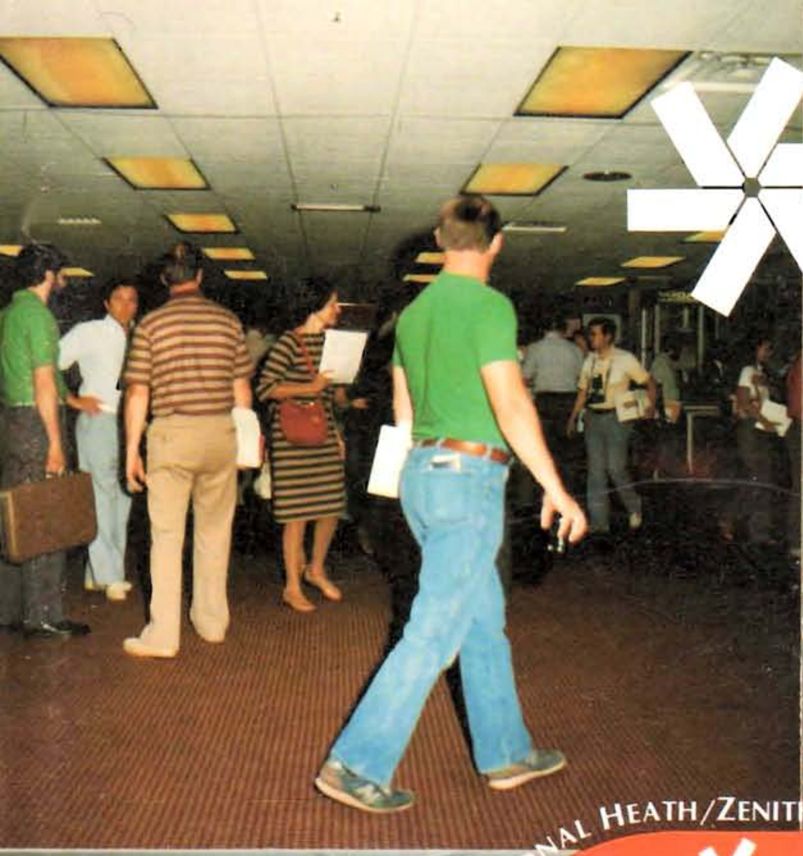
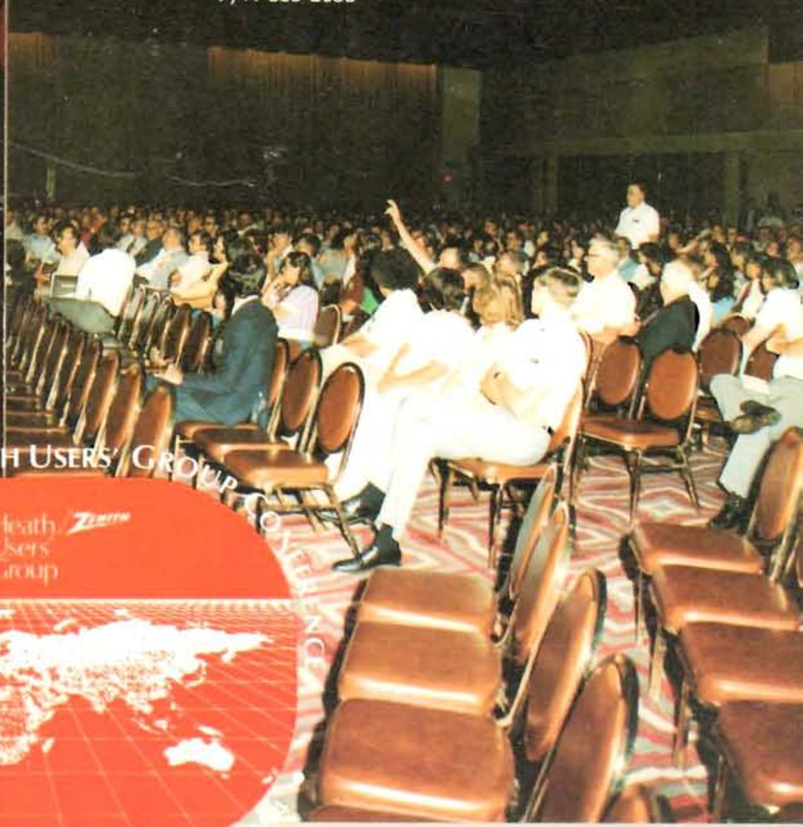
$2.50

REMark®

Volume 5, Issue 6 • June 1984
P/N 885-2053

INTERNATIONAL HEATH/ZENITH USERS' GROUP CONFERENCE
Heath/Zenith Users' Group
Saint Charles, Illinois • July 27, 28 & 29, 1984

Official magazine for users of HEATH ZENITH computer equipment.

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

PRICE BREAKTHROUGH

Super Sale on New Disk Drives

Starting at ~~$169.00~~! NEW LOW PRICE

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

40 or 80 Tracks — Single or Dual Head — New 3½" Drivette™
Our Disk Drives are Capable of Single and Dual Density Operation
The **NEWEST** Technology Capable of Operating on Most Popular Computers

Drive a Hard Bargain!!™ For your TRS/80, Color Computer, IBM, Apple, Franklin
5 M.B.-20 M.B. Complete Systems......from $999.95

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

SAVE!! PLEASE CALL FOR OUR MOST CURRENT PRICE REDUCTIONS.

TOLL FREE ORDERING 1-800-343-8841

GENERAL AND TECHNICAL 1-617-872-9090

Disk Drives (0123) TRS/80-IBM-Apple - TI Franklin-Max/80-LNW ..... ◄
Model I/III/IV Upgrade (Disk Drives - Memory).................... ◄
Printers — Daisywheel/Dot Matrix .... ◄
Percom Double Density Controller (Model I) .... ◄
Color Computer Printer Interfaces .... ◄
Disk Drive Operating Systems .... ◄
Repair Services Now Offered — FAST Turn-a-Round................ ◄
Apple/Franklin Compatible Add-On Drives with Case & Cable.......... ◄
Diskettes in Library Cases .... ◄

CALL TOLL FREE FOR NEW PRICES

DISK DRIVE CASES AND POWER SUPPLIES .... starting at $59.95
Printer **Buffers** 8K to 512K .... starting at $143.95
Holmes Model I/III Speed-up Mod .... starting at $90.00
Cables — Printer/Disk Drive .... starting at $23.00

Warranty on Disk Drives — 6 Months to 1 Year

SOFTWARE SUPPORT, INC.

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

DEALER INQUIRIES INVITED.

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

CANADA
MICRO R.G.S. INC.
751, CARRE VICTORIA, SUITE 403
MONTREAL, QUEBEC, CANADA, H2Y 2J3
Regular **Tel. (514) 845-1534**
Canadian Toll Free 800-361-5155

Service! Service!

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

TRS/80 Registered Trademark Tandy Corp. IBM-PC Registered IBM Corp. Apple Registered Trademark Apple Computer Corp. Franklin Registered Trademark Franklin Corp. Max/80 Registered Trademark Lobo Int.

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

Heath/Zenith Users' Group

HUG Manager .... Bob Ellerton
Software Engineer .... Pat Swayne
Software Developer and HUG Bulletin Board ...... Jim Buskziewicz
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.

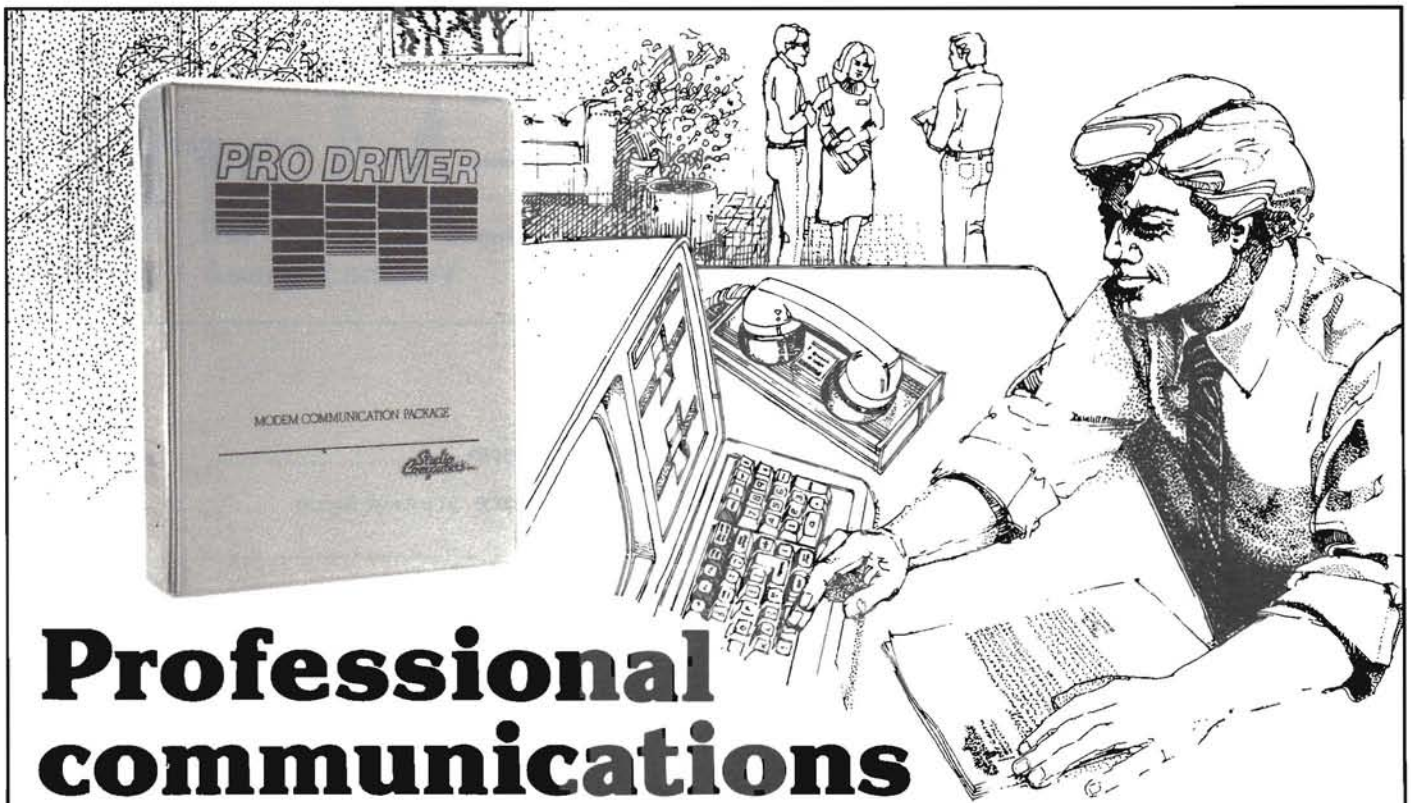
Send Payment to: Heath Users' Group
Hilltop Road
St. Joseph, MI 49085

616-982-3463

REMark®

Volume 5, Issue 6 • June 1984

on the stack



Professional communications

Pro Driver is a new professional communications package for Z100 computers using the ZDOS operating system. It allows you 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. Pro Driver lets you perform operating system commands such as renaming files, deleting files, directory listings or resetting disks. The clear, uncluttered menu-driven displays make using Pro Driver a pleasure.

Here are just some of Pro Driver's features:

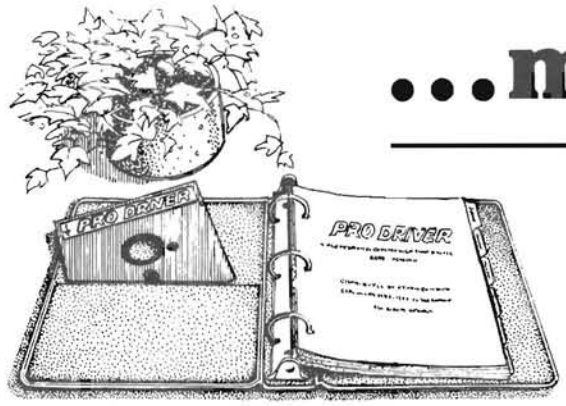
- Menu-driven request and control options
- Direct support for Hayes Smartmodems
- Report Generator for detailed tracking of calls
- XMODEM, Compuserve B and X/ON X/OFF protocols
- 32 User-Definable Automatic Logon Sequences
- Printer Trace function for hardcopy of all data
- User-Definable Control Sequences within Pro Driver
- Selectable Baud rates from 45 up to 19,200
- Configurable entry and exit foreground/background colors, cursor modes and keyclick options
- Help files for quick assistance on Pro Driver functions
- Detailed documentation of all functions

Pro Driver is a professionally designed package from top to bottom. Simple to operate even for the novice, yet with many features an experienced computer user would demand. For example, the 'Reporter' feature not only logs all calls made and the length of time per call, but gives you the option of outputting your report to a printer, a disk file or the video console. And, you can have that report sorted by telephone number or dates! Imagine being able to quickly check at the end of a week, month or even year just how long you spent connected to any specific time-sharing or bulletin board service. A great way to verify the connect-time bills they send to you!

Pro Driver comes with over 180 pages of documentation and pictorial screen displays, but don't let that scare you. Pro Driver is easy to operate! All major sections are organized with tab dividers for quick locating. And a complete index is also included.

The price? Just $49 for the entire package with all of the above features and more. Pro Driver may be purchased at our retail store location, by mail (please add $2 for shipping) or check your local Heathkit Electronic Center for availability.

Pro Driver — the "professional" communication interface for your Z100 computer.



...made simple!

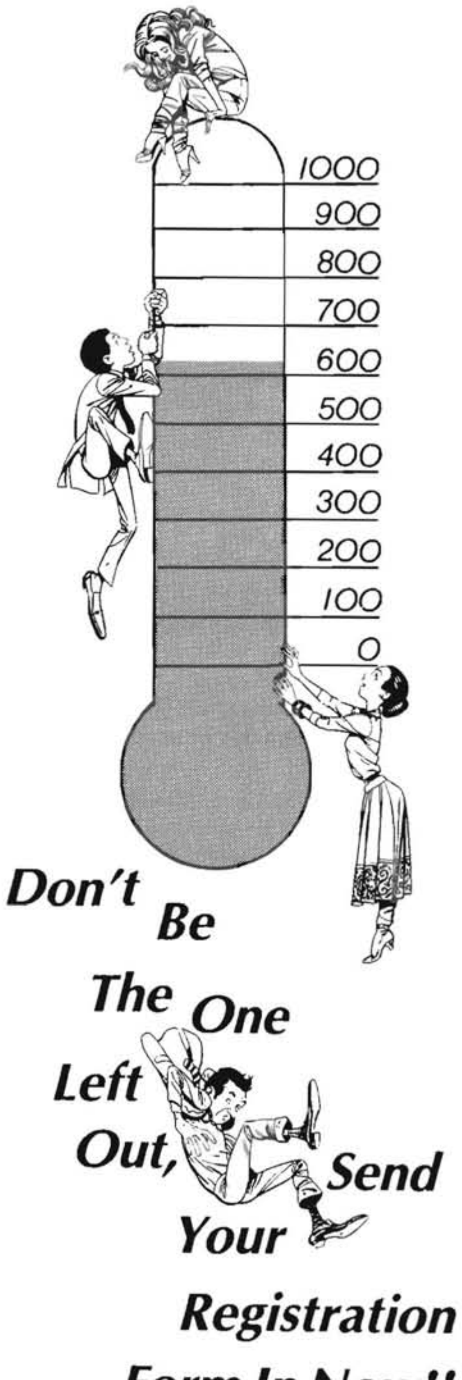
Studio Computers inc. | 999 South Adams
Birmingham, MI 48011
Phone (313)-645-5365

ZENITH data systems

Call or write for our complete catalog, or additional information on Pro Driver.

Be There!

International Heath/Zenith Users' Group Conference
Heath/Zenith Users' Group
Saint Charles, Illinois • July 27, 28 & 29, 1984


1000
900
800
700
600
500
400
300
200
100
0
Don't Be The One Left Out, Send Your Registration Form In Now!!

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.

Next Month .....

The International HUG Conference

INTERNATIONAL HEATH/ZENITH USERS' GROUP CONFERENCE
Heath/Zenith Users' Group
Saint Charles, Illinois • July 27, 28 & 29, 1984

Get ready for some computing fun!

Margaret Bacon
HUG Secretary

The International HUG Conference is just around the corner so we though it would be a good idea to review the events that are scheduled for the weekend of July 27, 1984, at Pheasant Run Resort. This year promises to be very exciting for those attending the Conference or those of you that just want to get away for a great vacation. Pheasant Run is equipped with everything from a dinner theater to a myriad of sports activities. And, what you can't find at the Resort itself, you will be able to get to within a few minutes of Pheasant Run.

So, what's going on at the Conference?

The Vendor Exhibit Area is always one of the most popular areas of the Conference. This year we have another strong showing from the vendors we have come to know as supplying some of the best user support for any computer available on the market. As of this writing, the list continues to grow. We have included the vendors that have come onboard so far elsewhere in this article. You will recognize many of the names and you will see that some new faces are going to be there with some exciting products never shown at the Conference before. Many of the vendors have donated prizes in the form of software or hardware to be given away Saturday evening. We have counted at least 77 prizes so far, with promises of even more prizes to come.

The Heathkit Electronics Centers (VEC) will be represented this year in a separate room next to the Vendor Exhibit Area. As always, reliable sources indicate that some great buys are going to be available. Also, our source tells us that there will be "a lot of non-product stuff". (!!!!?) So, it might be a good idea to pack your wallet, purse, suitcase, etc. with that green stuff and make sure you bring those funny plastic cards. The Heath Stores have always come through with excellent bargains during the two previous conferences and, this year is no exception from the information we have been able to come up with to date.

By popular request, we will again have bulletin boards available for the following purposes:

Wanted/Trade - This bulletin board will be available to those of you wishing to purchase or trade items.

For Sale - This bulletin board will be available to those of you wishing to sell computer related hardware or software.

General Messages - This bulletin board will be available for general messages of any nature.

These bulletin boards are reserved for users only. Commercial ads should not be placed on these boards. Special forms will be made available for the various boards. We ask that only ads appear for personal equipment and that the ad only be removed by the individual who placed it there. Your cooperation is appreciated.

How about the schedule?

Friday, July 27, 1984, starts with early registration at 1:00 pm. The Vendor Exhibit Area will open to all users from 4:00 pm to 9:00 pm. Sessions (outlined elsewhere in this article) will begin at 6:00 pm in both the Tower Suites and in the Governor's Hall. Friday's activities are being held to a minimum so that each of you will have the chance to explore Pheasant Run and the Vendor Exhibit Area. There are plenty of fine places in the Resort to catch a quick meal or full dinner.

Saturday, July 28, 1984, begins with the Grand Opening Breakfast served in the Governor's Hall. Our Keynote speaker will be Mr. Phil Cole, Director, Heathkit Product Development. Phil's subject will be "Heathkit Computer Products - What's Happening?". Phil is an exciting speaker with a good topic for those of us wondering what Heathkit computer products might be like in the future.

Following Saturday's Grand Opening Breakfast, we will take a little break while the exhibitors prepare for the busy day. Some of the talks will begin however, at 10:30 am in the Tower Suites. One of the featured talks will be Multitasking Operating Systems by Barry Watzman. Barry is well known to Heath users. Later in the afternoon, Barry will present a session on 16-Bit Programming for 8-Bit Programmers. He has suggested that attendees to his session be well versed in assembly language to get the most from his talks.

Saturday's sessions will feature Tom Dornback, Vice President, Zenith Data Systems Software Development. Tom's talk will actually cover three separate subjects including a review of currently available Zenith software products, an overview of Zenith's software direction, and finally a brief non-technical introduction to the new Z-100 PC's. Tom has requested that we let you know he will be open to questions and answers after his presentation.

What ever happened to HDOS 3.0?

This question is probably one of the most popular received at our previous conferences. Afternoon sessions may give us a long awaited answer to this menacing question! Finally!!! Our understanding is that we will get a chance to hear some of the actual specifications for HDOS 3.0 from some of the individuals selected to prepare this latest update to the product. Come prepared with some good questions!

As the events of Saturday move on toward the closing of the Vendor Exhibit Area, we will either be worn out or ready to party. And, party we will (one way or another)! Saturday evening will bring all attendees together at the International Heath Users' Group Cocktail Party. The activities scheduled for the party include bunches of food, casual dress, prizes and lots of fun. The Atrium, a portion of Pheasant Run, is a pleasant atmosphere where HUG types can get together and discuss bits, bytes, the kids, etc. over some great snacks.

What can we expect from Zenith, Heath, and VEC?

Almost a tradition, prizes donated from Zenith Data Systems, Heath Company, and the Heathkit Electronics Centers have been outstanding. This year, all users and vendors will be qualified to win these prizes. I know! What are they? Well, let's put it this way:

Bill Johnson, President of Heath Company, says "Let's give away the latest in computer kit technology available."

Joe Schulte, President of VEC (Heathkit Stores), "Ditto."

Don Moffet, President of Zenith Data Systems, says "I think one of these new things with the Winchester would be nice."

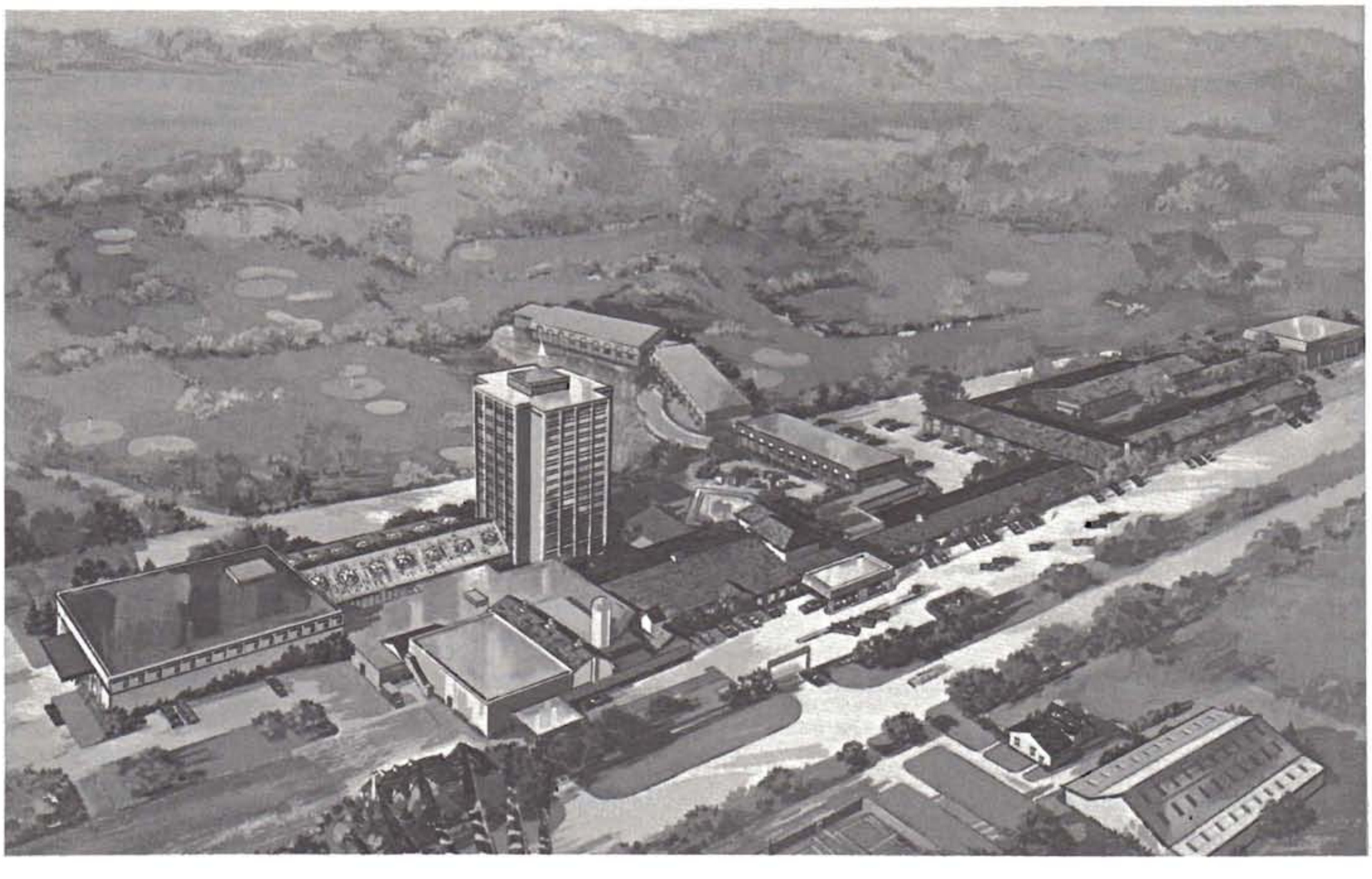
And there you have it! It would appear that some lucky individuals are going to come away with a new PC series computer to play with! Don't miss the Saturday night gathering!

We have received many requests in the past to have a casual gathering place where HUG types can get together to chat. The Atrium is just the place. If you want to cool off, you can take advantage of the pool or just sit around and enjoy the company of friends and new acquaintances. If the crowd gets to be too much, you can slip away to some of the other facilities available to you during your stay at Pheasant Run.

Talks will begin early on Sunday, July 29, 1984, so that you can squeeze in that last little piece of information before departing or going out for a good round of golf (?!). The Vendor Exhibit Area will be open from 9:00 am to 3:00 pm. All remaining Conference activities will end at 3:00 pm.

