY NUMBER	PRINT HYPHENS SSS-SS-SSSS
46-47	FILLER	
48-71	EMPLOYEE ADDRESS	
72-73	FILLER	
74-86	EMPLOYEE CITY	
87-88	FILLER	
89-96	EMPLOYEE TELEPHONE NUMBER	PRINT HYPHENS TTT-TTTT

INPUT RECORD FORMAT

FIELD LOCATION	FIELD NAME	DATA CLASS	COMMENTS
1-2	RECODE CODE	ALPHANUMERIC	CODE "L2"
3-11	EMPLOYEE SOCIAL SECURITY NUMBER	NUMERIC	
12-23	EMPLOYEE LAST NAME	ALPHANUMERIC	
24-34	EMPLOYEE FIRST NAME	ALPHANUMERIC	
35	EMPLOYEE MIDDLE INITIAL	ALPHANUMERIC	
36-59	EMPLOYEE ADDRESS	ALPHANUMERIC	
60-72	EMPLOYEE CITY	ALPHANUMERIC	
73-79	EMPLOYEE TELEPHONE NUMBER	NUMERIC	
80	SPACE	ALPHANUMERIC	

Note: Input Transaction Files do not contain the hyphens in the Social Security and Telephone Numbers for several reasons:

1. They would take up extra file space.
2. Added work typing when creating Transaction File.
3. Later when we test input data for validity, we will want to keep the numbers as numeric only.

Programmer's Job

With the above information, you the programmer, are now ready to develop the Employee List program Phase by Phase! Please go back to your previous COBOL Corner articles for a description of the Phases with their Steps if you do not remember them. I will name the Phases you should start now:

1. Print Chart.
2. Record Chart.
3. General Specification.
4. System Flowchart.
5. Structure Chart.
6. Program Flowchart.
7. Structure Walkthrough.

The next Phase, after the above have been completed, will be the Coding Phase (remember this does not mean KEYING). Let's start this together.

Coding Help

Remember we said that YOU were going to do most of this Program. You will note that Program #2 varies very little from Program #1. I will provide you with some Coding hints to get you started with this program's refinements. The added "hyphens" will require some changes in the Record-Description Entries and in the Procedure Division.

Input Record-Descriptions

```
01  ER-EMPLOYEE-RECORD.
    05  RECORD-CODE                          PIC X(02).
    05  ER-EMPLOYEE-SOC-SEC-NO.
        10  ER-SOC-SEC-3                     PIC 9(03).
        10  ER-SOC-SEC-2                     PIC 9(02).
        10  ER-SOC-SEC-4                     PIC 9(04).
    05  ER-EMPLOYEE-LAST-NAME                PIC X(12).
    05  ER-EMPLOYEE-FIRST-NAME               PIC X(11).
    05  ER-EMPLOYEE-MIDDLE-INIT              PIC X(01).
    05  ER-EMPLOYEE-ADDRESS                  PIC X(24).
    05  ER-EMPLOYEE-CITY                     PIC X(13).
    05  ER-EMPLOYEE-PHONE-NO.
        10  ER-PHONE-3                       PIC 9(03).
        10  ER-PHONE-4                       PIC 9(04).
    05  FILLER                               PIC X(01).
```

Output Record-Descriptions

```
01  EL-EMPLOYEE-LIST-LINE.
    05  FILLER                               PIC X(05).
    05  EL-EMPLOYEE-LAST-NAME                PIC X(12).
    05  FILLER                               PIC X(02).
    05  EL-EMPLOYEE-FIRST-NAME               PIC X(11).
    05  FILLER                               PIC X(01).
    05  EL-EMPLOYEE-MIDDLE-INIT              PIC X(01).
    05  FILLER                               PIC X(02).
    05  EL-EMPLOYEE-SOC-SEC-NO.
        10  EL-SOC-SEC-3                     PIC 9(03).
        10  EL-SS-HYPHEN-1                   PIC X(01).
        10  EL-SOC-SEC-2                     PIC 9(02).
        10  EL-SS-HYPHEN-2                   PIC X(01).
        10  EL-SOC-SEC-4                     PIC 9(04).
    05  FILLER                               PIC X(02).
    05  EL-EMPLOYEE-ADDRESS                  PIC X(24).
    05  FILLER                               PIC X(02).
    05  EL-EMPLOYEE-CITY                     PIC X(13).
    05  FILLER                               PIC X(02).
    05  EL-EMPLOYEE-PHONE-NO.
        10  EL-PHONE-3                       PIC 9(03).
        10  EL-PHONE-HYPHEN                  PIC X(01).
        10  EL-PHONE-4                       PIC 9(04).
    05  FILLER                               PIC X(36).
```

Procedure Division

```
MOVE "-"                          TO EL-SS-HYPHEN-1
                                     EL-SS-HYPHEN-2.
MOVE "-"                          TO EL-PHONE-HYPHEN.
```

Explanations

There are many ways to add Format Characters to Social Security Numbers and Telephone Numbers. The method I picked for this program will help you see what we are doing. We will use other methods with short cuts in future programs.

Looking at the Input Record-Description above you will notice that we broke the "group item", ER-EMPLOYEE-SOC-SEC-NO, into three (3) sub-fields using the Level-Number "10" for each sub-field. Remember that your Social Security Number is usually shown as "1###"; so we now have it in three (3) sub-fields. We did the same thing for the "group item", ER-EMPLOYEE-PHONE-NO, except we broke it into two (2) fields.

NOW, looking at the Output Record-Description you will find that we have added the "hyphen" to several sub-fields. This provides a character space to MOVE a "hyphen" into.

Review

Record-Descriptions -- entries are located just below the FD-- File Description. It is the Record-Description that tells the Compiler how to set up a record area for each file in which the Input Record can be stored and processed and the Output Record assembled and written. Thus, it provides the Compiler with the Format, or PICTURE, of one record of the File. Each entry begins with a Level-Number followed by two (2) spaces, the name of the Data Item, and a sequence of independent clauses descriptive of the Item. The last clause must be terminated by a period.

Position -- the Record is always described from left to right, i.e., from print position 1 to print position 132 for the case of an Output Line. Every one of the 132 columns must be accounted for.

Level -- the Record-Description entry must have Level-Numbers assigned. These Level-Numbers are used to show the hierarchy of the data within the logical record. There can be forty-nine different levels specified for a record, numbered from 01 thru 49. The name assigned to the entire record always has the Level- Number (01). The Level-Number 01 must be in area A, and for "good" programming style, the "0" must be in column 8! Major divisions (fields) within the Record are assigned a Level-Number, such as 05. These Level-Numbers must be in area B (NEVER in area A)! Again, for "good" style, each change of Level-Number should be indented by four (4) spaces. Also, for "good" style, we will identify the fields and sub-fields with Level-Numbers in increments of 5 (01, 05, 10, 15 and so on). Thus, to show that an elementary item (or field) belongs to a group item, we must assign it the next higher Level-Number than the group item.

```
    05  ER-EMPLOYEE-PHONE-NO.
        10  ER-PHONE-3                       PIC 9(03).
        10  ER-PHONE-4                       PIC 9(04).
    05  FILLER                               PIC X(01).
```

Notice that you show the end of the group item by using a Level-Number equal to or less than the Level-Number of the group item.

Name -- every field and sub-field within the Record-Description MUST be assigned a unique name (be sure it is a self-documenting name for "good" program style). This name is used to reference the field or sub-field in the PROCEDURE DIVISION statements. Any unreferenced fields are assigned the name FILLER (Reserved Word) to define those columns.

Format -- (Following the Level-Number and Data Name are a series of independent clauses.) Each field (or its sub-field) MUST be described as to size and type (numeric or alphanumeric), location of actual or assumed decimal point, and any editing desired. This description is given in the form of a PICTURE (or PIC), using the special PICTURE symbol. We will go into these in a future article.

Move Verb -- let's review the COBOL "MOVE" verb. Do you remember its Format? Here it is again:

```
      identifier-1
MOVE      or       TO  identifier-2 [identifier-3]....
        literal

WHERE
      identifier-1—represents the sending field.
      literal—(actual value specified), also a sending field.
      identifier-2 & identifier-3—represents a receiving field.
```

In this Program #2 we will use the MOVE verb to send the literal, "-", to the named output data sub-fields. We will use one (1) MOVE verb to send the literal to the two (2) receiving fields. Do you like this program style?

```
XXXXXX    MOVE "-"              TO EL-SS-HYPHEN-1  (NO PERIOD!)
XXXXXX                             EL-SS-HYPHEN-2. (NOW A PERIOD)
```

Coding & Compiling

With the information and hints that I have supplied, you should be able to write your Code on the COBOL Coding Forms. After writing the Code, perform the "walkthrough" referring to your Program #1 Listing. When you are satisfied that you have a "good" error-free program design, Compile the program. Use the instructions that we have used in previous COBOL Corner Articles. If you find that you have Errors, correct them as we have in earlier projects.

Link & Execute Program #2

Your HUG COBOL Corner Disk-I has the Transaction File--FILEL2.DAT--for this program. Using PIP, copy this File to your PRGM02 Disk A with the same name. LINK and EXECUTE Program #2 as we have done before in earlier COBOL Corner programs. You should obtain a Print-Out that will Match yours and my specifications (and of course no RUN-TIME ERRORS). If you do not get a correct Print-Out, review your Code Listing and your COBOL methods.

Closing

As a last resort, if you cannot find your Errors, prepare a "NEW" Disk A as we have done previously (NOT the Disk A you have been working with!). Again using PIP, copy PRGM02.COB and FILEL2.DAT from your HUG COBOL Corner Disk-1 to the "NEW" Disk A you have just prepared. Now Compile this Program #2 with a Listing from your printer. Next Link and Execute the Program #2. You should now have a Print-Out that will match the specifications! Compare the Listing and Print-Out from this Disk with the one that you obtained from your KEYED-IN Program #2. A comparison of Coding line by line should enable you to find your Errors! Be sure to do this ONLY after you have really tried to find you own Errors! This is the "Real World" way and the only way to learn Cobol!

The next COBOL Corner will start Program #3. We will design a program to print a Sales Report with Total Lines. This Program will use a Counter and Accumulators. We will also study and use the following COBOL verbs:

ADD
MULTIPLY
DIVIDE

We are going to find additional uses for the Working-Storage Section.

For Homework, you might want to get a head start by reading about these subjects in your COBOL-80 Manual.

GOOD LUCK with Program #2. Remember if all else fails, I will be able to help you! Do not get discouraged!

COBOL Corner NOTICE!!!

I have purchased and reviewed the NEVADA COBOL package by Ellis Computing. I find that it can be used to do some of the COBOL programs that I have planned for COBOL Corner, but not all of them. It does not adhere to all of the ANSI-74 Standards and thus will not produce transportable programs which is one of the big advantages of COBOL! The price of $29.95 might be an advantage to some of the COBOL Corner readers that want to find out if they really want to study COBOL. I have also reviewed many inquiries from COBOL Corner readers concerning what can they do if they have an H/Z-100. The NEVADA COBOL System will work on the 8085 side of the H/Z-100 as well. Here we have a better answer with the "NEW" COBOL-86 System!

I am preparing a Software Product Review Article for REMark that will cover the Pro and Con of NEVADA COBOL in detail along with HOW TO use it with COBOL Corner where it will be possible. Watch for it!

On The Leading Edge
Wm. M. Adney

MS-DOS 2.0
News Flash

I have just received the following quote from Tom Dornback of ZDS.

"The release date for MS-DOS 2.0 is June 1984. There has been a lot of excitement and demand for it. We intend to continue to support the Z-110 and Z-120 modules very heavily now that 2.0 is released."

"My Favorite Subroutines"

Dear HUG,

I have a neat little two line subroutine that I use to insert a one second pause in a program. It works just great on my H-100 (ZBASIC).

```
10 A = TIME
20 IF A = TIME THEN 20
```

It takes exactly one second to run whether interpreted or compiled, and it will not get confused and lock up at midnight. The accuracy of this routine is only dependant on the accuracy of the H-100 internal clock.

If pauses greater than one second are required, the following additions will do the trick in the same two lines, accuracy as before.

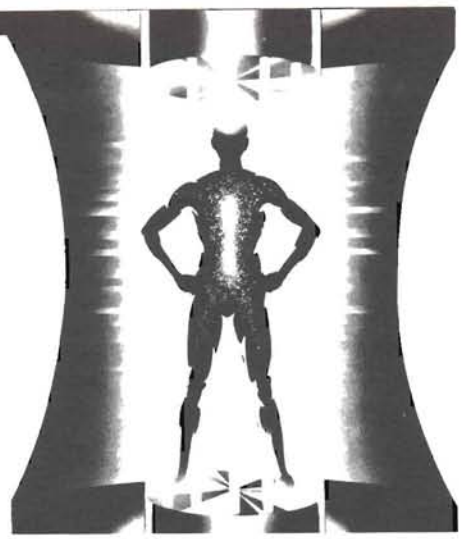
```
10 FOR J = 1 TO Q : A = TIME
20 IF A = TIME THEN 20 ELSE NEXT J
```

Where Q is the desired delay in seconds.

There are many possible uses and embellishments for this routine which I will leave to the imagination of the reader.

Arthur Calhoun
16 Cedar Valley Lane
Huntington, NY 11743

Managing Files - Help From AutoDex!



William M. Adney
P. O. Box 1477
La Mirada, CA 90638-1477

Have you looked at a disk directory lately and forgotten which file contains what? Or worse...when did I last update that file? And then deleted it only to discover that it was an absolutely critical file which took hours to reconstruct?

Without a doubt, file management is one of the most difficult tasks in any computer environment. As disk capacities expand, it's more difficult to remember exactly what a file contains. If you have well over 100 floppy disks as I do, it gets more than a little difficult to remember what all of the files contain. When did I last back up that critical file? And the file name limitation of up to eight characters with a three letter extension invites cryptic file names which are limited only by your imagination. Now what did I write that EA.COM file for? Let's test it and find out. Rats! Now I remember...it was a program that Erased All of the files on the target disk! Now where is my latest backup? What was the date on the backup?

Will this never end? How can I keep track of all of these files?

AutoDex Saves the Day!

Help is on the way with an amazing program called AutoDex. It has a screen display which is divided into two areas: the disk area and the file area as shown in Figure 1 at the end of this article. The disk area at the top has information about the disk as well as commands. The individual files are listed below and the default is to display the files in alphabetical order by file name. As you can see, each line contains the file size, change date, and a description of your choice which may be up to 42 characters long.

The cursor can be moved from the disk area to the file area by simply pressing the escape key. In addition, the escape key is used to abort any command in progress. Cursor keys are used to access the different commands in the disk area. The change date and the description in the file area can be modified by using the right arrow cursor key to move to the correct position and entering the appropriate information. Although there is no restriction on what characters you can enter in the Description field, the date is edited for validity.

The Disk Area

The Sort command allows you to sort the files on the display by file name (default), file type, size, and change date. The advantages of those options for sorting the directory are obvious so I won't spend any time discussing that.

The MULTiple command is used with the file area. It provides the capability to backup, list, or erase any number of files which are marked with an M preceding the file name. This capability is very similar to the popular SWEEP program, but is much more powerful.

The Current field displays the current disk drive. And of course the Backup field displays the backup disk drive. The current and/or backup drives may be changed by simply entering the designation of the new current or backup drive. AutoDex takes care of logging in the new disk so you don't have to worry about a CTRL-C.

The Date and DiskID fields are the keys to the power of AutoDex. The DiskID is the file name that contains all of your information on each file: the change date and the description as well as the file name. Up to six characters for the file name are allowed. After that, AutoDex creates a DiskID file which is in the form of--CPM-85.DID. When you enter the Date, the change date in the DID file line is automatically updated. This particular feature allows you to automatically update the change date for each file when it is accessed through the use of the eXecute command. I'll talk about that later.

The Exit command allows you to exit from AutoDex and return to the CP/M operating system.

FileList (FLst) will print the AutoDex screen to your system printer which is a handy thing to have. I keep all of the filelists in a three ring binder which I update periodically. It's a super way to keep track of what's on which disk.

Selecting User simply allows you to change the user number. What else can I say?

And no program would be complete without a Help screen. It displays enough information to help you remember the commands without the documentation, but AutoDex uses easy to remember commands so that you won't have to refer to help very often.

The File Area

The disk area contains the individual files which can be accessed by moving the cursor up and down. Since the H/Z-89 and H/Z-100 support reverse video, the line where the cursor appears is displayed in reverse video for ease of use.

If you have more than one screenful of files (20), how are they displayed? You can display the next page by pressing N or the previous page by pressing P. The screen also scrolls when the cursor is moved beyond the top or the bottom of the screen. And any line on the screen can be made the top line by pressing T.

AutoDex has a number of extremely useful command functions like C(opy), B(ackup), E(rase), V(iew - a file...like the Type command),

R(ename), and execute(X). The screen display includes the file name, file size, date of last update (ChgDte), and a Description field which allows you to enter a descriptive comment about the file. These commands are entered in the first column of the screen as shown in Figure 1.

Did you forget what was in the file? Use the View command to see the contents. Isn't that easier than entering the Type command? One letter does the trick. It even works with COM files in a way similar to the CP/M dump command, and it shows the ASCII equivalent on the screen where possible.

Entering the Copy command allows you to copy any file to a new file name on the current disk. And it asks you for the new file name. Rename does the obvious with a prompt for the new file name.

But you know that you should always back up your files...right? Now what was the command to copy files? Was it PIP B:=A: or PIP A:=B:? You won't have to remember all of the commands because AutoDex allows you to enter B for backup. The file is then copied to the backup disk under the same name. AutoDex makes the backup process so easy that there is absolutely no excuse for violating the prime directive...You WILL make backup copies of ALL critical disks (or files)!! And General Order Number 1...You WILL write protect ALL distribution disks as soon as you unwrap the package (if not sooner)! If you're really clever, you'll enter the new date on the backup disk, but I'll talk about that later.

Erase allows you to delete a file from the directory, and it even has a prompt asking if you are sure that is the file you want to erase.

The nicest part about the program is that it quite effectively replaces PIP, REN, ERA, STAT, and DIR with some very easy-to- remember commands. No more trying to remember all of the commands with the required syntax.

The Multiple command is very useful for backing up any number of files as I mentioned earlier. Each file is marked with an M in column 1 of the display. Escape to the disk area. Press return to select Mult, and press return again to select the backup option. Files can also be printed or erased by selecting the appropriate function with the cursor before pressing return.

If you have a very large disk, the Goto function allows you to move around quickly. Press G and you are asked to select Name (first 8 letters) or type (last 3 letters). For example, the disk file for my column begins with REM. I press G, select Name by pressing return, and enter REM. AutoDex finds the first entry in the directory beginning with REM. That may not sound like a big deal until you work with a disk containing about 80 files. It's a super quick way to move around the directory, and it's one of those thoughtful features that are very seldom found in a lot of programs.

And now to one of the best features in AutoDex...The eXecute. Would you believe that you can execute a data file? Without a shot being fired at dawn? No, seriously, it's one of those features that has all sorts of hidden applications, so I'll take a minute to explain how it works. The basic object here is to set up AutoDex so that by entering an X in front of a file, you are presented with a prompt to allow you to select a program to use with that data or text file. For example, if you type X and select the WS (WordStar) command, AutoDex will automatically start WordStar to edit the selected file. And if you have been careful to always enter the current date in the disk area, Auto Dex will automatically update the change date for the file with the current date. Now you don't have to remember when that file was last updated. It's done for you! All of this magic is done with a program called FINETUNE which allows you to enter the command (like WS or EDIT for Magic Wand). Then it asks if you want the date changed when the program is invoked, and finally if you want the file name sent with the command. There are some other options that you can select with this program including the date format which is normally displayed in MMDDYY order.

Installing AutoDex

Installing AutoDex is straight forward using their INSTALL program. The selection of the H/Z-19 terminal works for either the H/Z-89 or the H/Z-100, so there's not much to worry about for the installation. You must, however, have two disk drives since the Autodex installation program expects that the master disk will be in drive B. Remember to copy the master disk BEFORE you attempt the installation.

Also included is a terminal definition (TERMDEF) program which allows you to do custom installation if you wish, but it really isn't needed for the Heath/Zenith systems.

The tutorial included with the disk is very helpful, and all of the lessons may be deleted after you have finished. It took me about an hour to install AutoDex and run through the tutorial. Since the commands are easy to remember, it doesn't take too long to become familiar with the operation.

Do You Need AutoDex?

This is one of those programs that makes you wonder why no one thought of it before. In my book, it's one of those "gotta have" things that makes life much easier. The ease of making backup copies, the descriptive comments on the files, and the automatic update of the change date for each file are features that I couldn't live without at this point. By the way, this program used to be known as SimpliFile, and I bought it a year ago at CP/M-83. The distributor has been changed as has the name, but the features are still great.

AutoDex requires two disk drives for installation and takes 32K of disk space. That's a small price to pay for the features that it provides, but it could cause a space problem on the 100K hard- sectored drives.

The documentation is a little sparse, but it does include all of the necessary information. It consists of a fold-out reference card, but the tutorial takes care of most of the training. I suppose that a whole lot of documentation isn't really required since the program is so easy to learn and use.

AutoDex is priced at $150.00 and is available from Automatic Software in Santa Barbara. Additional ordering information, including a telephone number, is listed at the end of this column. It's available for CP/M only and may be ordered in the hard or soft sectored 5-1/4" Heath formats or the standard 8" format. The CP/M version works with CP/M-80 on the H/Z-89 or CP/M-85 on the H/Z-100. Although it's available for the IBM PC- DOS, I have not been able to get it running under Z-DOS. And I have tried using some of the emulators, but I guess that there are too many "hooks" set up for the IBM PC operating system.

CP/M-86 From Zenith

Zenith has implemented CP/M-86 on the H/Z-100 with the usual expertise that we have come to expect from ZDS. As I mentioned last month, it supports both 8 and 16-bit software, and at this point in my testing, I haven't found any bugs. Most of the command names are identical to those used in CP/M-80 and CP/M-85. The few that are different are no big deal, and I have listed the most obvious ones:

CP/M-80/85	CP/M-86	Comment
DUP	COPYDISK	
n/a	HELP	Provides on-screen command help
n/a	TOD	Time of Day (sets date and time)
SYSGEN	LDCOPY	
n/a	RDDOS	Read Z-DOS files to CP/M-86 disk
STAT	DIRS	Directory display of system files ($SYS)

The MOVCPM command is not used with CP/M-86 since the memory sizing is done when you cold boot the system. And the ASM86 and DDT86 perform the same functions as their 8-bit cousins with some changes to allow for the additional capabilities of the 16-bit processor.

There are, of course, some modest restrictions on the support of 8-bit application programs under CP/M-86. For those of you who would like to look up more detail in the documentation, they are located on page 22 of the Z-100 Utilities section. Restrictions are:

1. The disks are not automatically reset when an application program terminates. You must warm boot (CTRL-C) the system EVERY time you change disks or the new disk will not be logged into the system. This also means that some programs, like AutoDex, will not run properly in the 8-bit emulation mode since they apparently rely on the disk reset and a $$$.SUB file.

2. Any 8-bit programs which modify the operating system and stay resident (e.g. KEYMAP, XSUB, and DESPOOL) will not run.

3. All calls to the BDOS and BIOS that return the address of tables now return a pointer to a copy of the table. CP/M-86 supports the disk header table, the disk parameter table, and the allocation vector. One copy area is provided for each type of table.

4. CP/M-86 supports only the standard Digital Research BDOS and BIOS entries. I wouldn't expect this to be a problem with most software.

Chaos Reigns Supreme!

Not being content with leaving well enough alone, I decided to change jobs. I have accepted a position as senior consultant for Total Assets Protection, Inc. in Arlington, Texas. Yes, that means that I am moving to Texas. Although most of my background is directly related to computer security and disaster recovery, my new company also provides services for data center design and construction management as well as other related areas. Needless to say, I'm quite pleased with my new job. At this point, I'm quite involved in the development of a computer disaster recovery plan for a major Texas savings and loan institution.

I've been commuting to Texas for the last month, and I thought the drive to Pasadena was bad! Oh well, things will settle down one of these days. You can still send mail to me at the above address, and it will be forwarded to me when we move...whenever that is!

Reviewing New HUG Software

If you're still looking for a good reason to join HUG, take a look a some of the great software they offer. It is, without question, one of the best (if not THE best) software values around. For around $20.00, you can get some really dynamite software! As one of the regular (hopefully!) features in this column, I will be reviewing some of the latest HUG software. It's difficult to get a user perspective on the software when the product announcements are all in the same format. But enough of that...let's look at some software.

Z-DOS Keymap (885-3010-37)

I bought the CP/M Keymap (885-1230-37) some time ago, and it's cousin, the Z-DOS Keymap, is also a tremendous value. Both programs allow you to alter the codes produced by the Function and keypad keys. The CP/M version allows up to ten (10) characters on each key, and the Z-DOS version allows up to twenty (20). Both versions allow you to create labels for the function keys on the 25th line. By the way, the CP/M Keymap was created for the H/Z-89, so it can not define all of the function keys for the H/Z-100.

Both versions include some configured keymaps that can be used straight from the disk. KEYBAS can be used to generate commmands and program lines for BASIC (CP/M) and Z-BASIC (Z-DOS). A KEYSYS version is also included with the disks which allows you to press a function key to display a directory, format a disk and other commonly used commands. And of course, neither disk would be complete without a pre-configured keymap for WordStar.

For $20.00, either one (or both) is an excellent timesaver for those frequently used commands. I've used the CP/M version with WordStar for about eight months, and I couldn't live without it now. In addition, I've created custom versions for the system commands which I also use a lot. If you have WordStar running under CP/M or Z-DOS, you "gotta have" these programs! By the way, both of these programs were written by Pat Swayne who did his usual fine job in creating some very useful programs.

Z-DOS Utilities (885-3008-37)

The Z-DOS Utilities disk is another one that has a CP/M (885-1226-37) cousin that I got some time ago. My favorite program is the directory that displays an alphabetized list of files on a disk (DIR19 for CP/M and DIR100 for Z-DOS). Details of the Z-DOS disk were announced in the December 1983 REMark, and the CP/M version was announced in March 1983.

In addition to DIR100, the Z-DOS disk has TERMZ100 which is a modem control program. DTERM is a "dumb terminal" modem program which does not intercept any control characters or escape sequences except for CTRL-E which is used to return to Z-DOS. One of these days I guess I'll have to see about getting a modem, but I really haven't had much time to do any research on them.

Another program, PSET25, is very useful for changing the pitch and line spacing on the H/Z-25 printer. Although I'm not much of a BASIC fan, a neat program is included to help you set up menus for your programs. If you're setting up your own system, these menu programs could be just the ticket if you don't want to learn assembly language.

Other programs included allow you to do some clever things with other printers such as the IDS Prism color printer, Printek, and the Epson MX-80. One program not listed in REMark was COLOR.COM which allows you to change the foreground and background colors on the CRT. With all of the emphasis on color these days, I suppose that I'll have to get a color monitor too, but I think I'll wait to see if Heath develops a kit for the ZVM-133 or ZVM- 135 monitors. I'm currently using the ZVM-122 monitor because I found that the amber display seems to be easier on my eyes.

Since it looks like this column is getting longer than I had originally estimated, I'll talk about the HUG CP/EMulator and Cheapcalc for Z-DOS in the next column.

Where's the Book?