It looks like another good year for the International HUG Conference. The atmosphere provided by Pheasant Run, the company of fellow computer buffs and the continued support of the Heath-/Zenith vendors promises to make this event even more exciting than previous gatherings of HUGgers in Chicago. So, be sure to plan your vacation early. Bring the entire family! And, remember, we all hope to see you soon at the International HUG Conference.

Note: This article presents the schedule for the Conference as it is now planned. Changes that occur between now and the actual date of the Conference will be announced in the program that you can obtain when registering with the Heath/Zenith Users' Group at Pheasant Run. Any last minute changes will be announced during the Conference Sessions.



1984 International HUG Conference Discussion Group Schedule

Friday, July 27, 1984

Time	Room	Subjects	Speakers
6:00 pm	Turquoise	Operating Systems Extensions	Pat Swayne
6:00 pm	Ruby	Computer Chef	Susan Hayes
6:00 pm	Govnrs Hall	MS-DOS Operating System	Brian Barnes
6:30 pm	Ruby	C Lanugage	Walt Bilofsky
7:30 pm	Govnrs Hall	H8/H89 Hardware	Bruce Denton
8:00 pm	Turquoise	Introduction to Computers	Ron Hackney
8:00 pm	Ruby	Z-100 Graphics	Dale Wilson

Saturday, July 28, 1984

10:30 am	Ruby	Multitasking Operating Sys	Barry Watzman
10:30 am	Sapphire	Public Software	Bob Todd
10:30 am	Turquoise	Robotics	Ron Johnson
12:30 pm	Ruby	Disk Care	DYSAN
12:30 pm	Turquoise	Introduction to Computers	Ron Hackney
1:00 pm	Govnrs Hall	HDOS 3.0 Operating System	W. Parrott & D. Carroll
2:30 pm	Ruby	16 Bit For 8 Bit Programmers	Barry Watzman
2:30 pm	Govnrs Hall	Zenith Software/Intro to Z150	Tom Dornback
3:00 pm	Turquoise	Operating Systems Extensions	Pat Swayne
3:30 pm	Govnrs Hall	Software Consultants Panel	5 on Panel
4:30 pm	Ruby	CP/M 80-85 CP/M 86 Comparison	Bill Adney
4:30 pm	Turquoise	Z-100 Graphics	Dale Wilson

Sunday, July 29, 1984

8:30 am	Govnrs Hall	Software/Hardware Panel	3 on Panel
8:30 am	Sapphire	Public Software	Bob Todd
8:30 am	Turquoise	Operating Systems Extensions	Pat Swayne
8:30 am	Ruby	Introduction to Computers	Ron Hackney
10:00 am	Govnrs Hall	Z150 Hardware	Mark Foster
10:30 am	Turquoise	C Language	Walt Bilofsky
10:30 am	Ruby	Z-100 Graphics	Dale Wilson
11:30 am	Govnrs Hall	Z100 Hardware	Mike Cogswell
11:30 am	Turquoise	Computer Chef	Susan Hayes
1:00 pm	Close		

1984 International HUG Conference Exhibitors

Aid Data Systems, Inc.
CDR Systems, Inc.
Cherry Engineering
Cleveland Codonics, Inc.
Colorworks
Computer Consultants to Business
D-G Electronic Developments Co.
Dysan Corporation, CE Division
First Capitol Computer
Fusaro Associates, Inc.
Generic Software
Groffics Innovative Graphic Software
Hampton Business Machines Co.
Hilgraeve Inc.
Hoyle & Hoyle Software Inc.
Husker Systems of Nebraska, Inc.
Jay Gold Software
MPI

International Heath/Zenith Users' Group Conference
Heath/Zenith Users' Group
Saint Charles, Illinois • July 27, 28 & 29, 1984

Maps for Micros
Microservices
Micro Widget Works, Inc.
Micro World Publishing
Newline Software

New Orleans General Data Services, Inc.
Quikdata Computer Service, Inc.
Redwood Development
S & K Technology, Inc.
Sextant Publishing Company
SigmaSoft and Systems
Soft Shop
Software Wizardry, Inc.
Studio Computers, Inc.
Sunflower Software, Inc.
Technical Advisors, Inc.
Technical Micro Systems, Inc.
The MSW Company
The Software Toolworks
Viking Software
Barry A. Watzman
Zeducomp Incorporated
ZPAY Payroll Systems

Schedule of Events

Friday, July 27, 1984

1:00 pm to 9:00 pm	Registration Booth Open
4:00 pm to 9:00 pm	Vendor Exhibit Area Open
6:00 pm	Conference Sessions Begin

Saturday, July 28, 1984

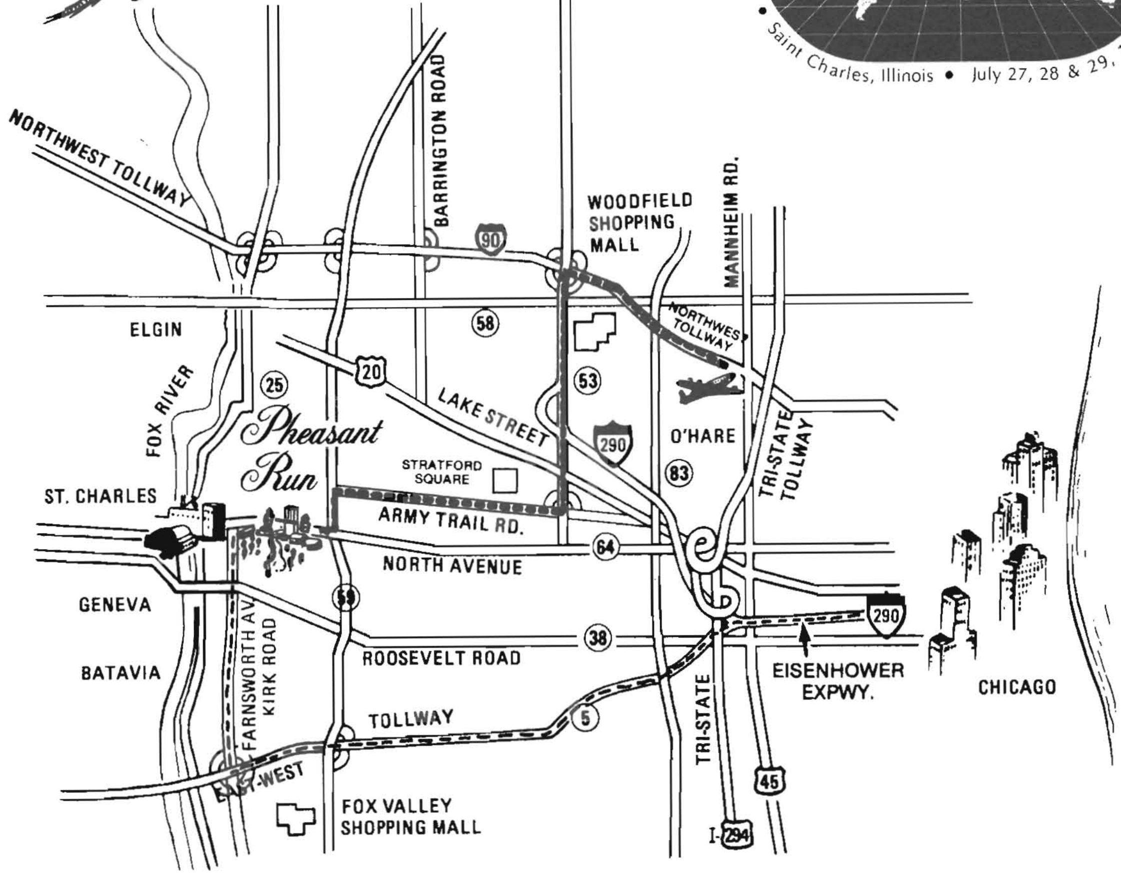
7:30 am to 8:30 am	Registration Booth Open
8:30 am to 10:00 am	Grand Opening Breakfast (Governor's Hall)
10:00 am to 5:00 pm	Registration Booth Open
10:30 am to 11:30 am	Vendor Exhibit Area Open (Vendors Only)
10:00 am	Conference Sessions Begin
11:30 am to 6:00 pm	Vendor Exhibit Area Open
12:00 pm to 3:00 pm	Buffet Lunch (New Orleans Ballroom)
8:00 pm	Cocktail Party and Prizes (Atrium)

Sunday, July 29, 1984

8:00 am to 1:00 pm	Registration Booth Open
8:30 am	Conference Sessions Begin
9:00 am to 3:00 pm	Vendor Exhibit Area Open
3:00 pm	Close of International HUG Conference

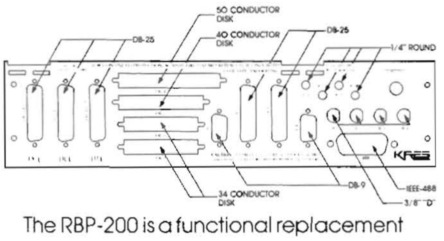
IT'S SO EASY TO REACH

International Heath/Zenith Users' Group Conference
Heath/Zenith Users' Group
Saint Charles, Illinois • July 27, 28 & 29, 1984



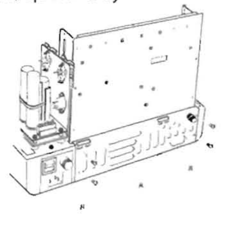
PHEASANT RUN is just 50 minutes from O'Hare International Airport. Take the Northwest Tollway to Route 53, continue South on Route 53 (following the signs to Joliet) to Army Trail Road. Go West on Army Trail Road for 13 miles to Route 59. Take Route 59 South for 3 miles to Route 64, West on Route 64 for three miles and you're there.

Bus service is provided to and from O'Hare Field. Private planes, including small jets, may land at adjoining DuPage County Airport.



The BACKPLATE a neater solution from KRES

DB-25	Printers, modems, parallel ports, RS-232, general purpose
34 Conductor Disk	5¼" disk drives (H-77, H-37 and similar cables)
40 Conductor Disk	H-47, H-67 and similar cables
50 Conductor Disk	Standard 8" drive cables. A must for Magnolia and CDR users
DB-9	Joysticks, color monitors, analogue signals A/D, D/A, RS-422
1/4" Round	Audio output (music, sound generators, speakers) microphones, BSR system X10, control
3/8" "D"	Video, color monitors, analogue signals A/D, D/A, any application requiring shielded cable
IEEE-488	Instrument control and laboratory communication



The RBP-200 is a functional replacement for the existing backplate on the H/Z89-90. It installs in minutes with just a screwdriver. For very old H88s an adaptor is available to allow use of the RBP-200 Backplate.

RBP-200 **$35.00** plus shipping
California residents add 6% sales tax

KRES ENGINEERING • P.O. Box 17328 • Irvine, CA 92713
(818) 957-6322

BUGGIN' HUG

Comments On the ZD Program For the Z-100

Dear HUG,

Finally you came through with something for us Z-100 owners with Jeff Kallis' ZD program. I was pretty bored with the only other program that was ever presented, particularly since I did not want to get rid of my key click. I am speaking about assembly language programs, of course. Mr. Kallis' program worked right after I had gotten rid of all my boo boos. However, I have a bone to pick with him and several other people who have written books and programs for ZDOS. It is not necessary in my estimation to clutter your disk with LST, CREF, and MAP copies until the assembly is completed. In this respect I agree with Pat Swayne. It may be necessary to get them later to help debug the program, but when attempting to assemble the program, they just take up time and use up space on the disk.

I commend Jeff Kallis for his very useful program. Now I hope somebody grabs it and expands it to make a catalog file so that ZDOS disks can be cataloged.

Warren F. Earles
2919 Flores Ave.
Laredo, TX 78040

A Problem With ZBASIC

Dear HUG,

Being fairly new to computing, I have come across a problem in using ZBASIC that no one locally has been able to help me solve. I would like very much to be able to write educational programs for my H-100 computer. The problem is with ZBASIC's random number generator. I am unable to write a program that will automatically reseed the RANDOMIZE function each time through, and still produce new random numbers. Manually reseeding the numbers each time is difficult for second grade handicapped students to understand and it limits my ability to select random reenforcers for the children. Is there anyone in HUG that has figured out a way to circumvent this need to manually reseed the number generator? I would appreciate hearing from you and I would be happy to pay for your postage or copying expenses if necessary.

On a different note, I too made an error while attempting to make backup copies of my Peachtext 5000 and erased the first disk. However, unlike Mr. Bartram (Buggin' HUG Vol. 5, Issue 3), I first called the Peachtree service number. They told me to return my erased disk and they would send me a new one free of charge. They did and I felt the turn-around time to be quite reasonable. I guess the lesson is that if you have problems with Peachtext, it might be of benefit to call first.

David Harvey
168 G Chilvers Rd.
Chehalis, WA 98532

ED) Dave, try using the seconds portion of TIME$ under ZBASIC to gain your seed. This would be a truly random number from 00 to 59.

Anybody Out There Have Customization Notes For WordStar Ver. 2.26??

Dear HUG,

For some time now we have attempted to obtain the Customization Notes for WordStar version 2.26. MicroPro has discontinued them and they are not available from Heath/Zenith or any dealer we have contacted.

If anyone can supply the above Customization Notes, please contact Chuck Hansen at Bendix, Electric Power Division, Rt. 35, Eatontown, NJ 07724, (201) 542-2000 ext. 460.

Thank you for your help in this matter.

Chuck Hansen
Bendix
Electric Power Division
Rt. 35
Eatontown, NJ 07724

Another View On Peachtext 5000

Dear HUG,

I am very glad that I purchased Peachtext 5000 before some of the comments came out in REMark. In fact, I still suggest purchasing the product, for it is a very good product. Everybody has the right to his own opinion, and I have one of my own that I think should be voiced.

The comments made about Peachtext 5000 are true, but what happened to the positive aspects?

Support from Peachtext is far better than any other software I have purchased. I know because I bug them all the time and get very good response from them, or to say it politely, they answer the questions.

If I were to buy based on all the features that are available today for computer software, I might have decided to buy an eighteen wheeler to run to work, rather than a pick-up truck. Maybe even a Cray computer rather than a Heathkit. I even noticed your IBM 370 does not know what to do with a two year subscription to REMark.

Based on the price from Heathkit for Peachtext 5000, and what it can do, it's a good deal. The documentation and lessons, plus the methods presented by Peachtext go a far way beyond most of the other software that I have seen in 1983 or 1984.

You cannot be everything to everybody.

Bert Rathkamp
5950 Park Road
Cincinnati, OH 45243

Another Answer to the Dreaded HT Bug

Dear HUG,

I am writing concerning L. T. Scotney's letter on page 66 of the April issue of REMark about "The Dreaded HT Bug". I, too, was bitten and quite perplexed when I began experimenting with my Epson FX-80 printer. After a couple of hours of research into the problem and a phone call to the Heath support personnel, my suspicions were confirmed.

There is, however, a way around the problem for those printers that can handle escape codes greater than dec 127 (hex 7F). Add 128 (hex 80) to the 09 and send the printer a dec 137 (hex 89). It works on the Epson, and if your printer handles codes the same way, it should work for you. It's not a permanent cure for his bite, it just relieves the sting.

Frank D. Torchia
16 Ramblewood Drive
Gales Ferry, CT 06335

Goodies For the '89 User

Dear HUG,

Just when I was beginning to think you were leaving the HDOS H89 users' out in the cold, along came an article in REMark that was just what the doctor ordered. I refer to "Device Drivers and Communication" in REMark March 1984. I just completed interfacing a Brother electronic typewriter to my 89 and used the article to construct a driver. Since I already did the cable modification for handshaking signals, all I wanted to do was eliminate the spurious characters that appear at the beginning of a printout. I followed the instructions but when the program assembled, I was left with (2) "ERRNZ" errors that had me puzzled (the same 2 mentioned in Buggin' HUG of the same issue - unfortunately I didn't read that letter in time). I discovered that when I used PIE (Software Toolworks) to modify the program, the errors would pop up. But if I used Heath's "EDIT" as supplied on HDOS 2.0, the program would assemble without error. Anyway it works now and I'm happy. All that's left is to modify the driver to stop at page breaks so I can load paper. Any suggestions on how to do this for a novice assembly programmer?

Jim Doherty
832 S. Beechwood
Rialto, CA 92376

Another Patch For "MAILPRO"

Dear HUG,

I thoroughly enjoyed Tom Best's article "Bells and Whistles For MAILPRO" in the April 1984 REMark. As Tom states, MAILPRO is an excellent value at a low price. His patches do add some nice new features to the program. I guess my issue of MAILPRO was an early version as I found that Tom's line numbers did not mesh in properly with those of my program. I made suitable adjustments and a couple of minor modifications and agree that it is a nice enhancement. Tom did not mention any patches to the "DELETE" subprogram of the "UPDATE" module, but one is necessary to keep the records accurate regarding number of items added. When deleting, each deletion must reduce the number of items added and items in the data base. A statement Y=Y-1 is required, and in my case, I called it Line 505. It immediately follows the GOSUB 700 in the DELETE code.

REMark is a great service! Keep up the good work! REMark and Sextant complement each other perfectly, and for us HUGgies, is a perfect combination.

S. K. Magee
234 Oak Street
Audubon, NJ 08106

How Compatible is "IBM Compatible"?

Dear HUG,

This letter is in response to a letter in the March 84 edition of REMark from Tim Johnson. I feel he has pinpointed a major area of confusion for Z-100 users created by the term "IBM compatible". This term gives use to the question: How compatible? What IBM software can I use? And what modifications must be made to the software?

I feel an article that details the extent of compatibility and possibly lists the usable software would be a savior to all of us Z-100 users. Please, give this some consideration.

Andy Jeffrey
S. U. 2275 Williams College
Williamstown, MA 01267

Some Help From the Author of "An Intro to Assembly Language on the H/Z-100" (Jan. 1984 REMark)

Dear HUG,

Recently, a Canadian Z-100 user wrote to me about my article on assembly language programming for the 8088 under ZDOS (REMark, Jan. 84). He had discovered the fact that spacing of characters in the Define Byte instructions can be critical. He had typed the following line:

```
DB      CC_ESC,  'x2,  '$' ;Control string to kill
                           ; keyclick
```

where the correct line was:

```
DB      CC_ESC,'x2,'$'     ;Control string to kill
                           ; keyclick
```

Note the different spacing. The first line will generate MASM error message 50, "Value is out of range". This is all too typical of the kind of error messages generated by today's utility software, and my correspondent couldn't decipher it. Neither could I, until I tried some experiments.

My correspondent also made an extremely valid comment almost in passing. He mentioned that he had difficulty in making use of the article because at first he didn't have the necessary programs on his disk. My article didn't cover this crucial but basic step, and I would like to rectify that now.

First, a few definitions. There are two basic types of assemblers, regular and macro. A regular assembler, such as ASM.COM that comes with CP/M-80 and CP/M-85, operates on symbolic input data to produce machine instructions. It does this by translating the symbolic codes into computer instructions, assigning locations in memory for each successive instruction, or computing absolute memory addresses from symbolic instructions. Most assemblers make provision for including other source code files (by use of the INCLUDE assembler directive, or pseudo instruction). A macroassembler does all that, but in addition, allows the user to preassemble subroutines that can be gathered together into a library for use with many different programs. Microsoft's MASM is an example of this type, and provides extensive library facilities.

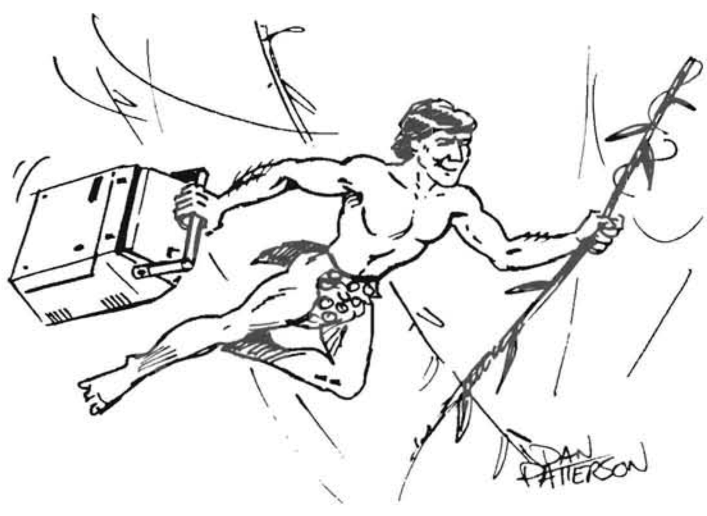
In the original article on the use of MASM, no use was made of the macro feature. Only the INCLUDE pseudo was used. This leads into a discussion of how to use the INCLUDE feature to best advantage. It is possible, of course, to create a SYSGENed disk that contains the assembler, and use the command line to identify what disk contains the files to be INCLUDEd, but this can rapidly become tedious, and assembly fails when the assembler can't fine an INCLUDE file. A better approach is to create a working disk that contains all the files you might need.

There are two ways to create such a working disk. The first, and easiest for those with only one disk drive, is to make up a SYSGENed disk that contains the operating system-related files, MASM, LINK, EXE2BIN, whatever editor you are using, and the minimal INCLUDE files. For ZDOS, you can usually get away with the following:

Vectored to 96 ☞

SWING INTO ACTION

with the new Z-160 portable PC from Zenith!



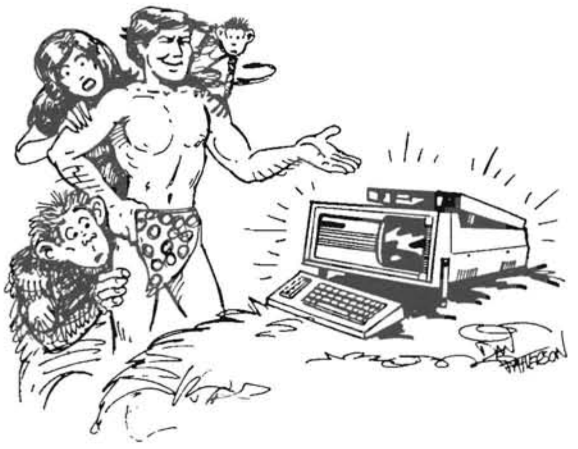
You say you want portability and PC compatibility?
Zenith quality and performance?
You say you want a good price, and you want it now?
Then say Software Wizardry!

The Z-160 is the new portable personal computer from Zenith, and Software Wizardry has them, at a price that will make you give a jungle yell!

The Z-160 is Zenith's hot new PC addition to the Z family, and as usual, they do it better—text and full color graphics without funky add-in cards, up to 320K RAM on the main board, and MS-DOS included!

Don't stampede through the jungle with the Big Blue Elephants; just take a look at these features!

- Fully IBM™ PC hardware and software compatible
  - will run Flight Simulator, considered an acid test
- More standard features
  - 9-inch high resolution amber monitor
  - detachable low profile keyboard
  - four open slots
  - extensive self diagnostics
  - RGB color and monochrome monitor interfaces
  - two serial RS-232 ports
  - Centronics type parallel printer port



  - 128K RAM, expandable to 320K on the main board
  - Total memory expandable to 640K

- Extra features
  - smooth scroll on monitor
  - booting from either drive
  - superior configuration program
  - four selectable display typestyles
  - MS-DOS 2.1 operating system included standard

DID SOMEONE SAY MEMORY?

They say in the jungle that elephants never forget; well, your Zenith won't forget either with RAM chips from Software Wizardry.

Guaranteed first quality, to work in your Z-100, Z-150, or Z-160

ZENITH | data systems
AUTHORIZED SALES & SERVICE

Zenith and Software Wizardry are synonymous with quality, capability, and support. Whether the top of the line Z-100 dual processor or the new PC systems, we know Zenith; so grab a vine, and swing our way!

Software Wizardry
THE MAGIC TOUCH

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

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

A vendor uses a Z-150 to demo Bluebush Chess, a program "for the IBM PC".

Faire Weather Computers

Thoughts on the 9th West Coast Computer Faire

Pat Swayne
HUG Software Engineer

This year I was the lucky HUGgie who got to go out to San Francisco for the annual Computer Faire held there. It had been two years since my last visit, and a few things had changed since then. For one thing, Radio Shack no longer had the large booth at the entrance to Brooks Hall (a prime booth location). Apple had it, with IBM next door. At the Apple booth, a huge replica of a Macintosh computer towered over the crowd (I told them, "That's how big you should have made it".) It was actually a rear projection monitor, and displayed the output from a real Mac somewhere in the booth.

Radio Shack's booth was a much smaller one hidden somewhere in the maze of Brooks Hall. Things never stand still in the world of microcomputers. But the good old Heathkit booth was where it has always been in the Faire, in the Civic Auditorium where folks would probably see it before they ever got to Brooks Hall (which is downstairs). Surrounding the Heathkit booth were the usual "friends of the family", such as Magnolia, Kres, CDR, Sextant, Software Toolworks, and others who support Heath and Zenith computers.

Attendance at the Faire seemed a bit down. Perhaps that is because of some changes made this year. For one thing, the Faire was open for 4 days instead of 3. Another possible factor is that the Faire's founder, Jim Warren, sold the whole works to Prentice-Hall publishers, who put the show on this year. I heard that they encouraged exhibitors to dress up (many used to wear t-shirts), and the show didn't seem quite as "down homey" as before. But I thought it was still a good show, with plenty of goodies to see. The exhibitors were just as enthusiastic as ever, and a few kept their t-shirts.

The dominant system this year was the IBM PC, with IBM compatible software or hardware everywhere you looked. Apple came in second, with quite a lot of Apple II stuff still around. At our own booth, the new Z-150 and Z-160 were on display for the first time in a major show, and there were handouts on the kit versions that drew quite a bit of attention. IBM'ers were very impressed with the new machines, and with the price of the kit version. I suspect that it will do well. Many folks brought over IBM software to try out on the 150's, and I don't think that there was anything it could not run. It is very compatible, and will even boot PC-DOS and the IBM version of CP/M-86. However, it will not run IBM's BASIC. Could it be that they use an environment check similar to that used in the old version of H89 MBASIC that would not run on Z-100's? What ever the case, Zenith's BASIC for it works the same, and was used to run BASIC programs that people brought over to try out.

I have a few words of commentary on the H/Z-150. Some HUGgers have expressed concern that it will replace the H/Z-100. I see it as more of a replacement for the H89 than for the 100. For one thing, the kit 150 costs about the same as the kit 89 did a few years back. It is also an excellent hobby machine, with a whole lot of boards and software already available from scores of vendors for the hobbiest to play with. I view the 100 as more of a professional machine than a hobby machine, even in kit form. The 150 is either a hobby machine (kit) or just what they call it, a "personal computer".

The H/Z-150 and H/Z-160 were designed in a way that makes them easily mass producible, and that design lends to their appeal as hobby machines. Instead of having a mother board with most of the functions on it as in the 89 and 100, the 150 and 160 have a simple bus board (like an H8 or old style S100 machine) with a CPU board, a disk controller board, a video board, and a RAM/IO board plugged into it. This design lends itself to the addition of new video boards, CPU boards, etc. In other words, an easy machine to "tinker" with, and to expand. How about a CP/M board, similar to the "Baby Blue", but 100% H89 compatible?

For years now we have had a de facto standard operating system for 8-bit computers (CP/M) that allowed wide spread distribution of software. It is far from being the best operating system, but it is a standard, and that says a lot. Now that 16-bit computers have arrived, we have not only a standard operating system (MS- DOS), but a hardware standard (IBM) as well. Like CP/M, they are both far from being the best available, but they are standard. The Heath/Zenith implementation of that standard is, in my opinion, the best there is. Evidence of that could be seen at the Faire. Even though the 150 had only been available for sale a few weeks, I saw a few at some software vendors' booths being used to demo software written for IBM PC's. Vendors I talked to were enthusiastic about the 150.

I guess that's enough commentary on that subject. The (3) 150's and (1) 160 were not the only things we had in the Heathkit booth. We also had (3) Z-100's, and many IBM types who came to look at the 150's were ushered over to a 100 to show them some "real" graphics. We had the HUG game disk (885-3004-37) on one machine, and the slot machine game was a big hit. I found an independent Zenith dealer with his own booth in another part of the Faire who was also using the HUG disk for a demo.

This year there was a 10 foot extension to the Heathkit booth where some actual construction work on a Heathkit was done. A TV camera and a large monitor (Zenith, of course) set up on a pedestal allowed the crowd to watch the action. In addition to the 100's and 150's in the main booth, there was also one lonely Z-90 running a demo and two HERO robots, one with his clothes off.

One of the best things about working a show such as the Computer Faire is that, in your spare time, you get to see what the other guys are doing. I looked at, and in some cases played with, some of the competitors' machines. I tried drawing with Mac Paint on an Apple Macintosh (using the mouse). My evaluation after that brief encounter is that unless you are an accomplished artist using conventional methods, you aren't going to be able to do much with Mac Paint. I think I could do better with a light pen than with a mouse.

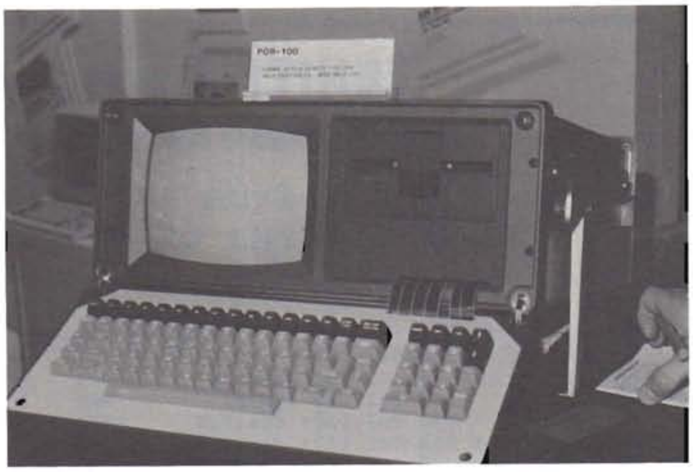
One item that impressed me a lot is DEC Talk, a speech synthesis module from Digital Equipment Corp. for their new computers (Rainbow, Professional). It can translate ordinary text on the screen into speech, can change pitch to sound like several different men's and women's voices, and can even sing. It is so "smart" that if $38 occurs in the text, it sounds out "thirty eight dollars", and if it sees $38 million, it sounds out "thirty eight million dollars". They had no brochures to hand out on it, but took my name to mail me something (which I haven't received yet).

Among the Heath/Zenith support vendors, there weren't many new things that haven't already been mentioned in REMark articles or ads. One pair of items that caught everyone's eyes were conversion kits to convert H-89 or Z-100 computers to transportables. These kits were introduced by Kres Engineering (P. O. Box 17328, Irvine, CA 92713 (213) 957-6322). They allow you to put the "guts" of your computer into a metal box along with a 9-inch monitor, some thin drives, and a hinged cover that houses the keyboard.

The Software Toolworks (15233 Ventura Boulevard, Suite 1118, Sherman Oaks, CA 91403) displayed a new version of their MYCHESS computer chess game that can run on a Z-100 or IBM-type computer. A configuration program lets you set it up for the kind of computer you are using. They gave me a copy of it to try out and demo on the computers in our booth. Since the show I have tested it extensively, and found a couple of bugs. Hopefully they will be fixed before I do a full review (hint!). They also provided a copy of their new MYCALC spreadsheet program. It is the successor to their previous ZENCALC, has several new features, and is still reasonably priced at $59.95. Watch for a review in REMark.

One of the personal highlights of the Faire for me was that I got to meet Dr. Jerry Pournelle who writes the "User's Column" in Byte Magazine. He is one of Heath/Zenith's biggest fans out there in the "real world", and has something good to say about Heath/Zenith equipment in just about every issue of Byte. If you get Byte Magazine and haven't been reading Jerry's column, you are missing the best part of the magazine.

The Faire was, as always, a great chance to see what the rest of the micro world is like. Now I'm looking forward to the HUG Conference, where I can see how the best part of the micro world is doing.



The POR-100 from Kres Engineering. It is a Z-100 installed in a transportable case.

San Francisco Electronics, an independent Zenith dealer who had their own booth, used HUG software for demonstrations.

Some of the many people who visited the Faire. This is the MicroPro booth, where continuous lectures and demonstrations on products such as WordStar were given.

ZENITH Users:
Catch These Bargains!
from
Husker Systems of Nebraska

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

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

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

FINE HARDWARE ACCESSORIES

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

Call for our sale price!

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

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

Call for our sale price!

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

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

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

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

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

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

SPECIFY PRODUCTS DESIRED AND FORMAT   Payment Method: ___Check ___Visa ___MC Send Catalog:___

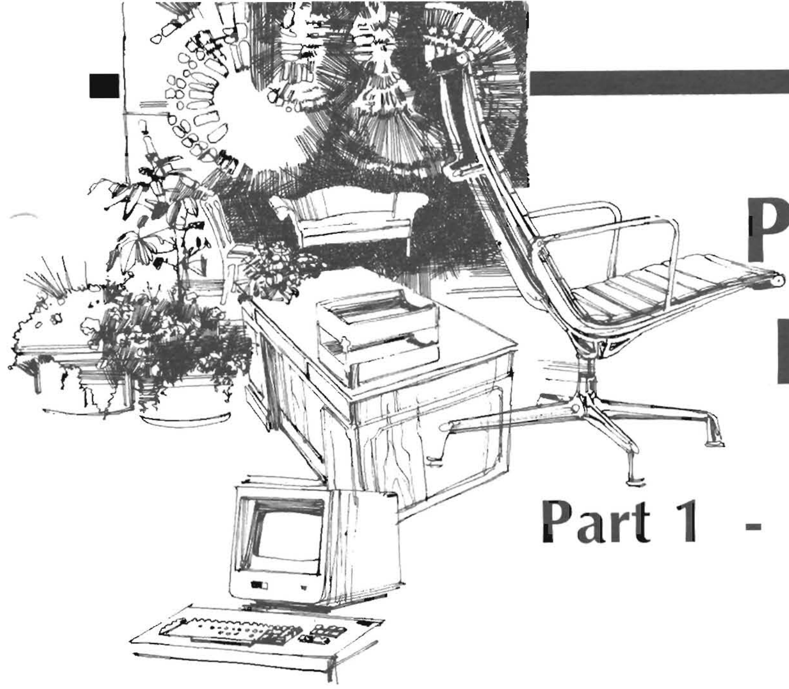
Product(s)_____ Format_____ Price_____   ___(x) ZMAG Easy-Pay Current Date:_____

_____ _____ _____ Name_____ Address_____

_____ _____ _____ City_____ State_____ Zip_____

_____ _____ _____ *Total_____ *Nebraska Residents Add 5% Tax.

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



Practical File Management

Part 1 - A File Handling Program

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

The last series of articles for this column presented random files. We were able to create and sort them, then get the data we stored from them. This article begins a series on the practical use of this information along with other programming techniques we've used in past articles and some new twists to keep life interesting.

We'll build a data base of information. We'll add to it, delete from it, change it, sort it, and then look up any data we want from it or print a chart of the information we've stored. When it's all working we'll look at some techniques to make it smaller (take up less space in RAM when running). As we write the program, we'll use some methods of doing things which we've ignored in the past to make it run faster.

When the series is complete, each reader should have a thorough enough understanding of what we did and how it works to adapt this file handling system to any need which he or she may have. And you'll do it quite easily, too. You'll be able to tailor it to your exact needs. This will be made simple by modular programming. That just means that our total package will be built of small easy-to-understand sub-programs which can be modified with a minimum effort. The WORDS game and biorhythm chart from past articles were built this way.

The program we'll write to demonstrate all of this will let us follow several stocks, bonds, or anything else which changes value periodically. I've chosen this subject as I feel it can be used by many readers, is relatively simple, and can be adapted easily for other uses. The files are not complex and we can get real data on which to try our work.

If you have any questions as we go along, be sure to write. It is imperative that you include a stamped, self-addressed envelope if you need a reply. Sometimes I'm amazed at the volume of mail a column like this can generate. If you have a problem with one of these programs running on your machine, you'll probably want to send a listing to me to check. Most problems I've helped solve were typing errors. I can't overemphasize this. Don't feel bad if it happens as we all make them and overlook them, but proof read the line you're having trouble with. Also check your MBASIC manual. Remember, this work is written in MBASIC Version 5.21 running under CP/M 2.2.03. There are some differences in version 4.82 and the HDOS MBASIC. They are minor but very important and are all covered very clearly in your manuals.

Conversion of this work to Benton Harbor BASIC would be a major undertaking, and unfortunately, time does not permit me to support that.

These articles and the programs are copyrighted by the author and are the property of Applied Computing. All HUG members may use them for their personal non-commercial applications.

On with our program. Figure 1 is a flow chart of the overall program. It is made up of thirteen blocks, each representing one of the sub-programs or modules we mentioned above. The first block, program title, is simple.

```
2   '**********STOKFILE.BAS**********
4   '********DAVID E. WARNICK********
6   '*********COPYRIGHT 1983*********
```

We follow this block with all the screen control characters we'll need. These were explained in issue 46. In order to conserve memory (our finished program will get quite large), we'll only use the variables our program will need, rather than using all of CONTROL2.BAS as we have done in the past. I'll include REMarks here, but suggest that you leave them out of your work. They'll be here for your reference and needn't take space on your disk or in RAM. This is not to say that REMarks should be ignored. They should be included in your programs. Document files explaining the operation of a program should be written, too. The screen controls our program will use are:

```
10 E$=CHR$(27)        'ESCAPE
20 CA$=E$+"Y"         'CURSOR ADDRESSED DIRECTLY
30 ED$=E$+"E"         'ERASE DISPLAY
40 EE$=E$+"J"         'ERASE END OF PAGE
50 EL$=E$+"l"         'ERASE LINE
60 LE$=E$+"K"         'LINE END ERASE
70 SH$=E$+"["         'SCREEN HOLD MODE
80 SS$=E$+"\"         'SCREEN SCROLL MODE
90 GM$=E$+"F"         'GRAPHICS MODE
100 GO$=E$+"G"        'GRAPHICS OFF
```

These lines will permit us all the control of the screen we'll need.

The third block of our flow chart provides for the defining of functions. The only one we have at this time is the function for cursor control as explained in issue 46.

```
200 DEF FNCA$(Y,X)=CA$+CHR$(31+Y)+CHR$(31+X)
```

If you're in doubt about how this works, refer back to issue 46, page 26 and check it out. This is a good time to point out how invaluable those back issues of REMark are. For those of you who joined HUG recently, the HUG price list shows volume 1, 2, 3, & 4 for $20.00 each. They're an excellent buy and an inexpensive way to build your library of back issues. They're full of information. Some individual back issues are also available from HUG or your local Heath Electronic Center or Zenith Computer Dealer. So much for the sales pitch. Back to our program.

As we need to DIMension arrays to hold our data for sorting and handling we'll do that on line 400. This leaves space to define additional functions if required. Then on line 500 we'll erase the screen and enter the hold-screen mode.

```
500 PRINT ED$           'ERASE DISPLAY
510 PRINT SH$           'SCREEN HOLD MODE
```

We are now to the body of our program. There are two ways to proceed. One is top-down development. In this method, all the sections at the top of the program are developed first. When the program branches to another subroutine, that subroutine is substituted for by a temporary "DUMMY" routine which tells us that we got there OK. That way we can test main program control and add all the subroutines later.

The second method of program development is the bottom-up method. Here each subroutine is written, data is developed for it, and it is tested independently of the rest of the program. When all subroutines are working, they are combined and the control portion is written. This is what we did in the last two issues when we worked on the sort programs.

For this application, we'll use top-down development. In this case I think it's easier to follow and will make more sense as we add modules. We'll start with the menu.

A menu, just like those in a restaurant, should present the user with all the options available. It should be appealing to the eye, easy to read, and include instructions for its use. It should also include a provision to handle any operator errors. As for appealing to the eye and easy to read, a CRT layout form is an invaluable aid. This is merely a sheet of paper with 24 or 25 lines each containing 80 character spaces. You can fill in all the information you want to appear on the screen and rearrange it until it's most appealing to you. While this may involve some trial and error on your part, it's better to do it on paper than to try it by programming and re-programming a portion of a BASIC program. Lines 1000 through 1160 provide our menu on the screen. We begin each subroutine by erasing the screen and adding new information. We use cursor positioning instructions to move about the screen and place all the data where we want it.

```
1000 PRINT ED$                    'ERASE THE DISPLAY
1010 PRINT FNCA$(2,39);"MENU"
1020 PRINT FNCA$(4,21);"SELECT THE DESIRED OPTION BY
PRESSING THE"
1030 PRINT FNCA$(5,21);"APPROPRIATE LETTER FROM THE LIST
BELOW"
1040 PRINT FNCA$(8,21);"A - ADD INFORMATION TO THE FILE"
1050 PRINT FNCA$(10,19);"* C - CHANGE INFORMATION IN THE
FILE"
1060 PRINT FNCA$(12,19);"* D - DELETE INFORMATION FROM
THE FILE"
1070 PRINT FNCA$(14,19);"* L - LOOK UP INFORMATION IN
THE FILE"
1080 PRINT FNCA$(16,21);"N - NEW FILE NAME TO BE WORKED
ON"
1090 PRINT FNCA$(18,19);"* P - PRINT INFORMATION FROM
THE FILE"
1100 PRINT FNCA$(20,21);"S - SORT THE FILE"
1110 PRINT FNCA$(22,21);"X - EXIT TO THE OPERATING SYSTEM"
1120  PRINT  FNCA$(25,5);"**** THE FILE BEING WORKED ON
MUST BE SORTED ";
1130 PRINT "TO USE OPTION C,D,L,N, OR P";FNCA$(1,1)
1140 PRINT FNCA$(1,1);"FILENAME - ";F$
1150 PRINT FNCA$(1,45);"FILE ON DRIVE ";D1$;":"
1160 PRINT FNCA$(1,66);"KEY ON DRIVE ";D2$;":"
```

The rest of our menu sub-program is the decision-making process. We'll take an input, test it for any of the possible choices offered by the menu, and direct program execution to the correct subroutine. If our tests for inputs don't find anything by line 1260, we add an error message and go back to the menu for another try.

We'll start our subroutines at line 2000 and add another every 3000 after that. By counting by tens as we do when programming, this allows us 300 lines for each subroutine. You can do a lot in that space so it should be enough. Lines 1170 through 1300 do our decision making for us.

```
1170 M$=INPUT$(1)               'GET AN INPUT
1180 IF M$="A" GOTO 2000        'ADD OPTION CHOSEN
1190 IF M$="C" GOTO 5000        'CHANGE OPTION CHOSEN
1200 IF M$="D" GOTO 8000        'DELETE OPTION CHOSEN
1210 IF M$="L" GOTO 11000       'LOOKUP OPTION CHOSEN
1220 IF M$="N" GOTO 14000       'NEW FILE NAME OPTION
                                 CHOSEN
1230 IF M$="P" GOTO 17000       'PRINT OPTION CHOSEN
1240 IF M$="S" GOTO 20000       'SORT OPTION CHOSEN
1250 IF M$="X" GOTO 23000       'EXIT TO OPERATING SYSTEM
1260 PRINT ED$                  'ERASE THE DISPLAY
1270  PRINT FNCA$(15,11);"THE KEY YOU PRESSED WAS NOT
ONE OF THE MENU OPTIONS"
1280 PRINT FNCA$(13,26);"PLEASE ENTER YOUR CHOICE AGAIN"
1290 FOR X=1 TO 500:NEXT X      'DELAY FOR TIME TO READ
                                 MESSAGE
1300 GOTO 1000                  'BACK TO THE MENU
```

Note on lines 1180-1250 that I've only allowed for capital letters to be input. If you want either upper or lower-case letters to work write these lines as

```
     IF M$="A" OR M$="a" GOTO 2000
```

etc.

We've now got a heading and a menu. We can test this much of the program by providing those dummy routines I talked about a few paragraphs ago. At each place the menu sends execution (line 2000, 5000, 8000, etc.), we'll add one of these. The dummy routine will tell us what option we've selected. Then it will wait for an input. This just gives us time to think about what we've done. Press any key and we'll be returned to the menu at line 1000. We will provide a real exit subroutine at line 23000. Both the data and the key files are closed (I'll explain the use of key files later), the cursor is moved to the home position, and the terminal is put back in the scrolling mode. When we've completed and tested this much of the program, we can replace any of the dummy routines with the final program module. Additionally, if we come up with a better method of performing one of the optional jobs (a faster sort, more convenient add routine, etc.), we just replace that module. It'll be easy to find and test. The dummy routines look like this:

```
2000 PRINT ED$              'ERASE DISPLAY
2010 PRINT FNCA$(11,26);"YOU SELECTED THE ADD OPTION"
2020 PRINT FNCA$(13,26);"PRESS ANY KEY TO CONTINUE"
2030 X$=INPUT$(1)           'WAIT TILL OPERATOR IS READY
2040 GOTO 1000              'BACK TO THE MENU

5000 PRINT ED$              'ERASE DISPLAY
5010 PRINT FNCA$(11,26);"YOU SELECTED THE CHANGE OPTION"
5020 PRINT FNCA$(13,26);"PRESS ANY KEY TO CONTINUE"
5030 X$=INPUT$(1)           'WAIT TILL OPERATOR IS READY
5040 GOTO 1000              'BACK TO THE MENU
```

```
8000 PRINT ED$                'ERASE DISPLAY
8010 PRINT FNCA$(11,26);"YOU SELECTED THE DELETE OPTION"
8020 PRINT FNCA$(13,26);"PRESS ANY KEY TO CONTINUE"
8030 X$=INPUT$(1)             'WAIT TILL OPERATOR IS READY
8040 GOTO 1000                'BACK TO THE MENU

11000 PRINT ED$               'ERASE DISPLAY
11010 PRINT FNCA$(11,26);"YOU SELECTED THE LOOKUP OPTION"
11020 PRINT FNCA$(13,26);"PRESS ANY KEY TO CONTINUE"
11030 X$=INPUT$(1)            'WAIT TILL OPERATOR IS READY
11040 GOTO 1000               'BACK TO THE MENU

14000 PRINT ED$               'ERASE DISPLAY
14010 PRINT FNCA$(11,26);"YOU SELECTED THE NEW NAME
OPTION"
14020 PRINT FNCA$(13,26);"PRESS ANY KEY TO CONTINUE"
14030 X$=INPUT$(1)            'WAIT TILL OPERATOR IS READY
14040 GOTO 1000               'BACK TO THE MENU

17000 PRINT ED$               'ERASE DISPLAY
17010 PRINT FNCA$(11,26);"YOU SELECTED THE PRINT OPTION"
17020 PRINT FNCA$(13,26);"PRESS ANY KEY TO CONTINUE"
17030 X$=INPUT$(1)            'WAIT TILL OPERATOR IS READY
17040 GOTO 1000               'BACK TO THE MENU

20000 PRINT ED$               'ERASE DISPLAY
20010 PRINT FNCA$(11,26);"YOU SELECTED THE SORT OPTION"
20020 PRINT FNCA$(13,26);"PRESS ANY KEY TO CONTINUE"
20030 X$=INPUT$(1)            'WAIT TILL OPERATOR IS READY
20040 GOTO 1000               'BACK TO THE MENU

23000 PRINT ED$               'ERASE DISPLAY
23010 CLOSE #1                'CLOSE THE DATA FILE
23020 CLOSE #2                'CLOSE THE KEY FILE
23030 PRINT FNCA$(1,1)        'CURSOR HOME
23040 PRINT SS$               'SCREEN SCROLL
23050 END
```

The programming presented thus far provides a menu and control features for data handling. Dummy modules are included so that testing can be performed. Enter all the programming lines shown into your computer and save them. Then run the program to test it. When asked for an input, try each of the menu choices. Also be sure to try entries the program is not looking for. This will verify that our error routine on line 1260 is working correctly.

Testing of a new program or sub-program is every bit as important as writing it. We must remember to verify that every desirable feature works as it should and to test for operator errors. There's little worse than spending a lot of time getting a program to do everything we want, then having it crash at a critical time just because we rushed our testing or forgot to make sure it can survive operator errors. You can't prevent everything such as a system reset, but most common errors (OOPS, I pressed the wrong key) are easy to take care of with a little imaginative programming.

That takes care of control and menu functions. Next month we'll begin adding the individual modules we need to work with our files, and we'll look at some simple things we can do to save RAM and disk space, and to help our program execute faster. Keep a backup of this portion of the program. It can be used as the beginning of any file handler you write.

See you next month.

Heath/Zenith Users' Group

USE ALL YOUR SPECIAL FUNCTION KEYS

With WORDSTAR™

WSKEY™: Now you can take the mystery out of WordStar with SKILL DATA's WordStar enhancement, which implements all twenty-one of the H/Z89-19 function/pad keys or all twenty-three Z-100 labeled key commands.
Function key commands are labeled by a twenty-fifth line banner, which can be toggled on and off by you during your session.

With dBASE II™, ZIP™, SuperCalc™

DBKEY™, ZPKEY™, and SCKEY™: Just type your favorite SKILL DATA key command. All function key commands are labeled by a twenty-fifth line banner, which can be toggled on and off by you during your session. Pad keys also function and send multiple key inputs with a single stroke. All previous command key sequences are still available for the old and painful ways.

Just $29.95 for each program (plus shipping charges shown below).
H/Z89-19 diskettes are 5.25" 10HS, require CP/M 2.2.03.
Z-100 diskettes are 5.25" SS; specify CP/M-85 or Z-DOS.

Available in many Heathkit Stores and Quikdata Inc., Sheboygan, WI...or complete and mail this order form with your check or money order today!

Mail to SKILL DATA, P.O. Box 1943, Olympia, WA 98507.

Yes, I want to use all my special function keys! Send me:

☐ WSKEY—keys to WordStar: ☐ for H/Z89-19 ☐ for Z-100	$29.95
☐ SCKEY—Keys to SuperCalc (for CP/M only)	$29.95
☐ DBKEY and ZPKEY—keys to dBASE II: ☐ for H/Z89 ☐ for Z-100	$29.95

For Z-100 diskettes, specify operating system: ☐ CP/M-85 ☐ Z-DOS

Name ____________________

Street ____________________

City ____________ State ________

Zip ____________ Phone ____________

Visa/MC Card #____________ Expires ________

All orders shipped by first class mail. Include $3.00 per order for shipping and handling, $5.00 for non-U.S. orders. Payment by check, money order, or Visa/MasterCard. Allow 2-4 weeks for delivery. 206/352-0669 (evenings).

SKILL DATA, P.O. Box 1943, Olympia, WA 98507

The ILLUSTRATOR

The **ILLUSTRATOR** is a full-featured graphics drawing program for use with the Z-100 pixel graphics (w/wo color) or the H89/H19 IMAGINATOR pixel graphics option. No need for a lightpen.

Features Include:

• Turtle Graphics	• Area Fill	• 64 Colors
• Rubber Banding	• Box Fill	• Color Define
• Line Drawing	• Circle Fill	• Color Pattern
• Box Drawing	• Diamond Fill	• Screen Save
• Circle Drawing	• Copy Area	• Screen Restore
• Ellipse Drawing	• Erase Area	• Area Save
• Diamond Drawing	• Text Mode	• Area Restore
• Screen Print	• Invert Mode	• Compacted Files
• Dot Cursor Control	• Help Display	• more. . .