In the February column, I mentioned the CP/M and Z-DOS FlipFast

Command Guides that I had written. The Zenith/Heath CP/M-80/85 FlipFast Guides were shipped in February. I've received a number of nice comments (and letters) about the CP/M-80/85 book, but it's always followed by: "Where's Z-DOS?". Although the Z-DOS book is written, it turns out that I seriously underestimated the time required to complete all of the gory details. Typesetting, layout, proofreading, printing, and binding all take much longer that I originally thought. Have patience...it's actually written, and I hope that it'll be available by the time you read this, but I won't promise anything.

Hints and Kinks

Are you interested in a way to renew nylon ribbons for your H/Z-25 (or any other printer) for just a few cents? Actually, your initial investment will be a couple of dollars for a can of WD-40 which is a light spray oil available in most automotive parts stores.

Place the old cartridge upside-down on something so that there is no pressure on the ribbon advance twist knob. Carefully (and I mean CAREFULLY!) separate the cartridge halves or you'll have an interesting job (not to mention messy) trying to put it back together. Saturate the ribbon (don't flood it) with the WD-40. About five passes of back and forth spraying ought to do it. Replace the bottom half of the cartridge, and let the ribbon set upside-down for about a week.

Just before you use it in your printer, be sure to advance the ribbon enough so that the renewed ribbon is in front of the print head. Run a sheet or two of paper through the printer in the test mode so that any excess oil will not ruin a good printout. You'll have to experiment a bit to find the right amount of spraying time, but this works until the ribbon is totally beyond repair. And remember, if you accidentally ruin the cartridge, you haven't lost anything because the ribbon was useless to begin with. Then you'll know how to do it the next time.

Transferring Files Between the H/Z-89 & H/Z-100

My thanks to Jim Johnson for his nice letter in the March 84 issue of REMark about the December 83 column on trading the H/Z-89 for an H/Z-100. For those of you interested in the IBM compatibility with the H/Z-100's, I had already planned a column on that which will appear next month. See below for more information on that.

Jim also asked about file transfers between the two systems. Since I have already been through it, I'll spend a few paragraphs on it.

ZDS has thoughtfully provided an appendix (Appendix H) in the Z-100 User's Manual (not the Tech Manual) on "Using Z-89 Software on the Z-100". Three conversion procedures are discussed in detail, but they use the same general techniques. If you have 8 inch drives, you can use the same data disks. Note that system disks are NOT interchangeable...that is CP/M 2.2.03/04 (CP/M-80) cannot be used on the H/Z-100. You must use CP/M-85.

For the 5 1/4" disks, the procedure is a little more complex. I'll assume that your system has an internal H-17-1 single sided drive, a Z-89-37 soft-sectored disk controller, and an HS-37 floppy disk system. If you have a soft-sectored disk controller and the H-17-1 disk drive (standard H/Z-89 internal drive), you can do the conversion without too many problems. You will also need at least two disk drives. Your objective is to create 48 TPI, soft-sectored disks that can be read by the H/Z-100. But these disks CANNOT be created on the double sided drives which are part of the HS-37 or ZC-37 floppy disk system. Even though you can configure the system to write 48 TPI disks, disks created by the HS-37/ZC-37 drives can not be read by the H/Z-100. This has to do with the way the quad density drives write to the disk at 48 TPI, but don't try it. Incidentally, this is not documented very well in the User's Guide.

First of all, you must have all of your disks in a soft-sectored format. If your internal drive is connected to the hard-sectored disk controller, change it to a soft-sectored drive by connecting it to the H-37 controller. Detailed instructions for doing this are included in the H-37 controller manual. I assume that your external drives can read all of your existing disks at this point. All you have to do now is format a sufficient number of disks on the internal drive to contain all of the data from the other disks, and then copy it to the disk in the internal drive. You don't need to copy the CP/M utilities since they are included with CP/M-85, but don't forget to copy your word processors, spreadsheets, and so on.

All of that may sound relatively simple, and it's really not too bad except for one thing. I had virtually all of my software on the 640K H-37 disks. But the H-17-1 drive will only format a single sided, soft-sectored drive to something on the order of 148K! Not all of my old disks had 640K of data on them of course, but I had to buy about 2 disks for every old one to do the conversion.

At this point, I had zillions of disks formatted on one side which my new H-100 could read. But the H-100 "normally" uses a double sided disk which has over 300K of storage. Back to the conversion again to convert the single sided disks to double sided.

Corrections, Changes or Whatever

With all of the job changes, book writing, and other normal chaos, it seems that my magic fingers managed to pick up the old versions of the programs published in the February column. For those of you who have played with those programs, it seems that there was a slight omission in Listings 1 and 2 for the printer form feed function. Obviously the BDOS call 9 and the Z-DOS DOSF_OUTSTR function will not send characters to the printer. And so the formfeed command would not work. Rather than go into a lengthy technical explanation, I have included only the "MAIN" part of the programs here as Listings 1 and 2 at the end of this article.

Since the CP/M call to send a character to a printer is different than a CRT, note that the lines which are "commented out" (i.e. preceded by a semicolon) are for printer control. All you have to do is to delete those semicolons and place semicolons in the first two lines of the program to send the character to the printer.

The Z-DOS version of the program is shown as Listing 2. You will get a "No STACK segment" error message when you use the LINK command...ignore it. That error message is normal (in fact required) when you develop a COM program for Z-DOS. I've explained the reasons for that in the Z-DOS FlipFast Guide under the LINK and EXE2BIN commands.

In The Mail

Software Toolworks, in the person of Susan Hayes, has provided me with some of their excellent programs to review in the column. I've already looked at a number of them, and they have a very impressive line. Aside from being good quality software, they make it available at reasonable prices which usually is less than fifty dollars. We'll be taking a look at their software which includes editors, spreadsheets, games, assemblers/compilers, and other general goodies over the next few months.

Ed Percy of Micro-Systems Software has also sent their new word processor, MSCRIPT, for Z-DOS. He told me that he thought it was "novice friendly", and I agree. It sells for $79.95, and I'll tell you more about it when I've had a chance to give it a little more testing.

Next Month

With the Zenith release of the new Z-150 series of "IBM compatible"

computers, it seems appropriate that we look at some of the compatibility problems next time. Although having been in data processing for a number of years, I do agree that a standard is necessary. But I don't agree with the one that IBM established in the de facto mode for microcomputers. It makes hardware and software compatibility difficult, to say the least, but I guess that's the standard that we'll have to live with.

As a part of the column, I'll also look at some programs (IB-Em and the Z-UTIL package) that help the Heath/Zenith world run some of the IBM-PC software on the H/Z-100. If you're interested in IBM compatibility with the H/Z-100, see next month's column.

Listing 1

```
;Clear Screen Program for H/Z-89 and H/Z-100 terminals
;         For CP/M-80 and CP/M-85

MAIN:
        LXI     D,CLS    ;Load Heath clear screen
;                          function(line 1)
        MVI     C,9      ;CP/M print string
;                          function(line 2)
;       MVI     E,FORMF  ;Substitute for line 1 for
;                          printer form feed
;       MVI     C,5      ;Substitute for line 2 for
;                          printer form feed
        CALL    BDOS     ;Call CP/M
        RET              ;Return to CP/M
```

Figure 1

Sample AutoDex command screen

```
*Disk Area *      Sort      Mult      Current:A      Backup:B  Date:022884
Disk Left: 214K   Exit      FLst      User: 0                  ID: REMARK
File: 2 of 37
-------------------------------------------------------------------------
CMD    Name       Type    Size   ChgDte    Description
-------------------------------------------------------------------------
     | —REMARK   DID  |  4K  | 022884  |
     | REM2-84   ART  |  10K | 112883  |Escape sequences
     | REM3-84   ART  |  10K | 121283  |ANGEL print buffer
_=>  | REM5-84   ART  |  10K | 031184  |AutoDex article
     | REM6-84   ART  |  8K  | 031184  |Software Toolworks, MSCRIPT
```

Listing 2

```
;Clear Screen Program H/Z-100 terminals
;         For Z-DOS ONLY


MAIN:
        MOV     DX,OFFSET CLS    ;Load Heath clear screen
;                                  function(line 1)
        MOV     AH,DOSF_OUTSTR   ;Z-DOS print string
;                                  function(line 2)
;       MOV     DX,FORMF         ;Substitute for line 1
;                                  for printer form feed
;       MOV     AH,DOSF_PRINTOUT;Substitute for line 2
;                                  for printer form feed
        INT     DOSI_FUNC        ;Call Z-DOS to send
;                                  message
        INT     DOSI_TERM        ;Return to Z-DOS
```

Products Reviewed

AutoDex — $150.00
Automatic Software
1035 Santa Barbara St.
Santa Barbara, CA 93101
(805) 963-5861

CP/M-86(OS-63-2) — $250.00
Heath Company
Benton Harbor, MI 49022
(800) 253-0570

Z-DOS Keymap(885-3010-37) — $ 20.00
CP/M Keymap(885-1230-37) — $ 20.00
Z-DOS Utilities(885-3008-37) — $ 20.00
CP/M Utilities(885-1226-37) — $ 20.00
Heath Company Parts Dept.
Hilltop Road
St. Joseph, MI 49085
(616) 982-3571

MSCRIPT — $ 79.95
Micro-Systems Software
4301-18 Oak Circle
Boca Raton, FL 33431
(800) 327-8724

HELP Is Here!

Jennifer McGraw
12741 SW 68th Tr.
Miami, FL 33183

This is a good program which I wish I had had five years ago. It works as advertised, and is easy to use. I have only one complaint about the package itself: there's almost too much documentation. However, it was easy enough to find exactly what I needed to know, once I got over the shock.

This package is actually a substitute for all those charts and notes which I have tucked between the computer and external drives, like the terminal escape codes. Most of these cost only some time, one or two cost a couple of bucks. INSTANT HELP strikes me as being overpriced.

However, the program does give instant help. Suppose that in MBASIC you want to use most variables as integers, and a few as single precision (an instruction I just can't remember how to spell). Well, if you had Instant Help, you would simply type HELP, still in MBASIC, mind you, and before your eyes would appear a list of almost all the commands. Takes up more than one screen, but the clever little program holds the first screen. If the command is there you can skip the next screen by typing an 'S'. You don't even have to remember that because the program tells you so. If you want to see the next screen, simply enter any other key.

OK, so now you know DEFINT and DEFSNG. What follows that? Type HELP DEFSNG and get some pointers on that command, to wit:

```
HELP DEFSNG
DEFSNG A-D,X
Defines variables starting with
A thru D or X as single precision.
END INSTANT HELP
```

When you purchase Instant Help, you receive one disk and 26 pages of documentation which give complete directions for making a backup copy and for installing INSTANT HELP. On any given disk, once you have learned how to use the program, you need only HELPON and one .HLP (keyword) file. And if you are running a multi-drive system, the .HLP file can be on any other disk in an active drive.

To use it, type HELPON before running any other program. HELP loads itself right underneath the system in the top of memory, and is available at almost any time. According to the documentation, it takes about 2K. Then just type HELP or HELP KEYWORD to get your information. HELP.HLP is the default keyword file. If you want to change this file simply type:

```
HELP /B:MBASIC.HLP   (or whatever file you choose.)
```

The file will be logged on, but is not moved into memory, so that every time you use it, there will be some drive action. The program will remain active until the next cold boot or until you type HELP-OFF.

Suppose that you need a file that doesn't come with the package. You write your own. The documentation makes it very easy and you can type anything you want to explain the keywords. After developing the file, using any editor, you then run PREHELP.COM. It makes a formatted copy of the file, adding a machine language header to it. This copy, or .HLP file is then ready to be used.

After playing with the program a bit, I wrote a file on TEXT.COM, which took a while, but now there it is, ready for the next time I want to make a beautifully formatted letter, and can't remember some of the more esoteric commands. Actually, I think there's a little too much in the file, but by re-editing the original, and again running PREHELP, I have something very useful. Another thing I noticed, at the moment, as far as TEXT is concerned, I know a lot about it, having looked everything up. But six months from now - well, you know how that is. Can't even remember what was up in programs that I wrote.

I have tried HELP with CP/M, PIE, MBASIC, and Magic Wand's EDIT and PRINT and it works fine. Although the authors do not recommend that the program be used with full screen editors, I didn't have any problems, so long as I moved the screen below the actual text area before calling for HELP. All the words that HELP puts on the screen are ignored by PIE or EDIT other than the word HELP as entered by you. But, don't delete those lines. Back up one page then come one page forward again, because you never know, you may delete some of the text. The word HELP has to be at the left margin in order to work, by the way. And, every now and then, it doesn't work the first try. Try again and there it is.

CP/M users should note that the program has not been tested with CP/M versions earlier than 2.2. It probably will not work with a non-standard CCP and might not work with programs that intercept BIOS calls 2 or 3. (Anyone who writes CP/M programs that make calls to the BIOS should have his head examined. This almost guarantees that the program is not transportable to a different computer, even though it's using the same version of CP/M. My battle cry is "Use the BDOS".)

The files included on the CP/M disk are:

HELPON	.COM	PREHELP	.COM
HELP	.HLP	HELP	.TXT
CPM	.HLP	CPM	.TXT
IDEA	.HLP	MBASIC	.TXT
ED	.HLP	SYSCALL	.HLP
MBASIC	.HLP	ED	.TXT
SYSCALL	.TXT		

I assume that similar files are on the HDOS version. It was nice of the J. E. Brancheau Engineering Company to put the text files in, so that you can edit them to your own specifications and re- format. For instance, the MBASIC I use does not have a command called SPACES$ but it is easy enough to substitute STRING$.

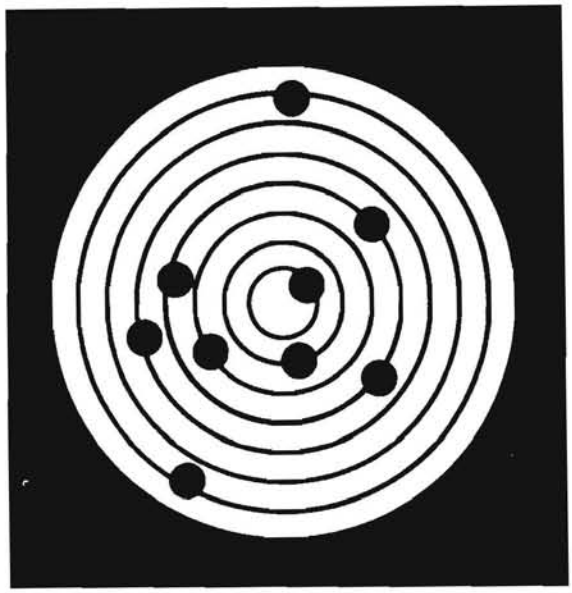
INSTANT HELP
Copyright by Brancheau and Campbell
J. E. Brancheau Engineering Company
Box 67
Trenton, MI 48183

Available for H/Z-89
CP/M or HDOS

Price: $39.95

Random Files

Sorting - Part 2



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

Last month we looked at two sorting routines. This month we'll look at two more and provide a set of files to be sorted so we can compare the efficiency of different sort routines under three different sets of conditions for both short and long files. We'll also add some improvements to our sorting methods so that they can be readily adapted to any random file.

We can read many articles, each of which extolls the virtues of a different sorting routine, stating that this is the best (that usually means the fastest) one. What gives? Who is right? Depending on the circumstances of the test, each could be right. A routine which is fast for many items may be a poor performer when sorting a short list. One which can sort and re-order many items in a flash may come in second best when most of the items were already in order as when re-sorting an updated file. The problem is that there is no standard for comparison. Even sorting the same file as we did last month is not fair as only one set of conditions is represented.

A file to be sorted may be represented by one of three conditions:

1) Most items are in order.
2) Most items are out of order.
3) The items are arranged in a random fashion.

We can set up files to represent each of these conditions, then we can sort each type of file with the routines we want to compare and thus produce meaningful data on their performance. The first case will have numbers, all in order, the second will be in reverse order, and the third will be in random order. We will produce two files of each type, one with 100 items and one with 1000 items. This will give results showing relative performance for long and short files.

The program FILEGEN1.BAS generates the file SORTTST1.DAT. Type, SAVE, and RUN it.

```
2 '**********FILEGEN1.BAS**********
4 '********DAVID E. WARNICK********
6 '*********COPYRIGHT 1983*********
1000 OPEN "R",#1,"SORTTST1.DAT",5
1010 FIELD #1, 5 AS A$
1020 FOR X=1 TO 100
1030 LSET A$=MKI$(X)
1040 PUT #1,X
1050 NEXT X
1060 CLOSE #1
1070 END
```

Now change lines 1000 and 1020 to:

```
1000 OPEN "R",#1,"SORTTST2.DAT",5
1020 FOR X=1 TO 1000
```

Run this program also. You now have files of 100 and 1000 in order. Next change lines 1000 and 1040 and run the program again.

```
1000 OPEN "R",#1,"SORTTST4.DAT",5
1040 PUT #1,(1001-X)
```

For the fourth test file change program lines 1000, 1020, and 1040 as follows and run it one more time.

```
1000 OPEN "R",#1,"SORTTST3.DAT",5
1020 FOR X=1 TO 100
1040 PUT #1,(101-X)
```

You have now generated two more files. These are made up of 100 or 1000 records in exactly the opposite order from that desired. They will test worst-case operation of your sort programs. Finally, we'll generate two files of randomly ordered numbers. To do this type, SAVE, and RUN FILEGEN5.BAS. Line 1050 is included as these programs run a relatively long time (several minutes) and you can see that they are progressing.

```
2 '**********FILEGEN5.BAS**********
4 '********DAVID E. WARNICK********
6 '*********COPYRIGHT 1983*********
H000 DIM A(100)
1010 FOR X=1 TO 100
1020 A(X)=X
1030 NEXT X
1040 FOR X=100 TO 1 STEP -1    'FOR EACH ITEM IN ARRAY
1050 PRINT "X NOW EQUALS ";X   'TELL OPERATOR WHAT'S
                                HAPPENING
1060 Y=INT(X*RND(1))+1         'PICK A RANDOM NUMBER FROM
                                LIST
1070 Z=A(Y)                    'TAKE NUMBER AT THAT
                                POSITION
1080 FOR W=Y TO X-1            'FROM NUMBER PICKED TO END
                                OF LIST
1090 A(W)=A(W+1)               'MOVE REMAINING NUMBERS
                                DOWN
1100 NEXT W                    'CONTINUE TILL DONE
1110 A(X)=Z                    'INSERT NUMBER PICKED
1120 NEXT X                    'CONTINUE TILL ALL NUMBERS
                                JUMBLED
1130 OPEN "R",#1,"SORTTST5.DAT",5  'OPEN A RANDOM FILE
1140 FIELD #1, 5 AS A$         'DESCRIBE A RECORD
1150 FOR X=1 TO 100            'FOR EACH NUMBER IN ARRAY
1160 N=A(X)                    'GET RANDOM NUMBER FROM
                                THAT POSITION
```

```
1170 LSET A$=MKI$(N$)          'PREPARE NUMBER FOR FILE
1180 PUT #1,X                  'PUT NUMBER IN FILE
1190 NEXT X                    'CONTINUE TO END OF ARRAY
1200 CLOSE #1                  'CLOSE THE FILE
1210 END
```

When this has been run, change lines 1000, 1010, 1040, 1120, and 1140 as follows and run this program a final time.

```
1000 DIM A(1000)
1010 FOR X=1 TO 1000
1040 FOR X=1000 TO 1 STEP -1
1120 OPEN "R",#1,"SORTTST6.DAT",5
1140 FOR X=1 TO 1000
```

Use pip or a similar program to copy your six data files to a backup disk so you don't lose them. You now have random files available to test your sort routines as follows:

SORTTST1 .DAT	100 items in order
SORTTST2 .DAT	1000 items in order
SORTTST3 .DAT	100 items in inverse order
SORTTST4 .DAT	1000 items in inverse order
SORTTST5 .DAT	100 items in random order
SORTTST6 .DAT	1000 items in random order

Before we get into our last sort routines and test them, there are some things we must change. In the past, every time we wrote a sort program, we wrote the name of the file to be sorted into that program. We should be able to specify the file name from the console and the following module will let us do that. We will always use the file extension .DAT. The extension could be a manual input too. If you want it, you should have no trouble adding the feature. While we're at it, we'll also make the drive name for the file a manual input, too. This will be a convenience to multi-drive users.

In sorting, we can input from one file name and output to a separate name. This saves the original file if something goes wrong. It's also necessary here as we don't want to destroy the standard files we just developed. Let's take a console input for the output file too. Type and SAVE the following module. It's a good idea to save a copy on a backup disk, and keep it in ASCII format so the MERGE command can be used. Use the command line:

```
SAVE "SORTHDR",A
```

Here's the program module.

```
2 '**********SORTHDR.BAS**********
4 '********DAVID E. WARNICK*******
6 '*********COPYRIGHT 1983********
1000 INPUT "WHAT IS THE NAME OF THE INPUT FILE ";FI$
1010 INPUT "WHICH DRIVE IS THAT FILE ON ";DI$
1020 INPUT "WHAT NAME SHALL I GIVE THE OUTPUT FILE ";
     FO$
1030 INPUT "WHICH DRIVE SHALL I PLACE THE OUTPUR ON ";
     DO$
1040 IN$=DI$+":"+FI$+".DAT"    'ASSEMBLE INPUT FILE NAME
1050 NO$=DO$+":"+FO$+".DAT"    'ASSEMBLE OUTPUT FILE NAME
```

You can refer to prior articles from this series for screen control information and make this routine as fancy as you like.

Another thing we've done in the past is assume a set length (total number of records present) for a file to be sorted. This is OK in the testing environment, but when we apply these programs to real situations, all kinds of files will be encountered. The following routine opens the file specified in SORTHDR.BAS and determines how many records it contains.

```
2 '**********FILELEN.BAS**********
4 '********DAVID E. WARNICK*******
6 '*********COPYRIGHT 1983********
2000 OPEN "R",#1,IN$,5         'OPEN FILE NAMED BY IN$
2010 FIELD #1, 5 AS A$         'DESCRIBE THE RECORD
2020 A=1                       'SET BOTTOM LIMIT OF
                                SEARCH
2030 C=32767                   'SET UPPER LIMIT OF SEARCH
2040 B=INT((A+C)/2)            'SET SEARCH POINT
2050 GET #1,B                  'GET THE RECORD TO TEST
2060 IF EOF(1) THEN C=B-1 ELSE A=B+1    'CHECK WHETHER
                                RECORD EXISTS.  IF IT
                                DOES NOT, MOVE TOP OF
                                SEARCH DOWN ELSE MOVE
                                BOTTOM OF SEARCH UP.
2070 IF A<C GOTO 2040          'NOT FOUND YET, TRY AGAIN
2080 GET #1,C                  'SEARCH WITHIN DISK SECTOR
2090 IF ASC(A$)=0 THEN C=C-1:GOTO 2080       'IF NOTHING
                                IN RECORD, CHECK NEXT
                                RECORD DOWN
2100 PRINT "THE LAST RECORD IN THE FILE ";IN$;
     " IS NUMBER ";C           'MESSAGE TO TELL WHAT LAST
                                RECORD NUMBER IS.
2110 CLOSE #1                  'CLOSE THE FILE
2120 END
```

Let's take a look at how this routine works. It all centers around line 2060. EOF(1) is the MBASIC END-OF-FILE function. If we ask the computer to get a record number which is higher than the last existing record in that file, EOF will be -1 which is the same as true. If the record number is not beyond the last record in the file, EOF will be +1, or false. You'll recognize from past search programs we've written that lines 2020 thru 2070 set up a BINARY SEARCH ROUTINE to find the highest numbered record which does not return EOF as true. Note too that this does not determine how many records are present, but rather it determines the highest numbered record. In random files, some of the lower numbered records could be missing.

If we've found the highest record by line 2070, what are we doing on 2080 and 2090? If you read your MBASIC and CP/M manuals (I'm sorry I haven't kept up on HDOS. Please read your manuals for this.), you'll find that your system writes to and reads from random files 128 bytes at a time. It does this with an area of memory called the random buffer. When we generated a file with records less than 128 bytes long the buffer waited until it was full, then wrote to the disk. This 128-byte chunk of data fills one sector on the disk and your operating system (CP/M) takes care of all the details for you. It can even split a record across two sectors of the disk if necessary and still keep track of it. The buffer size (and the 128-byte sector on the disk) is the reason the MBASIC manual cautions against records longer than this.

If we're finished putting information into a random file and the buffer is not full yet, it must write all 128 bytes anyway. If there is room for another record on the disk sector, the computer assumes it is there and will not return true for the EOF test. What we really found on line 2070 is a record number which could be stored on the last sector of the disk used for our file, and which is at least as high as the last existing record in that file. To check whether something was actually written in the record we get it and use the ASC function to see whether something is there. If it's blank the ASC value will be zero, so we'll reduce the record number by 1 and try again. When information exists in the record it will have an ASCII value greater than zero and we'll have found the highest numbered record in the file.

Input and SAVE FILELEN.BAS then MERGE SORTHDR.BAS and SAVE again as TEST1.BAS in ASCII format. RUN this program. When asked for an input file name and drive, specify one of the files we created above. A carriage return is sufficient for the output file name and drive as we won't need one now. As written, this test is only accurate for files with a record length of 5 characters. You should get answers of 100 or 1000 in each case.

We've generated six test files and can determine their length, but if you tried to print them using LIST or TYPE you got a surprise. The converted integers don't print as you'd expect. The following program when merged with TEST1.BAS will print these six files and their

sorted versions which we'll generate later. The printout will include ten numbers on each line.

```
2 '**********READOUT.BAS**********
4 '********DAVID E. WARNICK********
6 '*********COPYRIGHT 1983********
3000 Z=INT(/10)*10              'SET NUMBER OF LINES
3010 FOR X=0 TO Z STEP 10       'FOR EACH LINE
3020 LPRINT "#";(X+1);TAB(10);      'SET UP LINE
3030 FOR Y=1 TO 10              'FOR EACH CHARACTER ON
                                 LINE
3040 GET #1,(Y+X)               'GET THE RECORD
3050 LPRINT CVI$;" ";           'CONVERT AND PRINT NUMBER
3060 NEXT Y                     'CONTINUE ACROSS LINE
3070 LPRINT                     'MOVE PRINTER TO NEXT LINE
3080 NEXT X                     'CONTINUE THRU FILE
3090 CLOSE #1                   'CLOSE THE FILE
3100 END
```

Type the program, MERGE TEST1.BAS. Delete lines 2110 and 2120 which stopped the old program. SAVE as READOUT.BAS and run the program. Make the run six times, once for each of the TSTFILE data files we created, and you'll see what the next two programs will have to sort.

At last we can do some more sorting. After typing each of the next two sort routines, you'll have to MERGE the sub-programs SORTHDR.BAS and FILELEN.BAS to it. Then delete lines 2110 and 2120 so it doesn't stop in the middle. When running these sort programs, specify one of the six files we created above as the input file and any name you like for the outputs. I'd suggest making the number 1 to 6 part of the name to keep track of which file created the output. After sorting you can run READOUT.BAS and specify the output of the sort as the input file name and print out the results of the sort routines work. As we did last month, an input is required to start the actual sort so you can time it if you like.

Our first sort routine is called the HEAP sort. It works by comparing one item in a list with the larger of two other items which are twice as far down the list and placing the largest in the first position. This orders the file from largest to smallest items. The order is then reversed. As an example, in a 10-item file, item 5 would be compared to item 10 and the largest item placed in position 5. Next, item 4 would be compared to the larger of items 8 & 9 and the largest placed in position 4, etc.