ONLY $89.95!

HDOS version for H89, H8, requires IMAGINATOR
ZDOS version for Z-100, color memory optional
Supports many dot graphics printers. Call for info.

NEWLINE SOFTWARE
P.O. Box 402, Littleton, MA 01460 (617) 486-8535

NAME ____________________ CHECK ONE

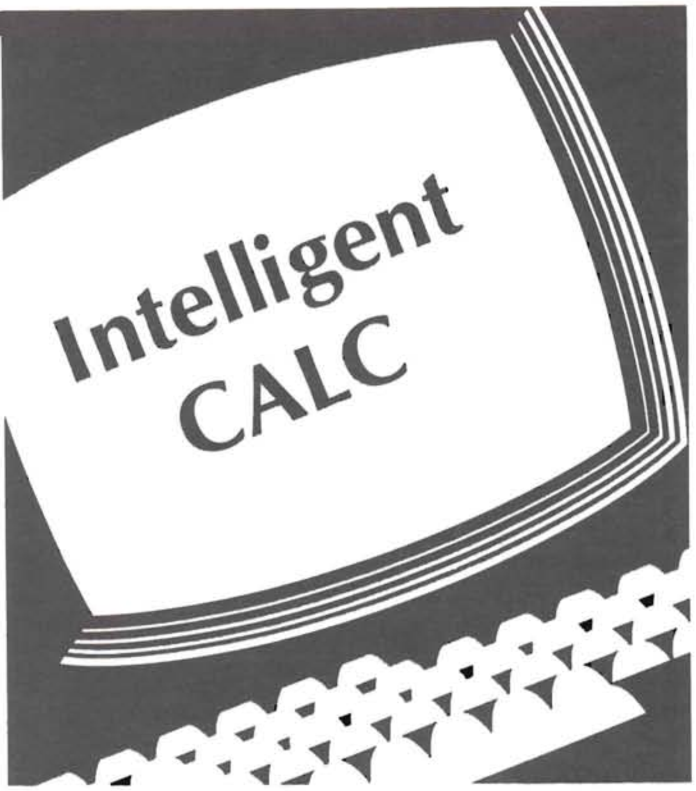
STREET ____________________ ☐ H89, H8/H19, HDOS

CITY ____________________

STATE ____________ ZIP ________ ☐ Z-100, ZDOS

Send me ______ "The ILLUSTRATOR" program(s) at $89.95 each.
Check one: ☐ payment enclosed ☐ send COD (add $4.00)
Send order to:
NEWLINE SOFTWARE, P.O. BOX 402, LITTLETON, MA 01460
Foreign orders: add $3.00 Airmail, $10.00 for non-U.S. checks

HDOS is a trademark of Heath Company
ZDOS, Z-100 are trademarks of Zenith Data Systems, Inc.
IMAGINATOR is a trademark of Cleveland Codonics, Inc.



Adding intelligence to your home financial programs using a BASIC subroutine

Robert L. Sanders
Programming Group
5101 McNarney
Midwest City, OK 73145

Intelligent CALC is a BASIC subroutine that uses BASIC's for-next loop and subscripted variables to perform smart calculations. The RAM of your computer is used to replace the pencil, pad, and calculator in your home budgeting, forecasting or financial planning.

Its final accuracy is based on the user's judged approximation of monthly utility bills (phone, lights, gas, water, etc.). These values can only be closely guessed at.

The word intelligent describes the most important attribute of this routine. This separates it from many of the other calculating routines I've used. It doesn't use simple lump sum methods to perform its calculations. For example, if I wanted to determine the number of months required to save a sum of money to make a purchase, using the lump sum method I would say:

```
           Income is   $950 per month
           Expenses    $400 per month
                       --------------
Therefore  Savings are $550 per month
```

If the item I am interested in purchasing costs $5,500, I would say it will take ten months to save the money for the purchase (10*550=5,500). If two of the items that make up my total expenses are loans that have a balance of 200 and 300 dollars respectively (Sears and VISA) and the payment due on each balance is $100 per month, after the second month's calculation the balance for the loan that started out with a balance of $200 (Sears) would be 0. And after the third month the loan that started out with a balance of $300 (VISA) would also be 0.

First Month:

Income $950

	Before Calculation		After Calculation	
	amount due	balance	amount due	balance
Sears	100	200	100	100
VISA	100	300	100	200
Phone	100	0	100	0
Electricity	100	0	100	0
Total	400			

Savings $550

Second Month:

Income $950

	amount due	balance	amount due	balance
Sears	100	100	0	0
VISA	100	200	100	100
Phone	100	0	100	0
Electricity	100	0	100	0
Total	400			

Savings $550

Third Month:

Income $950

	amount due	balance	amount due	balance
Sears	0	0	0	0
VISA	100	100	0	0
Phone	100	0	100	0
Electricity	100	0	100	0
Total	300			

Savings $650

Therefore the amount subtracted for expenses would be $200 less at the end of the third month. From the fourth month on, the total amount saved each month would be $750 using this routine. The time to reach our goal of $5,500 is actually less than ten months when we use the Intelligent CALC method.

Thus using the Intelligent CALC routine, it takes only 8 months to accumulate the required $5,500, not the 10 months when using the lump sum method.

I have attempted to have calculations such as the one described done at two local computer stores. One salesman used VisiCalc and the other Lotus 1-2-3. After more than twenty minutes of using complicated input protocols and row, column arrangements, neither salesman was able to give me a run that would tell me what I was asking. So I wrote this simple routine that performs this quickly and easily by answering prompts on the screen without worrying about which row or column I put my values in.

```
10 FOR I=1 TO BO
20 LET EX=EX+BI(I)
30 LET BA(I)=BA(I)-BI(I)
40 IF BA(I)=>0 THEN GOTO 80
50 LET BA(I)=BA(I)+BI(I)
60 LET EX=EX-(BI(I)-BA(I))
70 LET BA(I)=0:LET BI(I)=0
80 NEXT I
```

Line 10 FOR I=1 TO BO
This sets up the for-next loop. The number of loops is determined by the number of bills with a balance owed (BO).

Line 20 EX=EX+BI(I)
This line adds the payment for bill BI(I) to the expense total (EX), the number of locations in array BI(I) are the same as the number of bills with a balance.

Line 30 LET BA(I)=BA(I)-BI(I)
This handles the function of decreasing the balance by the amount of the payment. The number of locations in array BA(I) are the same as in array BI(I).

Line 40 IF BA(I)=>0 THEN GOTO 80
This is used to detect when the balance for the payment being calculated has been paid off and the variables need to be set to 0.

Line 50 LET BA(I)=BA(I)+BI(I)
This restores the balance to the value it held before line 30 was encountered. This is necessary because line 40 detected a negative value for BA(I) (the payment made in lines 20 and 30 was greater than the balance due on the bill).

Line 60 LET EX=EX-(BI(I)-BA(I)
This line decreases the value of variable EX (expenses) so that the amount added to expenses equals only that required to pay off the balance, which now is less than the previous monthly payments. This is the final payment.

Line 70 BA(I)=0:LET BI(I)=0
Because the balance has been paid the payments are set to 0.

Line 80 NEXT I
We are now set to go through the next loop in the routine.

Some of the useful applications of a routine such as this are:

a) Tells the actual amount saved over a period of time.

b) Tells the actual amount of time required to make a purchase based on the cost of the item and your actual savings.

c) Tells the balance of an account after (x) number of payments.

d) Tells when an account balance will actually be paid off.

Here is an example of an application. It's the one described here in the article.

```
10 DIM ACCT$(15),BA(15),UP(15),BI(15)
20 M=0:EX=0:MO=1
30 PRINT :INPUT"AMOUNT NEEDED TO MAKE PURCHASE ";ND
40 PRINT:INPUT "CASH OR SAVINGS ON HAND ";CASH
50 PRINT:INPUT "TOTAL MONTHLY NET INCOME ";INCOME
180 PRINT:PRINT:
    INPUT"NUMBER OF NON BALANCE PAYMENTS ";NBP
190 FOR I= 1 TO NBP
200 LET M=M+1
210 PRINT:PRINT"NAME OF NON BALANCE ACCOUNT # ";I:
    INPUT ACCT$(M)
220 PRINT:INPUT "AMOUNT DUE PER MONTH ";UP(I)
230 NEXT I
240 PRINT:INPUT"NUMBER OF BALANCE OWED PAYMENTS ";BO
250 FOR I= 1 TO BO
260 LET M=M+1
270 PRINT:PRINT"NAME OF BALANCE OWED ACCOUNT # ";I:
    INPUT ACCT$(M)
280 PRINT:INPUT"AMOUNT DUE PER MONTH ";BI(I)
290 PRINT:INPUT"BALANCE OWED ON ACCOUNT ";BA(I)
300 NEXT I
310 FOR I=1 TO NBP
320 LET EX=EX + UP(I)
330 NEXT I
340 FOR I= 1 TO BO
350 REM
360 LET EX=EX+BI(I)
370 LET BA(I)=BA(I)-BI(I)
380 IF BA(I)=> 0 THEN GOTO 420
390 LET BA(I)=BA(I)+BI(I)
400 LET EX=EX-(BI(I)-BA(I))
410 LET BA(I)=0:LET BI(I)=0
420 NEXT I
430 LET CASH=CASH+(INCOME-EX)
435 LET EX=0
440 IF CASH => ND GOTO 470
450 LET MO=MO+1
460 GOTO 310
470 PRINT "IT TAKES";MO; " MONTHS TO ACQUIRE "; ND
480 PRINT (CASH-ND);" WILL REMAIN AFTER THE PURCHASE."
490 END
```

In this program, when prompted to input NON BALANCE payments you should include such items as utilities, rentals, food, gas, insurance, paperboy, cable TV, and the like. BALANCE OWED payments are those of the type described in the article such as Sears and VISA.

You will notice several variables used in the program (ACCT$(M),M) that are not of any apparent use in this program as is. They are used to provide full screen formatting such as shown in the example calculations in this article and to provide the other functions described in the article (actual amount saved over a period of time, balance of an account after (x) number of payments, and when an account balance will be paid off). These extended functions of Intelligent CALC are explained in full detail in the Programming Group's BASIC Subroutines Reference Guide. This and other routines including their practical applications, plus examples of how to incorporate them in your programs, all written BASIC, are available for $10.00. The purchase of the reference guide includes a one year membership in the Programming Group, Programmers Association (PGPA), at the address at the beginning of this article.

About the Author:

<b>Robert Sanders</b> is a Computer System Specialist for the USAF on the Airborne Warning and Control System. He has experience in BASIC, Fortran, and assembly language programming. Also with HDOS, CP/M, and CDOS operating systems. Robert is also a chartered member of the Tinker Computer Club and co-founder of the Programming Group.

Patch Page

Pat Swayne
HUG Software Engineer

This article presents patches for CP/EMulator (885-3007-37), DIR100 (on 885-3008-37), and the Heath/Zenith version of CP/M-86. When you patch a program, be sure that you have at least one backup copy of it in case something goes wrong.

CP/EMulator Patch

The original release version of the CP/EMulator program will not work with MS-DOS version 2.0 in cases where the operating system requires memory above FFF:FFFF. CP/EMulator requires memory starting at 1000:0 for the CP/M programs. The following patch will allow it to use memory at a higher location if necessary. The patch must be done to the assembly code (CPM.ASM), which must then be re-assembled to produce a new CPM.COM. **Note:** This patch has been installed in our master disks.

To make the patch, locate the data area indicated by the comment shown here, and add the line defining CPMSEG. (If CPMSEG is already there, the complete patch has already been made in your copy.)

```
;
;       Flags, Pointers, and Data
;
CPMSEG  DW      1000H
SERADR  DW      0
```

Next, locate the message area and add a "not enough memory" message as shown below.

```
;
;       Messages
;
NMMSG   DB      13,10,'ERROR - Not enough memory'
        DB      ' for this program.',13,10,'$'
ABTMSG  DB      0DH,0AH
        DB      'ILLEGAL CP/M FUNCTION CALL: $'
```

Now, find the label LODR_ENTR, and add the additional lines shown below, before the CLD statement.

```
LODR_ENTR:
        MOV     AX,OFFSET LEND          ;POINT TO END OF PGM.
        MOV     CL,4
        SHR     AX,CL                   ;DIVIDE BY 16
        MOV     BX,CS                   ;GET THIS SEGMENT
        ADD     AX,BX                   ;ADD TO PGM END
        INC     AX                      ;ADD A PARA. FOR GOOD MEASURE
        PUSH    AX
        AND     AX,0FFFH                ;CHECK FOR EVEN 64K BOUNDARY
        POP     AX
        JZ      EVEN                    ;IT'S AN EVEN BOUNDARY
        AND     AX,0F000H
        ADD     AH,10H                  ;ELSE, MOVE UP TO THE NEXT ONE
EVEN:
        MOV     BX,WORD PTR CS:2        ;GET UNUSED SEG. ADDRESS
        CMP     AX,BX                   ;GOT ENOUGH RAM?
        JNB     NOT_ENOUGH_RAM          ;NO, ERROR
        MOV     Word Ptr CPMSEG,AX      ;SAVE CP/M SEGMENT
        MOV     ES,AX                   ;PUT ES HERE
        MOV     BX,0FFFFH               ;POINT TO LAST BYTE IN SEGMENT
        MOV     AL,BYTE PTR ES:[BX]     ;GET A BYTE
        INC     BYTE PTR ES:[BX]        ;TRY TO CHANGE MEMORY
        CMP     AL,BYTE PTR ES:[BX]     ;SEE IF IT CHANGED
        MOV     BYTE PTR ES:[BX],AL     ;FIX RAM
        PUSH    CS
        POP     ES                      ;FIX ES
        JNZ     RAM_OK                  ;RAM IS OK
NOT_ENOUGH_RAM:
        MOV     DX,OFFSET NMMSG
        MOV     AH,9
        INT     21H                     ;ELSE, SAY "NOT ENOUGH RAM"
        INT     20H
RAM_OK:
        CLD
        MOV     SI,OFFSET BUF
```

Locate the label NOT_FIRST_TIME and change MOV AX,1000 to what is shown.

```
NOT_FIRST_TIME:
        MOV     BYTE PTR CS:FTFLAG,1    ; MARK FIRST TIME DONE
        MOV     AX,WORD PTR CPMSEG
        MOV     ES,AX                   ; ES at 1000H (2nd bank of 64K)
```

Locate the comment "SET IN DUMMY..." and replace MOV AL,1 with the 4 lines shown below before the OUT statement.

```
INT     21H                     ; SET IN DUMMY CONTROL-C ROUTINE
MOV     AX,WORD PTR CPMSEG      ; GET CP/M SEGMENT
MOV     CL,4
SHR     AX,CL                   ; SHIFT IT DOWN
XCHG    AL,AH                   ; PUT RESULT IN AL
OUT     OFDH,AL                 ; Set 8085 to address proper 64K bank
```

Find the comment "NOT AT ZERO..." and change the code from there to the label TIMEXIT to what is shown below.

```
        JZ      TIMEXIT                 ; NOT AT ZERO, EXIT
        MOV     AX,WORD PTR CS:CPMSEG
        MOV     ES,AX                   ; ELSE, SET ES TO CP/M SEGMENT
        INC     WORD PTR ES:TICCNT      ; INCREMENT TIC COUNTER
        JNZ     TIMEXIT                 ; IF NOT, EXIT
        INC     WORD PTR ES:TICCNT+2    ; ELSE, UPDATE HIGH TIC COUNTER
TIMEXIT:
```

At the end of the program, add a label LEND as shown below.

```
STACK:
;
LEND:
CPM     ENDS
        END     START
```

After you make the patches, assemble CPM.ASM to a .COM file using the procedure on page P.5 of your Z-DOS manual.

DIR100 Patch

The DIR100 program on the Z-DOS UTILITIES disk (885-3008-37) has a bug in it that will not allow it to list more than 256 files correctly. To fix the problem, locate the label BLDORD in the source code and change the lines from there to the line with NEXTT so that they appear as below. If the code already looks like this, all patches have already been installed.

```
BLDORD: MOV     AX,Word Ptr COUNT       ;GET COUNT
BLDORD1:MOV     Word Ptr [BX],DX        ;SAVE ORD ADDR
        INC     BX
        INC     BX
        ADD     DX,CX                   ;POINT TO NEXT ENTRY
        DEC     AX                      ;MORE?
        JNZ     BLDORD1                 ;..YES
        MOV     Word Ptr NEXTT,Offset ORDER;SET TABLE POINTER
```

While you are working on DIR100, you may also want to make the next patch, which causes the files to be listed on the screen more rapidly. Locate the label TYPEIT and change the subroutine there so that it appears as below.

```
TYPEIT: MOV     AL,M
        OR      AL,AL                   ;TEST FOR HI BIT
        JNS     TYPEIT1                 ;NONE THERE
        PUSH    AX                      ;SAVE CHARACTER
        CALL    TYPTX
        DB      27,'p'+80H
        POP     AX
        AND     AL,7FH                  ;STRIP PARITY BIT
        CALL    TYPEC
        CALL    TYPTX
        DB      27,'q'+80H
        INC     BX
        DEC     CH
        JNZ     TYPEIT
        RET
TYPEIT1:CALL   TYPEC
        INC     BX
        DEC     CH
        JNZ     TYPEIT
        RET
```

After making the patches, assemble DIR100 to a .COM file.

CP/M86 Patch

The initial release of Heath/Zenith CP/M-86 (release 1.10) has a bug in the 8-bit emulation part of it that causes it to occasionally "kick out" of running programs back to the operating system. This problem occurs rarely, but it can be a real headache if you are in the middle of an editing session. To fix it, duplicate disk 3 of your CP/M-86 distribution disks and remove the files CLDR207.A86, FORMAT.A86, and LBIOS207.A86 from it. Copy the file CPM.H86 from disk 2 of the distribution disks to the new disk. Make sure you have an editor (such as ED.CMD), ASM86.CMD, and GENCMD.CMD on your system disk. In the file R85PKG.LIB on the new disk, locate the label SWAP85 and add the line shown below.

```
SWAP85:
        MOV     ES,R85MCBA
        MOV     ES:COMFLG,0     ;<-- THIS LINE ADDED
        MOV     ES:COMWHO,ZPSPPS5+ZPSPI88
```

Re-assemble the BIOS by entering

```
A>ASM86 B:BIOS86 $PZ SZ
```

Combine the resulting hex file and CPM.H86 with PIP as follows. (You can now delete BIOS86.A86 if you need space.)

```
A>PIP B:CPMX.H86=B:CPM.H86,B:BIOS86.H86
```

After combining the files, you can delete CPM.H86 and BIOS86.H86 if you need space. Now, run GENCMD to make a CMD file:

```
A>GENCMD B:CPMX 8080 CODE[A40]
```

Rename the resulting file to CPM.SYS:

```
A>REN B:CPM.SYS=CPMX.CMD
```

Now, you are ready to replace the existing CPM.SYS file on your system disk with the new one. Make sure you have a backup system disk before you start. First, remove write protection from the old CPM.SYS with STAT, as follows.

```
A>STAT CPM.SYS $R/W
```

Now, delete it and copy on the new one:

```
A>ERA CPM.SYS
A>PIP A:=B:CPM.SYS
```

Reset your computer and boot up on the system with the new CPM.SYS to test it. If it works OK, you can set the write protect and system attributes:

```
A>STAT CPM.SYS $R/O
A>STAT CPM.SYS $SYS
```

Now you can run 8-bit software under CP/M-86 with confidence that you will not get "kicked out".

"My Favorite Subroutines"

Dear HUG,

Here is a short one liner that produces the first 30 Fibonacci numbers. The sequence can be extended by declaring X and Y as double precision, and greater program flexibility can be attained by substituting "N" for "15" and changing the first PRINT statement to (INPUT"Fibonacci numbers";N: etc).

```
10 PRINT"Fibonacci numbers:":Y=1:FOR I=1 TO 15:
   X=X+Y:Y=Y+X:PRINT X,Y:NEXT
```

George Holt
403 2nd St. West
BAFB, LA 71110

Dear HUG,

For your "My Favorite Subroutines" column, I have a quickie that I have used many times in the past. I needed to develop it as a result of transferring a dairy management program from another machine into the H-89. They have a PRINT @ statement that allows you to print any specific location on the screen using either a fixed constant or a variable name. The screen is numbered similarly to the H-89 with location 1 being in the upper left corner and the highest number 1920 being at the lower right corner of the screen. You can print at any location on the screen using either fixed constants or any variable name that results in a decimal number of any magnitude. The quickie is as follows.

In the beginning of the program you have the following statement:

```
DEF FN Q$(X)=CHR$(27)+"Y"+CHR$(INT(X/80)+32)
+CHR$(XMOD80+32)
```

The `CHR$(INT(X/80)+32)` portion takes the decimal number and divides by 80 to get the integer of the row and adds 32 to offset for the 31 special codes.

The `CHR$(XMOD80+32)` portion of the direct cursor addressing statement gets the remainder after dividing by 80 for column location and also adds 32 to offset for the 31 special codes.

Then in the body of the program you would just use the following to print at any specific location using a fixed constant or a variable.

```
50 PRINT FNQ$(440) "Hello there" or
50 PRINT FNQ$(A) "Hello there"
```

I enjoy every magazine and hope that this column along with many other interesting ones continue.

Fred Schmidt
4578 Lantern Ct. N.W.
Comstock Park, MI 49321

Dear HUG,

Here is my favorite subroutine, which I call "The Defeat of Humbug Printer Buffering". Software buffering for printer output is very nice, when you want it. When you don't want it, it can be aggravating and cause you to wish misfortune to some arrogant software programmer. Sometimes, it is very useful to see exactly what has been printed before deciding what to do next. Here is a subroutine in C/80 (Software Toolworks) that works under HDOS to put out a single byte at a time to a printer (mine is an Epson MX-80) immediately, with no ifs, ands, or buts, and without waiting for an internal software buffer to fill up, or for the printer channel to be closed.

```
putclp(prchan,c) int prchan; char c;
     {static char a; a = c; write(prchan,&a,1) ; }
```

A printer channel must be opened before you start using a putclp, by a

```
prchan = fopen("LP:","w") ;
```

command. The HDOS and C/80 manuals say that the write command argument for buffer length must be a multiple of 256. Fortunately, that is not necessary for the LP: device handler that Heath supplied with my HDOS system. In the above command, the buffer length is specified as one byte.

Phillip L. Emerson
3707 Blanche
Cleveland Heights, OH 44118

Dear HUG,

After having read the "My Favorite Subroutines" column in the April 1984 issue, it seemed the right time for me to send in my favorite routine regarding PRINT and/or LPRINT.

I have seen so many routines where so many statements are used to perform such a simple task that I thought you might find this one much simpler.

No PEEK statement is needed and you can use either PRINT or LPRINT, whichever suits you.

If you wish to use a PRINT statement and then use them for both console and printer output, then use 'POKE 3,170' for output to the printer:

```
10 INPUT"Do you wish hardcopy printout (Y/N)? ",Z$
20 IF Z$="Y" THEN POKE 3,170
30 PRINT"Output to the correct device depending on Z$"
```

Any answer to Z$ other than 'Y' will output copy to the console device. The following routine uses an LPRINT statement and has the same effect:

```
10 INPUT "Do you wish hardcopy printout (Y/N)? ",Z$
20 IF Z$="N" THEN POKE 3,105
30 LPRINT"Output to the correct device depending on Z$"
```

Any answer to Z$ other than "N" will output copy to the printer.

After each such routine, or at the end of the program, another POKE statement is needed to return the I/O BYTE back to the normal configuration. Just use 'POKE 3,169' and all is well.

John Hafey
5636 Drake Ave.
Cleveland, OH 44127

LEARN A NEW WORD FOR YOUR VOCABULARY

ZPAY\zē pā\ n, 1: A computer payroll system for the Zenith and Heath computer systems 2: The act or fact of paying or being paid 3: The status of being paid by an employer

ZPAY PAYROLL SYSTEMS Presents

ZPAY available for ZDOS, IBM-PC Compatable or CPM

Some of the features available in the **ZPAY** Payroll System

- User Friendly — Menu Driven
- Pays by Hourly, Salaried & Commissioned
- No interpreters necessary
- Super FAST operation
- Attractive 3 ring binder manual
- User changeable State & Federal tax tables
- User selected pay periods
- Maintains payroll records required by law
- Prints out easy to use and read reports
- Choice of two different check formats
- *Plus* many features of systems costing much more

May be seen at many Heathkit Electronic Centers or order yours today from **ZPAY** Payroll Systems — **Only $100.00.**

Please send me **ZPAY** for ☐ ZDOS ☐ IBM-PC ☐ CPM

Name ____________________

Street ____________________

City ______________ State ________ Zip ________

Phone __________ Visa/MC Card # __________ expires ________

Orders shipped by UPS. Enclose $100.00 plus $4.00 per order for shipping and handling. Payment by check, money order, or Visa/Master Card. *Allow 2—4 weeks for delivery.*

ZPAY PAYROLL SYSTEMS

3516 Ruby Street • Franklin Park, IL 60131 • (312) 671-3364

HOW MUCH FREE SOFTWARE COULD YOU USE?

FIND OUT WITH OUR GIANT PUBLIC DOMAIN DIRECTORY

- SUPPLIED ON DISK FOR EASY COMPUTER ACCESS
- MORE THAN 4,500 ENTRIES
- SUBJECT AREAS INCLUDE:

ASTRONOMY, AVIATION, BUSINESS, EDUCATION, ENGINEERING, GAMES, GRAPHICS, HAM RADIO, MUSIC, PROGRAMMING, TEXT EDITING, VOICE SYNTHESIS, UTILITIES AND MUCH MORE.