In our program HEAPSORT.BAS, line 4200 finds the middle of the file for the first comparison and causes us to work down through the list. The actual sort begins at line 4500 which finds the first item for comparison. Lines 4510 and 4520 prevent errors if we start above the middle of the list. Lines 4530 thru 4560 compare items and make swaps if necessary. If a swap is required, lines 4570 and 4580 force us back up the list to make sure we haven't gotten anything out of order by making the swap. When the list has been ordered, lines 4240 thru 4280 re-order it to smallest item first. We then get the message "SORT IS DONE" so we can stop timing the run, and the rest of the program writes the sorted information to the output file. Don't forget to MERGE SORTHDR.BAS and FILELEN.BAS to this program. Then delete lines 2110 and 2120 and SAVE "HEAPSORT". When running this sort program with our test files, expect 100-item lists to take 30 - 40 seconds and 1000-item lists to take 9 or 10 minutes. Print the output files using READOUT.BAS and compare them to the printouts of the test files. This will verify that your program worked as it should.

```
2 '**********HEAPSORT.BAS**********
4 '********DAVID E. WARNICK********
6 '*********COPYRIGHT 1983********
4000 DIM X(C)                   'DIMENSION ARRAY FOR FILE
                                 SIZE
4010 FOR X=1 TO C               'FOR EACH RECORD IN THE
                                 FILE
4020 GET #1,X                   'GET THE RECORD
4030 X(X)=CVI(A$)               'CONVERT AND PUT NUMBER IN
                                 ARRAY
4040 NEXT X                     'CONTINUE TO END OF THE
                                 FILE
4100 PRINT "PRESS ANY KEY TO START THE SORT"
                                'MESSAGE ON SCREEN
4110 Z$=INPUT$(1)               'TAKE KEYBOARD INPUT TO
                                 START SORT
4190 B=C                        'SET TOP OF LIST INFO
4200 FOR A=INT(C/2) TO 1 STEP -1   'FOR RANGE OF SORT
4210 X=A                        'SET BASE FOR COMPARISON
4220 GOSUB 4500                 'DO COMPARISON
4230 NEXT A                     'CONTINUE DOWN LIST
4240 FOR B=C-1 TO 1 STEP -1     'RANGE FOR FINAL ORDERING
4250 SWAP X(B+1),X(1)           'SWAP VARIABLES
4260 X=1                        'SET COMPARISON START
4270 GOSUB 4500                 'DO COMPARISON
4280 NEXT B                     'CONTINUE FINAL ORDERING
4290 GOTO 4800                  'ALL DONE
4500 Y=2*X                      'SET COMPARISON ITEM 2
4510 IF Y>B THEN RETURN         'BASE ABOVE MIDDLE OF LIST
4520 IF Y=B GOTO 4550           'ITEM 2 IS END OF LIST
4530 IF X(Y)>=X(Y+1) GOTO 4550 'IF FIRST ITEM BIGGER,
                                 USE IT
4540 Y=Y+1                      'SET CONTROL FOR ITEM 2
                                 BIGGER
4550 IF X(X)>=X(Y) THEN RETURN      'ALL IN ORDER SO GO
                                     BACK
4560 SWAP X(X),X(Y)             'SWAP TO PUT IN ORDER
4570 X=Y                        'SWAP WAS MADE, FORCE UP
                                 LIST
4580 GOTO 4500                  'DO COMPARISON AGAIN
4800 PRINT "SORT IS DONE."      'MESSAGE TO STOP TIMER
4810 OPEN "R",#2,NO$,5          'OPEN OUTPUT FILE
4820 FIELD #2, 5 AS A$          'DESCRIBE RECORD
4830 FOR X=1 TO C               'FOR EACH ITEM IN THE FILE
4840 LSET A$=MKI$(X(X))         'GET ITEM READY FOR FILE
4850 PUT #2,X                   'PUT DATA IN OUTPUT FILE
4860 NEXT X                     'CONTINUE TO END OF FILE
4870 CLOSE #1                   'CLOSE INPUT FILE
4880 CLOSE #2                   'CLOSE OUTPUT FILE
4890 END
```

The last sort routine we'll look at is the very popular and highly efficient QUICK SORT. In operation it selects an arbitrary item from the list to be sorted and uses it as a key item. The routine we'll use selects the middle of the list. Having selected a key, it is compared to every other item in the list. Those items smaller than the key are placed below it while those larger than the key are placed above it. This creates two smaller unsorted lists and the key. Each of the smaller lists is then rearranged as the original was, and the process continues until all lists consist of only one item. When this happens, all items are in place.

The most separate lists we can expect to generate is one-third the number of items in the original list. We'll have to keep track of the starting and ending points of the sub-lists within the original list. We do this with arrays for the upper limit (UL) and the lower limit (LL) of these lists. We dimension these arrays to one-third the size of our original list on lines 5100 and 5110. Line 5200 selects our key item to be in the middle of the limits of the list or sub-list we're sorting. A stack pointer (SP) keeps track of where we are in the list.

Enter the following lines of programming, then MERGE SORTHDR.BAS and FILELEN.BAS. Delete lines 2110 and 2120, and SAVE QUIKSORT.BAS. Now you can run the quicksort specifying the files SORTTST1 thru SORTTST6 as input files, and any names you like as output files. You'll be surprised at the improvement in speed over previous sorting methods. On my machine the 100-item test ranged from 15 to 26 seconds and the 1000-item tests ranged from 189 to 337 seconds. This was two to three times faster than the heapsort.

```
2 '**********QUIKSORT.BAS**********
4 '********DAVID E. WARNICK********
6 '*********COPYRIGHT 1983*********
4000 DIM X(C)                    'ARRAY TO HOLD RECORDS
4010 FOR X=1 TO C                'FOR EACH RECORD IN FILE
4020 GET #1,X                    'GET A RECORD
4030 X(X)=CVI(A$)                'CONVERT AND PUT IN ARRAY
4040 NEXT X                      'CONTINUE TO END OF FILE
5000 PRINT "PRESS ANY KEY TO START THE SORT"
5010 X$=INPUT$(1)                'KEYBOARD INPUT TO START
                                  SORT
5100 NL=INT(C/3)                 'CALCULATE NUMBER OF LISTS
5110 DIM LL(NL),UL(NL)           'DIMENSION LOWER AND UPPER
                                  LIMITS
5120 SP=1                        'SET STACK POINTER
5130 LL(1)=1                     'SET LOWER LIMIT OF LIST 1
5140 UL(1)=C                     'SET UPPER LIMIT OF LIST 1
5150 LL1=LL(SP)                  'LOWER LIMIT SMALL ITEM
                                  LIST
5160 UL1=UL(SP)                  'UPPER LIMIT SMALL ITEM
                                  LIST
5170 SP=SP-1                     'DECREMENT STACK POINTER
5180 LL2=LL1                     'LIMITS OF NEW LIST
5190 UL2=UL1                     'LIMITS OF NEW LIST
5200 K=X(INT((LL1+UL1)/2))       'PICK KEY FOR COMPARES
5210 IF X(LL2)>=K GOTO 5240      'MAKE COMPARISON
5220 LL2=LL2+1
5230 GOTO 5210                   'CONTINUE THRU LIST
5240 IF K>=X(UL2) GOTO 5270      'MAKE COMPARISON
5250 UL2=UL2-1
5260 GOTO 5240                   'CONTINUE THRU LIST
5270 IF LL2>UL2 GOTO 5310
5280 SWAP X(LL2),X(UL2)
5290 LL2=LL2+1
5300 UL2=UL2-1
5310 IF LL2<=UL2 GOTO 5210
5320 IF LL2>=UL1 GOTO 5360
5330 SP=SP+1
5340 LL(SP)=LL2                  'SET LIST LIMIT IN ARRAY
5350 UL(SP)=UL1                  'SET UPPER LIMIT IN ARRAY
5360 UL1=UL2
5370 IF LL1<UL1 GOTO 5180
5380 IF SP>0 GOTO 5150
6000 PRINT "SORT IS DONE"        'MESSAGE TO STOP TIMER
6010 OPEN "R",#2,NO$,5           'OPEN RANDOM FILE FOR
                                  OUTPUT
6020 FIELD #2, 5 AS A$           'DESCRIBE THE RECORD
6030 FOR X=1 TO C                'FOR EACH RECORD IN THE
                                  FILE
6040 LSET A$=MKI$(X(X))          'PREPARE INFO FOR FILE
6050 PUT #2,X                    'PUT RECORD IN THE FILE
6060 NEXT X                      'CONTINUE TO END OF FILE
6070 CLOSE #1                    'CLOSE THE INPUT FILE
6080 CLOSE #2                    'CLOSE THE OUTPUT FILE
6090 END
```

The sort routines you now have can easily be adapted for use with any file you choose. We used integers here, so you may have to make the array to be sorted and the sort routine variables string variables for your application. You'll also have to change the record length as required and SWAP all variables associated with the sorted field as we did last month. Additionally, the input file could be used for the output and just rewritten in the desired order. We used a separate file here to preserve our unsorted files for later use with other routines we may wish to test. In a future article, we'll cut down sort time by using key files and show you how to sort on multiple fields, but next month we'll get back to random files.

Play around with these sort routines and make up tests of your own. The real power of your computer is at your fingertips as we continue to handle and manipulate files.

See you next month.

It's Contest Time At The Heath/Zenith Users' Group

Bob Ellerton
HUG Manager

Are you sitting there staring at a blinking cursor wondering what to do with your spare computer time?

Have you created a really neat spreadsheet that you feel could be useful to other members of the Heath/Zenith Users' Group?

Have you created a slick game for yourself or the kids that's the greatest thing since PAC-something?

Are you interested in picking up an extra $1000.00 Gift Certificate for Heath or Zenith Data Systems products absolutely FREE?

If you have answered yes to at least one of these questions, read on!

The Heath/Zenith Users' Group will be sponsoring not one, but two software contests beginning April 1, 1984 and ending July 1, 1984. You may enter both contests if you wish. And, you may enter these contests as many times as you like.

Heath/Zenith Users' Group Spreadsheet Competition

The first of the two contests will be based on currently available spreadsheet programs from Heath/Zenith (e.g. SuperCalc, Multiplan, etc.). Entries to this category should be worksheets that are composed using one of the major spreadsheet programs. Any topic for your worksheet will be accepted (e.g. Tax Calculations, Payroll, General Ledger, Inventory, etc.).

Specific Rules for the Spreadsheet Competition

1. You must include at least two files with each worksheet entered in the contest. The first file should include documentation and instructions on the use of your worksheet as well as a clear description of the results to be expected from the use of your creation. The second file should be the worksheet itself. You may include additional files if you feel examples or further explanations are required to get the most from your entry.

2. Your worksheet must be capable of running on the H8, H/Z-89 or the H/Z-100 series computers. The spreadsheet program used to create your work must be one currently available from Heath Company or Zenith Data Systems (described in the Heathkit Catalog).

3. Entries to the Spreadsheet Competition must be sent to:

Heath/Zenith Users' Group Spreadsheet Competition
Hilltop Road
Saint Joseph, MI 49085

The second contest will concentrate on your ability to use the graphics facilities of your computer to build a game. This competition will be open to all languages currently available from Heath-/Zenith or described in the Heathkit Catalog. Further, you may use languages from other sources providing that the finished software will run without having the user purchase software not found in the Heathkit Catalog.

General Rules:

1. All entries to either the Spreadsheet Competition or the Graphics Game Competition become the property of the Heath/Zenith Users' Group Software Library.

2. Each entry must be accompanied by the Program Submittal and Agreement Form found on page 27 of the January 1984 Issue of REMark. The form must be completed by you.

3. All entries must be submitted on disk and be accompanied by suitable documentation describing the purpose of the entry. Necessary information on setup and operation must be included for the reviewer.

4. Your entry must be clearly marked with the following words: **"Heath/Zenith Users' Group Contest Entry"**

If possible, these words should be included in your documentation file to ensure the proper handling of your contest entry.

Selecting the Winners:

1. The contest for both the Spreadsheet Competition and the Graphics Game Competition will be divided into two parts. During the first round, HUG Staff members will select those programs or worksheets that are thoroughly documented and perform as described by the author. These programs will then be placed on one or more disks containing similar games or worksheets.

2. Authors of programs selected to appear on a HUG Disk will then be informed that their work has been placed in the final competition with other similar programs.

3. AS A BONUS, authors selected for the final competition will receive any piece of Heath/Zenith or HUG software FREE along with a copy of the disk containing their work.

4. Final judging will be provided by the members of the Heath-/Zenith Users' Group via a postcard sent with each of the disks ordered from the HUG Library. The worksheet and graphics game receiving the most votes before November 1, 1984, will be chosen as the Grand Prize Winners in each of the two categories.

Grand Prize Winners

Two winners will be selected by popular vote, one from each category to receive a $1000.00 Gift Certificate from the Heath/Zenith Users' Group which can be used to purchase a variety of products available at any of your local Heathkit Electronics Centers or through Heathkit Mail Order Catalog. The two winners will be announced in the January 1985 issue of REMark.

Specific Rules for the Graphics Game Competition

1. Entries must include at least two files on the disk. One file should be the game itself. The remaining file must contain the necessary start-up instructions and documentation to allow proper operation of your game. Additional files may be included should you feel they are necessary for the end user to get the most from your creation.

2. Your entry must be capable of running on an H8, H/Z-89 or the H/Z-100 series computer using the various graphics modes available to each computer. Each game must be of your design and not a translation from another available computer video game.

3. Entries to the Graphics Game Competition must be sent to:

Heath/Zenith Users' Group Graphics Game Competition
Hilltop Road
Saint Joseph, MI 49085

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.

885-3011-37 ZDOS
ZBASIC Games Disk $20.00

Introduction: This ZDOS disk contains a few graphic games which will bring hours of entertainment to the young and old. Spend a few hours on the job as an Air Traffic Controller, then relax while playing a few games of Blackjack. Practice your typing speed and accuracy or shoot it out with other tanks on the battlefield.

Requirements: These games require the ZDOS operating system on a Z-110 or Z-120 computer. The programs are written in and require the ZBASIC interpreter. The programs are written using the color commands.

Note: You will need to have color memory chips in your Z100 in order to view the games as written. Without the color chips, the games may not be playable.

The following files are included on the HUG P/N 885-3011-37 ZDOS ZBASIC Games Disk:

README	.DOC
BLACKJCK	.BAS
ATC	.BAS
ATC	.TXT
ATC	.DOC
TYPING	.BAS
STOREWD	.BAS
WORD	.DAT
TYPING	.DOC
BATTLE	.BAS
BMENU	.BAS
HISCORES	.DAT

Authors:

BLACKJCK -- John Kappers
ATC -- Del Tapparo
TYPING -- Diana Hsu
BATTLE -- Nathan Vedder

BLACKJCK -- This version of Blackjack uses the full capability of the Z-100 graphics to display the playing cards. A maximum of three players is allowed.

ATC -- This program was inspired from the "Air Traffic Controller" game by David Mannering, distributed by Creative Computing Software. Although similiar in design, the program is the original work of Del Tapparo.

In this real time simulation, you are an air traffic controller responsible for directing the flow of air traffic over a city containing two major airports. You are given a 20 minute shift to direct 6 to 26 aircrafts safely to their destinations. You choose the number of aircrafts according to your ability as a controller.

Air Traffic Controller is a realistic simulation demonstrating the stress involved with the job. The major difference being, in the event of a tragedy, you simply "press return" for another game.

TYPING -- TYPING is a game designed to help you improve your typing speed and accuracy. The computer will randomly choose one word (from WORD.DAT) to descend down the screen. You must type the word as fast as possible before the word reaches the line. If you misspell the word, you receive no points. The score is based on the player's speed of spelling the word correctly.

STOREWD is a program which will allow you to store your own vocabulary of words in the random file WORD.DAT. You can edit or add your own words to the file. The new WORD.DAT file can then be used with TYPING.

BATTLE -- This Battlefield game sets your tank against a host of enemy tanks. You are to move your sites onto the enemy tanks and shoot. The enemy tanks are continually moving, therefore you may have to shoot in front of the tank to destroy it. The tanks also may shoot back, so you must move in hurry.

Note to ZBASIC programmers: This program has some real potential for any of you game writing enthusiasts. The program could be enhanced to include many other features, such as aircraft to fight, more tanks, different tank positions, etc. See what you can come up with.

Comments: This games disk offers a variety of graphic games for the game enthusiasts.
TABLE C Rating: (2),(9)

885-3012-37 Z-DOS
885-5002-37 CP/M-86
HUG Editor $20.00

Introduction: The HUG Editor is a fast command mode character editor originally derived from a public domain CP/M Users' Group editor. It resembles the CP/M ED editor somewhat in operation, but more closely resembles the Intel ISIS-II editor. It is not a "screen"

editor, and uses no function or arrow keys. It is designed mainly for writing source code for assemblers and compilers.

Requirements: The HUG Editor requires the Z-DOS or MS-DOS (for 885-3012-37) or CP/M-86 (for 885-5002-37) operating system, and will run on any computer compatible with those operating systems (it is not machine dependent).

The following files are included on the HUG Editor disks:

885-3012-37

README .DOC
EDIT .COM
SPEDIT .COM
EDIT .DOC
EDIT .ASM

885-5002-37

README .DOC
EDIT .CMD
SPEDIT .CMD
EDIT .DOC
EDIT .A86

Authors:

Z-DOS Version -- Patrick Swayne, from CP/M-86 version
CP/M-86 Version -- Jim Buszkeiwicz, modified by P. Swayne

These versions were translated from the HUG CP/M version, which originated from a CP/M Users' Group program.

EDIT.COM or EDIT.CMD -- This is the HUG Editor program. It is a command mode editor, which means that all text manipulation is done via commands, and none is done directly on the screen. All commands consist of only one letter each, and are easy to memorize. Command iteration is supported with nesting so that complex operations can be carried out with a single command line entry.

In translating this version from the CP/M version, text movement sections were optimized using 8088 string instructions with the result that this version is approximately three times faster when doing multiple search-and-replace commands (both versions running on a Z-100). For such operations, it is one of the fastest editors available.

The HUG Editor automatically creates a back up of the file you are editing. Files can be any size up to the size of one disk, and input and output can be on separate drives.

The HUG Editor supports true backspace, and backspaces correctly through tabs, and even through carriage returns to the previous line while you are inserting text.

SPEDIT.COM or SPEDIT.CMD -- This is a modified version of EDIT that works a little differently from the regular version if you backspace through a carriage return.

EDIT.DOC -- These are the instructions for using EDIT.

EDIT.ASM or EDIT.A86 -- This is the assembly source code for EDIT.

Comments: This editor has been around in some form or other for some time, and is popular with "old timers" in the microcomputer community. It is not as easy to use as some screen editors, but all of the commands are logical and easy to remember. The HUG Editor is an excellent replacement for EDLIN for use with Z-DOS.

Note: This editor is public domain and its use is not restricted by copyright or other legal hindrances.

TABLE C Rating: (10)

HUG Price List

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

Part Number	Description of Product	Selling Price	Volume - Issue
HDOS			
885-1030[-37]	Disk III, Games II	$ 18.00	5-2
885-1096[-37]	MBASIC Action Games	$ 20.00	5-2
885-8026	Space Drop	$ 16.00	5-2
885-8027	HDOS SCICALC	$ 20.00	5-3
CP/M			
885-1234[-37]	CP/M Ham Help	$ 16.00	5-2
885-5001-37	CP/M-86 KEYMAP	$ 20.00	5-4
885-8025-37	CP/M 85/86 FAST EDDY	$ 20.00	5-2
ZDOS			
885-3009-37	ZBASIC Dungeons & Dragons	$ 20.00	5-3
885-3010-37	ZDOS KEYMAP	$ 20.00	5-4
885-8028-37	ZDOS SCICALC	$ 20.00	5-2
MISCELLANEOUS			
885-0004	HUG 3-Ring Binder	$ 5.75	
885-4001	REMark Volume 1, Issues 1-13	$ 20.00	
885-4002	REMark Volume 2, Issues 14-23	$ 20.00	
885-4003	REMark Volume 3, Issues 24-35	$ 20.00	
885-4004	REMark Volume 4, Issues 36-47	$ 20.00	
885-4700	HUG Bulletin Board Handbook	$ 5.00	5-2

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.

Ordering Information

For Visa and MasterCard phone orders; telephone Heath Company Parts Department at (616) 982-3571. Have the part number(s), description, and quantity ready for quick processing. By mail; send order, plus 10% postage and handling, up to a maximum of $3.50 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.

Local HUG Club News

MIHUG
Michiana Heath Users' Group
52578 US 31-33 North
South Bend, IN 46637

This is a new HUG in South Bend that has been meeting since last fall.

219-255-3923
Group Size: 10
Contact Person: Mark L. Meidel
Meet 3rd Monday, 7:00 p.m.

Louisville Heath Users' Group (LHUG)
Contact: Ray Donner
6802 Crossmoor Lane
Louisville, KY 40222
Home: 502-426-9433
Work: 502-585-3727
Meeting Address:
Heathkit Electronic Center
12401 Shelbyville Road
Louisville, KY 40243
502-245-7811
Meetings are held last Sun. of mo. at 8:00 p.m.
With beginning Users' classes held one hour prior to each meeting.
Group Size: Growing!!!

Regarding the **Albuquerque HUG,** we have a new contact person:

Ken Benson
7909 Hendrix NE
Albuquerque, NM 87110
505-294-1658

ARE YOU MOVING?

Don't leave your REMark behind.

Send your change of address in now to:

Heath **U**sers' **G**roup
Hilltop Road
St. Joseph, MI 49085

Clock Watcher's Delight #2

Pat Swayne
HUG Software Engineer

Back in the November 1981 issue of REMark, I wrote an article describing modifications for HDOS and CP/M that caused a clock display to be maintained in the upper right corner of the screen, even while other programs were being run. I have been asked to do the same thing for Z-DOS, and the program in Listing 1 is the result of my efforts. It maintains a 24 hour time display in the upper right corner of the screen that is updated each second. One of the problems with the original HDOS-CP/M screen clock program is that it interfered with programs that used graphics, cursor addressing, etc. To keep this version from interfering, I designed it to write the clock display directly to video RAM. That made it easy to display the clock in any color, and even to use my own font for the numbers.

To use the program, first type in the souce code from Listing 1, and change the color in the line COLOR EQU RED to the color you want for your clock. If you do not have color memory chips installed, you can use white, yellow, or green for your color. If you want the clock numbers to be in the same font as normal numbers, change the line NORNUM EQU 0 to NORNUM EQU 1. With my font, the numbers are a little "heavier" than normal, and will stand out even with a full screen. When you have created a file from Listing 1, you can assemble it by entering:

```
A:MASM B:SCRNCLK,B:SCRNCLK;
A:LINK B:SCRNCLK,A:SCRNCLK;
A:ERASE B:SCRNCLK.OBJ
A:EXE2BIN SCRNCLK.EXE.COM
A:ERASE SCRNCLK.EXE
```

This example assumes that your source file is called SCRNCLK.ASM, and is on drive B:. When you have completed these steps, you will have a file SCRNCLK.COM on drive A:, which you can run by entering

```
A:SCRNCLK
```

The clock display should appear in the upper right corner of your screen. The program is locked into memory and cannot be removed unless you re-boot. To set the clock, just use the TIME command that is built into Z-DOS.

Turning Off the Clock

I have run several different programs with the clock on, and it does not seem to harm any program. However, some programs can cause the clock display itself to "tear" on the screen. Programs that use system function 6, direct console I/O, in a tight loop seem to be the worst offenders. Since no one has been able to come up with a reason for the clock display tearing, I decided to provide a way to turn the clock off and back on after it is loaded. If you assemble the short program in Listing 2 into CLOCK.COM, you can enter

```
A:CLOCK OFF
```

to turn the clock off, and

```
A:CLOCK
```

to turn it back on. The SCRNCLK program stays in memory, but a flag within it is set or reset to enable or disable the clock display.

How It Works

The first part of the SCRNCLK program that is executed when you run it is the set up routine at the end of the program. It alters the timer interrupt vector so that interrupts are processed by SCRNCLK and then sent on to the system after the time is checked and displayed, if necessary. After setting up the interrupt, the program exits via interrupt 27H, which causes all of the program up to the label LEND to remain in memory.

After the program is loaded, it does nothing until a timer interrupt comes along. I chose to intercept the actual timer interrupt vector instead of the software timer vector (see the file DEFIPAGE.ASM on your Z-DOS disk II) because it seemed that there would be slightly less of a drain on processor time. When the interrupt comes, the program tests to see if it is a timer zero interrupt and exits if not. (Timer zero maintains the real time clock.) Then it checks to see if the clock is enabled and exits if not. If the clock is enabled, it calls the BIOS GETDATE routine to get the date (which is ignored) and time. A check is made to see if a second has passed since the screen clock was last updated, and if not the program exits.

If it is time to update the screen clock display, the program converts the hour, minute, and second values to BCD (Binary Coded Decimal) format to separate the tens and ones digits and stores the result in memory. Then it zeros the area of video RAM where the time is to be written to clear out anything that is already there. Finally, the time is written to the screen by using the decoded BCD digits as pointers to a table of bytes to use to form the images of numbers.

Controlling Video RAM

Because the ability to write directly to video RAM is a useful technique for fully exploiting the H/Z-100's capabilities, I will explain it in more detail. Video RAM is controlled by a port (called VRPORT in this program). Of the eight bits in a byte written to the port, the lower four control the way the CRT controller can access the RAM, the higher four control the way the processor (and therefore you) can control it. The highest bit controls your access to the RAM. If it is high, you cannot write to video RAM, and if it is low, you can. The next three bits in descending order control the blue, green, and red video planes. If the bit for a particular plane is low and you write to another plane, the data is written to that plane also. If the bit is high, its plane can only be written to by addressing it directly. If the red and green bits are low and you write to either the red or green plane, you get yellow. (Don't forget that we're adding light, not dyes as you did in grade school art class.)

When you write a byte to video RAM, that byte controls 8 pixels on one scan line on the screen. The most significant bit controls the first pixel (from left to right), the next bit controls the next pixel, and so on. Each byte represents one line of a character position. Since the first line at the top of the screen starts at address 0 in a video plane, the 80

character positions are addressed at 0 through 79 decimal, or 0 through 4F hex. The next line, however, does not start at 50 hex, but at 80 hex, and its 80 positions are addressed at 80 through CF hex. The third line is addressed at 100 through 14F hex, and so on. Lines are addressed at boundaries evenly divisible by 80 hex, so figuring out the address of a particular line is not too difficult.

Each character position on the screen occupies byte-wide segments of 9 lines. The characters themselves are formed within a 5 bit by 7 line matrix, with an 8th line used for lower case descenders. Characters normally start on the second line within the character position, but in SCRNCLK I started characters on the first line, so that the clocks digits appear slightly higher on the screen than other characters on the first character line. I also used a 6 by 7 matrix for my own font, with each vertical line 2 pixels wide, so that the numbers appear a little fuller than normal numbers.