Yes! I need to know what free software is available. Send me the public domain directory on the Heath CP/M* 5¼ format checked ☐ ON 3 HARD SECTORED SD $17.00 ☐ ON 2 SOFT SECTORED DD $12.00 ☐ ON 1 DOUBLE SIDED DD $7.00

☐ SPECIAL UPDATE OFFER

I purchased your 3,000 entry P. D. directory and want to update it. I have enclosed a formatted disk with 40K of free space and my check or M.O. for either $2 domestic or $4 foreign S & H. Please put the update on my disk and rush it back. Offer expires 7-31-84.

HEAD WARE

HEADWARE
2865 AKRON STREET
EAST POINT, GA. 30344

TERMS: NO RISK, MONEY back guarantee. Add $2 domestic $4 foreign per order for S&H. Enclose your check or M.O. with your order. Sorry, no charge or phone orders.

Name ____________________

Address ____________________

City, State, Zip ____________________

*CP/M Reg. TM Digital Research Corp.

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 # ____________________

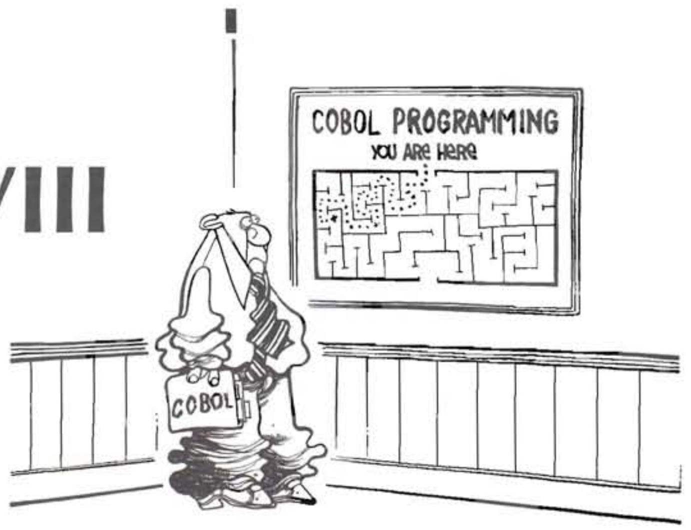
☐ M/C EXP DATE ____________________

ITEM	FORMAT	PRICE

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

COBOL Corner VIII

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



Introduction

How did your Program #2 work out? Did you need the HUG COBOL Disk-I PRGM02.COB file to find your errors? If you could not find your errors or you do not understand what we are doing, NOW IS THE TIME to get on top of your problems! This is your COBOL Corner, so let me hear from you (don't forget the SASE, business size).

Style Summary

Before starting Program #3, let's review the programming style we have and will use:

General Items

1. Make user-defined words meaningful and self-documenting.

2. Use hyphens in user-defined words to separate the English words and abbreviations.

3. Do not use commas or semicolons in user-defined words.

4. Provide vertical line spaces between Divisions, Sections and certain Paragraphs by inserting blank comment lines ("*" in column 7).

5. Write only one (1) COBOL sentence, statement, clause or phase per Coding Line.

6. When indentation is used, indent four (4) spaces. Exceptions are cases where we desire vertical alignment or when 4 spaces consume too much space on the Coding line. We will find this when we start using Nested IF statements. Two (2) spaces will then be used.

Identification Division

1. Limit program-names to six (6) characters plus the extension.

2. Vertically align the Program-Name and Comment-Entries below the "D" in Division.

3. A future ANS-COBOL Standard will be changing the following Paragraph Headers to Comment-Lines to retain their documentation value; thus, place an asterisk "*" in column 7 of each of these Coding lines:

a. Author
b. Installation
c. Date-Written
d. Security

Some compilers supply the Date-Compiled comment with a Date, so we will not use the "*" for this line.

Environment Division

1. Write the Select statement so the Assign clause begins on a separate line. Indent the Assign clause 4 spaces.

2. Sequence the Select statements so the Input Files are first followed by the Output File's Select statement.

3. DO NOT choose file-names that refer to specific Input/Output devices (such as Disk-File).

Data Division

1. Write each clause of the FD entry on a separate line. Indent each clause after the file-name.

2. Use the Record Contains clause of the FD.

3. Omit the Data Records clause of the FD. This clause will be removed from the next COBOL Standard.

4. Indent each Data-Item subdivision 4 spaces (Level-Number 05 at column 12, Level-Number 10 at column 16, and so on). Again if 4 spaces causes problems use 2 spaces.

5. Leave gaps between Level-Numbers (01,05,10,15 and so on).

6. Prefix all Data-Names of a record to help its documentation. The prefix for each record should be unique and meaningful. In later programs we will use suffixes so that you can decide which you like best.

7. Conserve space on the Coding line by using self-explaining abbreviations where possible -- PIC, rather then the word Picture for example.

8. Vertically align File Section Picture clauses.

9. Express the field length in the picture character string by a number enclosed in parentheses rather than by the repetition method -- PIC 9(04). vs PIC 9999. -- for example.

10. Do not use the 77 Level-Number. It is scheduled to be removed by the next COBOL Standard. We will use Working-Storage independent data-items in collections of logically related fields using Level-Numbers 01 thru 49 instead. Do not worry about this now. We will clear it up as we use it.

Do you agree with our "style"? Let me have your comments and suggestions. We will expand on our style conventions in future articles. You will learn these by repetition, repetition, etc.

Program Report Design

Up to now (Programs #1 & #2), we have designed programs to write (Print-Out) the Body Area of a Report. If we are going to design Reports that can be understood by more people, we must improve our report design. When designing a report, the programmer should provide three (3) general data areas:

1. Heading Area
2. Body Area
3. Total and Sub-Total Areas

Heading Area -- contains data that will identify the report. Examples of such data are Report Title, Organization Name, Report Date, and Report Page Number. The bottom heading area usually provides the Column Names for the following detail line fields printed in the Body Area. We will provide Headings in future programs.

Body Area -- contains the detail lines as we have produced in Programs #1 & #2. Sometimes this area will also have Summary Lines and/or Sub-Total Lines. We will do some of these in future programs.

Total Area -- is usually at the end of the report (as we will do in Program #3). Common items include Record Counts, Final Totals for columns and results of calculations (such as averages) which are made after all applicable Input Records have been processed. Descriptive words to identify these total figures are usually provided.

Arithmetic Statements & Report Totals

Notice that the Total Area described above will usually involve calculations. Therefore, if we are going to design a program with Report Total Lines, we will have to include COBOL arithmetic verbs into our program design. This will be one of the Program #3 problems. We will furnish a report with total lines. We will not attempt to do headers at this time! That will be a few COBOL Corner articles away.

Program #3

As COBOL Corner's System Analyst, I will supply you with the instructions that you, the programmer, will need for your program design. The program will be a Sales Report that will be printed from a Transaction Order File stored on disk. This Report must contain the Detail Lines and two (2) Total Lines.

```
                 Program Specifications
 ------------------------------------------------------
  PROGRAM NAME: SALES REPORT       PROGRAM ID: PRGM03
 ------------------------------------------------------
```

Program Description:

This program prints a Sales Report from the Order Record Data File.

Input File:

The FILEL3.DAT contains the Customer Name and the Purchase Amount (File has other data that we use for another report).

Output File

Sales Report as specified below.

List of Program Operations

A. Read each input order record.

B. For each record, the program should compute the Sales Tax Amount and the Transaction Amount.

1. Sales Tax Amount will be computed by multiplying Purchase Amount by 6.5% sales tax.
2. Transaction Amount will be computed by adding the Sales Tax Amount to the Purchase Amount.
3. Count the number of records processed and accumulate the count in Total Number Transactions accumulator.

C. For each record, print the following fields on the Sales Report detail line in accordance with the format specified for the Output Report Line:

1. Customer Name
2. Purchase Amount
3. Sales Tax Amount
4. Transaction Amount

D. Double-space each detail line.

E. After all the input records have been processed, the program shall print the following total fields on the Sales Report:

1. Total Line #1
   a. Total Number Transactions (computed by counting 1 for each order record).
   b. Total Purchase Amount (computed by summing each order record Purchase Amount into an Accumulator).
   c. Total Sales Tax Amount (computed by summing each record Sales Tax Amount into an Accumulator).
   d. Total Transaction Amount (computed by summing item b and item c into an Accumulator).
   e. Triple-space Total Line #1 down from the last detail line.
   f. Place one (1) asterisk "*" after each dollar total amount.
2. Total Line #2
   a. Average Purchase Amount (computed by dividing Total Purchase Amount by Total Number Transactions and round to nearest penny).
   b. Label this total.
   c. Double-space Total Line #2 down from Total Line #1.

F. COBOL will be the programming language.

OUTPUT REPORT LINE FORMAT

PRINT POSITIONS	FIELD NAMES	COMMENTS
	DETAIL LINE	
1-5	FILLER	PROVIDE LEFT MARGIN
6-29	CUSTOMER NAME	
30-32	FILLER	
33-44	PURCHASE AMOUNT	ZERO-SUPPRESS NON-SIGNIFICANT DOLLAR POSITION ZEROS INSERT COMMAS & DECIMAL POINT
45-48	FILLER	
49-58	SALES TAX AMOUNT	ZERO-SUPPRESS NON-SIGNIFICANT DOLLAR POSITION ZEROS INSERT COMMAS & DECIMAL POINT
59-62	FILLER	
63-74	TRANSACTION AMOUNT	ZERO-SUPPRESS NON-SIGNIFICANT DOLLAR POSITION ZEROS INSERT COMMAS & DECIMAL POINT
75-132	FILLER	
	TOTAL LINE #1	
1-9	FILLER	PROVIDE LEFT MARGIN
10-27		PRINT "TOTAL TRANSACTIONS"
28	FILLER	
29-32	TOTAL NUMBER TRANSACTIONS	ZERO-SUPPRESS NON-SIGNIFICANT ZEROS
33	FILLER	
34-46	TOTAL PURCHASE AMOUNT	ZERO-SUPPRESS NON-SIGNIFICANT DOLLAR POSITION ZEROS INSERT COMMAS & DECIMAL POINT
47		PRINT AN ASTERISK "*"

48	FILLER	
49-60	TOTAL SALES TAX AMOUNT	ZERO-SUPPRESS NON-SIGNIFICANT DOLLAR POSITION ZEROS INSERT COMMAS & DECIMAL POINT
61		PRINT AN ASTERISK "*"
62-63	FILLER	
64-76	TOTAL TRANSACTION AMOUNT	ZERO-SUPPRESS NON-SIGNIFICANT DOLLAR POSITION ZEROS INSERT COMMAS & DECIMAL POINT
77		PRINT AN ASTERISK "*"
78-132	FILLER	

TOTAL LINE #2

1-7	FILLER	
8-30		PRINT "AVERAGE PURCHASE AMOUNT"
31-32	FILLER	
33-44	AVERAGE PURCHASE AMOUNT	ZERO-SUPPRESS NON-SIGNIFICANT DOLLAR POSITION ZEROS INSERT COMMAS & DECIMAL POINT
45-132	FILLER	

INPUT RECORD FORMAT

FIELD LOCATION	FIELD NAME	DATA CLAUSE	COMMENTS
1-2	RECORD CODE	ALPHANUMERIC	CODE "L3"
3-5	FILLER		
6-29	CUSTOMER NAME	ALPHANUMERIC	
30-70	FILLER		
71-79	PURCHASE AMOUNT	NUMERIC	ASSUMED DECIMAL POINT BETWEEN COLUMNS 77 & 79
80	FILLER		

Note: Input Transaction File contains other fields that we will not use in Program #3. We will use them in our next program.

Programmer's Work

Using the above facts, you the programmer, are now ready to develop the Sales Report Program Phase by Phase. You should now know these. If not, go back to previous COBOL Corner articles for a review. Remember that we DO NOT want to proceed without doing these Phases first! If you do, you will not be prepared when we get to the complicated programs.

Coding Help

I will start you out on the new programming ideas we will be using. Here they are:

```
INPUT RECORD-DESCRIPTION

01  OF-ORDER-FILE.
    05  FILLER                              PIC X(05).
    05  OF-CUSTOMER-NAME                    PIC X(24).
    05  FILLER                              PIC X(41).
    05  OF-PURCHASE-AMT                     PIC 9(07)V99.
    05  FILLER                              PIC X(01).

OUTPUT RECORD-DESCRIPTION

01  SR-SALES-REPORT.
    05  FILLER                              PIC X(05).
   05  SR-CUSTOMER-NAME                    PIC X(24).
    05  FILLER                              PIC X(03).
    05  SR-PURCHASE AMT                     PIC Z,ZZZ,ZZZ.99.
    05  FILLER                              PIC X(04).
    05  SR-SALES-TAX-AMT                    PIC ZZZ,ZZZ.99.
    05  FILLER                              PIC X(04).
    05  SR-TRANSACTION-AMT                  PIC Z,ZZZ,ZZZ.99.
    05  FILLER                              PIC X(58).
01  TL-TOTAL-LINE-1.
    05  FILLER                              PIC X(09).
    05  TL-TRANSACTION-WORDS                PIC X(18).
    05  FILLER                              PIC X(01).
    05  TL-TOTAL-NO-TRANSACTIONS            PIC ZZZ9.
    05  FILLER                              PIC X(01).
    05  TL-TOTAL-PURCHASE-AMT               PIC ZZ,ZZZ,ZZZ.99.
    05  TL-ASTERISK-1                       PIC X(01).
    05  FILLER                              PIC X(01).
    05  TL-TOTAL-SALES-TAX-AMT              PIC Z,ZZZ,ZZZ.99.
    05  TL-ASTERISK-2                       PIC X(01).
    05  FILLER                              PIC X(02).
    05  TL-TOTAL-TRANSACTION-AMT            PIC ZZ,ZZZ,ZZZ.99.
    05  TL-ASTERISK-3                       PIC X(01).
    05  FILLER                              PIC X(55).
01  AT-TOTAL-LINE-2.
    05  FILLER                              PIC X(09).
    05  AT-PURCHASE-WORDS                   PIC X(23).
    05  FILLER                              PIC X(02).
    05  AT-AVERAGE-PURCHASE-AMT             PIC Z,ZZZ,ZZZ.99.
    05  FILLER                              PIC X(86).

WORKING-STORAGE DIVISION

01  WS-SWITCHES.
    05  WS-END-OF-FILE-SWITCH               PIC X(03).

01  WS-WORK-AREA.
    05  WS-WORK-TAX-AMT                     PIC S9(06)V99.

01  WS-TOTAL-ACCUMULATORS.
    05  WS-TOTAL-NO-TRANSACTIONS            PIC S9(04).
    05  WS-TOTAL-PURCHASE-AMT               PIC S9(08)V99.
    05  WS-TOTAL-SALES-TAX-AMT              PIC S9(07)V99.
    05  WS-TOTAL-TRANSACTION-AMT            PIC S9(08)V99.

PROCEDURE DIVISION

XXXXXX PRINT-SALES-REPORT.

XXXXXX      OPEN INPUT FILEL3
XXXXXX           OUTPUT SALES-REPORT.
XXXXXX      PERFORM INITIALIZE-VARIABLE-FIELDS.
XXXXXX      READ FILEL3
XXXXXX          AT END
XXXXXX              MOVE "YES"          TO WS-END-OF-FILE-SWITCH.
XXXXXX      PERFORM PROCESS-SALES-REPORT
XXXXXX          UNTIL
XXXXXX              WS-END-OF-FILE-SWITCH IS EQUAL TO "YES".
XXXXXX      PERFORM PRINT-TOTAL-LINE-1.
XXXXXX      PERFORM PRINT-TOTAL-LINE-2.
XXXXXX      CLOSE FILEL3
XXXXXX            SALES-REPORT.
XXXXXX      STOP RUN.

XXXXXX INITIALIZE-VARIABLE-FIELDS.

XXXXXX      MOVE "NO "                  TO WS-END-OF-FILE-SWITCH.
XXXXXX      MOVE ZEROS                  TO WS-TOTAL-ACCUMULATORS.

XXXXXX PROCESS-SALES-REPORT.

XXXXXX      MOVE SPACES                 TO SR-SALES-REPORT.
XXXXXX      MOVE OF-CUSTOMER-NAME       TO SR-CUSTOMER-NAME.
XXXXXX      MOVE OF-PURCHASE-AMT        TO SR-PURCHASE-AMT.
XXXXXX      MULTIPLY OF-PURCHASE-AMT
XXXXXX          BY .065
XXXXXX              GIVING WS-WORK-TAX-AMT ROUNDED.
XXXXXX      MOVE WS-WORK-TAX-AMT        TO SR-SALES-TAX-AMT.
XXXXXX      ADD OF-PURCHASE-AMT WS-WORK-TAX-AMT
XXXXXX          GIVING SR-TRANSACTION-AMT.
XXXXXX      DISPLAY SR-SALES-REPORT.
XXXXXX      WRITE SR-SALES-REPORT
XXXXXX          AFTER ADVANCING 2 LINES.
XXXXXX      ADD 1                       TO WS-TOTAL-NO-TRANSACTIONS.
XXXXXX      ADD OF-PURCHASE-AMT         TO WS-TOTAL-PURCHASE-AMT.
XXXXXX      ADD WS-WORK-TAX-AMT         TO WS-TOTAL-SALES-TAX-AMT.
XXXXXX      ADD OF-PURCHASE-AMT WS-WORK-TAX-AMT
XXXXXX                                  TO WS-TOTAL-TRANSACTION-AMT.
XXXXXX      READ FILEL3
XXXXXX          AT END
XXXXXX              MOVE "YES"          TO WS-END-OF-FILE-SWITCH.

XXXXXX PRINT-TOTAL-LINE-1.

XXXXXX      MOVE SPACES                 TO TL-TOTAL-LINE-1.
XXXXXX      MOVE "TOTAL TRANSACTIONS"
XXXXXX                                  TO TL-TRANSACTION-WORDS.
XXXXXX      MOVE WS-TOTAL-NO-TRANSACTIONS
XXXXXX                                  TO TL-TOTAL-NO-TRANSACTIONS.
XXXXXX      MOVE WS-TOTAL-PURCHASE-AMT
XXXXXX                                  TO TL-TOTAL-PURCHASE-AMT.
XXXXXX      MOVE WS-TOTAL-SALES-TAX-AMT
XXXXXX                                  TO TL-TOTAL-SALES-TAX-AMT.
XXXXXX      MOVE WS-TOTAL-TRANSACTION-AMT
XXXXXX                                  TO TL-TOTAL-TRANSACTION-AMT.
XXXXXX      MOVE "*"                    TO TL-ASTERISK-1
XXXXXX                                     TL-ASTERISK-2
XXXXXX                                     TL-ASTERISK-3.
XXXXXX      DISPLAY TL-TOTAL-LINE-1.
XXXXXX      WRITE TL-TOTAL-LINE-1
XXXXXX          AFTER ADVANCING 3 LINES.

XXXXXX PRINT-TOTAL-LINE-2.

XXXXXX      MOVE SPACES                 TO AT-TOTAL-LINE-2.
XXXXXX      MOVE "AVERAGE PURCHASE AMOUNT"
XXXXXX                                  TO AT-PURCHASE-WORDS.
XXXXXX      DIVIDE WS-TOTAL-PURCHASE-AMT
XXXXXX          BY WS-TOTAL-NO-TRANSACTIONS
XXXXXX              GIVING AT-AVERAGE-PURCHASE-AMT ROUNDED.
XXXXXX      DISPLAY AT-TOTAL-LINE-2.
XXXXXX      WRITE AT-TOTAL-LINE-2
XXXXXX          AFTER ADVANCING 2 LINES.
```

Closing

The next COBOL Corner will explain the many new programming ideas shown above. We will explore Picture clauses, working-storage uses, and some of the COBOL computing verbs. So, if you do not understand all of this article, we hope to clear it up next time! However, for your "homework", please do the following:

1. Develop Program #3 Phase by Phase.
2. Fill out your Code forms.
3. Key-In the Code.

To help with the above assignment, you may want to refer to your COBOL-80 Manual for information on Picture clauses, Add, Subtract, and Multiply verbs. Until next time, good COBOL computing.

Dear Readers,

Many "COBOL Corner" readers have asked me if and how they could use Ellis Nevada COBOL with this series. I advised the readers that I had purchased the software and that I had tested it with the H-8/H-89 and H/Z-100 (8085), and as a result of this testing that I had prepared a series of articles and a program/data disk for REMark.

Walt Gillespie, REMark's editor, approved the articles for publication, but now advises me that he cannot devote the REMark space for both "COBOL Corner" and "Nevada COBOL" articles in the same issue. Thus, to not cause an interruption in the "COBOL Corner" series, I am offering to supply the "Nevada COBOL" articles and the program/data disk.

I will provide the series on either two (2) CP/M hard sector disks or a CP/M soft sector disk for $25.00 including shipping. These will provide you with the necessary starting information and the working program/data files that you need to work along with the "COBOL Corner" series using the Ellis Nevada COBOL software on your H-8/H-89 or H/Z-100 (8085 CP/M). Be sure to advise me which format you require.

The regular COBOL-80 readers will still order their program/data disk from HUG.

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

FLOPPY DISK

CONTROLLER

Controls Any Combination Of Up To Four
8" and 5¼" Drives

This easy to install plug in board can control any combination of single or double sided, single or double density drives.

Designed especially for H88/H89 users

- Fully compatible Bios supplied for your CP/M 2.2 operating system
- Easy to follow instructions
- Contains controller board with boot prom
- Order cables for connections $15 (HFDC-110)
- Latest technology for enhanced reliability, minimal power consumption
- Quantity and user group discounts available

INTRODUCTORY OFFER

$395

Order HFDC-100

NORTH COAST INTELLIGENCE INC.

CP/M 2.2 Trademark of Digital Research Corp
H88/H89 Trademark of Heath Co.

ORDER FORM

Name ____________________

____
Street

____
City State Zip

Phone (_____) ____________________

☐ I am interested in what the HFDC-100 can do for my system, please send me more information.

☐ I need the expansion the HFDC-100 can provide. Please rush me:
_____ HFDC-100 _____ HFDC-110

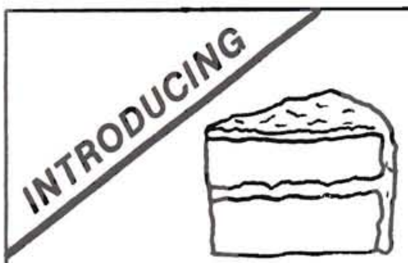
Please indicate payment below

☐ VISA Account # ____________

☐ M.C. Expiration Date ____________

☐ Check or Money Order Enclosed

Mail to: 1201 Cherokee Trail, Willoughby, Ohio 44094
Phone: 216-946-7756



CAKE (Cursor And Keypad Editor)

Now, you can get a full-screen text editor as easy to use as ***(PIE).***

You get the frosting on the **CAKE** with the following additional benefits:

- ★ file and command-key compatibility with the popular and reliable ***PIE*** editor.
- ★ system file control while running **CAKE** which allows you to obtain file directories, delete files, and list files.
- ★ you can merge other files with the file you are currently editing.
- ★ you can selectively save portions of the file you are currently editing.
- ★ you can easily format your text by setting margins, combining and dividing lines, and controlling TAB functions.

PIE is a registered trademark of Thomas Crosley.

Call or write for more information and our **FREE** catalog.

For Faster Delivery Call 906-249-9801

Now you can have your CAKE, and edit too!

And to sweeten the offer now you can **save $10.00**

CP/M version (requires 32K)	Regular price $49.95
	Special HUG price . . . . . . $39.95 (first 500 orders!)
ZDOS version (requires 128K)	Regular price $59.95
	Special HUG price . . . . . . $49.95 (first 500 orders!)

LIMITED TIME OFFER
CALL YOUR ORDER IN TODAY!

Add $2.00 for shipping. Michigan residents add 4% for sales tax.
PAYMENT IN U.S. FUNDS, PLEASE.

GENERIC SOFTWARE
P.O. Box 790 · Dept. 684R
MARQUETTE, MI 49855
906-249-9801
10 AM · 5 PM EST

Controlled Data Recording Systems Inc.

ANNOUNCING THE FDC-H8

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

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

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

NEW PRODUCTS FOR THE FDC-880H

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

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

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

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

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

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

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

Terminal Control With H/Z-100 Using MS: FORTRAN and MACRO-86

M. Manivannan
Department of Chemical Engineering
Clarkson College of Technology
Potsdam, NY 13676

BASIC remains the most popular language for microcomputer beginners. Unfortunately, it is also one of the slowest languages. This can be tolerated for programs which are run only a couple of times. But if you have a completely debugged program which is used frequently, then even a thirty second delay seems interminable. One of the possible solutions is to buy a BASIC compiler and compile the frequently used programs. The three hundred and odd dollars price tag on the compilers is not very encouraging, to say the least. Since I already have the Microsoft FORTRAN-86 compiler, I decided to translate my BASIC programs into FORTRAN to give me a compiled version.