When you are designing your own character font for a program such as SCRNCLK, it is helpful if you define the bytes for each scan line in binary instead of hex. For example, notice that the code for the number two in the first font is

```
DB      70H,88H,8,10H,20H,40H,0F8H
```

In binary, it would be

```
DB      01110000B
DB      10001000B
DB      00001000B
DB      00010000B
DB      00100000B
DB      01000000B
DB      11111000B
```

It is easy to see that the ones in the binary numbers form a figure "2". I originally did the numbers in binary, but converted them to hex so that the listing would not take up too much magazine space. Now that you know how characters are formed, you may want to experiment with your own fonts for the screen clock.

Listing 1

```
        TITLE   SCREEN CLOCK FOR Z-DOS
        PAGE    ,132
;       SCRNCLK — SCREEN CLOCK FOR Z-DOS
;
;       THIS PROGRAM PROVIDES A SCREEN CLOCK FOR Z-DOS THAT FUNCTIONS
;       LIKE THE HDOS AND CP/M SCREEN CLOCK PRESENTED IN REMARK #22,
;       NOV., 1981.
;
;       THE CLOCK IS TIMED BY THE SYSTEM CLOCK.  THE NUMBERS ARE WRITTEN
;       TO THE SCREEN BY WRITING DIRECTLY TO THE VIDEO RAM, AND CAN BE
;       ANY COLOR (DETERMINED BY THE LABEL "COLOR" BELOW).
;
;       BY P. SWAYNE, HUG  27-FEB-84

;       DEFINITIONS

NORNUM  EQU     0                       ;IF NON-ZERO, ASSEMBLE FOR NORMAL NOS.
                                        ;ELSE, ASSEMBLE FOR SPECIAL NUMBERS
WHITE   EQU     0                       ;VRAM COLOR CONTROL BITS
CYAN    EQU     00010000B
MAGENTA EQU     00100000B
BLUE    EQU     00110000B
YELLOW  EQU     01000000B
GREEN   EQU     01010000B
RED     EQU     01100000B

COLOR   EQU     RED                     ;CLOCK COLOR

TIMEINT EQU     108H                    ;Z100 TIMER INTERRUPT VECTOR
TIMSTAT EQU     0FBH                    ;TIMER STATUS PORT
TIMER0  EQU     1                       ;TIMER 0 BIT
VRPORT  EQU     0D8H                    ;VIDEO RAM CONTROL PORT
BRAM    EQU     0C000H                  ;BLUE VIDEO PLANE SEGMENT
RRAM    EQU     0D000H                  ;RED VIDEO PLANE SEGMENT
GRAM    EQU     0E000H                  ;GREEN VIDEO PLANE SEGMENT
        IF      COLOR LE BLUE
VRAM    EQU     BRAM
        ENDIF
        IF      (COLOR GT BLUE) AND (COLOR LE GREEN)
VRAM    EQU     GRAM
        ENDIF
        IF      COLOR EQ RED
VRAM    EQU     RRAM
        ENDIF
BIOSSEG SEGMENT AT 40H
        ORG     24H
GETDATE LABEL   FAR                     ;BIOS GET DATE/TIME FUNC.
BIOSSEG ENDS

CLK     SEGMENT
        ASSUME  CS:CLK,DS:CLK,ES:CLK,SS:CLK
        ORG     100H

START:  JMP     SETUP                   ;SET UP CLOCK INTERRUPT, ETC.

;       CLOCK DATA AREA

CLKFLG  DB      1                       ;CLOCK ON/OFF FLAG
OLDSEC  DB      0                       ;OLD SECOND
OLDIP   DW      0                       ;OLD INSTRUCTION POINTER
```

```
SYSSTK  DW      0                       ;SYSTEM STACK
SYSSTKS DW      0                       ;SYSTEM STACK SEGMENT
VPVAL   DB      0                       ;VIDEO CONTROL PORT VALUE
SCRPTR  DW      0                       ;SCREEN POINTER
HOUR    DW      0                       ;HOUR
        DB      ':'-'0'
MINUTE  DW      0                       ;MINUTE
        DB      ':'-'0'
SECOND  DW      0                       ;SECOND
        DB      0FFH
        IF      NORNUM
NUMTBL  DB      70H,88H,98H,0A8H,0C8H,88H,70H,0 ;BITS FOR ZERO
        DB      20H,60H,20H,20H,20H,20H,70H,0   ;ONE
        DB      70H,88H,8,10H,20H,40H,0F8H,0    ;TWO
        DB      0F8H,10H,20H,10H,8,88H,70H,0    ;THREE
        DB      10H,30H,50H,90H,0F8H,10H,10H,0  ;FOUR
        DB      0F8H,80H,0F8H,8,8,88H,70H,0     ;FIVE
        DB      30H,40H,80H,0F0H,88H,88H,70H,0  ;SIX
        DB      0F8H,8,10H,20H,40H,40H,40H,0    ;SEVEN
        DB      70H,88H,88H,70H,88H,88H,70H,0   ;EIGHT
        DB      70H,88H,88H,78H,8,10H,60H,0     ;NINE
        ELSE
NUMTBL  DB      78H,0CCH,0CCH,0CCH,0CCH,0CCH,78H,0      ;ZERO
        DB      18H,38H,18H,18H,18H,18H,3CH,0           ;ONE
        DB      0F8H,0CH,0CH,78H,0C0H,0C0H,0FCH,0       ;TWO
        DB      0F8H,0CH,0CH,0F8H,0CH,0CH,0F8H,0        ;THREE
        DB      0CCH,0CCH,0CCH,0FCH,0CH,0CH,0CH,0       ;FOUR
        DB      0FCH,0C0H,0C0H,78H,0CH,0CH,0F8H,0       ;FIVE
        DB      78H,0C0H,0C0H,0F8H,0CCH,0CCH,78H,0      ;SIX
        DB      0FCH,0CH,0CH,0CH,0CH,0CH,0CH,0          ;SEVEN
        DB      78H,0CCH,0CCH,78H,0CCH,0CCH,78H,0       ;EIGHT
        DB      78H,0CCH,0CCH,7CH,0CH,0CH,78H,0         ;NINE
        ENDIF
        DB      0,30H,30H,0,30H,30H,0,0         ;COLON
        DB      64 DUP (?)                      ;STACK SPACE
MYSTAK  EQU     $

;       PROCESS CLOCK INTERRUPTS HERE

MYTIME: POP     CS:OLDIP                ;SAVE INSTRUCTION POINTER
        PUSH    AX
        IN      AL,TIMSTAT              ;GET TIMER STATUS
        TEST    AL,TIMERO               ;TIMER ZERO INTERRUPT?
        JZ      TIMEXIT                 ;IF NOT, EXIT
        CMP     CS:CLKFLG,0             ;CLOCK ENABLED?
        JZ      TIMEXIT                 ;IF NOT, EXIT
        MOV     CS:SYSSTK,SP            ;ELSE, SAVE SYSTEM STACK
        MOV     CS:SYSSTKS,SS           ;SAVE SYSTEM STACK SEGMENT
        MOV     AX,CS
        MOV     SS,AX                   ;PUT STACK SEGMENT HERE
        MOV     SP,OFFSET MYSTAK        ;SET LOCAL STACK
        PUSH    CX                      ;SAVE REGISTERS
        PUSH    DX
        CALL    GETDATE                 ;ELSE, GET TIME FROM BIOS
        CMP     DH,CS:OLDSEC            ;HAS SECOND CHANGED?
        JNZ     UPDATE                  ;IF SO, UPDATE CLOCK
PEXIT:  POP     DX                      ;ELSE, RESTORE REGISTERS
        POP     CX
        MOV     SS,CS:SYSSTKS           ;RESTORE SYSTEM STACK
        MOV     SP,CS:SYSSTK
TIMEXIT:POP     AX
        PUSH    CS:OLDIP                ;RESTORE OLD IP
TIMEX   DB      0EAH                    ;FAR JUMP INSTRUCTION
        DB      0,0,0,0                 ;SYSTEM TIMER ADDRES GOES HERE
UPDATE: PUSH    BX                      ;SAVE SOME REGISTERS
        PUSH    SI
        PUSH    DI
        PUSH    DS
        PUSH    ES
        MOV     AX,CS
        MOV     DS,AX                   ;PUT DS HERE
        MOV     OLDSEC,DH               ;UPDATE OLD SECOND

;       A SECOND HAS PASSED, PRINT TIME ON THE SCREEN

PTIME:  MOV     AL,CH                   ;GET HOURS
        CALL    CONBCD                  ;CONVERT TO BCD
        MOV     HOUR,AX                 ;RESULT TO BUFFER
        MOV     AL,CL                   ;GET MINUTES
        CALL    CONBCD                  ;CONVERT TO BCD
        MOV     MINUTE,AX               ;RESULT TO BUFFER
        MOV     AL,DH                   ;GET SECONDS
        CALL    CONBCD                  ;CONVERT TO BCD
        MOV     SECOND,AX               ;RESULT TO BUFFER
        IN      AL,VRPORT               ;READ VIDEO RAM PORT
        MOV     VPVAL,AL                ;SAVE IT
        AND     AL,0FH                  ;STRIP VRAM CONTROL BITS
        OUT     VRPORT,AL               ;ENABLE VIDEO RAM
        MOV     AX,OFFSET VRAM
        MOV     ES,AX                   ;POINT ES AT VIDEO RAM
        XOR     AL,AL                   ;GET A ZERO
        MOV     DI,47H                  ;SET A VRAM POINTER
        MOV     CX,9                    ;SET A COUNTER
        MOV     DL,CL                   ;AND ANOTHER ONE
CLRLP:  PUSH    CX                      ;SAVE COUNTER
        REP     STOSB                   ;CLEAR A LINE OF VRAM
        ADD     DI,80H-9                ;MOVE TO NEXT LINE
        POP     CX                      ;RESTORE COUNTER
        DEC     DL
        JNZ     CLRLP                   ;LOOP UNTIL 9 LINES CLEARED
        IN      AL,VRPORT               ;GET VIDEO PORT VALUE
        OR      AL,COLOR                ;SET CLOCK COLOR
        OUT     VRPORT,AL               ;SET UP COLOR
        MOV     SCRPTR,48H              ;INITIALIZE SCREEN POINTER
        MOV     BX,OFFSET HOUR          ;POINT TO TIME STRING
        MOV     DL,8                    ;SET A COUNTER
        XOR     AH,AH                   ;GET ZERO IN AH
        PUSH    BP                      ;SAVE BP
        MOV     BP,7FH                  ;GET ADDITION CONSTANT
PTMLP:  MOV     AL,[BX]                 ;GET A DIGIT
        INC     BX                      ;INCREMENT POINTER
        MOV     CL,3
        SHL     AL,CL                   ;MULTIPLY RAW NO. BY 8
        MOV     SI,OFFSET NUMTBL        ;POINT TO NUMBER TABLE
        ADD     SI,AX                   ;POINT TO NUMBER DESIRED
        CALL    PDIGIT                  ;PRINT IT
        DEC     DL                      ;DONE?
        JNZ     PTMLP                   ;LOOP UNTIL DONE
PDONE:  MOV     AL,VPVAL                ;GET VIDEO PORT VALUE
        OUT     VRPORT,AL               ;RESET VIDEO CONDITION
        POP     BP
        POP     ES                      ;RESTORE REGISTERS
```

```
        POP     DS
        POP     DI
        POP     SI
        POP     BX
        JMP     PEXIT                   ;AND EXIT

;       CONVERT NUMBER IN AL TO BCD DIGITS IN AL, AH

CONBCD: MOV     AH,0                    ;CLEAR AH
        MOV     BH,10                   ;GET RADIX
        DIV     BH                      ;DIVIDE BY IT
        RET

;       PLACE A DIGIT IN VIDEO RAM

PDIGIT: MOV     CX,7                    ;SET A COUNTER (7 LINES/DIGIT)
        MOV     DI,SCRPTR               ;GET SCREEN POINTER
PDIGLP: LODSB                           ;GET A BYTE
        STOSB                           ;STORE IT IN VRAM
        ADD     DI,BP                   ;UPDATE VRAM POINTER
        LOOP    PDIGLP                  ;LOOP UNTIL DONE
        INC     SCRPTR                  ;UPDATE SCREEN POINTER
        RET
LEND:                                   ;END OF RESIDENT CODE

;       SET UP CLOCK VECTOR AND INITIALIZE TIME STRING
;      THEN EXIT WITH PROGRAM RESIDENT

SETUP:  MOV     AX,CS
        MOV     DS,AX                   ;ENSURE DS IS HERE
        MOV     AX,0
        MOV     ES,AX                   ;ES AT 0
        CLI                             ;TURN OFF INTERRUPTS
        MOV     BX,OFFSET TIMEINT       ;POINT TO TIMER INTERRUPT
        MOV     AX,ES:2[BX]             ;GET SEGMENT ADDR
        CMP     AX,40H                  ;IS CLOCK INSTALLED?
        JNZ     ITSIN                   ;IF SO, EXIT
        MOV     WORD PTR TIMEX+3,AX     ;PUT IT IN OUR VECTOR
        MOV     AX,ES:[BX]              ;GET TIME INT. ADDR
        MOV     WORD PTR TIMEX+1,AX     ;SET UP OUR VECTOR
        MOV     AX,OFFSET MYTIME        ;GET OUR TIMER ADDR
        MOV     ES:[BX],AX              ;PUT IT IN INT. VECTOR
        MOV     AX,CS                   ;USE CODE SEGMENT
        MOV     ES:2[BX],AX             ;FOR HIS VECTOR
        STI
        MOV     AX,CS
        MOV     ES,AX                   ;RESTORE ES
        MOV     DX,OFFSET LEND          ;POINT TO END OF RES. CODE
        INT     27H                     ;EXIT, LEAVE PROGRAM HERE
ITSIN:  STI
        INT     20H                     ;EXIT, NOTHING DONE
CLK     ENDS
        END     START
```

Listing 2

```
        TITLE   SCREEN CLOCK CONTROL PROGRAM
        PAGE    ,132
;       THIS PROGRAM ALLOWS YOU TO TURN OFF THE SCREEN CLOCK
;       CREATED BY THE SCRNCLK PROGRAM.  TO USE IT, ENTER
;
;       A:CLOCK OFF             TO TURN THE CLOCK OFF
;
;       A:CLOCK                 TO TURN IT BACK ON
;
;       BY P. SWAYNE, HUG   15-FEB-84

;       DEFINITIONS

TIMEINT EQU     108H                    ;TIMER INTERRUPT
CMDBUF  EQU     80H                     ;COMMAND LINE BUFFER
CLKFLG  EQU     103H                    ;CLOCK ON/OFF FLAG

CLOCK   SEGMENT
        ASSUME  CS:CLOCK,DS:CLOCK,ES:CLOCK,SS:CLOCK
        ORG     100H

START:  MOV     AX,CS
        MOV     DS,AX                   ;ENSURE DS IS HERE
        MOV     AX,0
        MOV     ES,AX                   ;PUT ES IN INTERRUPT SEGMENT
        MOV     AX,WORD PTR ES:TIMEINT+2        ;GET TIMER SEGMENT
        CMP     AX,40H                  ;IS CLOCK INSTALLED?
        JZ      EXIT                    ;IF NOT, EXIT
        MOV     ES,AX                   ;ELSE, PUT ES IN CLOCK SEGMENT
        MOV     BX,OFFSET CLKFLG        ;POINT TO CLOCK FLAG
        MOV     SI,OFFSET CMDBUF        ;POINT TO COMMAND LINE
        LODSB                           ;GET COUNT BYTE
        OR      AL,AL                   ;ANY ARGUMENT?
        JZ      CLKON                   ;NO, TURN CLOCK ON
SOS:    LODSB                           ;GET NEXT BYTE
        CMP     AL,' '                  ;SPACE?
        JZ      SOS                     ;IF SO, SKIP IT
        CMP     AL,'0'                  ;OFF?
        JNZ     CLKON                   ;NO, TURN CLOCK ON
        MOV     BYTE PTR ES:[BX],0      ;KILL CLOCK
        INT     20H                     ;AND EXIT
CLKON:  MOV     BYTE PTR ES:[BX],1      ;ENABLE CLOCK
EXIT:   INT     20H                     ;AND EXIT
CLOCK   ENDS
        END     START
```

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

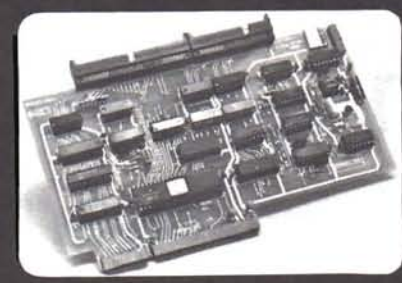
Limitations: MODIFY 89 is not a disk duplicate program. It is currently available for use with an H/Z89 or H/Z90 computer that has an FDC-880H double density 8" and 5.25" controller, using C.D.R.'s BIOS V.2.9 or with an H8 computer using the FDC-H8 by C.D.R. Systems, Inc.

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.

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

DISKSORT
Sort any Disk File

Fast and easy-to-use stand-alone SORT utility. 30+ page manual with many examples. Specify: ZDOS CP/M HDOS. $49.95

E.A.SY
Executive Appointment Secretary

Time management made EASY. Throw away your appointment book and keep your schedules the EASY way. Specify ZDOS CP/M HDOS IBM KAYPRO. $49.95

ZDOS TOOLS
Three Programs for your H/Z-100

UNDERLINE text on your screen
SORT your ZDOS directory
Display serial port status. $24.95

DUALPORT

Connect a 2nd Terminal to your Heath/Zenith computer. Specify ZDOS CP/M HDOS. $39.95

ZBERT

Fast Action Video Arcade Game and Graphics Editor. Requires ZBASIC and full video RAM. $29.95

S-BASIC
Write Structured BASIC Programs

You'll wonder how you ever wrote programs without S-BASIC. Specify ZDOS CP/M HDOS. $49.95

COLOR GAMES

Enjoy "Othello", "Battleship", "Blackjack", "Isolation", and "Yahtzee" in color. Requires ZDOS/ZBASIC and full video RAM. $24.95

ZDUMP

Edit your ZDOS disks in Hex or ASCII using cursor control keys. $29.95

CALL FOR CURRENT PRICING ON 'NEW' Z-150 COMPUTERS.

SUNFLOWER SOFTWARE, INC. (913) 631-1333
13915 Midland Drive, Dept. R4, Shawnee, KS 66216

VISA MasterCard

SUNFLOWER SOFTWARE, INC.
13915 Midland Drive, Dept. R4,
Shawnee, KS 66216 (913) 631-1333

☐ Please, send me a free catalog
☐ H8 ☐ H/Z89 ☐ H/Z100

NAME _____

ADDRESS _____ APT _____

CITY _____ STATE _____ ZIP _____

☐ VISA ACCT # _____

☐ M/C EXP DATE _____

ITEM	FORMAT	PRICE

Kansas 4½% tax _____
Shipping/Handling 2.00
Amount Enclosed _____

The Framework for the International HUG Conference

It's report time. The International HUG Conference is taking shape and it is time to share the plans with you. This article will give you the tentative schedule of events for the weekend of July 27-29, 1984 at the INTERNATIONAL HUG CONFERENCE to be held at Pheasant Run Resort in St. Charles, Illinois. I will also answer a few of the questions that have been asked which may be of interest to a number of you who are planning to come to the Conference.

Since first things should come first, here is the schedule as it stands today.

Friday, July 27, 1984

1:00 P.M.
Registration booth opens for sign-in until 9:00 P.M.

4:00 P.M.
Vendor Exhibit Area opens to Users until 9:00 P.M.

6:00 P.M.
First Session of the Conference talks begin (Speakers to be announced).

8:00 P.M.
Second Session of the Conference talks.

Saturday, July 28, 1984

8:30 A.M.
Breakfast in Governors Hall followed by the Conference General Meeting and Keynote Address.

10:30 - 11:30 A.M.
Vendor Exhibit Area open to Exhibitors only.

11:00 A.M.
Conference talks begin (Speakers to be announced).

11:30 A.M.
Vendor Exhibit Area open to Users until 6:00 P.M.

12:00 - 3:00 P.M.
Buffet Lunch served in the New Orleans Ballroom.

4:30 P.M.
Close of Afternoon Conference Talks.

6:00 P.M.
Close of Vendor Exhibit Area

8:00 P.M.
Fun Time in the Atrium - hors d'oeuvres, prize drawings, etc.

Sunday, July 29, 1984

8:45 A.M.
Conference Talks until 12:00 noon.

9:00 - 10:00 A.M.
Vendor Exhibit Area open to Exhibitors only.

10:00 A.M.
Vendor Exhibit Area open to users until 3:00 P.M.

Margaret Bacon
HUG Secretary

1:00 P.M.
Local HUG Club Gathering.

3:00 P.M
Exhibit Area and remaining activities close.

As in the past, attendance at the Conference is limited by the number of attendees who can be seated for the main meal of the Conference. Governors Hall at Pheasant Run is large enough to accomodate 1200 for a banquet. So this year we will be able to accept 1200 reservations for the Conference.

To date we have 13 definite speakers on a variety of subjects ranging from operating systems to applications including hardware and new technology. We are also looking at a few more possibilities. As we will have access to many more meeting areas this year, there will be space available for special interest groups to get together. Let us know who you are so that we can arrange the space you would like to have.

The Vendor Exhibit area will be about the same size as in 1983 and you will be seeing some new vendors as well as those you are familiar with. They are telling me that you will have some interesting things to see again this year.

Because Pheasant Run is a resort facility, many of you have questions about arrangements for the children you may be bringing. Of course, the first question that comes to mind is - what will it cost to have them in my room? If they are under 12 and do not require an extra bed, their accommodations will be at no cost to you. An extra bed or a child over 12 will cost $10.00 and there is a limit to 4 persons in a double room.

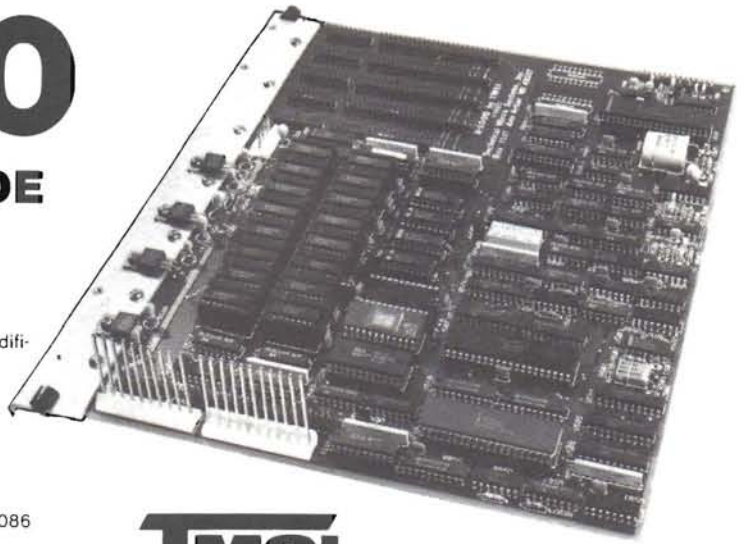
Baby-sitting is another popular subject. The staff at Pheasant Run tells me that they will have local young ladies available for that purpose and the charge is the young lady's regular rate. The staff at Pheasant Run would appreciate knowing about how many children will need that service so that they can make arrangements for a number of sitters and space for group sitting. So let them know when you make your reservations.

Local bus trips can be arranged. One possibility is a visit to Haeger Pottery. Another is a visit to either the St. Charles Shopping Center or Woodfield Mall. Dunham-Hunt House (Dunham Castle) would also fill an afternoon. Check at the Conference Registration Area for more information about these activities.

May I repeat myself for a moment? Pheasant Run really does need to know how and when you will be arriving. When you receive your Conference tickets, you will also receive a reservation form to be mailed to Pheasant Run for room reservations. This form has a space to be filled out with airport arrival information if you are flying. If you are not flying, please indicate in that area of the form. If you are planning to phone Pheasant Run to make reservations, please tell them at that time about your arrival plans. For your convenience, the Pheasant Run phone number is (312) 584-6300.

H-1000

A Z80/8086 UPGRADE FOR THE H89/Z89



HARDWARE

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

SOFTWARE

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

TMSI

Technical Micro Systems Inc.

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

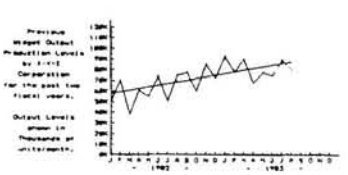
We accept MasterCard and VISA.
Serious Dealer Inquiries Invited

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

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

Powerful Graphics Tools for EPSON Printers!

DINOSAUR



Business Graph Example

• **MXGRAPH** Turn your EPSON printer into a mighty graphics plotter for business or personal needs with MXGRAPH! Plot an unlimited number of business graphs, geometric designs, and artistic sketches with easy one stroke menu driven commands. MXGRAPH available for the H/Z-89 under CP/M-80 or HDOS operating systems. Requires a SERIAL EPSON printer equipped with bit plot graphics. Specify O.S. and hard or soft sectored disk. MXGRAPH is $49.95

• **MXPRINT** Now **LETTER-QUALITY** printing is available on the EPSON printer equipped with bit plot graphics! MXPRINT prints your text and document files at 10 cpi by 6 lpi using letter-quality character font sets. Each character is formed using a user definable 12x24 matrix. MXPRINT includes MXFONT for easy, one stroke creation of your own character and graphic font sets. MXPRINT runs under CP/M on the H/Z-89 and H/Z-100 machines. Specify hard or soft sectored disk. MXPRINT is $19.95

A TRUE WORDSTAR ENHANCER

• **NOVA** The final WordStar enhancer enables full use of ALL your keypad and H/Z89-19 special function keys. The automatic one-time installation implements thirty-six function key enhancements, nearly twice the number of enhancements others provide! Text entry and editing speed is Greatly increased! Function key definitions are labeled on the 25th line. Requires Heath/Zenith WordStar version 2.26. Specify hard or soft sectored disk. NOVA is $24.95

PRINTING THAT ONLY A DIABLO CAN DO

• **LETTERIT** generates poster quality lettering on any DIABLO printer! Prints proportionally about 3 cpi by 2 lpi using a 32x24 matrix. LETTERIT includes DBFONT for generating your own lettering. Runs under CP/M on the H/Z-89 and H/Z-100 machines. Specify hard or soft sectored disk. LETTERIT is $19.95

INTERLACED VIDEO GRAPHICS PACKAGE

• **ILG GRAPHICS** Add Interlaced Graphics to your Z-100 and DOUBLE the graphic resolution of your display! ILG is a resident assembly language graphics package which easily interfaces 640x by 480y color interlaced graphics routines to any MicroSoft compiler (PASCAL, C, FORTRAN, COBOL, ZBASIC), Interpreter (ZBASIC), or assembler (MASM-86). Once loaded, ILG remains resident in system memory for easy interface by any high level language or assembly language module. ILG also includes an interlaced screen-dump utility for EPSON and Diablo printers. Manual includes examples. 64K video ram chips are required for operation in interlaced mode. Runs under Z-DOS on the H/Z-100. ILG GRAPHICS is ONLY $39.95.

Z-DOS COLOR SCREEN EDITOR

• **CSE- Color Screen Editor** CSE is a powerful high-speed, screen-oriented text editor. CSE makes use of the entire H/Z-100 function key row, the cursor positioning keys, and a switchable keypad mode. All functions are performed by quick, single-stroke operations. User settable tabs, text user definable colors, protected status line, block get and put operations, powerful MACRO capabilities, as well as dozens of other editing functions are included in CSE. Use CSE once and you'll never return to your basic editor again. Runs under Z-DOS on the H/Z-100. CSE Color Screen Editor is ONLY $49.95

********** Order Form **********

Quantity	Model	Format	O.S.	Price
_____	MXGRAPH:	___ Soft ___ Hard,	___ CP/M ___ HDOS	$49.95
_____	MXPRINT:	___ Soft ___ Hard		$19.95
_____	NOVA:	___ Soft ___ Hard		$24.95
_____	LETTERIT:	___ Soft ___ Hard		$19.95
_____	ILG Interlaced Graphics Package for Z-DOS			$39.95
_____	CSE Color Screen Editor for Z-DOS			$49.95

Micro Innovations

2455 Sylvania Avenue, Toledo, Ohio 43613
Visa/MasterCard 9 to 5 Eastern time 419-471-1245
Ohio residents add 6% sales tax

Downloading Binary and ASCII Files

A Simple Method of Using a Spare Computer

Alan D. Wilcox
60 South 8th St.
Lewisburg, PA 17837

What can I do with a perfectly good spare computer, but one without disk drives? Use it for experimentation at the workbench? This is an ideal application for my spare Heath H-8, but how to easily run large programs with it? Fortunately, the front panel allows direct entry of a small program which can be used to bootstrap in a larger program.

Here's how ...

Connect one of the H-8's serial ports to a serial port on a host computer, in my case the Heath H-89. Once connected together, the H-89 sends a binary file directly to the H-8. The H-8 has a small program in it to receive the file and load the received data into memory for later execution. The H-8 program also echoes each byte back to the H-89 for verification; any byte echoed incorrectly is indicated on the console in reverse video.

You can see from Figure 1 that the physical connection is quite simple. I already have a long cable between my H-89 and telephone modem, and all I did was disconnect the cable from the modem and hook it directly into the H-8 without changing the H-89 end. The H-89 end is configured DTE (Data Terminal Equipment) and has data-out on the RS-232C pin 2 and expects data-in on pin 3. The H-8 end is DCE (Data Communication Equipment) and receives data on pin 2 and sends on pin 3. None of the other RS-232C lines are used except, of course, the ground. The H-89 port address is internally decoded as 330Q. The H-8 port was decoded at 330Q for convenience only.

The software makes it happen ...

Before we get into the details of the programs, note that the H- 89 program is written to be used under the CP/M operating system and was assembled using ASM. The H-8 program was written using HDOS for one reason only: so that I would have the listing-file printout in Heath split-octal notation for easy front-panel code entry. Although I have Heath's hex EPROM set in the H-8, I find that I can "touch-type" octal much faster than I can hex.

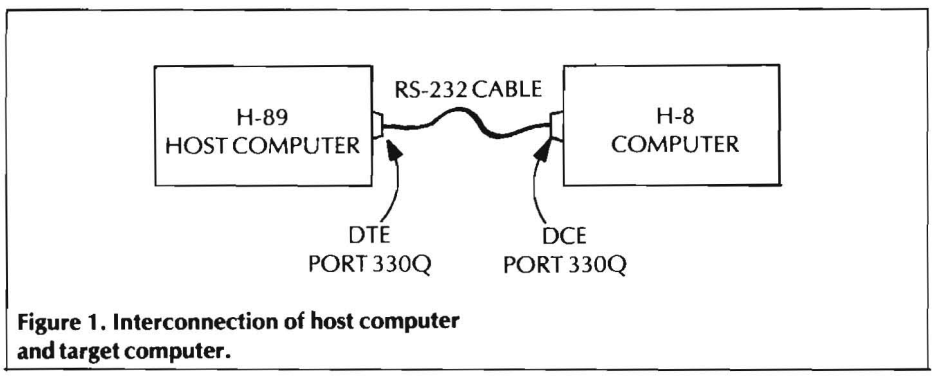


Figure 1. Interconnection of host computer and target computer.

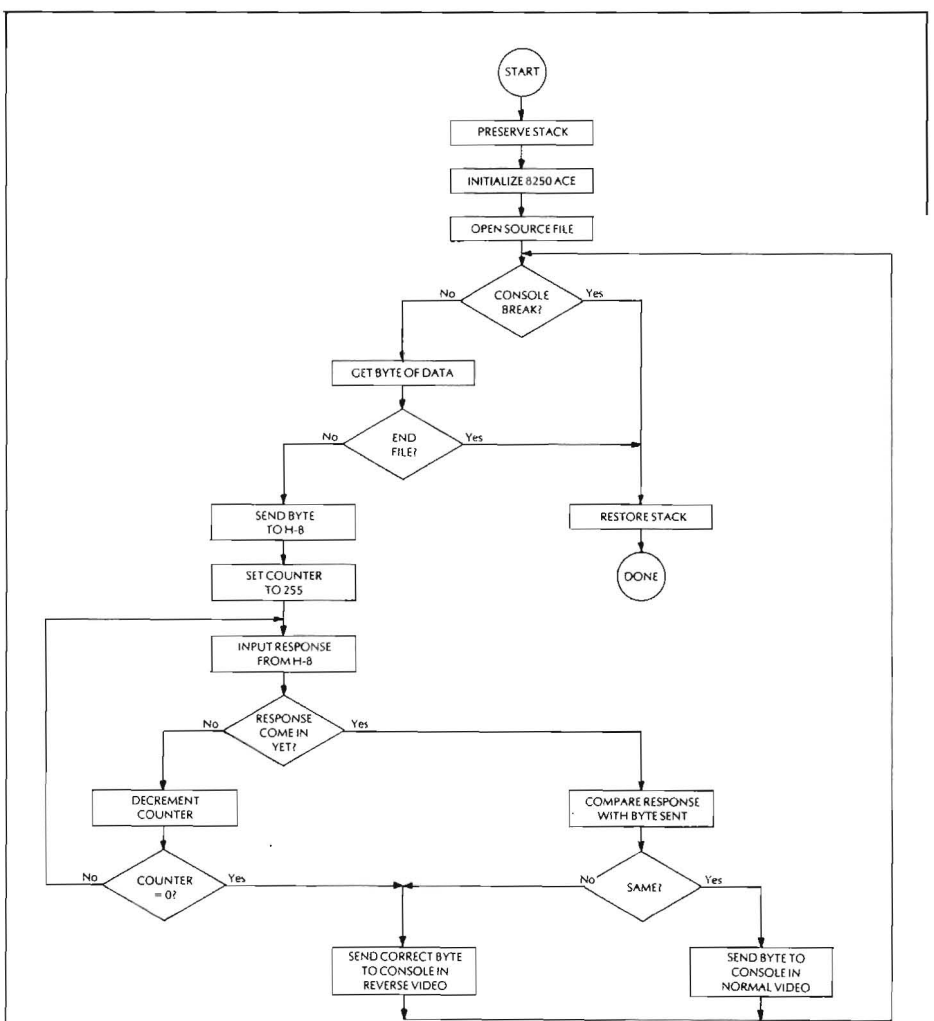


Figure 2. Flowchart for main program shown in Listing 1.

Note also, the purpose of the whole endeavor is to download a large program to the H-8 for execution. This large program should have already been assembled and loaded on disk as a .COM file. When the binary .COM file gets downloaded, it will exist in the H-8 memory in a form ready to be run. If ASCII data were downloaded, then extra code would be necessary in the H-8 to convert it to binary for execution. If, however, the intention is to download ASCII for later transmission to a printer, for example, then these downloading programs will suit perfectly.

Figure 2 shows the flowchart for the H-89 program given in Listing 1. Because the program uses the 128-byte default disk buffer space between 80H and 100H, the stack has to be moved before the disk file is opened. (Recall that the stack runs from 100H downwards.) Next, the H-89's INS8250 ACE (Asychronous Communications Element) gets initialized for 8-bit words. This allows transmission of 8-bit binary data as well as the normal ASCII data if desired. The data speed is set to 9600 baud.

After the file to be downloaded is opened successfully, the main program loop begins. A byte of data is sent to the H-8; the H-89 waits for up to 8.6 mS in a loop counting 255 to 0 until either there is an echo from the H-8, or there is no response. If an echo comes back from the H-8, it is compared with what was just sent; if they match, then the byte just validated is printed on the console screen. If an incorrect echo comes back, or no echo at all, then the byte which should have been received is printed on the console screen in reverse video. This allows an instant check of correct data to the H-8; it also allows identification of bad data for correction where needed by using the front-panel keyboard. After all the bytes in the source file have been sent, the program branches to a short routine which restores the stack and returns control to CP/M.

Listing 1

```
; DOWNLOAD      Program to read a file and send it out to H-8
;
; Program by    Alan D. Wilcox
;               13 Feb  83
;
        ORG     100H

FALSE   EQU     0
TRUE    EQU     NOT FALSE

BDOS    EQU     0005H   ;BDOS entry point

CONSOUT EQU     2       ;Console Output
PRINTST EQU     9       ;Print String Function
GETSTAT EQU     11      ;Console Status
OPENF   EQU     15      ;Open File
READF   EQU     20      ;Read Next Record

FCB     EQU     5CH     ;File Control Block Address
BUFF    EQU     80H     ;Input Disk Buffer Address

DPORT   EQU     330Q    ; Modem data port
BAUD    EQU     12      ; 9600 baud
LINELN  EQU     64      ; Chars per line printed to console

LF      EQU     0AH     ; Line Feed
CR      EQU     0DH     ; Carriage Return
ESC     EQU     1BH     ; Escape
;
;       FILE CONTROL BLOCK DEFINITIONS
FCBDN   EQU     FCB+0   ;DISK NAME
FCBFN   EQU     FCB+1   ;FILE NAME
FCBFT   EQU     FCB+9   ;DISK FILE TYPE (3 CHARACTERS)
FCBRL   EQU     FCB+12  ;Current Extent Number
FCBRC   EQU     FCB+15  ;Extent record count (0 TO 128)
FCBCR   EQU     FCB+32  ;Current (next) record for r/w
FCBLN   EQU     FCB+33  ;FCB LENGTH
;
;       8250 ACE CONTROL AND BIT DEFINITIONS

URRBR   EQU     0               ; RECEIVER BUFFER REGISTER (READ ONLY)
URTHR   EQU     0               ; TRANSMITTER HOLDING REGISTER (WRITE ONLY)
URDLL   EQU     0               ; DIVISOR LATCH (LEAST SIGNIFICANT)
URDLM   EQU     1               ; DIVISOR LATCH (MOST SIGNIFICANT)
URIER   EQU     1               ; INTERRUPT ENABLE REGISTER
UCEDA   EQU     00000001B       ;  ENABLE RECEIVED DATA AVAILABLE INTERRUPT

URLCR   EQU     3               ; LINE CONTROL REGISTER
UC8BW   EQU     00000011B       ;  8 BIT WORDS
UC2SB   EQU     00000100B       ;  2 STOP BITS
UCDLA   EQU     10000000B       ;  DIVISOR LATCH ACCESS (DLAB)

URLSR   EQU     5               ; LINE STATUS REGISTER
UCDR    EQU     00000001B       ;  (Received) DATA READY
UCTHE   EQU     00100000B       ;  TRANSMITTER HOLDING REGISTER EMPTY
;
;
; ***** MAIN PROGRAM *****
; *
; *
STACK:  LXI     H,0             ; Put stack in safe place while
        DAD     SP              ;   using disk buffer space.
        SHLD    OLDSTK          ; Save old stack position
        LXI     SP,NEWSTK       ; Use new stack pointer

        CALL    INIT8250        ; Initialize 8250 ACE

        CALL    SIGNON          ; Greetings, open the input file.

        CALL    MAINLOOP        ; Read, print, send file.

        CALL    QUIT            ; Graceful exit from program.
; *
; *
;
; ***** SUBR INIT8250 *****
;
;       This subr sets up ACE to communicate with distant H-8.
;       Note: 8-bit words required to transmit binary data.
INIT8250:
        MVI     A,0             ; Turn off interrupts
        OUT     DPORT+URIER     ; INTERRUPT ENABLE REGISTER

        MVI     A,UCDLA         ; Set DivLatch Access Bit
        OUT     DPORT+URLCR     ; LINE CONTROL REGISTER

        LXI     H,BAUD          ; Set baud rate
        MOV     A,L
        OUT     DPORT+URDLL     ; DIVISOR LATCH (LS)
        MOV     A,H
        OUT     DPORT+URDLM     ; DIVISOR LATCH (MS)

        MVI     A,UC8BW+UC2SB   ; 8-bit word,2 stop bits,reset DLAB
        OUT     DPORT+URLCR     ; LINE CONTROL REGISTER

        IN      DPORT+URRBR     ; RECEIVER BUFFER REGISTER
                                ;   (Clean it up)
        RET
;
;
; ***** SUBR SIGNON *****
;
SIGNON:         ; Sign on and open the file requested

        LXI     D,GREET
        CALL    PRINTIT

        XRA     A               ; Zero accumulator
        STA     FCBCR           ; Zero file record count
        LXI     D,FCB
        MVI     C,OPENF         ; Open the file
        CALL    BDOS            ; Get 0FFH in Accum if error
        CPI     0FFH
        JNZ     OKOPEN

        LXI     D,OPENERR       ; Had error in opening of file
        CALL    PRINTIT         ; Print error message
        CALL    QUIT            ; Restore stack and get out

OKOPEN: RET
;
;
; ***** SUBROUTINE MAINLOOP *****
;
MAINLOOP:       ; Will remain within this loop until either break
                ;  from console or the file is completely sent.

        MVI     A,LINELN
        STA     CONSCNT         ; Save width of line printed on console

        MVI     A,BUFF          ; Set position of input
        STA     INBUFPT         ;  buffer pointer

LOOP:
```

```
        ; Check console for break. Exit without losing stack
        MVI     C,GETSTAT
        CALL    BDOS            ; Accum LSB=1 if break request
        RRC                     ; Put LSB into carry
        JC      DONE            ; Leave program



READY:  CALL    GETNB           ; Get the next byte.
        JC      DONE            ; Carry set if EOF


        CALL    SENDIT          ; Send binary byte in A reg to H-8

        CALL    CHECHO          ; Check if echo came back OK

        JMP     LOOP

DONE:   RET
;
;
; ***** SUBR GETNB *****
;
;       Gets the next byte from memory or EOF flag if done
;
;       EXIT:   A = Next file byte fetched from memory
;               F = Carry set if End-of-File

GETNB:  LDA     INBUFPT         ; Get current buffer relative ptr
        CPI     BUFF            ; Is it up to end of record yet?
        JNZ     GET             ; Get next byte if not.

;       Read a 128-byte record from disk

        LXI     D,FCB           ; DE gets FCB adr. Reads 128 bytes
        MVI     C,READF         ;  into memory starting at BUFF adr.
        CALL    BDOS


        ORA     A               ; A=0 if record read successfully
        JZ      GET             ; So go get the info if OK.
                                ; Else if A#0, then must be EOF.
        STC                     ; Set carry to indicate EOF.
        JMP     GETEND
;
GET:    ; Read the byte at BUFF + Reg A's relative offset

        MOV     E,A             ; DE to contain relative position of
        MVI     D,0             ;  pointer into disk buffer space.
        INR     A
        STA     INBUFPT         ; Increment and save new pointer.

        LXI     H,BUFF          ; Start address of buffer space
        DAD     D               ;  plus relative offset ---> HL

        MOV     A,M             ; Get byte from memory ptd to by HL

        ORA     A               ; Reset carry bit. Keep data intact.

GETEND: RET
;
;
; ***** SUBR PRHEX *****
;
;       Converts binary value into two ASCII-hex characters
;       and prints on console
;
;       ENTRY:  A = Binary byte from disk buffer
;       EXIT:   A = Binary byte (no change)
;
PRHEX:  PUSH    PSW
        PUSH    PSW
        RRC                     ; Put 4 MSB's into 4 LSB's
        RRC
        RRC
        RRC
        CALL    PRNIB           ; Print hex char equiv to 4 MSB's
        POP     PSW
        CALL    PRNIB           ; Print hex char equiv to 4 LSB's
        POP     PSW

        RET
;
; ***** SUBR PRNIB *****
;
PRNIB:  ; Prints a nibble of Reg A

        ANI     0FH             ; Mask out top 4 bits
        CPI     10              ; Is it number or letter?
        JNC     LETR            ; Must be a letter if jump.
        ADI     '0'             ; Add offset to make number ASCII
        JMP     PRNT

LETR:   ADI     'A' - 10        ; Add offset to make binary value
                                ;  into ASCII letter

PRNT:   CALL    PCHAR           ; Send ASCII to console

        LDA     CONSCNT         ; How many columns left to print?
        SUI     1
        STA     CONSCNT         ; Save new remainder
        JNZ     PREND           ; Not end of line yet.

NEWLN:  MVI     A,LINELN        ; Reset console counter for new line
        STA     CONSCNT

        LXI     D,CRLF          ; Send CR, LF
        CALL    PRINTIT

PREND:  RET
;
;
; ***** SUBR PCHAR *****
;
PCHAR:  ; Prints Reg A to console

        MVI     C,CONSOUT
        MOV     E,A
        CALL    BDOS

        RET
;
;
; ***** SUBR SENDIT *****
;
;       Sends Reg A to output port; keeps Reg A intact
;
SENDIT: PUSH    PSW
        PUSH    PSW
OUTCK:  IN      DPORT+URLSR     ; LINE STATUS REGISTER
        ANI     UCTHE           ; Trans Hold Reg Empty yet?
        JZ      OUTCK           ; Wait for it
        POP     PSW
        OUT     DPORT+URTHR     ; TRANS HOLDING REG
                                ;   Send the char
        POP     PSW
        RET
;
;
; ***** SUBR CHECHO *****
;
;       Checks echo coming back from the H-8. If ok, byte is sent
;       to the console. If not ok, then byte which SHOULD HAVE BEEN
;       echoed is sent to console in reverse video.
;
CHECHO  PUSH    PSW
        MVI     A,0FFH          ; Loop counter. 68 t-states/loop
        STA     COUNTER         ;  for 8.6 mS max with 2 MHz clock.

DATAYET:
        IN DPORT+URLSR          ; Line Status Register
        ANI     UCDR            ; Received-data-bit set yet?
        JNZ     COMPARE         ; Compare data when bit set

        LDA     COUNTER         ; Wait for echo or timeout.
        DCR     A
        STA     COUNTER
        JNZ     DATAYET         ; Fall thru to bad-echo code if timeout.

        POP     PSW             ; Get correct data which had been sent.


BADECHO:
        LXI     D,REV           ; Reverse video
        CALL    PRINTIT
        CALL    PRHEX           ; Print the correct data to console
        LXI     D,NOR
        CALL    PRINTIT         ; Return to normal video

        JMP     CHEND

COMPARE:
        POP     PSW             ; Get correct data
        MOV     B,A             ; Save it in B

        IN      DPORT+URRBR     ; Receive Buffer Register. Get echo.
        CMP     B               ; Is it what was sent out?
        JNZ     BADECHO         ;  Z=0 if compares OK.

        MOV     A,B             ; Recover data
```

```
        CALL    PRHEX           ; Print data to console

CHEND:  RET
;
;
; ***** SUBR PRINTIT *****
;
;       ENTRY:  DE = Start address of string
;
PRINTIT:
        PUSH    PSW
        MVI     C,PRINTST
        CALL    BDOS            ; Print string to console
        POP     PSW
        RET
;
;
; ***** SUBR QUIT *****
;
QUIT:           ; Exit back to the CCP

        LHLD    OLDSTK          ; Recover the old stack pointer
        SPHL                    ; Put it on the stack

        RET                     ; Direct jump back to CCP
;
;
; ***** CONSOLE MESSAGE AREA *****
;
  GREET:    DB      CR,LF,'File Dump to H-8, Version 1.0',CR,LF,LF,'$'
  OPENERR:  DB      CR,LF,'Input file not found or
                                        not specified!',CR,LF,'$'
  REV:      DB      ESC,'p','$'         ; Console into reverse video
  NOR:      DB      ESC,'q','$'         ; Back to normal video

  CRLF:     DB      CR,LF,'$'

;
; ***** VARIABLE AREA *****
;
INBUFPT:  DS    2               ; Input buffer pointer

OLDSTK:   DS    2               ; Stack pointer to get back
                                ;  to the CCP from this pgm.

CONSCNT:  DS    1               ; Console column-position counter.
                                ;  Runs from LINELN to 0.

COUNTER:  DS    1               ; Keep count while waiting for echo.

;
; ***** STACK AREA *****

        DS      64              ; Reserve 32-level stack
NEWSTK:

        END
```

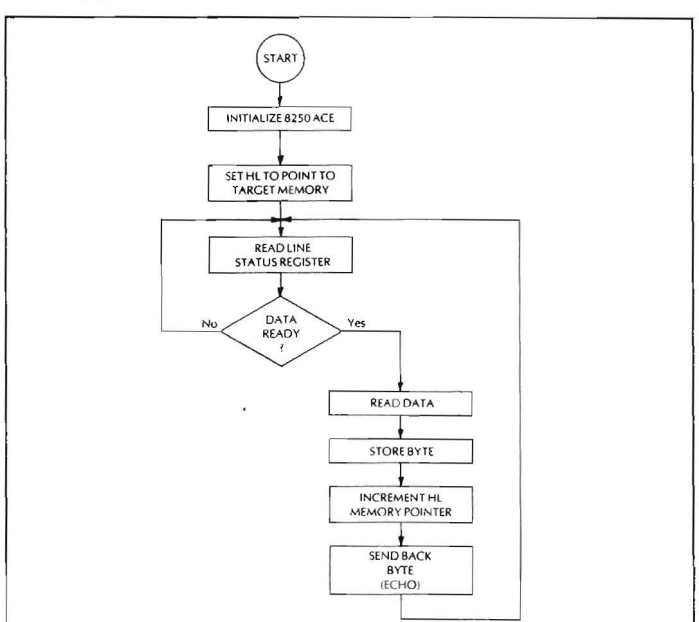


Figure 3. Flowchart for H-8 program shown in Listing 2.

Figure 3 shows the flowchart for the H-8 program printed in Listing 2. The first section of the program sets up the INS8250 the same as the H-89 and sets the memory pointer. The HL register is used as a pointer to the place in memory where each incoming data byte is to be stored. So, before bringing in data, this pointer is initialized to a target address.

The second section of the program is a loop to wait for data from the H-89. The H-8 checks the 8250 line-status register until a byte of data has been received. Once the data has been received, the byte is read and saved in the memory location pointed to by register HL. After incrementing HL, the byte is sent back to the H-89 for verification. Then the program goes back to wait for the next data byte.

How to use the programs ...

First, prepare the cable set to connect both computers. Only pins 2 and 3, plus ground, are used. One end of the cable should go to the H-89 serial port 330Q, the other goes to one of the H-8 serial DCE ports on the H-8-4 card. Set the address on the H-8 card for 330Q.

Next, using the front panel keyboard, enter the H-8 program from the octal code in Listing 2. If you want your large program to be downloaded at a different destination than I used for TARGET, make the necessary changes. Likewise, change the PGM equate if you have less than 64K of memory. Remember when you reassemble to allow at least 80 bytes at the top of memory for use by the Heath front panel system.

After the H-8 code has been entered, set the program counter to the beginning of the executable code (376.000A in Listing 2). Next set the front panel to display the HL register so that you can see the memory pointer change when data comes in. Then press "GO". The program will run until you return to the monitor by pressing "RTM/0".

Using the program in Listing 1, download the code you intend to run on the H-8. Your H-89 downloading program should be ready on disk as an executable file; let's call it DOWNLOAD.COM. Also, the code you want downloaded should be in binary form for H-8 execution; let's call that file BIGCODE.COM. Initiate the transfer by typing in this sequence:

A> DOWNLOAD BIGCODE.COM <cr>

After the greeting message, the download program will print to the console each byte sent to the H-8. Each line will have 64 characters representing 32 memory locations in the H-8. If any bytes are not echoed correctly, they will appear in reverse video. Try it out: send a large program down to the H-8 and disconnect and reconnect the cable while sending data, see what happens.

After the data is all loaded into the H-8, do an RTM/0 and check that the data went where you think it did. Then, assuming that the large program is ready to run, set the program counter of the H-8 to the start of executable code and run your program. The H-8 should act just like it would if you had put in the large program by hand.

Difficulties?

One of the most common difficulties that I've experienced with my H-8 is caused by poor board connections to the motherboard. Unless you have the gold pins in your H-8, when strange things seem to happen (displays go blank, the horn starts, the computer seems dead, etc.), first check the boards. I usually take off the board-top bracket and take out the screws holding each board to the bottom of the H-8. Then I move each board up and down to scrub the connections to the motherboard.

Another thing to watch for: don't try to program in the top 80 bytes of memory where Heath's panel monitor expects its code. Also,

check your monitor source code to see where it ends. For example, the hex EPROM set I have installed in the Z-80 board requires memory up to 041.126A (2156H); when I did my H-8 program, I left a comfortable margin between it and my load target.

Changes ...

You might wish to use my approach for different computers other than my illustration. In principle, you should have little difficulty. The program for the H-89 can be easily changed for other ports and serial devices. The H-8 program might need more work to adapt to other computers.

Whatever small computer you hope to download into must have some sort of monitor program that will allow entry of the bootstrap code. You must be able to initialize its serial communications unit to be compatible with the data format you intend to send; that is, be sure to match baud rate, word length, stop bits, and parity.

And in conclusion ...

With the above programs in operation, you should be able to find some new use for your old H-8. Now that you can easily put some programs into it, perhaps you can make it a valuable tool at your workbench or perhaps even at your main computing center.

Consider a possibility: write the code to allow the H-8 to be a buffer for your printer. Then send high-speed output to the H-8 and let the H-8 feed the printer at the printer's slower rate. Another thought: put a slightly modified version of the code for the H-8 downloading routine into an EPROM in the H-8 to replace the front-panel monitor. This would require modification of the Heath code to free up some EPROM space and you would need to burn a new EPROM with the combined code.

The possibilities could be endless ... the only limit is your imagination and your enthusiasm for computer design and system development. Have fun!

Additional Information

1. The hex EPROM mentioned in the article is an upgrade kit for the Heath Z-80 board and is available directly from Heath Company as Part #H-8-19 for approximately $20. The EPROM set allows the monitor program in the H-8 to display hex as well as split-octal addresses and data. The user can also display the Z-80 alternate register set.

2. The author will provide Listing 1 and Listing 2 source files ready for your assembly for $10 postpaid. Specify either 8" SD or 5 1/4" 10 HS disk; either size disk will be CP/M format.