It may come as a surprise to many beginners that FORTRAN, essentially a main-frame computer language, is the closest relative of BASIC which is widely identified as a microcomputer language and it is rather simple to translate one to the other. However, some of the most useful features of BASIC are simply not available in FORTRAN, though it is possible to find similar commands to do the same job. As an example, the 'FOR...NEXT' loop of BASIC is not available in FORTRAN but can be effectively replaced by the 'DO...CONTINUE' loop. String handling, which is one of the strong features of BASIC, is rather primitive in FORTRAN. Also FORTRAN does not have anything equal to the graphics and terminal control commands of BASIC such as 'LOCATE m,n' etc. This need not discourage us. Most of the terminal control commands can be added to FORTRAN by writing simple subroutines to do the job. Often these subroutines will have to use system specific escape codes and hence the subroutines as well as the main programs that utilize them are not transportable to other computers, sacrificing one of the important advantages of FORTRAN. The ESCAPE codes of ZORRO the H/Z-100 are organized very cleverly and we can use these very effectively to control the terminal output. In this article, I will try to illustrate

Listing 1. FORTRAN Graphics Subroutines CLS.FOR and LOCATE.FOR

```
         to clear Screen, Cursor to Row 1, Col 1
          SUBROUTINE CLS
          CHARACTER ESC
          ESC = CHAR(27)
          WRITE(1,10)ESC
       10 FORMAT(1H ,A1,'E')
          END

    C     To locate the cursor to IROW and ICOL
          SUBROUTINE LOCATE(IROW,ICOL)
          CHARACTER IR,IC,ESC
          COMMON /AREA/ESC
          IR=CHAR(IROW+31)
          IC=CHAR(ICOL+31)
          WRITE(*,10)ESC,IR,IC
      10  FORMAT(1H ,A1,'Y',2A1\)
          RETURN
          END
```

Listing 2. Required assembly coding for interfacing with FORTRAN.

```
DGROUP  GROUP DATA        ;See Note on FORTRAN manual page 76.
CODE    segment  'CODE'
        assume CS:CODE, DS:DGROUP ,SS:DGROUP
PUBLIC CLS                          ;'PUBLIC' declaration enables
                                    ; a FORTRAN main program to call
                                    ; the subroutine named CLS
CLS     PROC      FAR               ; 'FAR' declaration enables
                                    ; inter segment call
        PUSH      BP                ;Save caller's frame pointer
        MOV       BP,SP             ;Set up our frame pointer
                                    ;
*actual code goes here*             ;
                                    ;
        POP       BP                ;Restore caller's frame pointer
        RET                         ;This is a FAR return. This alone is
                                    ;sufficient for any assembly subroutine
                                    ;that does not pass any variable to
                                    ;or from the calling program.
CLS     ENDP

CODE    ends
DATA    SEGMENT PUBLIC  'DATA'

DATA    ENDS
        end
```

how to write the subroutines in FORTRAN and (don't be scared) assembly language.

Most of the readers of REMark magazine should be familiar with the use of escape codes for terminal control. Using these codes with MBASIC has been illustrated in several articles in REMark (for example see P. Swayne, Issue 43 page 15 and D. Warnick, Issue 46 page 25). Most of these terminal control commands are standard in ZBASIC and the user who starts with ZBASIC may not be familiar with the escape codes. These readers should have a good look at the appendix B, chart 3 of the H/Z-100 user's manual and appendix O of the ZDOS manual, vol. 2. Detailed explanations for these codes are also available in the Technical manual beginning at page 10.42 of vol. 1.

If you are not willing to devote some time to learn assembly language but still want to control terminal output with FORTRAN, take heart, you can write the subroutines in FORTRAN itself. For this, you would find the article by Mr. John Sams in the May 1983 REMark (Issue 40, page 44) very helpful. This article, 'ESCAPE with FORTRAN and PASCAL', dealt with the use of escape codes for the H/Z-89 in FORTRAN-80 and Lucidata Pascal. Some modifications are necessary to write these subroutines in FORTRAN-86 for ZORRO the H/Z-100. The modifications required are illustrated in Listing 1. (The readers are reminded that all the listings are for their personal use only and the author reserves all the commercial rights.)

There are two important modifications to note. First, in FORTRAN-86, we use Character Declaration instead of Byte for the variable ESC. Secondly, we use the intrinsic function CHAR to get the character value for the ASCII integer code. These two changes alone are enough to make all the FORTRAN-80 programs given by Mr. Sams to run equally well on the ZORRO with FORTRAN- 86. With FORTRAN-80, Mr. Sams had to adopt some clever programming tricks to position the cursor and write the message. As he explained, the format processor in FORTRAN forces a new line automatically before it executes the format specification of the WRITE statement. This automatic carriage return can be supressed with FORTRAN-86 by using a backslash ('\') descriptor, which tells the format processor to print on the current line. Using this, I was able to position the cursor with the write statement in the FORTRAN subroutine itself unlike Mr. Sams who was forced to use the calling program for that purpose. Hence the LOCATE subroutine positions the cursor just like the LOCATE statement of ZBASIC.

Listing 3. Assembly Language subroutine CLS.ASM.

```
        ;
        ;Clear Screen - Assembly language subroutine for FORTRAN and Pascal
        ;FORTRAN  calling sequence:
        ;       CALL CLS

        DOSF_OUTSTR     EQU     09H
        DOSI_FUNC       EQU     21H
                Page    , 132
        DGROUP  GROUP DATA      ;See Note on FORTRAN manual page 76.
        CODE    segment  'CODE'         ;Subroutine to clear screen
                assume CS:CODE, DS:DGROUP ,SS:DGROUP,ES:NOTHING
        PUBLIC CLS
        ;-------------------------------------------------------------

        CLS     PROC    FAR                     ;Allows inter segment call
                PUSH    BP                      ;Save caller's frame pointer
                MOV     BP,SP                   ;Set up our frame pointer
    START:  PUSH    DS                  ;Save caller's Data Segment
                                                ;Point DS:DX to beginning of
                MOV     AX,DATA                 ;  string to be printed out
                MOV     DS,AX                   ;Set up our Data Segment
                MOV     DX,OFFSET MESG          ;Get message address
                MOV     AH,DOSF_OUTSTR          ;Get function to output message
                INT     DOSI_FUNC               ;Print message
        END:    POP     DS                      ;Restore caller's Data Segment
                POP     BP                      ;Restore caller's frame pointer
                RET
        CLS     ENDP
        ;------------------------------------------------------------------

        CODE    ends
        DATA    SEGMENT PUBLIC  'DATA'
                        ;ESC, CLS, END_CHAR
        MESG    DB        27, 'E',   '$'
        DATA    ENDS
                end
```

Most of the terminal control commands one might need can be written in FORTRAN-86. Then, you might ask, why take the trouble to write these in assembly language. FORTRAN code is not exactly very efficient. It is slower than the straight forward assembly coding and produces a much larger code. The FORTRAN object code size for each of the terminal command subroutines varies between 500 to 600 bytes and the eventual executable file is quite large. There are about twenty five terminal control commands and the total size of all these subroutines will be at least 12K. I found that if I wanted to create a graphics library with the objective codes generated by subroutines written in FORTRAN, the library would be unnecessarily big. The same subroutines written in assembly language generate an object code 1/5th the size of the FORTRAN code. In addition, writing assembly language subroutines to do the same job is quite elementary. The ZDOS manual (appendix P) and the FORTRAN manual (Chapter 7) give some good example programs. Following the example in the FORTRAN manual, the assembly language subroutines can be easily interfaced with FORTRAN.

The minimum required assembly language coding that is necessary for any subroutine to be interfaced with a FORTRAN calling program, is shown in Listing 2. I have used comments liberally to make it very easy to follow the program. The FORTRAN compiler manual advises that the data used by assembly language routines must be placed in a segment whose name is DATA, whose class name is 'DATA', and should be grouped in DGROUP. This will become clear with an example I will be using shortly. The manual also requires the ASSUME statement. This statement reserves a maximum of 64K segment for the part of the program we name in the assume statement. We have to put the code in the CS statement and use DGROUP for DS and SS segment (CS, DS, SS, and ES are short for code, data, stack, and extra segments respectively). The rest of the code is self-explanatory.

Before I clarify the RET instruction, I should discuss the types of subroutines we will be concerned with. To give BASIC-like features to FORTRAN, we need to write three or four types of assembly language routines. The first and simplest type just needs to be called from the main program. It is essentially self sufficient. Routines that clear the screen, beep at the terminal, etc., belong to this type.

These routines do not pass any variables to and from the calling program and hence use a simple RET instruction to pass control back to the main program. In this segment, I will be primarily discussing this type of assembly subroutines.

All the other types of assembly routines 'communicate' with the main program by passing variables and hence additional parameters are pushed on the stack. Depending on the type and number of the passed variables, the number of bytes pushed on the stack will vary. This should be kept track of and the corresponding bytes should be popped back by an appropriate RET N instruction where N is the number of bytes to be returned. The routines for cursor positioning, color setting, string handling, and graphics are included in these types. I will discuss the first two routines of this type in a future article.

The simpler type routine is illustrated in Listing 3. It closely follows the model described in Listing 2. This routine uses the print string function of the interrupt command. We put the string to be printed out in the DATA segment and declare it public. The string is named as MESG and it ends with a dollar sign to indicate the end of the string to the interrupt routine. The ASCII decimal code for ESC is 27 (= 1B hex) and the code to clear the screen is ESC followed by 'E'. We start our program by loading DATA segment into DS. Since we cannot directly alter DS register, we load AX register with DATA and then move AX to DS register. Next, we get the address of the MESG string into DX register by using the OFFSET directive. Now DS:DX points to the string to be printed out. Next we load DOS function outstring code DOSF_OUTSTR (which is equal to 9) into AH register followed by the function interrupt call. We will be using this interrupt call INT DOSI_FUNC or INT 21H quite often so it is good idea to browse through appendix I of ZDOS manual. Incidentally, by replacing the MESG line of example 2 in appendix P (page P.5) with that given in Listing 3, and running that example, you will have a clear screen command available to you at the DOS level.

If you compare the example program to Listing 3, you would also notice that I am not including the DEFASCII.ASM and DEFMS.ASM programs. The example method is the preferred way to write the programs. However, the programs I will be describing use only two variables (DOSI_FUNC and DOSF_OUTSTR) which will need to be defined by the external include programs. Following the example program in this case only increases the assembling time since the assembler has to look up the include files. If

Listing 4. Graphics Subroutine TERMGRAF.ASM

```
;----------------------------------------------------------------------------;
;Screen Graphics - Assembly language subroutine for FORTRAN and Pascal;
;Program Name : TERMGRAF.ASM    (C)  1984        Version: 1.01        ;
;Author       : M. Manivannan                                          ;
;               Dept. of Chem. Engg., Clarkson College of Technology  ;
;               Potsdam, NY 13676                                      ;
;             - all commercial rights reserved by the author.         ;
;Created      : 28 Dec 83                                              ;
;Edited       : 31 Dec 83                                              ;
;Revised      : 5 Jan 84                                               ;
;                                                                      ;
;FORTRAN  calling sequence          :        Purpose                   ;
;       CALL CLS                    :To Clear Screen                   ;
;       CALL REVON                  :To turn Reverse Video On          ;
;       CALL REVOFF                 :To turn Reverse Video Off         ;
;       CALL KEYON                  :To enable the 25th line           ;
;       CALL KEYOFF                 :To disable the 25th line          ;
;       CALL SCRLON                 :To go into the Screen Scroll mode ;
;       CALL SCRLOF                 :To go into the Screen Hold mode   ;
;       CALL GRAFON                 :To turn Graphics mode on          ;
;       CALL GRAFOF                 :To turn Graphics mode off         ;
;       CALL CSRON                  :To turn Cursor on                 ;
;       CALL CSROFF                 :To turn Cursor off                ;
;       CALL BLKCSR                 :To set block cursor               ;
;       CALL LINCSR                 :To set line cursor                ;
;       CALL CLIKOF                 :To turn key-click off             ;
;       CALL CLIKON                 :To turn key-click on              ;
;       CALL BEEP                   :To give a BEEP at the terminal    ;
;----------------------------------------------------------------------------;
        PAGE     132
DOSF_OUTSTR     EQU     9           ;Function to output string
DOSI_FUNC       EQU     21H         ;Function interrupt

DGROUP  GROUP DATA          ;See Note on FORTRAN manual page 76.

GRAPHICS_CODE   segment  'CODE' ;Subroutine for screen graphics
assume CS:GRAPHICS_CODE, DS:DGROUP ,SS:DGROUP,ES:NOTHING

PUBLIC CLS,REVON,REVOFF,KEYON,KEYOFF,SCRLON,SCRLOF,GRAFON,GRAFOF
PUBLIC CSRON,CSROFF,BLKCSR,LINCSR,CLIKOF,CLIKON,BEEP
;------------------------------------------------------------------
START   MACRO
        PUSH    BP                  ;Save caller's frame pointer
        MOV     BP,SP               ;Set up our own frame pointer
        PUSH    DS                  ;Save caller's Data Segment
        MOV     AX,DATA             ;Set up our Data Segment
        MOV     DS,AX               ;Get address of message
        POP     DS                  ;Restore caller's Data Segment
        ENDM
;--------------------------------------------------------------------
PRINT   MACRO DOSF
        MOV     AH,DOSF             ;Get function to output message
        INT     DOSI_FUNC           ;Print message
        ENDM

FINISH  MACRO
        POP     BP                  ;Restore caller's frame pointer
        RET
        ENDM

PRINT_MSG       MACRO   MESG
        START
        MOV     DX,OFFSET MESG  ;Get message address
        PRINT   DOSF_OUTSTR     ;Print string
        FINISH
        ENDM
;-------------------------------------------------------------------
CLS     PROC    FAR
        PRINT_MSG MSG_CLS           ;Output message
CLS     ENDP
;-------------------------------------------------------------------
REVON   PROC    FAR
        PRINT_MSG MSG_REVON
REVON   ENDP
;-------------------------------------------------------------------
REVOFF  PROC    FAR
```

```
        PRINT_MSG MSG_REVOFF
REVOFF  ENDP
;-------------------------------------------------------------------
KEYON   PROC    FAR
        PRINT_MSG MSG_KEYON
KEYON   ENDP
;-------------------------------------------------------------------
KEYOFF  PROC    FAR
        PRINT_MSG MSG_KEYOFF
KEYOFF  ENDP
;-------------------------------------------------------------------
SCRLON  PROC    FAR
        PRINT_MSG MSG_SCROLLON
SCRLON  ENDP
;-------------------------------------------------------------------
SCRLOF  PROC    FAR
        PRINT_MSG MSG_SCROLLOFF
SCRLOF  ENDP
;-------------------------------------------------------------------
GRAFON  PROC    FAR
        PRINT_MSG MSG_GRAPHON
GRAFON  ENDP
;-------------------------------------------------------------------
GRAFOF  PROC    FAR
        PRINT_MSG MSG_GRAPHOFF
GRAFOF  ENDP
;-------------------------------------------------------------------
CSRON   PROC    FAR
        PRINT_MSG MSG_CURSORON
CSRON   ENDP
;-------------------------------------------------------------------
CSROFF  PROC    FAR
        PRINT_MSG MSG_CURSOROFF
CSROFF  ENDP
;-------------------------------------------------------------------
BLKCSR  PROC    FAR
        PRINT_MSG MSG_BLOCK_CURSOR
BLKCSR  ENDP
;-------------------------------------------------------------------
LINCSR  PROC    FAR
        PRINT_MSG MSG_LINE_CURSOR
LINCSR  ENDP
;-------------------------------------------------------------------
CLIKOF  PROC    FAR
        PRINT_MSG MSG_CLICKOFF
CLIKOF  ENDP
;-------------------------------------------------------------------
CLIKON  PROC    FAR
        PRINT_MSG MSG_CLICKON
CLIKON  ENDP
;-------------------------------------------------------------------
BEEP    PROC    FAR
        PRINT_MSG MSG_BEEP
BEEP    ENDP
GRAPHICS_CODE   ends
DATA    SEGMENT PUBLIC  'DATA'
                        ;ESC,MSG_CHAR,END_CHAR
ESC             EQU     27
END_CHAR        EQU     '$'
MSG_CLS         DB      ESC,'E'  ,END_CHAR
MSG_REVON       DB      ESC,'p'  ,END_CHAR
MSG_REVOFF      DB      ESC,'q'  ,END_CHAR
MSG_KEYON       DB      ESC,'x1',END_CHAR
MSG_KEYOFF      DB      ESC,'y1',END_CHAR
MSG_SCROLLON    DB      ESC,'\'  ,END_CHAR
MSG_SCROLLOFF   DB      ESC,'['  ,END_CHAR
MSG_GRAPHON     DB      ESC,'F'  ,END_CHAR
MSG_GRAPHOFF    DB      ESC,'G'  ,END_CHAR
MSG_CURSORON    DB      ESC,'y5',END_CHAR
MSG_CURSOROFF   DB      ESC,'x5',END_CHAR
MSG_BLOCK_CURSOR  DB    ESC,'x4',END_CHAR
MSG_LINE_CURSOR DB      ESC,'y4',END_CHAR
MSG_CLICKOFF    DB      ESC,'x2',END_CHAR
MSG_CLICKON     DB      ESC,'y2',END_CHAR
MSG_BEEP        DB          07  ,END_CHAR
DATA    ENDS
        end
```

you make those ubiquitous typographical errors as often as I do, you will be spending awfully long minutes waiting for the assembler to finish the job. That explains why I chose to define those variables in this program itself. However, if you will be using a lot of the system defined variables, you would be better off including the define files. Another thing that might interest the beginner is the comparison between the Listing 3 and the ".COD" file developed by the FORTRAN compiler for Listing 1. To get the latter, you should run PAS3 of the FORTRAN compiler. You would learn quite a bit from that ".COD" file!

We can write all the simpler type terminal command routines just like Listing 3 by using appropriate escape codes and renaming the program. We will then have about 15 individual object codes to be linked to the FORTRAN program. Nobody likes to type 15 names every time one wants to use the terminal command routines. To avoid this, we have two choices. The first one is to use the LIBrarian that comes with the ZDOS II disk. This bundles all of our routines into a neat little library and we can use this library along with FORTRAN library at the time of linking. More about this a little later. The second method is to optimize the code by bundling together all the similar routines into a single subroutine. This method may not always work, but when it does, it reduces the object code size quite a bit. Listing 4 shows how this could be accomplished.

It is obvious that all these subroutines use an identical procedure. Each of these push the same parameters on to the stack, perform the same interrupt and pop the same parameters back. The only difference comes from the fact that the data required by each of these subroutines is different. So, to make the programming simple, I decided to use the macro facility. This facility allows a set of frequently used instructions to be written as a block. The advantage of this method is the elimination of recoding of these instructions each time they are encountered. This facility does not, however, reduce the object code size; it simply saves us a lot of typing and reduces the source code size. Another advantage of the macro facility is the use of the dummy argument. With this, we can use the same macro block for several subroutines. In Listing 4, the macro PRINT_MSG is common to all the subroutines. This macro has the dummy variable MESG which is replaced by an appropriate message variable in each of the subroutines. The macro PRINT_MSG itself contains three additional macro calls. Of these, the macros START and FINISH just do the pushing and popping and the macro PRINT uses the interrupt routine to print a string. Now, it is easy to add any subroutines to this program simply by using the appropriate MACRO calls. This program should now be assembled using the "MASM TERMGRAF;" command and the TERMGRAF.OBJ file should be copied on the FORTRAN working disk.

The calling program in FORTRAN is given in Listing 5. To compile a FORTRAN program with a single drive is not exactly a pleasure. I do it this way. I have the files FOR1.EXE, PAS2.EXE, and PAS3.EXE all on a single non-system disk along with the RUNFORT.BAT file given in Listing 6. I also copy the FORTRAN source file to this disk and run the batch file RUNFORT. After compiling, I copy the object code to the FORTRAN working disk which contains the LINK.EXE, FORTRAN.LIB, and LIB.EXE files. An executable run file can now be created by LINKing SCREEN.OBJ, the compiled FORTRAN file to TERMGRAF.OBJ, the assembled version of TERMGRAF.ASM. This procedure requires only the FORTRAN.LIB as the library file. However, you might decide to have your terminal control command subroutine files to be linked to the FORTRAN object code. In a future article, I intend to give additional files such as LOCATE, COLOR, etc. You might want to include all these object codes as well. The best way to link all these files at the same time is by creating a library file of all the terminal control command routines. For this, we would use the LIB command, which is on the ZDOS II disk. It is easy to create a ".LIB" file; the LIB command prompts you for that. An example is given in Listing 7. The files are linked using the LINK command as shown in Listing 8. When you are linking a user created library with other files, the plus sign must be used to separate the library file names when library prompt is displayed. According the the ZDOS manual, a blank space can also be used to separate the library file names. This caused some problems when the user file name follows the FORTRAN library name producing the error message "Unresolved externals".

Just after I completed this, I came to know that Clarkson was developing a complete Graphics package for FORTRAN and Pascal with capabilities very similar to that of ZBASIC. It is now available to Clarkson students beginning this semester. I have not had any opportunity to test this package yet. But I like its capabilities. I am sure that this package would be an asset to anybody who likes to have BASIC-like features for FORTRAN programs. I expect that this graphics package will be marketed by Clarkson soon.

Listing 5. Fortran Calling Program SCREEN.FOR.

```
      PROGRAM SCREEN
      INTEGER LINNUM
      CHARACTER ESC
      COMMON /AREA/ESC
      CALL SCRLOFF
 1    WRITE(*,5)
      FORMAT(1H ,'ENTER THE NUMBER OF THE BOTTOM LINE')
      READ(*,*)LINNUM
      IF (LINNUM .EQ. 0) GOTO 999
      ESC =CHAR(27)
      CALL CLS
      CALL BEEP
      CALL LOCATE(12,35)
      WRITE(*,10)
 10   FORMAT(1H ,'CENTER OF SCREEN')
      CALL KEYON
      CALL BEEP
      CALL LOCATE(LINNUM,30)
      CALL REVON
      WRITE (*,20)
 20   FORMAT(1H ,'WELCOME TO 25TH LINE')
      CALL LOCATE(1,1)
      CALL REVOFF
      WRITE(*,30)
 30   FORMAT(1H ,'QUIT AT THE TOP')
      GOTO 1
999   CALL KEYOFF
      CALL CLS
      CALL SCRLON
      CALL BEEP
      END
```

Listing 6. RUNFORT.BAT FILE

```
A:FOR1 %1;
PAUSE .... If no errors, press RETURN to run PAS2  else press <CTRL C>.
A:PAS2
PAUSE *****  If no errors, press <CTRL C> else press RETURN to run PAS3
A:PAS3
```

Listing 7. RUNNING LIB COMMAND

```
A:LIB

      Microsoft Library Manager V1.02
    (C) Copy right 1981 by Microsoft Inc

Library File:TCONTROL <RETURN>
Library does not exist. Create? YES <RETURN>
Operations:+TERMGRAF+LOCATE+YOURFILE <RETURN>
List file: TCONTROL.LST <RETURN>
```

Listing 8. RUNNING LINKER

```
 LINKER PROMPT                 USER ENTRY            KEY PRESSED

Object Modules [.OBJ]:         SCREEN                RETURN
Run File [SCREEN.EXE]:                               RETURN
List File [NUL.MAP]:                                 RETURN
Libraries [.LIB]:          FORTRAN+TCONTROL          RETURN
```

About the Author:

M. Manivannan *is a graduate student from India completing his Ph. D. in Chemical Engineering at Clarkson College. He bought his H/Z-100 under the Clarkson-/Zenith plan. Mr. Manivannan has already created an alternate character set for his mother tongue "Tamil", a South Indian language and he is quite thrilled about it. He used to write short stories and poetry in Tamil; however he discontinued after he came to this country since he didn't have a Tamil typewriter and didn't have the patience to prepare handwritten manuscripts. He hopes to resume his literary career with the Zee and thanks the smart engineers at Heath-/Zenith who made this possible. Oh, by the way, the Zee has been of immense use in his engineering career as well!*

NAME

ADDRESS

PHONE NUMBER

MASTERCARD/VISA CARD NUMBER

EXPIRATION DATE

SIGNATURE

Clip and mail coupon

HOME FINANCE SYSTEM VERSION 2

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

Hardware: H8/H19 or H-Z89/90 with 64K RAM and two disk drives. Printer strongly recommended (any Heath®- Zenith® or other printer).

Software: HDOS 2.0 and Microsoft® MBASIC 4.82 for HDOS. —Complete system: $89† (specify hard-soft sector 5-¼", or 8"). Manual alone $21†.

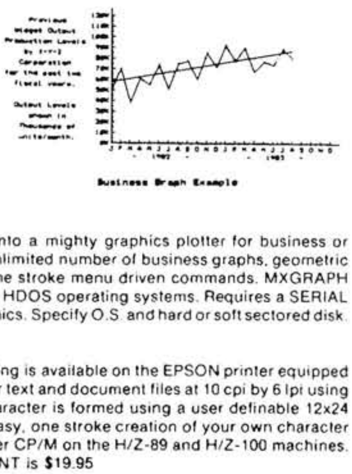
MasterCard/Visa accepted, please include your phone number.

†Prices include shipping.

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

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

A New Approach In MBASIC For Accepting Inputs From The H/Z-19 and H/Z-89 Keyboard

Richard E. Lucka
InchSoft
64 Fanchers Street
Pickerington, OH 43147

This article describes a new method for inputting characters from your H/Z-19 and H/Z-89 console using Microsoft MBASIC programs in the CP/M 2.2 operating system environment. This method will allow you to maintain one common console input routine capable of accepting all keys, including function keys, cursor control keys, and keypad keys in alternate mode.

This article is in response to an article entitled "Those Special Function Keys" by David E. Warnick (REMark Issue 47). The item of particular interest was in using the keypad keys on an H/Z-89 in alternate keypad mode. The author mentioned that some of the characters from the keypad keys in alternate mode are "lost" to MBASIC programs, perhaps due to timing between the Terminal Logic Board and MBASIC. Based on my experiences, I feel that the problem resides within the MBASIC interpreter and not the hardware.