About the Author:

Alan D. Wilcox *has a Ph.D. in electrical engineering from the University of Virginia and is an Assistant Professor at Bucknell University in Lewisburg, Pennsylvania. He is a licensed Professional Engineer and has been involved with computers since the mid-60's. His current research interests are microprocessor-based instrumentation and speech enhancement. His hobbies include home computing, ham radio, woodworking, photography, and Porsches.*

Listing 2

```
                00001   * LOOPRX         Program to initialize H-8 for
                00002   *                data reception and load into memory
                00003   *
                00004   * Program by:    Alan D. Wilcox
                00005   *                19 Feb 83
                00006   *
043.000         00007   TARGET  EQU     2300H           Beginning adr of downloaded pgm
376.000         00008   PGM     EQU     0FE00H          Begin ass'y of initialization pgm
                00009
000.330         00010   DPORT   EQU     330Q            Port for data entry into H-8
000.014         00011   BAUD    EQU     12D             9600 baud division in 8250 ACE
                00012
                00013   *     8250 UART CONTROL AND BIT DEFINITIONS
                00014
000.000         00015   URDLL   EQU     0               Divisor Latch (Least Significant)
000.001         00016   URDLM   EQU     1               Divisor Latch (Most Significant)
000.001         00017   URIER   EQU     1               INTERRUPT ENABLE REGISTER
                00018
000.003         00019   URLCR   EQU     3             LINE CONTROL REGISTER
000.003         00020   UC8BW   EQU     00000011B      8 BIT WORDS
000.004         00021   UC2SB   EQU     00000100B      2 STOP BITS
000.200         00022   UCDLA   EQU     10000000B      DIVISOR LATCH ACCESS (DLAB)
                00023
000.005         00024   URLSR   EQU     5             LINE STATUS REGISTER
000.001         00025   UCDR    EQU     00000001B      (Received) DATA READY
000.040         00026   UCTHE   EQU     00100000B      TRANSMITTER HOLDING REGISTER EMPTY
                00027
                00028   * ***** MAIN PROGRAM *****
                00029   * *
```

```
                        00030   * *
376.000                 00031   START:  ORG     PGM
                        00032
376.000  076 000       00033           MVI     A,0             Turn off interrupts
376.002  323 331       00034           OUT     DPORT+URIER     INTERRUPT ENABLE REGISTER
                        00035
376.004  076 200       00036           MVI     A,UCDLA         Set Div Latch Access Bit
376.006  323 333       00037           OUT     DPORT+URLCR     LINE CONTROL REGISTER
                        00038
376.010  041 014 000   00039           LXI     H,BAUD          Set baud rate
376.013  175           00040           MOV     A,L
376.014  323 330       00041           OUT     DPORT+URDLL     DIVISOR LATCH (LS)
376.016  174           00042           MOV     A,H
376.017  323 331       00043           OUT     DPORT+URDLM     DIVISOR LATCH (MS)
                        00044
376.021  076 007       00045           MVI     A,UC8BW+UC2SB   8-bit word, 2 stop bits
376.023  323 333       00046           OUT     DPORT+URLCR     LINE CONTROL REGISTER
                        00047
376.025  333 330       00048           IN      DPORT           RECEIVER BUFFER REGISTER
                        00049   *                               (Clean it up)
                        00050   *
376.027  041 000 043   00051           LXI     H,TARGET        HL points to load destination
                        00052   *
                        00053   *
                        00054   *       Wait until data byte comes in. Then save it and bump ptr HL.
                        00055   *       Echo back to sender what we received.
                        00056   *
376.032  333 335       00057   WAIT:   IN      DPORT+URLSR     Wait here until a byte
376.034  346 001       00058           ANI     UCDR             comes in from H-89
376.036  312 032 376   00059           JZ      WAIT
                        00060
376.041  333 330       00061           IN      DPORT           Read the byte
                        00062
376.043  167           00063           MOV     M,A             Put Reg-A into memory
376.044  043           00064           INX     H               Go to next memory location
376.045  365           00065           PUSH    PSW
                        00066
376.046  333 335       00067   ECHO:   IN      DPORT+URLSR     Line Status Register
376.050  346 040       00068           ANI     UCTHE           Trans Hold Reg empty?
376.052  312 046 376   00069           JZ      ECHO            Wait for it.
                        00070
376.055  361           00071           POP     PSW
376.056  323 330       00072           OUT     DPORT           Send the echo back
                        00073
376.060  303 032 376   00074   DONE:   JMP     WAIT
                        00075   *
                        00076   *
376.063  000           00077           END     START


00077 Statements Assembled
```

A CP/M BIOS Modification To Translate Non-Heath Programs

Rick A. Martin
8494 East Jamison Circle North
Englewood, CO 80112

One of the greatest advantages of CP/M is the vast collection of software available from both Heath and non-Heath sources. Local RCPM bulletin boards, for example, allow access to the huge library of the CP/M Users' Group, and that in turn opens up the challenge of trying to adapt programs not written for the H/Z-89. It can sometimes be done when source code is available, but what about .COM files when you don't have the code?

Even though the process quickly became as important as the result, I began this programming project with a useful result in mind. The common element of CP/M had already encouraged me to swap some files with a friend over our modems. Many of his Osborne 1 programs ran beautifully on my H-89 without any modification, but in a few cases the different terminal codes of the two machines made his programs unusable. I'd read that I could customize the CP/M BIOS, but I had never seriously considered it. Now I began to wonder if I could modify the BIOS to "fool" the software into thinking it was communicating with an Osborne 1. After all, the purpose of the BIOS is to interpret the standard CP/M for whatever computer is being used, so it seemed as if I ought to be able to modify the translation.

It turned out that I could. The modification described here is not a universal Osborne/Heath translator, but it does work beautifully with at least one piece of Osborne software. Providing a framework that others can use to solve similar problems, it also gives a step-by-step look at the process of BIOS modification from the perspective of someone who has never done it before.

Looking at the BIOS Listing

The primary function of the BIOS is to provide the communications link between the CP/M software and the I/O devices, a process that involves some translation and modification of data. In terminals without lower case, for example, ASCII characters are translated and "filtered" for the upper-case mapping that's necessary.

My plan, then, was to locate the places in the BIOS where characters are handed back and forth and insert my own little subroutines to intercept and translate any offending Osborne terminal codes. The BIOS listing provided with my CP/M software would be my tool.

What sounded like a simple task turned out, however, to be a long trial-and-error process. Although the CRT physical input and output routines are clearly identified by comments in the listing, it took quite a few tries to find the exact locations to insert my translation routines. It's probably obvious to some of you expert assembly language programmers, but remember this was my first crack at deciphering the BIOS.

Input Translation

On the input side, a routine called CRTIN1 is shown on page #108 of the BIOS listing. In particular, the input character is clearly available in register C in a statement MOV C,M that is noted ";PUT THE CHAR IN C". I chose this point to perform the modification. No input translation is really required because my particular piece of Osborne software doesn't need any special codes from the Osborne 1, but I wanted to use the "arrow" keys on my H-89 as follows:

Osborne Code	Heath Key	Heath Code
Ctl-E	up	ESC A
Ctl-D	right	ESC C
Ctl-X	down	ESC B
Ctl-S	left	ESC D

For this purpose I developed a translation routine called MOD2.ASM to insert in the BIOS after that ";PUT THE CHAR IN C" statement. The routine simply tests the character in register C to see if it is an escape character (27) or the second character of an escape sequence. If it isn't, the routine immediately branches to the end (ELENDO) and proceeds with normal processing. If the character is an escape, however, a flag (ESCEND) is set and the escape character is nulled. Then on the second pass through an escape sequence, the ESCEND flag directs the processing through a translation sequence to create the single- character control codes expected by the Osborne software (see the translation table above).

```
;MOD2.ASM routine to translate H19 keyboard codes to
;Osborne uses ESCEND
        MOV     A,C
        CPI     27
        JNZ     ELDOSO
        MVI     C,0
;if escape code, set indicator and proceed with input
        MVI     A,1
        STA     ESCEND
        JMP     ELENDO
;if not escape code, seo if second character of sequence
ELDOSO: LDA     ESCEND
        CPI     1
        JNZ     ELENDO
;if second character of escape sequence, translate it
        MOV     A,C
        CPI     'D'
        JNZ     IPXLT1
;if it is an ESC D, translate it
        MVI     C,13H
        JMP     ELTRES
IPXLT1: CPI     'B'
        JNZ     IPXLT2
;if it is an ESC B, translate it
        MVI     C,18H
        JMP     ELTRES
IPXLT2: CPI     'A'
        JNZ     IPXLT3
;if it is an ESC A, translate it
        MVI     C,05H
        JMP     ELTRES
```

```
IPXLT3: CPI     'C'
        JNZ     IPXLT4
;if it is an ESC C, translate it
        MVI     C,04H
        JMP     ELTRES
IPXLT4: CPI     'H'
        JNZ     IPXLT5
;if it is an ESC H, translate it
        MVI     C,1EH
        JMP     ELTRES
IPXLT5: MVI     C,0
ELTRES: MOV     M,C
;reset the escape indicator
        MVI     A,0
        STA     ESCEND
ELEND0: NOP
;end of MOD2.ASM routine
```

Before you move on to the output translation routine, locate some memory variable storage for both input and output routines at the beginning of the CRT input section of the BIOS. The following short table (MOD1.ASM) should be inserted just prior to the CRTIN statement on page #108:

```
;MOD1.ASM header for OSBORNE TO HEATH translator
ESCEND: DB      0
DIRCHR: DB      0
ESCINR: DB      0
OPLAST: DB      0
ESCTST: DB      0
OPBYP:  DB      0
OPOTHR: DB      0
;end of MOD1.ASM
```

Output Translation

The CRT output routines begin on page #111 of the BIOS listing, and a likely spot for the output translation routine is in CRTOUT, just before the JMP UO statement. It appears that the BIOS passes the characters to routine UO in register C, so you can just intercept that pass and translate it.

The Osborne 1 uses different screen control codes than the H-19 terminal, so a substantial amount of translation is required on the output side of this modification. Following is the translation table I used:

Osborne Function	Osborne Code	Heath Code
Backspace	08H	ESC D
Line feed	0AH	ESC B
Cursor up	0BH	ESC A
Cursor right	0CH	ESC C
Clear screen	1AH	ESC E
Home cursor	1EH	ESC H
Half intensity	ESC)	ESC p
End half intensity	ESC (	ESC q
Insert line	ESC E	ESC L
Start graphics	ESC g	ESC F
End graphics	ESC G	same
Insert character	ESC Q	ESC 40H
Delete line	ESC R	ESC M
Delete end of line	ESC T	ESC K
Delete character	ESC W	ESC N
Direct cursor address	ESC =++	ESC Y++

Of course, this output translation table introduces the problem of single-character codes from the Osborne that must be translated to two-character codes for the H-19, as well as the additional complexity of handling the four-character direct cursor addressing codes. This requires the ability to insert characters that have not been sent by the software. In the BIOS (page #119) I located a subroutine called PMSG that will transmit a string of characters defined by DB statements in the source code. BIOS messages like the sign-on are printed by this subroutine, and I have used it in this translation program.

First the translation routine tests each character to determine if it is an escape character or part of an escape sequence. If not, the routine is bypassed and the character is processed normally. If it detects the escape character (27), however, the routine nulls that character and sets a flag causing the next character to be stored in ESCINR, then nulled before being processed. Thus, all escape codes result in two consecutive nulls being transmitted to the terminal and the identifying second character being stored in ESCINR. This storing and nulling process is necessary in order to deal with the codes of different lengths.

After processing the second character and before returning to the software, the routine branches to the PMSG subroutine, where a string of characters representing the appropriate translated escape sequence (as identified by the content of ESCINR) are transmitted. Since the PMSG subroutine uses CRTOUT, a bypass flag (OPBYP) allows temporary elimination of the translation routine while this process is underway.

Finally, if the escape sequence is a direct cursor address code, DIRCHR flags the next two characters to be transmitted normally. Otherwise, OPOTHR flags these two following characters as belonging to an unknown four-character escape sequence and therefore nulls them.

The translation routines null, rather than pass, all escape sequences other than those expressly included in order to prevent unexpected codes from creating problems. This is because I spent several months searching in vain for the cause of a blank and an upper-case C following every letter entry before I realized that the Osborne software was sending some four- character escape code after every input letter.... and I was seeing the last two letters on my screen. Apparently, Osborne uses some ANSI codes such as ESC [p C to control cursor position. The "catchall" filtering should eliminate any future surprises.

Granted, there's a risk in this approach of inadvertently nulling the two characters following an unexpected two-character escape sequence, but I hope I've identified the ones you're likely to see. If not, a modification will be easy enough to make.

The following routine should be inserted just before JMP UO in CRTOUT on page #111. After insertion, delete the JMP UO statement (note that it is replaced by the last statement in the routine).

```
;MOD3.ASM routine to convert Osborne terminal
;codes to H19 codes for output
;uses OPLAST,DIRCHR,ESCTST,ESCINR,OPBYP,OPOTHR
;see if we're in the PMSG loop.
        LDA     OPBYP
        CPI     1
;if OPBYP is set, skip around to OPXLT7
        JZ      OPXLT7
;save registers
        PUSH    A
        PUSH    D
        PUSH    H
;get the character in C
        MOV     A,C
;is it an ESC?
        CPI     27
        JNZ     OPXLT1
;set escape sequence indicator
        STA     ESCTST
;null it and return
        MVI     C,0
        JMP     EXXIT1
;check for second character
```

```
OPXLT1: LDA     ESCTST
        CPI     27
;if not escape sequence, go on
        JNZ     OPXLT2
;reset ESCTST flag
        MVI     A,0
        STA     ESCTST
;go convert
OPXL20: MOV     A,C
        JMP     OPXLT3
;is it a single character code?
OPXLT2: MOV     A,C
        CPI     0BH
        JZ      OPXLT3
        CPI     0CH
        JZ      OPXLT3
        CPI     1AH
        JZ      OPXLT3
        CPI     1EH
        JZ      OPXLT3
;if not a single char code, check to see
;if we're in dir char or other code
        LDA     DIRCHR
        CPI     0
        JNZ     OPXL21
        LDA     OPOTHR
        CPI     0
        JNZ     OPXL99
        JMP     EXXIT1
;count off one character
OPXL21: DCR     A
        STA     DIRCHR
;pass the character as-is
        JMP     EXXIT1
;count off one character
OPXL99: DCR     A
        STA     OPOTHR
;null the 3rd and 4th characters of the unknown
;control code
        MVI     C,0
        JMP     EXXIT1
;save the character for the next time around
OPXLT3: STA     ESCINR
;set last character indicator
        MVI     A,1
        STA     OPLAST
;null the character and return
        MVI     C,0
        JMP     EXXIT1
;return the registers
EXXIT1: NOP
        POP     H
        POP     D
        POP     A
        CALL    U0
;redirect to OPXLT4
        PUSH    A
        PUSH    D
        PUSH    H
;is this the last character of an escape sequence?
        LDA     OPLAST
        CPI     1
        JNZ     OPXLT5
;if so, what is it?
        LDA     ESCINR
;send to the right PMSG code
        CPI     1AH
        JZ      OPXL1A
        CPI     1EH
        JZ      OPXL1E
        CPI     0DH
        JZ      OPXL0D
        CPI     0CH
        JZ      OPXL0C
        CPI     0BH
        JZ      OPXL0B
        CPI     0AH
        JZ      OPXL0A
        CPI     08H
        JZ      OPXL08
        CPI     ')'
        JZ      OPXLRP
        CPI     '('
        JZ      OPXLLP
        CPI     'E'
        JZ      OPXLE
        CPI     'g'
        JZ      OPXLLG
        CPI     'G'
        JZ      OPXLG
        CPI     'Q'
        JZ      OPXLQ
        CPI     'R'
        JZ      OPXLR
        CPI     'T'
        JZ      OPXLT
        CPI     'W'
        JZ      OPXLW
        CPI     '='
        JZ      OPXEQ
        JMP     OPXNUL
;Escape codes
OPXL1A: LXI     H,OPESCE
        JMP     OPXLT8
OPXL1E: LXI     H,OPESCH
        JMP     OPXLT8
OPXL0D: LXI     H,OPEOD
        JMP     OPXLT8
OPXL0C: LXI     H,OPNULL
        JMP     OPXLT8
OPXL0B: LXI     H,OPESCA
        JMP     OPXLT8
OPXL0A: LXI     H,OPESCB
        JMP     OPXLT8
OPXL08: LXI     H,OPESCD
        JMP     OPXLT8
OPXLRP: LXI     H,OPELP
        JMP     OPXLT8
OPXLLP: LXI     H,OPELQ
        JMP     OPXLT8
OPXLE:  LXI     H,OPESCL
        JMP     OPXLT8
OPXLLG: LXI     H,OPESCF
        JMP     OPXLT8
OPXLG:  LXI     H,OPESCG
        JMP     OPXLT8
OPXLQ:  LXI     H,OPAMP
        JMP     OPXLT8
OPXLR:  LXI     H,OPESCM
        JMP     OPXLT8
OPXLT:  LXI     H,OPESCK
        JMP     OPXLT8
OPXLW:  LXI     H,OPESCN
        JMP     OPXLT8
OPXEQ:  LXI     H,OPESCY
;there are two more characters coming
        MVI     A,2
        STA     DIRCHR
        JMP     OPXLT8
OPXNUL: LXI     H,OPNULL
;assume there are two more characters coming
;in unknown code
        MVI     A,2
        STA     OPOTHR
        JMP     OPXLT8
OPXLT8: NOP
        MVI     A,1
        STA     OPBYP
        CALL    PMSG
        MVI     A,0
        STA     OPBYP
        JMP     OPXLT5
;clean up and prepare to return
OPXLT5: MVI     A,0
        STA     OPLAST
        STA     ESCINR
        POP     H
        POP     D
```

```
        POP     A
        RET
OPXLT7: JMP     U0
;end of MOD3.ASM routine
```

The character strings used by the PMSG subroutine should be inserted in the BIOS with the other BIOS messages on page #120 of the listing. I inserted the following statements (MOD4.ASM) immediately after the statement labeled ERRMSG:

```
;start of MOD4.ASM routine
;character strings for PMSG
OPESCE: DB      27,'E',0
OPESCD: DB      27,'D',0
OPESCB: DB      27,'B',0
OPESCA: DB      27,'A',0
OPESCC: DB      27,'C',0
OPESCH: DB      27,'H',0
OPEOD:  DB      0DH,0
OPELP:  DB      27,'p',0
OPELQ:  DB      27,'q',0
OPESCL: DB      27,'L',0
OPESCF: DB      27,'F',0
OPESCG: DB      27,'G',0
OPAMP:  DB      27,40H,0
OPESCM: DB      27,'M',0
OPESCK: DB      27,'K',0
OPESCN: DB      27,'N',0
OPESCY: DB      27,'Y',0
OPNULL: DB      0,0
;end of MOD4.ASM routine
```

Finally, in order to identify your modified BIOS clearly upon boot, you should include a distinctive sign-on revision between the ENDIF and DB CR,LF,0 statements at the bottom of page #128 of the BIOS listing. Mine is as follows:

```
DB      CR,LF,LF
DB      27,')'
DB      27,'g'
DB      'faaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac'
DB      CR,LF
DB      '`',27,'G'
DB      ' Modified for use ONLY with Osborne 1 software '
DB      27,'g','`'
DB      CR,LF
DB      '`',27,'G'
DB      ' by Rick A. Martin                              '
DB      27,'g','`'
DB      CR,LF
DB      '`',27,'G'
DB      ' Ver 1.1  February 2, 1984                      '
DB      27,'g','`'
DB      CR,LF
DB      'eaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaad'
DB      27,'G'
DB      27,'('
```

Note that I've used some Osborne terminal control codes in the sign-on-- ESC (, ESC), ESC G and ESC g --since the modified BIOS will now translate these.

Use your word processor to incorporate the five MOD files described above into the complete BIOS.ASM listing provided with your CP/M software (see Distribution Disk #3). The BIOS source code is quite extensive, and this editing can be time consuming if done directly in the BIOS.ASM file. If your word processor has an "include file" function (like Magic Wand), I heartily suggest its use. Name this modified file BIOS.ASM.

Preparing the Disks

Prepare three disks for use in the modification process:

DISK #1 should contain the CP/M system as well as the following files:

SUBMIT.COM
XSUB.COM
OSBIOS.SUB (as follows):

```
; OSBIOS.SUB 02/02/84
; USE DISK #1 FOR A:
; USE DISK #2 FOR B:
; USE DISK #3 FOR C:
C:PIP A:=C:ASM.COM
B:MAKEBIOS B:1 B:OSBIOS.PRE
A:ASM BIOS.BAZ
REN A:OSBIOS.HX0=BIOS.HEX
B:MAKEBIOS B:2 B:OSBIOS.PRE
A:ASM BIOS.BAZ
REN A:OSBIOS.HX1=BIOS.HEX
B:PREL A:OSBIOS B:OSBIOS
B:MAKEBIOS B:3 B:OSBIOS.PRE
; SUBMIT ACTION COMPLETED.
```

DISK #2 should contain the following files (no system):

BIOS.ASM (our new revised BIOS)
PREL.COM
MAKEBIOS.COM

DISK #3 should be a duplicate of CP/M Distribution Disk #1, except for ASSIGN.COM, which should be erased.

Performing the Modification

1. Perform a cold boot with Disk #1 in Drive A: and Disk #2 in Drive B:.

2. Type SUBMIT OSBIOS <return>.

3. Follow the instructions that appear on the screen, and be patient. The assembly process is quite lengthy. There'll be plenty of time to go fix a snack while your disk drives clatter away.

4. When SUBMIT action is complete, perform a cold boot with Disk #3 in Drive A: and Disk #2 in Drive B:.

5. Type PIP A:BIOS.SYS=B:OSBIOS.PRE[RW] <return>.

6. Type MOVCPM17 * BIOS.SYS <return>.

7. Type SYSGEN <return>.

8. When SYSGEN asks for source drive, enter <return>.

9. When SYSGEN asks for destination drive, enter A, then <return>.

10. Reset computer <shift/reset>.

11. Perform a cold boot and answer CONFIGUR questions.

12. <shift/reset>, then cold boot again and check for proper graphics in sign-on message.

After the above steps are performed, you should have a properly sized CP/M system on Disk #3, with the modified BIOS. Use SYSGEN again to move this system and BIOS to a disk containing the Osborne program, then run it! Remember to keep this BIOS separate from your "normal" software, though, because the Heathkit H-19 terminal codes will no longer work with it.

As I stated earlier, this is far from a universal Osborne/Heath translator. For example, I have provided for translation of the start graphics and stop graphics codes, but I haven't attempted to translate the graphics characters between the two machines. The character sets are not the same, but there may be enough similarity to get a reasonable representation on the screen. If there isn't, I'll leave that little enhancement to fellow HUGgies. In the meantime, however, this BIOS does deal with the majority of common terminal codes for you.

Good luck!

ZBASIC
Machine Language Subroutines

Rex Klopfenstein, Jr.
400 Napoleon Rd., Apt. 332
Bowling Green, OH 43402

This procedure for the installation of machine language subroutines is very similar to the method described in Appendix C of the IBM PC BASIC manual. As with the IBM PC BASIC, there are two areas in memory which can be set aside for machine language subroutines. The first available memory area resides in the same 64K segment into which ZBASIC is loaded. To set aside memory with the 64K ZBASIC segment, the ZBASIC CLEAR statement or the /M switch on ZBASIC load and run command is used to set bottom limit of available memory for ZBASIC (see Appendix E of ZBASIC manual for complete description). As there are problems in determining where ZBASIC ends, this method will not be considered here. The second area in which machine language subroutines can be stored is outside the ZBASIC's 64K segment. This includes almost any block of memory which is not used by the system during execution of applications program.

After deciding where in memory to load subroutines, a method must be determined to load machine code into memory and then onto disk. One way (which is shown in Appendix E of the ZBASIC manual) is the use of POKE statements to insert machine code subroutines from binary files into memory. A second method will be discussed which allows the machine code routine to be loaded from a disk file with the ZBASIC BLOAD statement.

If more than one subroutine is required, it is recommended that a jump table be constructed which vectors (directs) calls to the proper individual subroutine. Also, all returns to calling ZBASIC program are vectored through this jump table. There are several advantages in using jump tables. The first being the ease of calculating offsets into the various subroutine entry points which are required for ZBASIC CALL statement. The second, and most important advantage is that the main subroutine body (the actual code which executes the function of the subroutine) can be lengthened or shortened without affecting the calculated offset into subroutine package. The reason for the stability of the offset values is that the jump table does not change length or location, only the actual subroutine code which is "pointed to" by the jump table.

An example of jump table construction is shown below for an application program which requires a three machine code subroutine called MASM, FARM, and RARM. These lines should be the first lines of the assembly code in the subroutine package.

```
SUB_PAK   SEGMENT   PARA PUBLIC
          ASSUME    CS:SUB_PAK,DS:SUB_PAK,ES:NOTHING
;
MARM      PROC      FAR             ; MOVE ARM ENTRY
                                    ;  POINT
          JMP       MOVE            ; GO TO ACTUAL
                                    ;  MOVE CODE
MARM_E    RET       6               ; EXIT & CLEAN
                                    ;  STACK
;
FARM      PROC      FAR             ; FIND ARM ENTRY
                                    ;  POINT
          JMP       FIND            ; GO TO ACTUAL
                                    ;  FIND CODE
FARM_E    RET       6               ; EXIT & CLEAN
                                    ;  STACK
;
RARM      PROC      FAR             ; ROTATE ARM
                                    ;  ENTRY POINT
          JMP       ROTATE          ; GO TO ACTUAL
                                    ;  ROTATE
RARM_E    RET       2               ; EXIT & CLEAN
                                    ;  STACK

          (body of subroutines)
```

It should be noted that there is a subroutine body for each entry in the jump table. The labels on the JMP instructions (MOVE, FIND, and ROTATE) are associated with the appropriate subroutine body code. Also, each subroutine body should contain a JMP instruction as the last instruction which would have the label of the appropriate return vector (MARM_E, FARM_E, or RARM_E). Note the numbers (6, 6, 2) associated with the RET in assembly code. These are required to clean passed parameters from stack on return. In addition to the above rule, all other rules listed in Appendix E of the ZBASIC manual regarding subroutine parameters must be followed.

The main ZBASIC program would have the following code to establish addressability to the machine language subroutine package:

```
100 DEF SEG = &Hxxxx            'Seg addr where
110                             ' subroutines are located
120 BLOAD "SUBPACK.EXE",0       'Load subroutine package
130                             ' into memory
140 MOVE_ARM = 0                'Offset to MARM vector
150 FIND_ARM = 6                'Offset to FARM vector
160 ROTATE_ARM = 12             'Offset to RARM vector
          .
          .
          .
200 CALL MOVE_ARM (X_axis,Y_axis,Z_axis)
          .
          .
          .
300 CALL FIND_ARM (X_loc,Y_loc,Z_loc)
          .
          .
          .
400 CALL ROTATE_ARM (Angle)
```

Now that you have your subroutine package written and assembled; an .OBJ file must be generated and loaded into memory in the proper format for ZBASIC BLOAD command. The easiest way to do this is to use the Z-DOS assembler (MASM) to generate an .OBJ file and then invoke the LINK program with the /HIGH switch (see LINK com-

mand in Z-DOS manual). Also, generate a MAP file when requested by linker so the starting address and length of the subroutine package can be determined when linked into memory.

```
A:MASM
{respond to file name requests}

A:LINK/HIGH
{respond to file name requests}
   {include .MAP request}
```

This operation will create an executable file which will load as high as possible in the memory that exists on the system. You may get an error message about a missing stack segment, this is O.K. Also, print the .MAP file for future reference. The binary file generated by the linker will have .EXE name extension.

The next step requires loading ZBASIC under the Z-DOS DEBUG program. When the ZBASIC is loaded and DEBUG responds with the prompt, display registers with R command. Record the values stored in the CS, IP, SS, SP, DS, and ES registers.

```
A:DEBUG ZBASIC.COM
>R
```

Now use the DEBUG program to load the subroutine file which is linked into high memory.

```
>N SUBPACK.EXE
>L
```

Again use the R command to display register contents to determine where in memory the subroutine package was loaded. The values stored in the CS (segment address) and IP (offset into segment) registers provide this information. Record the CS and IP register contents for future reference.