To illustrate, consider the following MBASIC program:

```
10 E$=CHR$(27): GR$=E$+"p":
   'Enter reverse video string
20 EGR$=E$+"q": 'Exit reverse video string
30 IN$=INKEY$: IF IN$="" THEN GOTO 30:
   'Wait for keyboard input
40 IF IN$ <> E$ THEN PRINT IN$;: GOTO 30:
   'If not ESC, print character
50 PRINT GR$ "e" EGR$;:
   'Have ESC, print "e" in reverse video
60 REM Next statement expects a function key identifier
70 IN$=INKEY$: IF IN$="" THEN GOTO 30:
   'Go if not a function key
80 PRINT IN$;: GOTO 30: 'Print function key identifier
```

This routine prints out what you type. Whatever the program gets from the console, or at least from the MBASIC interpreter, it will print it out to the console for you to see, including the characters representing function keys.

As many of you know, each function key on the top row of your keyboard sends two characters: an ESCape key and a function key identifier, which is an upper-case letter. For example, the f1 key sends an ESCape character followed by the letter "S". The above program will print a lower-case "e" in reverse video and the letter "S".

Enter the above program into your computer and run it. Type any key. You should see the character(s) of the key you pressed. Now, press a function key repeatedly. You will notice that sometimes the ESCape character is printed and sometimes it is not. For whatever reason (most likely the MBASIC interpreter), your MBASIC program will not receive all characters coming from your terminal using this programming method.

However, if you replace each INKEY$ statement with an INPUT$(1) statement, then all seems well. But, suppose you want to use the keypad in alternate mode? Enter and run the following program, and observe what you get:

```
10 E$=CHR$(27): GR$=E$+"p":
   'Enter reverse video string
20 EGR$=E$+"q": 'Exit reverse video string
30 PRINT E$ "x7";: 'Enter keypad in alternate mode
40 IN$=INPUT$(1): 'Wait for keyboard input
50 IF IN$ <> E$ THEN PRINT IN$;: GOTO 40:
   'If not ESC, print character
60 PRINT GR$ "e" EGR$;:
   'Have ESC, print "e" in reverse video
70 REM Next statement expects a function key identifier
80 IN$=INPUT$(1): PRINT IN$;:
   IF IN$ <> "?" THEN GOTO 40: 'Print fnctn key id
90 IN$=INPUT$(1): PRINT IN$;: GOTO 40
```

When you run this program, line 30 causes your keypad to be set in alternate keypad mode. Therefore, each key in the keypad will send three characters: an ESCape character, a question mark ("?"), and an identifier character. With this program, you will find that the "?" is missing nearly all the time. For example, if you press the ENTER key, you should see "e" in reverse video, a question mark, and the letter "M". Instead, you will see the "e" in reverse video and the letter "M", but not the question mark!

My solution to this dilemma is a small assembly language subprogram which "hides" itself behind the CP/M operating system nucleus and runs concurrently with most applications programs, including MBASIC. We will call this program CONSUSR. It is designed to capture all incoming characters and store them in its own type-ahead buffer, which can then be retrieved by MBASIC programs or any other program.

CONSUSR takes advantage of a hardware feature in your H/Z-89 such that each time a character is sent from the Terminal Logic Board, it interrupts the CPU and causes it to follow a different path (in this case, it goes to BIOS where it handles console inputs). At this point, CONSUSR captures the incoming character and stores it in it's own buffer. It will even capture all three characters sent by a keypad key in alternate mode.

Your MBASIC programs can call CONSUSR to return an input character in the same order that it receives from the console. No characters will be lost, and you can maintain one common console input routine to handle all your console inputs, including the function keys, etc.

Since CONSUSR "hides" itself behind CP/M, you will need to create a separate systems disk using MOVCPM and SYSGEN to create a CP/M configured to run on 1K less memory than the total available memory in your computer. The upper 1K of memory is the place where CONSUSR needs to operate. For example, if you have 64K in your machine, you will need to run MOVCPM to configure your CP/M systems disk to run on 63K so that CONSUSR will run on the upper 1K memory.

How CONSUSR Works

CONSUSR, as shown in Listing 1, relocates itself behind the CP/M nucleus code and intercepts and stores each incoming console character. It releases each character every time a program calls it, or it replies with a zero byte (CHR$(0)) if there are no outstanding input characters.

CONSUSR is written in Z80 assembly language, using the extended Intel 8080 mnemonics. Much of the Z80 code was chosen to make the program relocatable and not dependent upon a particular area of memory. The source code in Listing 1 can be assembled by the assembler supplied in your CP/M distribution disk. Note that the Z80 mnemonics are commented (since the CP/M assembler cannot handle Z80 instructions), but are defined by trailing DB statements. Also note that CONSUSR uses the Z80 prime registers. Generally, this should not be a problem if you are using commercial software. But if you write Z80 programs, using these registers, then avoid using these programs when CONSUSR is running.

Before the resident part of the program is relocated behind the CP/M nucleus, CONSUSR allocates and initializes three separate memory cells in low memory that does not appear to be in use by CP/M or any other program that I know of.

The first cell is in location 33H (H stands for hexadecimal), which contains a 3-byte JMP instruction to the CONSUSR program itself. This is the address which your programs use to call CONSUSR.

The second cell is located in locations 36H and 37H and contains the original entry address into BIOS that handles console inputs at interrupt time. CONSUSR stores this address in the second cell and also within itself so that after it has processed an input character, it jumps to BIOS to let it perform whatever console handling it has to do. The housekeeping code of CONSUSR checks the address value in the second cell to see if CONSUSR was loaded at least once since the last cold boot. This is done to avoid "re-connecting" the console interrupt vector the second time around which would cause your computer to crash if you type just any key.

The third memory cell is located in 3BH, and is the pass-off byte. This is the place where CONSUSR puts a console character from which your MBASIC program can retrieve with the PEEK command. If CONSUSR has no outstanding console character, then it plugs a null character (CHR$(0)) in this pass-off byte.

After the housekeeping code of CONSUSR has initialized these memory cells, it performs the following tasks:

1. Moves the resident code behind the CP/M nucleus code. This is assuming that you have already MOVCPM'd and SYSGEN'd the CP/M operating system.

2. Connects the console interrupt vector. Originally, the level-3 console interrupt jumps to BIOS to handle its own console inputs, but this vector is connected to CONSUSR, and CONSUSR connects to BIOS after it has read in the character and stored it in a separate type-ahead buffer.

3. Patches BIOS (in memory) to remove a check for the console interrupt identification register. This is necessary because since CONSUSR also reads data from the console port, it "satisfies" the 8250 UART interrupt identification register in your H-88-3 console serial port (or the H8-4 board in your H8 computer) which consequently would cause BIOS to ignore the console input character. Without this patch, you would not be able to run other programs. This patch will render the H8 computer using the H8-5 serial board totally useless because the patch causes BIOS to ignore input routines specifically for the H8-5 serial board.

When all the "housekeeping" is done, control returns back to the CP/M operating system. The resident CONSUSR code is now operating. Each time you press a key, CONSUSR will read the character from the console input data port and store it in its separate 256-byte type-ahead buffer. This type-ahead buffer is filled in a rotating manner, that is, if the buffer gets filled, CONSUSR starts at the beginning of the buffer. After storing the character, CONSUSR jumps to BIOS to let it handle its own console handling. This technique allows you to run other programs uninterrupted.

CONSUSR stands ready to accept calls from any program, including MBASIC programs. Each time you call CONSUSR via 33H, CONSUSR replies with a character in the pass-off byte at location 3BH, or it will reply with a null character if there is no outstanding character from the console. However, before you begin to use CONSUSR, POKE a CTRL-R (decimal 18) in the pass-off byte and call CONSUSR; the CTRL-R is a command for CONSUSR to clear its type- ahead buffer, or you will get all the type-ins (up to 256 characters) that you made prior to the first call to CONSUSR.

As an added feature, CONSUSR responds to the BREAK key and treats it as if you entered a CTRL-C character. In other words, you can press the BREAK key to "BREAK" an MBASIC program, or any other program that responds to the CTRL-C character. CONSUSR takes advantage of a special hardware feature found in the 8250 UART in your serial board that will allow it to substitute a character with another. Because of the nature of the BREAK key, the CTRL-C substitution feature takes place about 3/4 seconds after you press the BREAK key.

Using CONSUSR

Listing 2 shows an MBASIC program called RUNUSR, which demonstrates the power and effectiveness of CONSUSR. RUNUSR types out the character of each regular key that you typed, or the name of each function key you pressed. When running this program, you will find that you will not lose any character at all, even if you hold down a function key and the REPEAT key simultaneously. All keys will be identified except the CTRL-@ character (it produces a null character).

The following is a brief description of the most important parts of RUNUSR (the parts that you will want to include in your future programs):

Line 20
Defines variables UZR and BYTE. UZR points to memory location 33H where a JMP instruction has been installed (by CONSUSR) to "jump" to CONSUSR, whose location depends on the amount of memory you have in your computer. BYTE is the memory location of the "pass-off" byte.

Line 50
Plug a CTRL-R in the pass-off byte and CALL UZR to clear CONSUSR's type-ahead buffer. If you bypass this line, you will see all the type-ins you made before you got to this point.

Line 60
Use a GOSUB to call the common console input routine.

Lines 70-90
Upon return from the preceding GOSUB, the variable IN contains a value that represents the key that you pressed, except the CTRL-@ key. See Table-1 for the definition of each value for IN. The "ON expr GOTO" statement is ideal for selecting the appropriate routine for each input character.

Lines 100-4100
All lines within this range display the name of the key you pressed. All control characters will be displayed by the corresponding CTRL-character designation in reverse video.

Line 50020
Assign a value of 1 for the variable IN, the default value if you press a regular key (alpha, numeric, or special character).

Line 50030
CALL CONSUSR to see if there is a pending character from the console. If there is no pending character, then CONSUSR returns with a null character (CHR$(0)) in the pass-off byte.

Line 50040
Check to see if the input character is an ESCape character or not. If not, the variable IN is already "1", therefore RETURN.

Line 50050
Since the first character was an ESCape character, CALL CONSUSR again to see if there is another character.

Line 50055
If there is no other character following the ESCape key, then assume that the user pressed the ESC key only.

Line 50060
Check if the second character is a "?". If so, GOTO 50120 to look for a third character representing a keypad key in alternate keypad mode.

Lines 50065-50110
The lines within this range check the various characters that represent the function keys and the cursor control keys. For each of these keys, a value is assigned for the variable IN.

Lines 50120-50160
These lines process the third character that represents a key in the keypad in alternate keypad mode. For each key, a value is assigned for the variable IN.

The variable IN is used to indicate the key that was typed, whether it is a regular key, a function key, or a keypad key in alternate mode. The chart in Table 1 shows the different values and the associated key each value represents. For example, upon return from a GOSUB 50000 and IN=19, that means the user (you) typed an f4 function key. If IN=1, then the user typed in a letter or a number. Lines 70 through 90 in RUNUSR show the easiest way to GOTO the appropriate routines for each special key.

One precautionary note: If you are running RUNUSR on a 4MHz machine, when you press a function key, your MBASIC program should pick up the ESC key, then drop down to line 50050 to pick up the identifier character(s). However, without a small delay loop, RUNUSR may even miss the identifier from line 50050 because the CPU is fast enough to call CONSUSR before CONSUSR even gets the identifier, therefore RUNUSR assumes that you pressed the ESC key only, then after CONSUSR finally got the identifier and you call it from line 50030, you will assume that the user suddenly typed a letter, and will show this on the screen. To prevent this from happening, insert a small delay routine, such as:

```
50045 FOR I=1 TO 5: NEXT: 'Delay awhile...
```

This technique will give the fast computer a chance to capture all the key-ins.

Installation

CONSUSR has been verified to operate in the CP/M 2.2.03 operating system environment using the UNMODIFIED BIOS supplied in the CP/M distribution diskettes from Heath. CONSUSR does modify BIOS, and it is very likely that CONSUSR will not run (as presented in this article) if you are using a BIOS from another source.

Create a separate system disk, run MOVCPM and SYSGEN. Remember to specify 1K less memory than is available in your computer when running MOVCPM. For example, if you have 48K memory, type MOVCPM 47 * (see your CP/M reference manual for more details).

Enter the CONSUSR program from Listing 1. Fill in the correct value for the variable MEMSIZE. If you have 48K of memory, use the value 48.

Run ASM and LOAD to assemble the program. You may wish to configure the new system disk to run CONSUSR.COM automatically during cold boot.

Now, if you haven't already done so, run CONSUSR. It should load and return the CP/M command mode. CONSUSR is now in memory and running. As you type, it is storing your key-ins, but will not interfere with what you are doing.

Enter the RUNUSR program from Listing 2. Run this program to see what CONSUSR can do. In the future, you may use portions of RUNUSR as a foundation for a common console routine which can be merged in with future MBASIC programs.

Conclusion

I hope this article will prove useful to many in the HUG readership. With the technique described in this article, you can write effective programs that take advantage of all the keys and hardware features in your H8 or H/Z-89 computer using the MBASIC interpreter.

If you do not wish to take the trouble of entering CONSUSR and all the associated MBASIC routines, I will be glad to provide the CONSUSR source code, the associated MBASIC routines, including RUNUSR and the CONSUSR.COM file, if you send me a formatted 5 1/4" hard sectored diskette and $5.00. There will be three CONSUSR.COM files, one for 48K, the second for 56K, and the third for 64K. The $5.00 will be used to cover the cost of copying, postage, handling, and materials. Send your diskette to:

InchSoft
Attn: Richard E. Lucka
64 Fanchers Street
Pickerington, OH 43147

Table 1

Assignments for variable IN% for each H/Z-19 and H/Z-89 key as used in RUNUSR:

IN%	Key Representation
1	Single Key - alphabetic, numeric, special characters, and CTRL
2	Up arrow
3	Down arrow
4	Right arrow
5	Left arrow
6	HOME
7	ERASE
8	IC on (turn on insert character mode)
9	IL (insert line)

10	DL (delete line)
11	DC (delete character)
12	IC off (turn off insert character mode)
13	BLUE function key
14	RED function key
15	GRAY function key
16	f1 function key
17	f2 function key
18	f3 function key
19	f4 function key
20	f5 function key
21	ENTER key (keypad in alternate mode)
22	. key (keypad in alternate mode)
23	0 key (keypad in alternate mode)
24	1 key (keypad in alternate mode)
25	2 key (keypad in alternate mode)
26	3 key (keypad in alternate mode)
27	4 key (keypad in alternate mode)
28	5 key (keypad in alternate mode)
29	6 key (keypad in alternate mode)
30	7 key (keypad in alternate mode)
31	8 key (keypad in alternate mode)
32	9 key (keypad in alternate mode)

Listing 1

```
;  CONSUSR - Load Special Console Handler Routine
;            for MBASIC Programs.
;
MEMSIZE EQU     64                  ;Enter the amount of
                                    ; memory of your machine
;                                     here
USRMEM  EQU     1024*MEMSIZE-1024     ;Start address for
;                                      CONSUSR
TICCNT  EQU     0BH                 ;TICCNT
;
BRKEY   EQU     1                   ;BREAK key will have a
;                                     purpose
CTRL$S  EQU     0                   ;BREAK key will activate
;                                     CTRL-S/CTRL-Q (if 1)
;                                   ; else CTRL-C (if 0)
;                                     (↑S/↑Q UNTESTED)
;
;  ** NOTE **
;
;       Location 0033H = JMP USRMEM  (where CONSUSR
;                                     starts)
;       Location 0036H = address of CP/M entry for
;                        level-3 console interrupts
;       Location 003BH = pass-off byte - communications
;                        area between calling programs
;                        and CONSUSR
;
;  Initialization code (housekeeping).
;
        ORG     0100H
;
        MVI     A,0C3H              ;Install a JMP followed
;                                     by CONSUSR entry
        STA     33H                 ; address in location
;                                     0033H (for calling
;                                     programs)
        LXI     H,USRMEM+(USR-USR$ST)
        SHLD    34H
;
        LXI     D,USRMEM            ;If already connected
;                                     from a previous run,
        LHLD    19H                 ; do not re-connect
;                                     level-3 vector
        ORA     A
;       DSBB    D
        DB      355Q,122Q
        LHLD    36H                 ;If not to connect, use
;                                     previously stored
;       JZR     $+5                 ;  entry address to CP/M
        DB      050Q,3
        LHLD    19H                 ;Connect level-3 vectors
;                                     thru CONSUSR routine
        SHLD    JMP$CPM+1           ;Install CP/M console
;                                     interrupt handler
;                                      address
        SHLD    36H                 ;Also, save CP/M entry
;                                     address in location
;                                      0036H
        LXI     H,USRMEM            ;CONSUSR to be entered
;                                     for each level-3 int.
        SHLD    19H
;
        LHLD    36H                 ;Patch BIOS console
;                                     interrupt handler to
        LXI     D,21                ; bypass check for
;                                     interrupt identification
;                                      register.
        DAD     D                   ;  We are using the 8250
;                                      USART, base address is
        MOV     D,H                 ;  350Q (see page 109 of
;                                      BIOS source listing
        MOV     E,L                 ;  supplied with the CP/M
;                                      package from Heath)
        INX     D
        MOV     M,0
        LXI     B,9
;       LDIR
        DB      355Q,260Q

        LXI     B,USR$E-USR$ST  ;Move USR routines to
;                                 high memory (behind
;                                  CP/M)
        LXI     D,USRMEM
        LXI     H,USR$ST
;       LDIR
        DB      355Q,260Q
;
        DI                          ;Set up prime registers
;       EXX                         ;  *NOTE*  Other programs
;                                      that uses prime
        DB      331Q                ;  registers will fail
;                                      along with CONSUSR
        MVI     B,0                 ;B' = counter for INI
;                                      instruction
        MVI     C,350Q              ;C' = console data port
        LXI     D,USRMEM+256        ;DE' = buffer pointer
;                                     for fetch routine
        LXI     H,USRMEM+256        ;HL' = buffer pointer for
;                                     storing cons. inputs
;       EXX
        DB      331Q
        EI
;
        LXI     H,USRMEM+256        ;Clear console buffer to
;                                     nulls (256 bytes)
        XRA     A
        MOV     B,A
        MOV     M,A
        INX     H
;       DJNZ    $-2
        DB      020Q,252
;
        RET                         ;Done, return to CCP
;
;  The following code is relocated behind the CP/M
;    nucleus code.
;
;  Level-3 interrupt handler.
;
USR$ST  EQU     $
;       EXAF                        ;Switch to prime
;                                     registers
;       EXX
        DB      10Q,331Q
        IF      BRKEY
        IN      355Q                ;BREAK key?
        ANI     20Q
;       JNZR    USR$CTR
        DB      40Q,(USR$CTR-$)-1
        ENDIF
        IN      350Q                ;If CTRL-@, bypass
        ANA     A
;       JZR     $+7
;       INI                     ;Input data from console
;                                 port
;       JNZR    $+3             ;Cause wrap-around if at
;                                 end of console buffer
        DB      50Q,5,355Q,242Q,40Q,1
        DCR     H
;       EXX                     ;Restore registers and
;                                 return to CP/M
;       EXAF
        DB      331Q,10Q
JMP$CPM JMP     0
;
;  USR routine called from a MBASIC program to fetch a
;   data from buffer.
;
USR     DI                      ;Disable interrupts,
;                                 switch to prime regs
;       EXX
        DB      331Q
        LDA     03BH            ;See if user requesting
;                                 buffer clear
        CPI     012H            ;Buffer clear if user
;                                 sent ↑R
;       JER     USR1
        DB      050Q,(USR1-$)-1
        LDAX    D               ;Fetch data and store in
;                                 transfer area
        STA     03BH
        ORA     A               ;If null character, do
;                                 not bump to next addr
;       JZR     $+5
        DB      50Q,3
        XRA     A               ;Replace with null and
;                                 incr pointer
        STAX    D
        INR     E               ;Scheme supports
;                                 wrap-around
;       EXX                     ;Restore regs and return
        DB      331Q
        EI                      ;Enable interrupts
        RET                     ;Return to calling
;                                 program
;
USR1    XRA     A               ;Clear console buffer to
;                                 null and
        MVI     L,0FFH          ;  reset prime registers
        MOV     M,A
        MVI     E,0FEH
        LXI     B,254
;       LDDR
        DB      355Q,270Q
        STAX    D
        MVI     C,350Q
        DCR     L
;       EXX
        DB      331Q
        STA     03BH            ;Return with null in
;                                 pass-off byte
        EI
        RET
;
;  Special BREAK key handler.
;
USR$CTR PUSH    H               ;Save HL'
        PUSH    D               ;Save DE'
        LHLD    34H             ;Find out where D$CTRL is
        LXI     D,D$CTRL-USR
        DAD     D               ;HL = address of D$CTRL
        IF      CTRL$S
        MOV     A,M             ;Toggle CTRL-S/CTRL-Q,
;                                 if implemented
        XRI     3
        MOV     M,A
        ENDIF
;
        IN      351Q            ;Get current console
;                                 interrupt status
```

```
        PUSH    PSW             ;Save it
        XRA     A               ;Disable console
;                                 interrupts
        OUT     351Q
        IN      354Q            ;Set console UART in
;                                 loopback mode
        ORI     20Q
        OUT     354Q
        IN      350Q            ;Dummy read
        EI                      ;Allow CPU interrupts
;                                 for TICCNT
        XCHG                    ;DE = CTRL-char.,
;                                 HL = TICCNT
        LXI     H,TICCNT        ;Small delay
        MOV     A,M
        ADI     110
        CMP     M
;       JNER    $-1
        DB      40Q,253
        IN      350Q            ;Dummy read
        LDAX    D               ;Output the
;                                 CTRL-character
;       OUTC    A
        DB      355Q,171Q
;
        IN      355Q            ;Wait till data "looped"
;                                 around to input port
        RRC
;       JNCR    $-3
        DB      60Q,251
        IN      354Q            ;Reset UART loopback
;                                 mode
        XRI     20Q
        OUT     354Q
;
        MOV     A,M             ;Small delay again
        ADI     110
        CMP     M
;       JNER    $-1
        DB      40Q,253
;       INC     A               ;Dummy read to satisfy
;                                 pending data ready
        DB      355Q,170Q       ;  interrupts
        DI                      ;Disable CPU interrupts
;                                 for now
        POP     PSW             ;Restore original console
;                                 interrupt status
        OUT     351Q
;
        LHLD    34H             ;Find out where
;                                 JMP$CPM-2 is
        DCX     H
        DCX     H
        DCX     H
        DCX     H
        DCX     H               ;HL = address of
;                                 JMP$CPM-2
        POP     D               ;Restore DE' and HL',
;                                 go to JMP$CPM
        XTHL
        RET
;
        IF      CTRL$S
D$CTRL  DB      17              ;CTRL-Q
        ENDIF
;
        IF      NOT CTRL$S
D$CTRL  DB      3               ;CTRL-C
        ENDIF
;
USR$E   EQU     $
;
        END
```

Listing 2

```
10 REM RUNUSR - CONSUSR Exerciser Program
20 UZR = &H33: BYTE = &H3B: 'CONSUSR entry address and
   transfer byte
30 E$ = CHR$(27): RV$ = E$+"p": ERV$ = E$+"q"
40 PRINT E$ "=": 'Set keypad in alternate mode
50 POKE BYTE,18: CALL UZR: 'Flush console buffer
60 GOSUB 50000: 'Go get a character from console
70 ON IN GOTO 100,200,300,400,500,600,700,1200,
    800,900,1000,1100,1300,1400
80 ON IN-14 GOTO 1500,1600,1700,1800,1900,2000,2100,
    2200,2300,2400,2500,2600
90 ON IN-26 GOTO 2700,2800,2900,3000,3100,3200
100 IF IN$ < " " THEN PRINT RV$ CHR$(ASC(IN$)+64) ERV$:
    GOTO 60
110 PRINT IN$: GOTO 60
200 PRINT "Up arrow": GOTO 60
300 PRINT "Down arrow": GOTO 60
400 PRINT "Right arrow": GOTO 60
500 PRINT "Left arrow": GOTO 60
600 PRINT "HOME": GOTO 60
700 PRINT "ERASE": GOTO 60
800 PRINT "Insert Line key": GOTO 60
900 PRINT "Delete Line key": GOTO 60
1000 PRINT "Delete Character key": GOTO 60
1100 PRINT "Exit Insert Character mode": PRINT E$ IN$;:
     GOTO 60
1200 PRINT "Enter Insert Character mode": PRINT E$ IN$;:
     GOTO 60
1300 ID$ = "Blue": GOTO 4000
1400 ID$ = "Red": GOTO 4000
1500 ID$ = "Gray": GOTO 4000
1600 ID$ = "f1": GOTO 4000
1700 ID$ = "f2": GOTO 4000
1800 ID$ = "f3": GOTO 4000
1900 ID$ = "f4": GOTO 4000
2000 ID$ = "f5": GOTO 4000
2100 ID$ = "ENTER": GOTO 4100
2200 ID$ = ".": GOTO 4100
2300 ID$ = "0": GOTO 4100
2400 ID$ = "1": GOTO 4100
2500 ID$ = "2": GOTO 4100
2600 ID$ = "3": GOTO 4100
2700 ID$ = "4": GOTO 4100
2800 ID$ = "5": GOTO 4100
2900 ID$ = "6": GOTO 4100
3000 ID$ = "7": GOTO 4100
3100 ID$ = "8": GOTO 4100
3200 ID$ = "9": GOTO 4100
4000 PRINT ID$ " function key": GOTO 60
4100 PRINT "Keypad " ID$ " key in alternate mode":
     GOTO 60
50000 REM Common console input routine
50010 REM Exit: IN$ = last character, IN = index value
50020 IN=1: 'Default regular character index
50030 CALL UZR: IN$ = CHR$(PEEK(BYTE)):
      IF IN$ = CHR$(0) THEN GOTO 50030
50040 IF IN$ <> CHR$(27) THEN RETURN:
      'Regular character
50050 CALL UZR: IN$ = CHR$(PEEK(BYTE))
50055 IF IN$ = CHR$(0) THEN IN$ = CHR$(27):
      RETURN: 'ESC only
50060 IF IN$ = "?" THEN GOTO 50120:
      '? = alternate keypad character
50065 IF IN$ = "@" THEN IN = 8: RETURN:
      'Enter IC mode character
50070 IF IN$ < "H" THEN IN = ASC(IN$)-63: RETURN:
      'Arrow key indexes
50080 IF IN$ = "H" THEN IN = 6: RETURN: 'HOME
50090 IF IN$ = "J" THEN IN = 7: RETURN: 'ERASE
50110 IN = ASC(IN$)-67: RETURN: 'All the rest
50120 CALL UZR: IN$ = CHR$(PEEK(BYTE)):
      'Alt. keypad character
50130 IF IN$ = "M" THEN IN = 21: RETURN: 'ENTER
50140 IF IN$ = "n" THEN IN = 22: RETURN: '.
50150 IN = ASC(IN$)-89: RETURN: 'All the rest
```

IBM-PC/ZENITH Z-100 users.