Now use the DEBUG R command, reset the registers back to the original values which were recorded when ZBASIC was originally loaded.

Start ZBASIC by using DEBUG's G command, then load the ZBASIC applications program using ZBASIC LOAD command. When the program is loaded, edit the DEF SEG to match the value of the CS register when the subroutine package was loaded via LINK. We have already taken care of the variable value (offset) of CALL statements, the IP register should contain the value of 0 when the subroutine file was loaded into memory.

Using the direct mode, execute a DEF SEG command with the value of the subroutine starting address (value in CS register when subroutine package was load in above operation) as argument. Now use the BSAVE command in the direct mode in ZBASIC to create a file that can be used by BLOAD. Use the value returned in the LINK .MAP file for code length.

```
BSAVE "SUBPACK.SUB",0,<length from link map>
```

Edit your applications ZBASIC program to contain a BLOAD command after the DEF SEG which defines the proper segment. Make sure to use the ZBASIC SAVE command to save the modified ZBASIC applications program before exiting to DEBUG monitor.

Enter SYSTEM command and control will return to DEBUG monitor. Exit DEBUG with Q command which will then return control to Z-DOS, you have then completed the ZBASIC subroutine process.

A few lines of code in the ZBASIC applications might be included to allow the user to select the drive on which the subroutine file is located. When the drive is selected, it can then be concatenated with the file name in the BLOAD statement.

```
10 INPUT "On what drive is subroutine package located
       (A or B)";Dr$
20 BLOAD Dr$+":SUBPAC.SUB",0
```

Another suggestion is to write the subroutines in relocatable code, they then can be loaded into memory areas other than where linker originally placed them. All that is necessary is that the DEF SEG, BLOAD, etc. parameters are modified to the segment which will contain the subroutine when loaded.

References

IBM Personal Computer Hardware Reference Library - BASIC, Second Edition (May 1982), Version 1.10, Copyright IBM, 1981.

MICROSOFT Z-BASIC (Z-DOS) Volume I & II, Copyright Microsoft, 1979 and Zenith Data Systems, 1982.

Z-DOS Volume I & II, Copyright Microsoft, 1982 and Zenith Data Systems, 1982.

Caution for Users:

Recently, it has come to our attention that a foreign local users' group is claiming "official" affiliation with the Heath Users' Group as HUG's representative to all members. This group is collecting a membership fee in return for a subscription to a newsletter and membership represented as the official membership to the Heath/Zenith Users' Group. Although we do encourage user activities at the local level, HUG has no official affiliation with local organizations.

Put Some Style In Your Epson

J. F. Smith
Ford Aerospace
2101 Blair Mill Road
Willow Grove, PA 19090

About the Author:

***John F. Smith** is Vice President of Ford Aerospace & Communications International, Inc., currently residing in Cairo, Egypt. An old timer in electronics, with a BSEE degree from San Jose State College (CA) in 1951 and an active amateur radio operator (W3JF) since 1946, John is a newcomer to the world of computers. A one-year-old H-89 at home, and a brand new Z-100 (ZW-110-32) in the office, are providing a rapid and fascinating introduction to the technology. After 35+ years of exposure to "hard wiring", learning to exploit the power of software is an experience with equal parts of delight (when something works) and frustration (when it crashes).*

The Epson RX-80, at this writing, is a relatively unknown addition to the Epson product line, having been overshadowed by the slightly earlier introductions of Epson's FX series of line printers. Basically, it's an upgraded MX-80 - faster, with full graphics capability and additional print fonts. It sells for about the same price as the MX-80, and represents an excellent value.

As the new owner of an RX-80, and anxious to exploit its wide array of software-driven print control capabilities, I knew there had to be a better way than using Microsoft BASIC commands. I was particularly anxious to be able to directly control the format of hard copy output from frequently used application programs, like SuperCalc, without the inconvenience of exiting the program to run an MBASIC routine or, worse yet, fool around with dip switches.

Pat Swayne's excellent REMark series, "Getting Started With Assembly Language", provided both the inspiration (and just enough) knowledge for a first attempt at writing a CP/M assembly language program. Thus was "STYLE80.COM" born, which later begat "STYLE25.COM" when the original program was modified to provide similar features with the Z-100/Z-125 combination in my office running under CP/M-85.

Following Pat's advice, STYLE80.ASM (Listing 1) is heavily annotated, in the the hope that even a neophyte like myself will have little difficulty in following the program in enough detail to adapt it to another type of printer.

The program makes maximum use of "built-in" CP/M functions, another bit of advice from the "Getting Started With Assembly Language" series. Since it returns to CP/M without a warm boot, it can be used to supplement the WordStar print control commands by simply using the "R" command.

As a "first effort", the program undoubtedly lacks elegance. It has, however, one overriding virtue - it works! The resulting executable CP/M .COM file uses only 1K of disk space (2K on a double density soft-sectored disk), certainly small enough to tuck it on any disk from which you may wish to control the print style.

The original program was revised after acquisition of a new Z-100 computer in my office to provide software control of a Z-125 printer. STYLE25.ASM is shown in Listing 2. The features and print control codes are different, but the approach is identical. Now, I have to decide whether or not to wait for Pat's series to cover Z-DOS assembly programming before I tackle the 16-bit side of the Z-100!

Listing 1

```
;PROGRAM: STYLE80.ASM
;
; Permits software selection of print styles/modes
; on Epson RX-80 Printer
;
;                       ************************
;                       * by J.F. Smith        *
;                       *    64, el Zahraa St. *
;                       *    Dokki, Cairo      *
;                       *    30-Jul-83         *
;                       ************************
;
;
;       ASSEMBLY CONSTANTS
;       ******************
BASE    EQU     0
BDOS    EQU     BASE + 005H
TPA     EQU     BASE + 100H
;
;
;       CP/M I/O FUNCTIONS
;       ******************
CRT     EQU     9           ;CP/M Function: print string
;                            at the console
;
INPUT   EQU     1           ;CP/M Function: Move one
;                            char from keyboard to the
;                            "A" register
;
LIST    EQU     5           ;CP/M List Output Function
;                            - moves one character at a
;                            time from the "E" register
;                            to the printer
;
;       MISCELLANEOUS EQUATES
;       *********************
CR      EQU     0DH         ;Carriage Return = 0D Hex
```

```
LF      EQU     0AH             ;Line feed = 0A Hex
BELL    EQU     07H             ;Bell = 07 Hex
;
;
        ORG TPA                 ;Start program at 0100 Hex
;
;       PRINT MENU AND GET INPUT
;       ************************
;
START   LXI     D,PROMPT        ;Point to the prompt string
        MVI     C,CRT           ;Call the console print
;                                function
        CALL    BDOS            ;Put it on the screen
        CALL    SCIN            ;Move char from keyboard to
;                                "A" register
        CPI     '1'             ;Is it a 1?
        JZ      ITALIC          ;If so, go to Italics
;                                Routine
        CPI     '2'             ;Is it a 2?
        JZ      ELITE           ;If so, make it Elite
        CPI     '3'             ;Is it a 3?
        JZ      COND            ;If so, go to Condensed
;                                Print
        CPI     '4'             ;Is it a 4?
        JZ      ENLG            ;If so, go to Enlarged mode
        CPI     '5'             ;Is it a 5?
        JZ      CNDEL           ;If so, go to Condensed +
;                                Enlarged
        CPI     '6'             ;Is it a 6?
        JZ      REDIT           ;If so, go to Reduced
;                                Italics
        CPI     '7'             ;Is it a 7?
        JZ      QUIET           ;If so, half speed print
        CPI     '8'             ;Is it an 8?
        JZ      DBLSTRK         ;If so, go to Double-Strike
        JMP     DFALT           ;If any other, go back to
;                                Pica
;
SCIN    PUSH    H               ;Single character input
;                                routine
        PUSH    D
        PUSH    B
        MVI     C,INPUT
        CALL    BDOS
        POP     B
        POP     D
        POP     H
        RET
;
;       PRINTER STYLE SELECTION ROUTINES
;       ********************************
;
ITALIC  MVI     E,1BH           ;Put Hex 1B <ESC> in the
;                                "E" register
        MVI     C,LIST          ;Tell CP/M it goes to the
;                                LST: device
        CALL    BDOS            ;Send it
        MVI     E,34H           ;Put Hex 34 (ASCII "4") in
;                                "E" register
        MVI     C,LIST          ;Tell CP/M it goes to the
;                                printer
        CALL    BDOS            ;Send it
        JMP     ACK             ;Sent <ESC> "4" -- see if
;                                it arrived
;
REDIT   MVI     E,1BH           ;Enter Italics code
        MVI     C,LIST
        CALL    BDOS
        MVI     E,34H
        MVI     C,LIST
        CALL    BDOS            ;and
        JMP     COND            ;condense it
;
ELITE   MVI     E,1BH           ;Set up the Elite codes
        MVI     C,LIST
        CALL    BDOS
        MVI     E,4DH
        MVI     C,LIST
        CALL    BDOS
        MVI     E,BELL      ;Emphasized elite invalid, so
        MVI     C,LIST      ;ring the bell
        CALL    BDOS        ;and
        RET                 ;Return to CP/M
;
COND    MVI     E,0FH       ;Go to condensed print mode
        MVI     C,LIST
        CALL    BDOS
        MVI     E,BELL      ;Emphasis invalid with this mode
        MVI     C,LIST      ;so ring the bell
        CALL    BDOS        ;and
        RET                 ;get out of here
;
ENLG    MVI     E,1BH       ;Go to Enlarged mode
        MVI     C,LIST
        CALL    BDOS
        MVI     E,57H
        MVI     C,LIST
        CALL    BDOS
        MVI     E,01H
        MVI     C,LIST
        CALL    BDOS
        JMP     ACK
;
CNDEL   MVI     E,1BH           ;Go to Condensed + Enlarged
;                                mode
        MVI     C,LIST
        CALL    BDOS
        MVI     E,57H
        MVI     C,LIST
        CALL    BDOS
        MVI     E,01H
        MVI     C,LIST
        CALL    BDOS
        MVI     E,0FH
        MVI     C,LIST
        CALL    BDOS
        MVI     E,BELL      ;Invalid to emphasize this mode,
        MVI     C,LIST      ;so sound the bell
        CALL    BDOS        ;and
        RET                 ;return to CP/M
;
QUIET   MVI     E,1BH       ;Set up quiet (half-speed)
;                            printing
        MVI     C,LIST
        CALL    BDOS
        MVI     E,73H
        MVI     C,LIST
        CALL    BDOS
        MVI     E,01H
        MVI     C,LIST
        CALL    BDOS
        MVI     E,BELL      ;Acknowledge the message
        MVI     C,LIST
        CALL    BDOS
        RET                 ;Return to CP/M
;
DBLSTRK MVI     E,1BH       ;Set up double-strike mode
        MVI     C,LIST
        CALL    BDOS
        MVI     E,47H
        MVI     C,LIST
        CALL    BDOS
        JMP     ACK         ;See if emphasis wanted
;
ACK     MVI     E,BELL
        MVI     C,LIST
        CALL    BDOS        ;Send the bell to the printer
        LXI     D,PROMP2    ;then point to the
;                            "Emphasize" prompt
        MVI     C,CRT
        CALL    BDOS        ;Print it
        CALL    SCIN        ;Get the single character
;                            answer
        CPI     'Y'         ;Is it a Y?
        JZ      EMPH        ;If so, go to the emphasize
;                            routine
        CPI     'y'         ;Is it a y?
        JZ      EMPH        ;Jump to Emphasize routine
        MVI     E,BELL      ;If not, ring the bell
        MVI     C,LIST      ;and
        CALL    BDOS        ;
        RET                 ;return to CP/M
;
EMPH    MVI     E,1BH       ;Send the Emphasized mode
;                            message
        MVI     C,LIST
        CALL    BDOS
        MVI     E,45H
        MVI     C,LIST
        CALL    BDOS
        MVI     E,BELL      ;Got here OK
        MVI     C,LIST
        CALL    BDOS
        RET                 ;Go back to CP/M
;
DFALT   MVI     E,1BH       ;Return the printer to
;                            default (Pica) mode
        MVI     C,LIST
        CALL    BDOS
        MVI     E,40H
        MVI     C,LIST
        CALL    BDOS
        JMP     ACK         ;Acknowledge and check for
;                            emphasis
```

☞

```
;
;       PROMPT STRINGS
;       **************
;
PROMPT  DB 'You may select print styles on the RX-80.
        DB  You will also',CR,LF
        DB 'have an opportunity to emphasize the print
        DB  on modes for',CR,LF
        DB 'which this option is valid.  Please make
        DB  your choice:',CR,LF
        DB CR,LF,'   (1) Italics',CR,LF
        DB '    (2) Elite',CR,LF
        DB '    (3) Condensed',CR,LF
        DB '    (4) Enlarged',CR,LF
        DB '    (5) Enlarged + Condensed',CR,LF
        DB '    (6) Reduced Italics',CR,LF
        DB '    (7) Quiet (Half Speed) Printing',CR,LF
        DB '    (8) Double-Strike Printing',CR,LF
        DB '    (Any other character) Pica',CR,LF,CR,LF
        DB 'Please enter your choice:',CR,LF,'$'
;
PROMP2  DB CR,LF,
        DB 'Do you want emphasized print (Y/N)?',CR,LF,'$'


Listing 2

;PROGRAM: STYLE25.ASM
;
; Permits software selection of printer functions on
; the Heath/Zenith H/Z-25 and H/Z-125 Line Printers
;
;                  ************************
;                  * By J.F. Smith        *
;                  *    64, el Zahraa St. *
;                  *    Dokki, Cairo      *
;                  *    Egypt             *
;                  *    31-Oct-83         *
;                  ************************
;
;
;       ASSEMBLY CONSTANTS
;       ******************
BASE    EQU     0
BDOS    EQU     BASE + 005H
TPA     EQU     BASE + 100H
;
;
;  CP/M I/O FUNCTIONS
;  ******************
CRT     EQU     9       ;CP/M Function: print string
;                        at the console
;
INPUT   EQU     1       ;CP/M Function: Move one char
;                        from keyboard to the "A" register
;
LIST    EQU     5       ;CP/M Function - moves one
;                        character at a time from the "E"
;                        register to the LST: device.
;
;  MISCELLANEOUS EQUATES
;  *********************
CR      EQU     0DH             ;Carriage Return = 0D Hex
LF      EQU     0AH             ;Line feed = 0A Hex
BELL    EQU     07H             ;Bell = 07 Hex
;
;
        ORG TPA                 ;Start program at 0100 Hex
;
;  PRINT PROMPT AND GET INPUT
;  **************************
;
START   LXI     D,PROMPT        ;Point to the prompt string
        MVI     C,CRT           ;Call the console print
;                                function
        CALL    BDOS            ;Put it on the screen
        CALL    SCIN            ;Move char from keyboard
;                                to "A" register
        CPI     '1'             ;Is it a 1?
        JZ      DBLW            ;If so, go to Double
;                                Width routine
        CPI     '2'             ;Is it a 2?
        JZ      PICA            ;If so, make it Pica
        CPI     '3'             ;Is it a 3?
        JZ      ELITE           ;If so, go to Elite print
        CPI     '4'             ;Is it a 4?
        JZ      REDELT          ;If so, go to Reduced Elite
        CPI     '5'             ;Is it a 5?
        JZ      COND            ;If so, go to Condensed
;                                print
        CPI     '6'             ;Is it a 6?
        JZ      LPI6            ;If so, go to 6 Lines
;                                per Inch
        CPI     '7'             ;Is it a 7?
        JZ      LPI8            ;If so, set it for 8 lines
;                                per inch
        CPI     '8'             ;Is it an 8?
        JZ      GRAPH           ;If so, start graphics
;                                mode
        CPI     '9'             ;Is it a 9?
        JZ      NOGRAPH         ;If so, turn off graphics
        CPI     '0'             ;Is it a 0?
        JZ      DFALT           ;If so, initialize printer
;                                to default settings
        CPI     0DH             ;Is it a <return>?
        RZ                      ;If so, close out and return
;                                to CP/M
        CPI     ' '             ;None of the above
        JMP     ERROR           ;Send bell and return to
;                                the menu
;
SCIN    PUSH    H               ;Single character input
;                                routine
        PUSH    D
        PUSH    B
        MVI     C,INPUT
        CALL    BDOS
        POP     B
        POP     D
        POP     H
        RET
;
ERROR   MVI     E,BELL          ;Send a bell to the console
        MVI     C,CRT
        CALL    BDOS
        JMP     START           ;Return to the menu, try
;                                again
;
;       PRINTER STYLE SELECTION ROUTINES
;       ********************************
DBLW    MVI     E,1BH           ;Put ESC (Hex 1b) in the
;                                "E" Register
        MVI     C,LIST          ;Direct it to the printer
        CALL    BDOS            ;Send it
        MVI     E,63H           ;Put ASCII "c" in the
;                                "E" Register
        MVI     C,LIST          ;Direct it to the printer
        CALL    BDOS            ;Send it
;
;       *** NOTE:  The above sequence sent ESC "c" --
;                  the Z-25/125 initialization sequence...
;
        MVI     E,0EH           ;Place code for double width
;                                print in E register
        MVI     C,LIST          ;Tell CP/M where it goes
        CALL    BDOS            ;Send it to the printer
        MVI     E,BELL          ;Put the bell code in the
;                                E register
        MVI     C,LIST          ;Tell CP/M where it goes
        CALL    BDOS            ;Send it to the printer for
;                                verification
        JMP     START           ;Any more?
;
PICA    MVI     E,1BH           ;Send code for horizontal
;                                pitch=10 CPI
        MVI     C,LIST          ;
        CALL    BDOS            ; ***************************
        MVI     E,5BH           ; *NOTE: The H/Z-25 software *
        MVI     C,LIST          ; *code for 10 CPI horizontal*
        CALL    BDOS            ; *is:                       *
        MVI     E,77H           ; *  Esc [ w in ASCII        *
        MVI     C,LIST          ; *      or                  *
        CALL    BDOS            ; *  1B 05 77 in Hex        *
        MVI     E,BELL          ; ***************************
        MVI     C,LIST          ;
        CALL    BDOS            ;
        JMP     START           ;Any more changes?
```

☞

```
ELITE   MVI     E,1BH       ;Send code for horizontal
;                            pitch=12 CPI
        MVI     C,LIST
        CALL    BDOS
        MVI     E,5BH
        MVI     C,LIST
        CALL    BDOS
        MVI     E,32H
        MVI     C,LIST
        CALL    BDOS
        MVI     E,77H
        MVI     C,LIST
        CALL    BDOS
        MVI     E,BELL
        MVI     C,LIST
        CALL    BDOS
        JMP     START       ;Return to the menu
;
REDELT  MVI     E,1BH       ;Send code for horizontal
;                            pitch=13.2 CPI
        MVI     C,LIST
        CALL    BDOS
        MVI     E,5BH
        MVI     C,LIST
        CALL    BDOS
        MVI     E,33H
        MVI     C,LIST
        CALL    BDOS
        MVI     E,77H
        MVI     C,LIST
        CALL    BDOS
        MVI     E,BELL
        MVI     C,LIST
        CALL    BDOS
        JMP     START       ;Return to the menu
;
COND    MVI     E,1BH       ;Send code for horizontal
;                            pitch=16.5 CPI
        MVI     C,LIST
        CALL    BDOS
        MVI     E,5BH
        MVI     C,LIST
        CALL    BDOS
        MVI     E,34H
        MVI     C,LIST
        CALL    BDOS
        MVI     E,77H
        MVI     C,LIST
        CALL    BDOS
        MVI     E,BELL
        MVI     C,LIST
        CALL    BDOS
        JMP     START       ;Return to the menu
;
LPI6    MVI     E,1BH       ;Send code for lines per
;                            inch = 6
        MVI     C,LIST
        CALL    BDOS
        MVI     E,5BH
        MVI     C,LIST
        CALL    BDOS
        MVI     E,78H
        MVI     C,LIST
        CALL    BDOS
        MVI     E,BELL
        MVI     C,LIST
        CALL    BDOS
        JMP     START       ;Return to the menu
;
LPI8    MVI     E,1BH       ;Send code for lines per
;                            inch = 8
        MVI     C,LIST
        CALL    BDOS
        MVI     E,5BH
        MVI     C,LIST
        CALL    BDOS
        MVI     E,32H
        MVI     C,LIST
        CALL    BDOS
        MVI     E,78H
        MVI     C,LIST
        CALL    BDOS
        MVI     E,BELL
        MVI     C,LIST
        CALL    BDOS
        JMP     START       ;Back to the menu
;
GRAPH   MVI     E,1BH       ;Tell printer to enter
;                            Graphics mode
        MVI     C,LIST
        CALL    BDOS
        MVI     E,5BH
        MVI     C,LIST
        CALL    BDOS
        MVI     E,0AH
        MVI     C,LIST
        CALL    BDOS
        MVI     E,6DH
        MVI     C,LIST
        CALL    BDOS
        MVI     E,BELL
        MVI     C,LIST
        CALL    BDOS
        RET                 ;Can't mix with others,
;                            return to CP/M
NOGRAPH MVI     E,1BH       ;Exit Graphics mode
        MVI     C,LIST
        CALL    BDOS
        MVI     E,5BH
        MVI     C,LIST
        CALL    BDOS
        MVI     E,0BH
        MVI     C,LIST
        CALL    BDOS
        MVI     E,6DH
        MVI     C,LIST
        CALL    BDOS
        MVI     E,BELL
        MVI     C,LIST
        CALL    BDOS
        JMP     START       ;Return to menu
;
DFALT   MVI     E,1BH       ;Initialize printer to
;                            default modes
        MVI     C,LIST
        CALL    BDOS
        MVI     E,63H
        MVI     C,LIST
        CALL    BDOS
        MVI     E,BELL
        MVI     C,LIST
        CALL    BDOS
        JMP     START       ;Return to menu
;
;       PROMPT STRINGS
;       **************

PROMPT  DB '   You may select print characteristics on
        DB  the H/Z-25.  If',CR,LF
        DB 'you wish to combine commands, i.e. change
        DB  both horizontal ',CR,LF
        DB 'and vertical pitch, the program will permit
        DB  you to do so',CR,LF
        DB 'by re-prompting with the menu.  It will
        DB  not re-prompt  ',CR,LF
        DB 'if the "Enter Graphic Mode" option is
        DB  selected. It will',CR,LF
        DB 'sound the bell at the console and re-prompt
        DB  if an invalid',CR,LF
        DB 'selection is made.',CR,LF,CR,LF
        DB '   To exit the program, type
        DB  <RETURN>...',CR,LF,CR,LF
        DB 'Please make your choice: ',CR,LF
        DB '   (1) Double Width (cancelled by
        DB  <RETURN>)',CR,LF
        DB '   (2) Pica (10 CPI)',CR,LF
        DB '   (3) Elite (12 CPI)',CR,LF
        DB '   (4) Reduced Elite (13.2 CPI)',CR,LF
        DB '   (5) Condensed (16.5 CPI
        DB          <132 column>)',CR,LF
        DB '   (6) Vertical Pitch - 6 LPI',CR,LF
        DB '   (7) Vertical Pitch - 8 LPI',CR,LF
        DB '   (8) Enter Graphics Mode',CR,LF
        DB '   (9) Exit Graphics Mode',CR,LF
        DB '   (0) Initialize Printer to Default
        DB          Settings',CR,LF,CR,LF
        DB 'Please enter your choice:',CR,LF,'$'
```

Vectored from 8

problem in Rick Swenton's "I/O Baud Rate Programmer" (REMark, March 1984). As published, it won't work properly if the "F" (3600 baud) option is selected from the baud rate menu.

The fix is easy; simply change the 3600 baud value under the label BRDVT: from 0024H to 0020H. This will provide the proper divisor for loading into the 8250's programmable baud rate generator, and correct the baud rate mismatch which was most obvious when the Console speed was changed to 3600 baud.

While I have your ear, I'd also like to suggest an easy change to Mr. Swenton's very useful program. Under the label SENDB:, after the 3rd line (CALL INC"), insert the code "ANI 5FH". The listing will then look like this:

```
SENDB:  LXI     D,BAUDM
        CALL    OUTS
        CALL    INC
        ANI     5FH
        SUI     A
        (etc.)
```

What this small modification accomplishes is to allow keyboard input of either upper or lower case letters in response to the Select Baud Rate prompt. Not an earthshaking change, but it made the program easier for me to use, and it might help others as well.

Thanks for your unique support to all of us Heath/Zenith users, and keep up the good work!

Robert P. Moroney
656 Chapel Gate Drive
Odenton, MD 21113

Beware - Power Surges!

Dear HUG,

This letter doesn't have anything to do with computing, but it may (I hope) keep some fellow readers from experiencing the computer down time that I have.

I live in the rain and thunderstorm capitol of the nation (in my opinion), Mobile, AL. We received approximately 85 inches of rain last year. Now that is a lot of thunderstorms!

As we all know, when thunderstorms are present it is a wise move to power the system down. Around here, it can be nice in my front yard and storming around the block. That's when you can get surges and spikes up into the thousands of volts.

I know a lot of you, like me, have bought those plug-in type surge suppressors. These are fine, but in my view, not quite enough. Most of these only take out between 150 and 450 volts and can be damaged by spikes above these. This of course depends on the type of MOV that is used. I've had one of these $65 devices get zapped and short out, along with the video board and my microwave oven.

I'm an electrician by trade, and for the past two years I've installed quite a few MOV's on different mainframes and mini computers. According to some of the maintenance reports, they seem to work if they are sized properly. After some checking around, I found that GE has published a data sheet on MOV's that was very helpful. I found that you could cascade these MOV's, starting with a small 130 volt model that can be attached to the receptacle that you wish, up to models that can handle several thousand volts. These larger models can handle in excess of 50,000 amps for a few micro seconds.

The biggest advantage of installing these type of MOV's is that you can not only protect your computer, but your whole house. I've cascaded three different sizes on my house that go from 150 volts to 1300 volts for about $145. Of course me being an electrician, installation was of no cost. If you are not sure about how these devices should be installed, get advice from someone that knows.

I recommend putting the larger MOV's in the Main Distribution Panel, if you have room. I put the two larger MOV's in my main panel, the next smaller size in the indoor breaker panel, and the smaller ones on the receptacles themselves. The smaller MOV's are small enough to be attached directly to the back of any receptacle you wish, without being seen. I used several of these on the '89, microwave, and TVs. They only cost a couple of dollars.

The following is a list of the MOV's I've been talking about. All of these are GE numbers, you may find another manufacturer and use these as a cross reference. You may not want all three levels of protection, but I recommend at least the first two.

MOV #	RMS VOLTAGE	CLAMPING VOLTAGE MIN	CLAMPING VOLTAGE MAX	PEAK CURRENT
V130LA10A	130	184	340	4500
V250PA40A	250	354	675	6500
V480PA80A*	480	670	1280	6500
V480HE500*	480	670	1320	25000

*I used the V480PA80A mainly because of the cost. The V480HE500 costs about $30 more than the PA series.

Here are some instructions for you or your electrician, if he's not sure how to install these MOV's.

1) Be sure to insulate the small leads on the MOV's you install on any receptacle.
2) The larger MOV's have heat sinks on the back that are also used as a mounting means. Be sure it is securely mounted to the grounded metal frame of your panel.
3) If you mount the larger MOV's in the main panel, DO NOT connect the leads directly to the Power Co. meter. Either connect it to a new circuit breaker or the existing breaker (or fuse block) that feeds your house panel.

The information above came from GE pamphlet 600.60 1/83 (50M) S.L. The pamphlet also lists MOV's for low voltage DC protection, in the 5.5, 8.0, 14.0, and 18 volt range. I haven't tried these yet, but I intend to. Every little bit of protection helps.

I hope this information will help some of you, and not make the manufacturers of Surge Protectors too mad. If you have any comments or suggestions, please write at the address below.

Johnny Dunn
5217 Azalea Cir.
Mobile, AL 36608

Running a Heath H-8-2 Parallel Interface In CP/M

Dear Pat Swayne,

On page 30 of REMark Issue 29, you stated that you would like to hear from anyone who had tried running a Heath H-8-2 parallel interface in CP/M. I recently acquired a printer that came with a parallel interface and I had an H-8-2 that I decided to use rather than having to buy a serial interface, thereby saving a little bit of money.

Following is an outline of the modifications made to BIOS that seem to result in a satisfactory operation of the H-8-2 parallel interface to communicate with the new printer assigned to port 0D0H as the LPT:.

THIS IS FOR CP/M VERSION 2.2.03 AND THE BIOS LISTING FURNISHED BY HEATH/ZENITH IS THE REFERENCE

Page #013

The line that reads:

```
H84LPT  EQU     0E0H
```

Change to:

```
H84LPT  EQU     0D0H
```

Page #112

The section headed LINE PRINTER OUT was rewritten as follows:

```
LPTOUT: CALL    LPTOU2
        ORA     A
        JZ      LPTOUT
        JMP     UO
LPTOU2: LXI     H,H84PT3
        LXI     D,LPTCTS
        JMP     CRTOS3
```

This will direct the line printer out to the 8251 output routines.

Page #114

At:

```
CRTOS3:  IN         H85CRT+1
```

Change the H85CRT+1 to H84PT3+1 to check the status of the correct port.

Page #125

```
11th line       POP     PSW
12th line       CALL    PIN
13th line       CPI     3
```

Delete the 5 lines after CALL PIN to cause the H8-5 card to be initialized regardless. The line CALL PIN could be deleted too, I assume, but I left it in. The main thing is to avoid the jump past the 8251 initialization.

In the 8251 part, change at 4 places the H85CRT+1 operand to H84PT3+1 so the correct port will be initialized.

Page #138

On the very first line of the 8250 initialization subroutine, remove the label IN8250: from the statement

```
IN8250: MOV     B,A
```

And immediately preceeding this line, insert the following two lines:

```
IN8250:  CPI        H84PT3
         RZ
```

This prevents the 8251 from being initialized as an 8250 when the 8250s are done in sequence.

With these changes in the BIOS.ASM file, make a working BIOS.SYS with MAKEBIOS, CONFIGUR it to your system, and start printing.

An LA38 DECwriter or similar machine can be connected to port 340Q as the TTY: device if desired.

For HDOS, I assembled the ATH84.ASM for an 8251 USART, and renamed it LP.DVD. I was even able to squeeze in a SET option for TABX - NOTABX.

The CPU is a D-G Z80 running at 4MHz.

Daniel A. Schlichtig
18832 W. Cabral Street
Canyon Country, CA 91351

Correction to BASMAPER

Dear HUG,

I have received several letters concerning my article BASMAPER which appeared in your February 1984 issue and thought you might also have had inquiries. The listing in Figure 2 has two truncation problems: a "+" at the end of the second line of statement 65503, and a ":" at the end of the second line of statement 65506.

```
65503 ... +5*(X=29)+9*(X=31))
65506 ... WEND: WHILE FNMORE AND JX ...
```

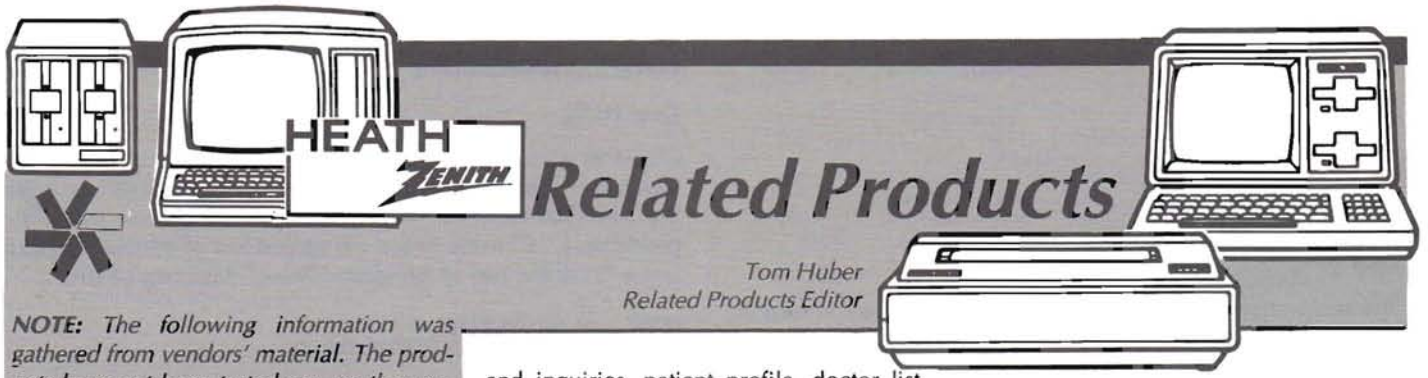
In addition I have been informed by one reader that the program does not work on a 128K RAM configuration. I apologize for this problem. I have a 192K RAM model and didn't think to look into that type of problem. If any of your readers comes up with a solution, please let me know. I will look into it when time is again available.

Ted Miller, Jr.
7749 Granada Dr.
Buena Park, CA 90621

Correction

In the April 1984 issue, please make the following correction to the article "Random Files, Sorting - Part 1", on page 60. In the listing, LOOKUP.BAS, line 1110 should read as follows:

```
1110 IF C$<I1$ THEN A=B+1:GOTO 1070  'TOO SMALL. MOVE UP
```



Related Products

Tom Huber
Related Products Editor

***NOTE:** The following information was gathered from vendors' material. The products have not been tested nor are they endorsed by HUG. We are not responsible for errors in descriptions or prices.*

With the advent of the Z-100 PC Series (Z-150 Desktop Computer and Z-160 Portable Computer), Zenith Data Systems is producing two IBM PC compatible computers. Compatibility is almost complete. However, due to differences in the monitor ROM's of the Zenith and IBM computers, 100 percent compatibility cannot be guaranteed. Therefore, this column will limit coverage of IBM software to only those packages of high interest or unusual flavor or those packages that are specifically mentioned by the vendor to be suitable for use with the Zenith computers.

Custom Graphics for CP/M, MS-DOS

Mosaic Software has released their SOFTPLOT/BGL device- independent graphics extension system, which allows users to create custom graphics applications in BASIC. It features two dimensional viewing with windowing (scaling), three dimensional plotting, dashed and color lines, image rotation, and automatic text justification. Emuplot, a general purpose plotter emulator for dot-matrix printers, is included. SOFTPLOT provides functions comparable to CORE and GKS base level graphics packages running on mini and mainframe computers. The package requires 64K, CP/M or MS-DOS (specify), and MBASIC. For a complete list of compatible computers, printers, and plotters, contact the vendor.

Vendor: Mosaic Software Inc.
1972 Massachusetts Ave.
Cambridge, MA 02140
(617) 491-2434
Price: $99.00

Pharmacy System for dBASE II

The Superior Pharmacy System is a complete pharmacy management system that was written under the guidelines of the American Pharmaceutical Association's "Computers and Pharmacists" publication and runs under Ashton-Tate's dBASE II. It handles prescription filling and automatic refills, prescription labels and Medicaid forms, pricing and inquiries, patient profile, doctor list, drug inventory and listing, warnings, archiving, and can be optionally tied to accounts receivable. For more information, contact the vendor.

Vendor: Superior Software Corp.
202 13th Street, Suite 206
Augusta, GA 30901
(404) 722-0831
Price: Complete System: $1,295.00
Demo w/user manuals: $49.00
User Manuals alone: $9.00

Valley Data Sciences Adds New Programmers to Line

Supporting the H/Z-89/90 series of computers, Valley Data Sciences has added several lines of Memory and Logic Device programmers, including production units that can program up to sixteen devices at the same time. Full software support, including logic and memory editors, is available for CP/M machines, including the H/Z-89. For more information, contact the vendor.

Vendor: Valley Data Sciences
Charleston Business Park
2426 Charleston Road
Mountain View, CA 94043
Phone: (415) 968-2900

ZPAY Payroll Systems for CP/M-80, CP/M-85, CP/M-86, Z-DOS, and PC-DOS

ZPAY has been designed exclusively for the Zenith and Heath computer line and supports all of the lines special features such as graphics and printers. It supports both old and new style HUG checks, quarterly reports, and job costing information. Support for some states (AK, FL, IN, MI, NH, SD, TX, WY, CT, IL, LA, NV, PA, TN, WA) is supplied with the CP/M system and (apparently) custom support for other states is available upon request (our Z-DOS sample was for California only). Contact the vendor for individual state and computer needs.

Vendor: ZPAY Payroll Systems
c/o Paul Mayer
3516 Ruby Street
Franklin Park, IL 60131
(312) 671-3130
Price: $100.00 + $4.00 S&H

New Book Offers True Proportional Printing for WordStar

Proportional Spacing on WordStar provides all the details on how to enable proportional printing with most Daisy wheel and thimble printers. In addition, the book also tells how to print two or more justified columns on a page, and underlining between words. The book was written after three years of research and covers all versions, both 8- and 16-bit. For more information, contact the vendor.

Vendor: Writing Consultants
11 Creek Bend Drive
Fairport, NY 14450
(716) 377-0130
Price: $19.95 + $2.50 S&H
(NY state residents add
7% sales tax)

Doodler Graphics Package for the H/Z-100

Paul F. Herman has announced the Doodler Graphics Package, a sophisticated design tool. Two-dimensional drawing and design in color or monochrome is simplified with menu driven single-key commands for lines, boxes, circles, ovals, and mirror images. Text is variable width, proportionally spaced, and may be scaled. The user may select italic or reverse italic, and design his own character fonts using the included font editor. Doodler includes drivers for Gemini, Epson, C. Itoh, and similar printers to produce the graphics displayed on the screen.

Vendor: Paul F. Herman
Data Systems Consultant
P.O. Box 535
St. James City, FL 33956
(813) 283-2227
Price: $79.95

Winchester Backup for H/Z-100 Computers to Become Available Soon

Systems Innovations, Inc. has announced the Guardian 25 cartridge backup system for H/Z-100 Winchester computers. The system uses a DC600 Data Cartridge and provides both selective file and/or total disk backup through fully integrated utility software. The software is menu driven and

prompts the user to select which files to back up or restore based on a number of different factors including time, date, and by using wild card characters. The total 11 megabytes of the Winchester can be backed up or restored in about eight minutes. Optional Archival Management Software (AMS25) allows the unit to function as an extension to the Winchester system, providing 35 megabytes of storage. For some applications, the Guardian 25 can be used as a stand alone system with 25 megabytes of storage. For more information, contact the vendor. (Note: Availability is 2nd quarter for evaluation units and 3rd quarter for production units, according to the vendor.)

Vendor: Systems Innovations, Inc.
N.R. Prevett
P.O. Box 2066
505 Westford St.
Lowell, MA 01851
(617) 459-4449
Price: $1,295.00

The Naked Computer Chronicles Trivia and Other Arcane Facts

It has been said that if automobile technology had advanced at the rate of computer technology of the last 30 years, a Rolls Royce would cost $2.50 and get two million miles to the gallon. True, unless the particular computer technology one had in mind were that of the GE Fluid Computer, which attempted to use water instead of electrons for switching circuits; or, the Interplex round teleprinter computer that could multiply 12 times 12 and never get anything but 143; or, the RCA BIZMAC, a vacuum tube dinosaur that took so long to build it was obsolete before it was done (it was so big that its operators wore roller skates). These excerpts are from The Naked Computer, by Jack B. Rochester and John Gantz, which is subtitled, "Layperson's Almanac of Computer Lore, Wizardry, Personalities, Memorabilia, World Records, Mind Blowers and Tomfoolery". 335 pages, hardcover only.

Vendor: William Morrow & Co.
Price: $15.95 through most bookstores
Full-Text:DELPHI Videotex system:
(617) 491-3393

Sorcim Access for SuperCalc

Sorcim Access is a clearinghouse for ideas on how to make SuperCalc and other Sorcim products more usable. It is a catalog of products selected by Sorcim, the vendor of SuperCalc, that is felt to be the best of what's available in templates, books, and accessories, and make them available by mail if you can't obtain them elsewhere. Contact the vendor for a copy of the catalog.

Vendor: Sorcim Access
P.O. Box 32505
San Jose, CA 95125
(408) 942-0771 (8:00 AM
to 5:00 PM Pacific Time Zone)
Price: contact vendor

Graph-Pac I and Graph-Pac II for H-8 and H/Z-89/90

Graph-Pac I and II are two graphics software support packages for the HA-8-3 and HA-89-3 color graphics and sound generation boards for the H-8 and H/Z-89/90 computers.

Graph-Pac I supports all the capabilities of the graphics boards including the Votrax and DAC's (if installed). It consists of a graphics version of Tiny Pascal (for HDOS) and two GSL's (Graphics Support Libraries), one each for the H-8 and H/Z-89/90. The disk contains the Tiny Pascal Compiler, configure program and include files, a 123 sector Tiny Pascal documentation file, H8 and H89 GSL's, and test and demonstration programs which illustrate the capabilities of the graphics board and routines.

Graph-Pac II adds, in addition to Graph-Pac I features, character rotation, Greek alphabet, font creation, random number generation, CP/M & HDOS optional support for Pascal MT+, C/80 (from Software Toolworks), MACRO-80, FORTRAN, COBOL, and MBASIC (both interpreter and compiler).

Vendor: Fred Pospeschil
3108 Jackson St.
Bellevue, NE 68005
(402) 291-0795
(7:00 PM -10:00 PM
Central Time Zone)
Prices:
Graph-Pac I (with Tiny Pascal on HDOS hard-sectored only): $39.00
Graph-Pac I (MACRO-80 source code on three disks): $30.00
Graph-Pac II (one DOS and one language, your choice, specify hard or soft sectored disk format): $39.00
Other operating system: add $12.00
Each additional language: add $10.00
(NE residents add sales tax to your order)

Instant Help Utility for H-8, H/Z-89/90, and H/Z-100

Instant Help is a utility for computer programmers. It allows access to reference material from MBASIC and CP/M and HDOS editors without exiting the language editor. Help files are supplied for MBASIC, CP/M, ED, and CP/M system calls (a similar set is supplied for HDOS). The user may modify any of the existing supplied help files or create his own custom help files. All H-8 and H/Z-89/90 HDOS and CP/M and H/Z-100 CP/M-85 formats are available (specify).

Vendor: J. E. Brancheau Engineering Co.
P.O. Box 67
Trenton, MI 48183
(313) 675-5585
Price: $39.95

Four Emulators from KEA Systems for H/Z-100 Computers

KEA Systems has added the XMODEM transparent error correcting protocol to ZSTEM-VT100 (the DEC VT100 and VT52 emulator), ZSTEM- VT52 (the DEC VT52 emulator), ZSTEM-D200 (the Data General D100/200 emulator) and ZSTEM-HOBBY (a limited version of ZSTEM- VT52). The XMODEM implementation includes single and multiple file transfers, directory display and file deletion (both have wild card support), and CRC or checksum error detection. ZSTEM- Hobby does not have direct printer support but does emulate DEC VT52 escape sequences, user configuration for speed (45.5 to 1200 baud), character size, parity, flow control protocols, and so on.

Vendor: KEA Systems Ltd.
#311-811 Beach Avenue
Vancouver, B.C. Canada V6Z 2B5
(604) 687-2744
Prices: ZSTEM-VT100: $148.95
ZSTEM-D200: $124.95
ZSTEM-VT52: ..... $ 98.95
ZSTEM-Hobby: ..... $ 39.95

H-8, H/Z-89/90, and H/Z-100 Products from Newline Software

Newline Software has announced a number of new products available in a number of different formats for H-8, H/Z-89/90, and H/Z-100 computers. They are available in hard or soft sector 5.25 inch formats or the 8 inch format (specify disk format and operating system). Contact the vendor for a full list of their offerings.

Vendor: Newline Software
P.O. Box 402
Littleton, MA 01460
(617) 486-8535

Pro Driver Z-DOS Communication Package

Studio Computers, Inc., has just announced Pro Driver, a new communications package for H/Z-100 computers using Z-DOS. It allows the user to talk to remote computers using any modem or directly to another computer through a null modem cable, and allows transmitting and receiving of both ASCII and binary files. It allows normal operating system functions, such as renaming files, deleting files, directory listings or resetting (exchanging) disks under menu-driven commands. Requires Z-DOS and 128K on H/Z-100 computers. (Note: Vendor indicates that versions for the Z-150 and CP/M-85 will be released later in the year.)

Vendor: Studio Computers, Inc.
999 South Adams
Birmingham, MI 48011
(313) 645-5365
Price: $49.00 + $2.00 S&H

Error, Error, Error!

I goofed! In last month's issue, I reviewed **Vega-Bound I** and mentioned its price at $49.00. While the vendor would, I suppose, be happy to sell it to you at that price, it was in error (my apologies all the way around). The originally announced price (in Heath Related Products, March, 1984) of $44.95 is still correct, I think... Contact the vendor to make sure before you order.

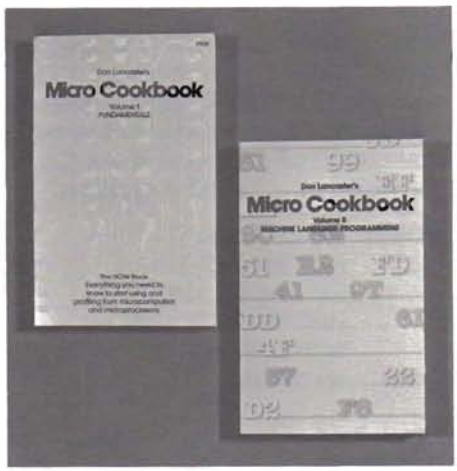
Vendor: Interdiscipline, Inc.
403 S. Brandon
Seattle, WA 98108
(206) 763-2099
Price: $44.95

Print Personal Checks on Your Tractor Printer

PaperCaper II allows you to use your own personal checks with your computer, printer, and personal finance software instead of purchasing continuous-form checks. The carrier will handle up to seven personal size checks or two business size letterheads through your 9.5-inch wide tractor or friction printer such as the H-14, H/Z-25/125, MPI-99, MPI-150, WH-54B, and WH-55. It is precision die cut and printed on a tough synthetic paper that is almost impossible to tear.

Vendor: Services Squared
Box 2665
Las Cruces, NM 88004-2665
(505) 522-4925
(evenings only, please)
Price: $20.00
(NM res. add 4.75% sales tax)

Volume 2 of the Don Lancaster's Micro Cookbook -- Machine Language Programming



Howard W. Sams & Co. has announced the second volume of the Micro Cookbook by Don Lancaster, author of *TTL Cookbook, CMOS Cookbook, Cheap Video Cookbook, Son of Cheap Video, TV Typewriter Cookbook, Active Filter Cookbook,* many magazine articles and other books. Second of a series on microprocessors and microcomputers, this book uses a group of "discover modules" to explain machine language programming fundamentals the reader can use with any microcomputer or microprocessor family. Virtually all available opcodes are explored, as are the details of flowcharting, using a stack, testing individual bits, creating text messages, using files, subroutines, interrupts, and more. The practicalities of addressing, memory maps, registers, I/O, and the simple circuitry needed to connect ports successfully with the outside world are all covered. For more information, contact the vendor.

Vendor: Howard W. Sams & Co., Inc.
4300 West 62nd St.
Indianapolis, IN 46268
(317) 298-5400
Price: $15.95

DISK-TRAN Software Expanded to Include Z-150/160 Computers

Computer Consultants to Business sells a line of disk-format conversion programs for CP/M, CP/M-85, Z-DOS, and MS-DOS to and from various other manufacturer's microcomputers and has expanded the line to include Z-150 and Z-160 computers. Contact the vendor for full details.

Vendor: Computer Consultants to Business
1033 Bishop Walsh Rd.
Cumberland, MD 21502
(301) 759-1260
Prices: $30.00 each, 2 for $50.00
(Add $1.00 S&H to any order)

H/Z-89/90 and H/Z-100 Sorting Utility From Sunflower

Sunflower Software is offering DISKSORT Version 2.0 for H/Z-89/90 computers under either HDOS or CP/M or the H/Z-100 under Z-DOS. DISKSORT version 2.0 is a new sort/merge program that can be used to create and maintain all kinds of ordered lists. It can sort text files by variable-length lines, multiple line groups, or fixed-length records, and on up to five user-specified fields in either ascending or descending order for each field. Files too large for memory are sorted in segments, using temporary files on a user-specified disk. DISKSORT can also merge two sorted files into a single file. Specify operating system (HDOS, CP/M, or Z- DOS) when ordering.

Vendor: Sunflower Software
13915 Midland Drive
Shawnee, Kansas 66216
(913) 631-1333
Price: $59.95 + $2.00 S&H
(Kansas residents add sales tax)

H/Z-19 and H/Z-89/90 Improved Graphics Resolution

NORCOM is featuring G-Prom, a new character generator for the H/Z-19 terminal and H/Z-89/90 computers. Twenty-four of the original graphic symbols are modified to enhance vertical resolution and to improve the vertical to horizontal ratio in the line drawing characters. Twenty-three ASCII characters are modified to improve character formation. The G-Prom can address 125 pixels vertically and any one of ten (five by two) pixels in each character location can be turned on. G-Prom is a direct plug replacement for the original character generator and includes documentation, installation instructions, and a demonstration program listing.

Vendor: NORCOM
9630 Hayes
Overland Park, KS 66212
Price: $19.95

General Ledger Interfaces with Multiplan

Taranto & Associates has announced the release of General Ledger version 4.0 for CP/M and MS/DOS operation systems. Integration with Multiplan permits virtually unlimited report formatting and financial analysis of General Ledger data. For more information, contact the vendor.

Vendor: Taranto & Associates, Inc.
P.O. Box 6216
121 Paul Drive
San Rafael, CA 94903
(800) 227-2868 or
(415) 472-2670 inside CA
Price: $200.00

EPROM Programmer for H-8, H/Z-89/90, and H/Z-100 Computers

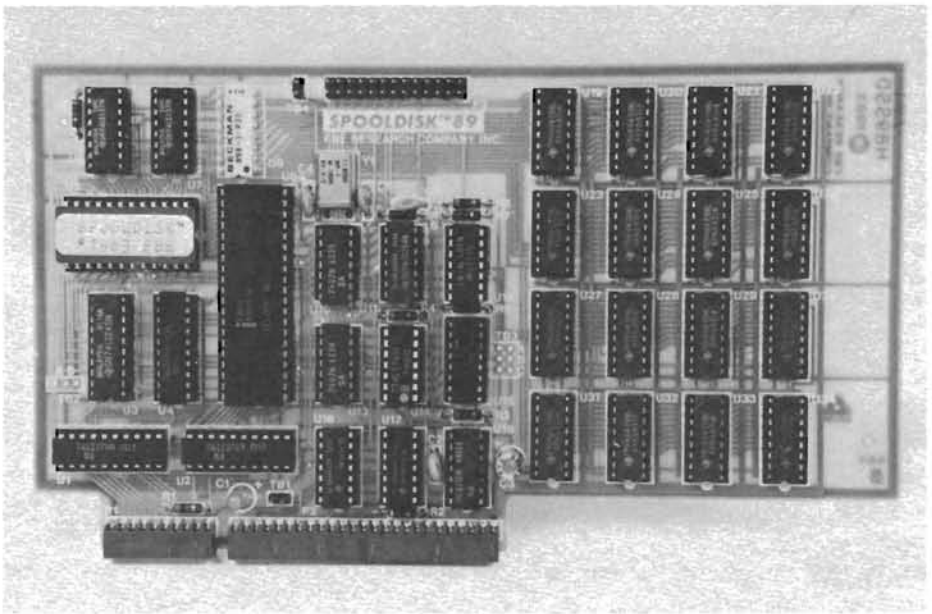
Ross Custom Electronics has introduced the IntelliBurner EPROM programmer. Two communications modes are featured: "DumBurner Emulation" mode, which allows data to be transferred to or from disk files under software control; and, "Intel Hex" mode, which allows EPROM data to be transferred in ASCII format under control of modem software. In this last mode, the DUMP command causes the EPROM contents to be sent to the computer where applicable modem software may be used to save the data in memory (and/or disk). The PGM command will cause the EPROM to be programmed with the data that follows the command. Baud rates of 1200 to 19200, XON/XOFF, and Ready/Busy protocols are supported. HDOS and CP/M software is available in any format (8 inch or 5.25 inch, hard or soft sectored). Specify operating system and disk format when ordering. Contact the vendor for details on this and other programmers.

Vendor: Ross Custom Electronics
1307 Darlene Way, Ste. A-12
Boulder City, NV 89005
(702) 293-7426

Prices: IntelliBurner: $269.00
RS-232 Interconnect Cable: $9.00
Add $2.00 S&H to order

SPOOLDISK-89 Now Available for H/Z-89/90 Computers

FBE Research Company has announced that the SPOOLDISK-89 is now available, thanks to the arrival of the Intel 8031 microcontroller used in the design of this product. SPOOLDISK89 is an "electronic RAM disk"



with 128K of RAM memory. The RAM acts as an electronic disk of 128K in size. Significant speed improvement can be realized from either CP/M or HDOS using the RAM as a temporary electronic disk, eliminating delays due to rotation speed, head movement, or startup time. The card includes a parallel printer port compatible with Epson and other "Centronics Standard" parallel interface printers. Up to 127K of the RAM may be used to queue document files for printing, allowing the user to reset his computer and boot another system. In addition, the card also contains a 64K FIFO conventional printer buffer (optionally enabled). Software for both CP/M and HDOS support is provided. Specify disk type when ordering. Contact the vendor for more details.

The vendor also offers other products and has announced the H89PIP dual port parallel interface card for H/Z-89/90 computers. One port is buffered for use as a "Centronics" printer interface. The second port may be programmed for either input or output through the 8255 programmable parallel interface chip. Contact the vendor for full details and information on other available products.

Vendor: FBE Research Company, Inc.
Box 68234
Seattle, WA 98168
(206) 246-9815 (6:00 PM to 10:00 PM, Pacific Time Zone)

Prices: SPOOLDISK-89: $395.00; $315.00 in qty. of five or more.
H89PIP: $50.00; $40.00 in qty. of five or more.

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

CUT ALONG THIS LINE

HUG MEMBERSHIP RENEWAL FORM

When was the last time you renewed?

Check your ID card for your expiration date.

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

Name _____

Address _____

City-State _____

Zip _____

REMEMBER - ENCLOSE CHECK OR MONEY ORDER

CHECK THE APPROPRIATE BOX AND RETURN TO HUG

	NEW MEMBERSHIP RATES	RENEWAL RATES
US DOMESTIC	$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.

Index of Advertisers

Heath/Zenith Users' Group

Hilltop Road

Saint Joseph, Michigan 49085

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

Volume 5, Issue 5

POSTMASTER: If undeliverable, please do not return.

885-2052