Expand your computer universe with—

micro/VERSAL ™

A utility program to READ/WRITE over 20 different 5¼ (CPM, CPM/86, MS-DOS & USER DEFINABLE) disk formats. Now you can easily transfer text, data, or programs between many different micro computers by simply loading **micro/VERSAL**™ and the disk you want to READ/WRITE from or to. **micro/VERSAL**™ also includes comprehensive utilities to DUMP any 5¼ disk by track, RDSECT to read disk sectors and FAPP, a program to append files together to produce a large file. For disk formats not directly supported, **micro/VERSAL**™ provides customization routines that allow users to write their own directory routines.

micro/VERSAL™ **$79.99**
plus $4.00 shipping & handling

Also **COED**™ full screen editor................**$34.97**
Includes: Stack arithmatic, MACRO commands, multiple files, definable function keys and much more.

CREDIT CARD ORDERS:	Master Charge / VISA
MAIL ORDERS:	Checks or money orders. N.J. resident add 6% sales tax.

ADVANCED SOFTWARE TECHNOLOGIES
417 Broad Street
Bloomfield, N.J. 07003
(201) 783-7298

Z-DRAW a development tool for H/Z-100 Users

- DEVELOP interesting graphics screens quickly and easily that you can display from your BASIC pgms.
- USEFUL applications are:
  - Games
  - Instructional Software
  - Process Description Software
  - Online Documentation
- PROGRAMS included are:

Z-DRAW	A graphics editor
Z-LEARN	An online tutorial
Z-MERGE	ZBASIC source for a file display routine

Send Check or M.O. for $49.95 to:
REDWOOD DEVELOPMENT
7 Redwood Dr., BUTTE, MT 59701
(406) 494-5610 Business Hrs. 5-9 pm MST
Mastercard & Visa accepted

Visit our booth at the HUG Conference

Z-100, ZBASIC are trademarks of Zenith Data Systems, Inc.

WITH THE PASSWORD™ MODEM AND TELPAC™ BEN FRANKLIN COULD HAVE PUBLISHED THE FRIDAY EVENING POST.

The Password™ modem and Telpac™ software deliver text fast, far, cheap, and letter-perfect. Fast? Ten times faster than an expert typist (and four times faster than most other modems). Far? Crosstown or crosscountry. Letter perfect? Multiple accuracy checks of your text are just one editorial benefit. Cheap? Thousands of words by phone lines, for less than express mail. And if the text is to be typeset, the cost will be half or less – the proofreading zero!

Password is USR's virtually automatic modem: 300/1200 baud, auto dial/answer, auto mode/speed select, two-year warranty. $449.* Telpac, the USR friendly telecommunications software package, $79. Write or call for complete descriptions – both Telpac and Password do far more than this!

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

*Suggested list for Password complete with power adapter, phone cable, RS232 interface.

Password, Telpac, and USR logo are trademarks of U.S. Robotics Inc.

How To Build An RS232 Circuit For HERO I

J. P. Weichert, Jr.
5347 Edith
Houston, TX 77096

Overview

This article presents information necessary to construct an RS232 circuit for HERO. For brevity, no attempt was made to give a complete step by step assembly. The modifications to HERO are minor and easily reversed. The circuit is constructed on the experimental circuit board. Selected cables within HERO have been used to bring the additional lines needed up to the experimental circuit board.

Refer to Heathkit's Microprocessor Interfacing Course for additional information on RS232 circuits.

The basic steps in construction of the RS232 circuit are as follows.

- Remove the head panel and rear panel to gain access to the cables.
- Make the necessary changes to the cable.
- Construct the circuit on the experimental circuit board.
- Build the RS232 cable that is appropriate for your microcomputer.
- Test the circuit.

Parts List

Part numbers are given when available. The list is complete except for the tools needed and such items as electrical tape and solder.

Qty.	Description	Part Number
1	6850 ACIA	Heath 443-1019
1	1488	Heath 443-794
1	1489	Heath 443-795
1	74LS00	Heath 443-728
1	AY-5-8166 Dual Baud Rate Generator	Radio Shack 276-1795
1	5.068 MHz Crystal	
1	10 Pin Connector	Heath 432-958
1	27K OHM (1/4 watt)	Heath 6-273-12
1	4.7K Ohm (1/4 watt)	
2	2C battery Connector	Heath 432-798
2	9 Volt batteries	
nFT	Three lead cable	
1	RS232 Connector for your Microcomputer	
3FT	Wire for Experimental Board	

Circuit Diagrams

The diagrams do not indicate the normally low lines such as CS2 on the 6850. Refer to appropriate data sheets for the full specification of a given chip. The diagrams show +12v and -12v as the ideal voltages to be used. Although the two 9 volt batteries will be used and are satisfactory.

```
-9v to 1488 PIN 1  -- 9V BAT ++++ TO GND
+9v to 1488 PIN 14 ++ 9V BAT ---- TO GND
```

```
                        6850 ACIA
                       ____________
            GND ---|1           CTS 24|---- GND
     1489 PIN 3 ---|2           DCD 23|---- GND
AY-5-8116 PIN 3 ---|3               22|-- D0
     6850 PIN 3 ---|4               21|-- D1
            GND-|5                  20|-- D2
     1488 PIN 2 ---|6               19|-- D3
            N/C-|7                  18|-- D4
           A6---|8  CS0             17|-- D5
6000H CHIP SELECT-|9  CS2           16|-- D6
           A7--|10 CS1              15|-- D7
           A0--|11 RS               14|-- 02(E)
          +5V--|12                  13|-- PIN 3 - 74LS00
                       ____________
```

Note: The two address select lines are normally high and go low when an address falls within the indicated range (4000-5FFF or 6000-7FFF).

```
                        74LS00
                       ________
               R/W --|1    14|-- +5V
   TO PIN 1 74LS00 --|2    13|
      6850 PIN 13 ---|3    12|      NOTE: This circuit
                     |4    11|      is needed to restore
                     |5    10|      the read/write(R/W)
                     |6     9|      signal since it is
                GND-|7      8|      inverted.
                       --------
```

```
                         1489
                       ________
            RS232 IN--|1    14|-- +5V
1488 PIN 1--27K OHM --|2    13|
       6850 PIN 2 ---|3    12|
                     |4    11|
                     |5    10|
                     |6     9|
                GND-|7      8|
                       --------
```

```
                         1488
                       ________
            -12V ----|1    14|--- + 12V
      6850 PIN 6 ---|2     13|
      RS232 OUT ----|3     12|
                     |4    11|
                     |5    10|
                     |6     9|
                GND-|7      8|
                       --------
                     AY-5-8166

                   ___5.068____
                   | MHZ CRYSTAL|     NOTE: As shown the
                   |  ________  |     circuit provides a 1200
                   '—|1    18|—'      baud clock signal for
                  +5V-|2    17|-N/C   the 6850. Refer to the
      PIN 3 6850 ——|3    16|-GND      data sheet for other
                  N/C-|4    15|-GND   baud rates.
                  N/C-|5    14|-GND
                  N/C-|6    13|-GND
                  GND-|7    12|-GND
+5V-4.7K OHM————————|8    11|-GND
                  N/C-|9    10|-N/C
                     ________
```

Experimental Board Layout

The asterisks represent the connector blocks. The R/W and INT are in the same position on the new block as they were on the old block. The other connector blocks that were changed are marked with their new values. The pin 1 locations were chosen to minimize the number of lines that had to be overlapped.

```
    6850        1489  1488  AY-5-8116 74LS00
 ___________  _____  _____  _________  _____
|          1| |   1| |   1| |        | |     |
|         24| |    | |    | |1       | |1    |
 ___________  _____  _____  _________  _____
 **** ****  **** **** **** **** **** ****
 D         D A        A               46  ERI
 7         0 0        7               00   WN
                                      00    T
                                      00
```

Connector Charts

Before constructing any of the connectors, remove the fuses from inside HERO. Also, to reduce any chance of a short circuit do not wear any jewelry (watches, rings, ets.). Unplug P316 and remove the leads from pins 150(BRN), 152(WHT), 153(VIOL), and 154(WHT/BRN). Leads can easily be removed with any pointed object small enough to be inserted into the back of the connector and simultaneously depress the raised catch and slip it back into the connector. After four leads are removed from P316, reinsert P316. Please note that the pin connectors and colors of the lead were taken from the block interconnect diagram. The assembly manual gives the pins as 149(BRN), 152(WHT), 154(VIOL), and 155(WHT/BRN). After the leads are disconnected, they should be verified as being connected to the same colors on P1304 through the use of a volt ohm meter.

Next, unplug P308. The BRN and WHT leads just removed from P316 will be used. Insert the BRN lead into slot 110 and the WHT lead into slot 109. At the lower edge on the far end of the CPU board when viewed from the access panel, is a ten pin plug. Insert the old P308, the new P406, in that position. The label on the plug should be visible and slot 109 should be on top.

Originally, P1301 was connected to P306. P1301 will be left unchanged. Unplug the original P306 and remove the lead from slot 130. The 'NEW P306' will be made from lead in the equivalent slot of a ten pin connector (see parts list). Plug the 'NEW P306' into the old P306 position. Insert the WHT/BRN lead from P316 into slot 130. Slot 129 will not have a connection. There are two ten pin brackets at the leading edge of the CPU board as viewed from the access door, plug the old P306 into the uppermost ten pin bracket. The label should be visible and slot 129 should be on top. This completes the new P407. (Note: Had the WHT/BRN lead on my HERO been connected to slot 24 of P403 to obtain the R/W signal, there would have been no need to move the connection.)

Three of the four leads from P316 have now been used. The remaining lead should be the violet(VIOL) lead. Remove the connector from the violet lead and strip about 5/16 inch of insulation. Carefully remove about 1/4 inch insulation from the grey lead in slot 25 of P403, twist the bare end of the violet lead around the bare spot on the grey lead, and solder. Wrap the two wires with electrical tape. This will bring the clock signal E up to the experimental board.

Of the four plugs on the experimental circuit board, only P1304 will be modified. Pins 7 through 10 remain the same. Move the WHT lead to pin 1, the BRN lead to pin 2, the VIOL lead to pin 5, and the WHT/BRN lead to pin 6 if not already in this position. Pins 3 and 4 should have no connection.

The following charts illustrate the new/old experimental circuit board connector block/cable connections to the old/new connections.

NEW	ORIGINAL CONNECTOR BLOCK	ORIGINAL PLUG P1301	TO	ORG. P306	OLD P306 NEW P407
D7	DI0	1		138 —	D7
D6	DI1	2		137 —	D6
D5	DI2	3		136 —	D5
D4	DI3	4		135 —	D4
		5-N/C			
		6-N/C			
D3	DI4	7		134 —	D3
D2	DI5	8		133 —	D2
D1	DI6	9		132 —	D1
D0	DI7	10		131 —	D0
				130-	R/W WHT/BRN TO P1304 PIN 6
				129-	N/C

```
     NEW
    P306
130-SLEEP-130
129- N/C -129
```

NEW	ORIGINAL CONNECTOR BLOCK	ORIGINAL PLUG P1302	TO	ORG. P308	OLD P308 NEW P406
A0	D00	1		118 —	A0
A1	D01	2		117 —	A1
A2	D02	3		116 —	A2
A3	D03	4		115 —	A3
A4	D04	5		114 —	A4
A5	D05	6		113 —	A5
A6	D06	7		112 —	A6
A7	D07	8		111 —	A7
		9		N/C-110 —	Address select 6000 BRN P1304 PIN 2
		10		N/C-109 —	Address select 4000 WHT P1304 PIN

Note: Pin 10 of P1302 has no connection. The abort line is a separate lead that connects to P1302.

ORIGINAL CONNECTOR BLOCK	ORIGINAL PLUG P1303
GND	1
GND	2
+5V	3
+5V	4
+12V	5
+12V	6
—	7

```
          ---           8
          ---           9
          ---           10

        ORIGINAL       ORIGINAL   NEW
       CONNECTOR        PLUG      PLUG
     NEW    BLOCK       P1304     P1304
CS 4000-5FFF  ---           1 --- WHT PIN 110 P406
CS 6000-7FFF  ---           2 --- BRN PIN 109 P406
                      GND   3 N/C
        N/C   ---           4 N/C
      02(E)   ---           5 N/C- VIOL TO P403
                                            PIN 25(GREY LEAD)
        R/W  R/W            6 R/W- WHT/BRN PIN 130 P407
        INT  INT            7 INT- INT

                      GND   8 N/C
                      GND   9 N/C
                    SLEEP  10 --- SAME
```

RS232 Cable

For my computer, I used an eight foot 3 conductor cable and a male RS232C connector. The cable is wired as a null modem cable. You may have to reverse lines 2 and 3, depending on your microcomputer. The leads that plug into the board were tinned with solder to make plugging and unplugging the leads easier.

EXPER. BOARD	RS232 CONNECTOR PIN
1489 PIN 1	2
1488 PIN 3	3
GROUND	7

Sample Software

As wired, the address of the 6850 ACIA is 60C0 hexadecimal.

The following program transmits characters A through Z from HERO to your microcomputer.

```
                          ORG   0100H
0100 86 03                LDA A  #03H   ; LOAD VALUE TO
                                        ;  RESET 6850
     B7 60 C0             STA A  60C0H  ; DUE SOFTWARE
                                        ;  RESET
     86 11                LDA A  #11H   ; DIVIDE CLOCK BY
                                        ;  16, 8 DATA BITS
     B7 60 C0             STA A  60C0H  ; PLUS TWO STOP
                                        ;  BITS
     C6 5B                LDA B  #5BH   ; CHARACTER Z+1
     86 40       RSET:    LDA A  #40H   ; BLANK CHARACTER
     B7 00 FF             STA A  00FFH  ; SAVE
     B6 00 FF    SEND:    LDA A  00FFH  ; LOAD CHARACTER
     4C                   INC A         ; INCREMENT
     B7 00 FF             CBA           ; COMPARE TO B
     2C F1                BGE    RSET   ; RESET IF GT ZERO
     B6 00 FF             LDA A  00FFH  ; ELSE RESTORE CHAR. TO
                                        ;  REGISTER A
     B7 60 C1             STA A  60C1H  ; AND SEND CHARACTER
     B6 60 C0    ASTAT:   LDA A  60C0H  ; LOAD STATUS
                                        ;  REGISTER
     46                   ROR A         ; SHIFT RIGHT TWO
                                        ;  BITS
     46                   ROR A
     24 F9                BCC    ASTAT  ; IF CARRY BIT CLEAR
                                        ;  CONT. CHECKING
     20 E7                BRA    SEND   ; STATUS ELSE GO
                                        ;  SEND NEXT
                                        ;   CHARACTER.
                          END
;
```

```
;  RECEIVE DATA FROM YOUR MICROCOMPUTER AND STORE IN HERO
;
                          ORG 0200H
0200 86 03                LDA A #03H   ; VALUE TO RESET
                                       ;  6850
     B7 60 C0             STA A 60C0H  ; RESET 6850
     86 11                LDA A #11H   ; CLOCK DIVIDE BY
                                       ;  16, 8 DATA BITS
     B7 60 C0             STA A 60C0H  ; PLUS 2 STOP BITS
     CE 03 00             LDX   0300H  ; LOAD ADDRESS TO
                                       ;  START SAVING
                                       ;   INCOMING DATA
     B6 60 C0    ASTAT:   LDA A 60C0H  ; GET STATUS BYTE
     46                   ROR A        ; SHIFT RECEIVED
                                       ;  BIT INTO CARRY
     24 FA                BCC   ASTAT  ; IF CARRY BIT
                                       ;  CLEAR CONTINUE
                                       ;   CHECKING
     B6 60 C1             LDA A 60C1H  ; LOAD INCOMING
                                       ;  CHARACTER
     A7 00                STA A 0,X    ; STORE CHARACTER
     08                   INX          ; INCREMENT INDEX
                                       ;  REGISTER
     20 F2                BRA   ASTAT  ; WAIT ON NEXT
                                       ;  CHARACTER
                          END
```

On my microcomputer I wrote a program that would read the assembly listing, extract the hexadecimal values of the instructions, convert the values to binary, and send the values out the communication line.

WATCHWORD

VERSION 2

The word processor and full screen editor for the Z100 and ZDOS.

NEW FEATURES:

- *Files of arbitrary length*
- *Extensive configuration facility*
- *Multiple fonts loaded from ALTCHR files*
- *Support for ZDOS Version 2 subdirectories*

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

For additional details, send a SASE.

Demo Disk: $3.00
WatchWord with Manual: $100.00

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

Assembly Language Subroutines With BASIC

Allen Gilchrist, Jr.
Route 2, Box 827
Rosharon, TX 77583

This article deals with the use of Microsoft's M80 assembler and assembly language subroutines with Microsoft's BASIC Compiler under CP/M. Assembly language subprograms can improve your BASIC programs in several ways. Speed of execution can be improved in critical areas, and individual bit manipulations can be easily done. There are applications for which assembly is still the best if not the easiest choice. Assembly routines can seem hard to write, but you don't need to become an accomplished "hacker" to use assembled subroutines.

Numerous programming languages are available for use on CP/M systems. There are several dialects of BASIC, COBOL, FORTRAN, RATFOR, PASCAL, C, LISP, LOGO, FOURTH, M80, RMAC, ASM, and others. Each of these has its supporters, some approaching religious zeal in their devotion. The fact is, any language has strengths and weaknesses. FORTRAN is well suited for precision floating point calculations, but doesn't handle other data types very well. PASCAL is good for teaching structured programming, but forces programmers to use its preconceived structure. BASIC is easy to learn, but is not modular and lends itself to sloppy programming techniques. C, a good modular systems programming language, lacks many mathematical functions like exponents or trigonometric functions. Assembly language is especially good for those applications requiring high speed (real time) data handling or very compact code. Some routines, however, can be very hard to write. The point is that each of these languages is particularly suited for certain problems and not to others.

A good mechanic would never go to work with only one screw driver or one wrench. The true professional will have a box full of different tools, each one specifically suited to a certain task. He will know which item to use for each job. Similarly, the professional programmer will have an arsenal of programming tools. He will know which ones to use, and when to use them. Most of us are not using our Heath computers professionally, but most have more than one programming language for our systems. We should not restrict ourselves to the use of only one language.

Particular advantages exist for the CP/M user who is at least somewhat familiar with assembly language. Over one hundred disk volumes of free public-domain software are available for virtually the price of the media. The programs range from games to compilers. One of the best modem programs available anywhere is MODEM7, available free! Many of the free programs include assembly source code and may require slight changes to work properly on a given system. Much can be learned just from reading the source code for these programs.

A knowledge of assembly language can be very useful to the BASIC programmer. BASIC is easy to learn and can be used for a variety of applications. Sometimes, interpreted BASIC is too slow for some real time tasks, and even compiled programs may need speeding up. In addition, there are certain tasks which are awkward in BASIC. Checking the status of individual ports requires bit checking which can be tedious. A particular program which is just a little too slow can benefit from the right assembly subroutine. For example, the BASIC source code listed below is a simple terminal emulator program.

```
1 '*************************************************
2 ' TERM300.BAS - January, 1984 - Allen Gilchrist  *
3 '*************************************************
5 DEFINT A-Z
6 ' Setup USART for 300 baud.
10 OUT 219,131
20 OUT 220,3
30 OUT 216,128
40 OUT 217,1
50 OUT 219,3
60 ' Main program loop
100 K$=INKEY$: 'check for keyboard input
105 ' if no input, go to 140
110 IF K$<CHR$(1) THEN 140
115 'get port status word and rotate right to fifth bit
120 S=INP(221)/32
125 'output character if port ready
130 IF INT(S/2)<>S/2 THEN OUT 216,ASC(K$)
135 'get port status byte
140 I=INP(221)
145 'get and display character if ready
150 IF INT(I/2)<>I/2 THEN PRINT CHR$(INP(216));
155 'loop back and do it again
160 GOTO 100
170 END
```

Lines 10 through 50 set the INS8250 asynchronous communications element to 300 baud, 8 bit operation as outlined on page 13-4 of the H-89 operating manual. The comments explain the operation of the main loop. Console input and output is handled by the INKEY$ function and the PRINT statement. Port status tests and I/O are not so easy. The status word for port number 216 is obtained in lines 120 and 140. If the least significant bit is a one, there is data at the port from the modem. If the sixth bit is a one, the port is ready to receive data for the modem. These conditions are determined in lines 130 and 150. This simple terminal emulator works quite well at 300 baud when compiled with BASCOM. It will not work at 1200 baud however. It is possible to carefully optimize this program to achieve 1200 baud operation, but that is not the purpose of this article. The use of two short assembly routines to handle the port I/O and status checks solves the problem. Before we can write the subroutines, however, we must know a little bit about assembly language.

Pat Swayne's series of REMark articles, "Getting Started With Assembly Language", is a good place to begin. A more complete treatment can be found in the "Assembly Language Programming" course available from Heath. The course, which can be applied to both CP/M and HDOS assemblers, starts at a beginners level but moves rapidly. The example programs won't all run the first time, but getting them to go can be part of the learning experience. Assembly source codes obtained with HUG or public domain software can also be helpful. There are numerous books on 8080 and Z80 programming available at most book stores. I have several of these, but don't think any are as good as the Heath course.

A first step in writing subroutines is to learn to use the Microsoft M80 macro assembler. This is not necessarily an easy task. A source file that assembles and loads with Digital Research's ASM may not necessarily do so with M80. The first difference is that most labels in M80 source files must be followed by colons (:). ASM does not require colons, but will allow them. It is probably a good idea to get in the habit of using the colon after each label. Be cautious, however, labels preceeding the EQU and SET pseudo-opcodes must not have the colon. The ORG assembler directive also causes problems and is best left out of the source files for M80. The following source listing is a very simple assembly language program which clears the H/Z-89 screen.

```
;       CLEAR.ASM       A VERY SIMPLE ASSEMBLY PROGRAM
;       ALLEN GILCHRIST - JANUARY, 1984
;**********EQUATES**********
ESC     EQU     027
BDOS    EQU     05
;**********MAIN PROGRAM**********
START   ORG     0100H
        LXI     H,0     N       ; LOAD H-L PAIR WITH 0
        DAD     SP              ; ADD STACK POINTER TO
                                ;  H-L
        LXI     SP,STACK        ; SETUP LOCAL STACK
        PUSH    H               ; CP/M SP ON LOCAL STACK
        MVI     A,ESC           ; PUT ESCAPE IN REGISTER
                                ;  A
        CALL    PCHAR           ; OUTPUT IT TO THE
                                ;  SCREEN
        MVI     A,'E'           ; PUT AN 'E' IN REGISTER
                                ;  A
        CALL    PCHAR           ; OUTPUT IT TO THE
                                ;  SCREEN
        POP     H               ; GET CP/M SP FROM STACK
        SPHL                    ; SWAP SP WITH H-L
        RET                     ; AND RETURN TO CP/M
;**********SUBROUTINE**********
PCHAR   PUSH    B               ; SAVE REGISTERS ON STACK
        PUSH    D
        PUSH    H
        MVI     C,2             ; SET REGISTERS FOR CP/M
        MOV     E,A             ; CONSOLE OUTPUT,
        CALL    BDOS            ; AND CALL BDOS
        POP     H               ; RESTORE REGISTERS
        POP     D
        POP     B
        RET                     ; RETURN TO MAIN PROGRAM
;**********DEFINE STORAGE**********
        DS      20              ; SPACE FOR 10 PUSHES
STACK   DS      0               ; IN LOCAL STACK
        END     START
```

This program was written for ASM. The first five lines of the main portion of the program setup a local stack and save the CP/M stack pointer on it. The next four lines send ESC E to the console to clear the screen. The subroutine PCHAR saves the registers on the stack, sets the C and E registers to the correct values, and calls BDOS to output the characters. The registers are then restored and control is returned to the main program. Some BDOS calls alter the registers, so it is a good idea to save the registers before the BDOS call and restore them afterward. After control returns to the main program for the last time, all that remains is to restore the CP/M stack pointer and return to the monitor. The use of a separately defined stack with a simple return at the end of the program allows a return to CP/M without the need for a warm boot. The second listing is the same program, but for M80.

```
;       CLEAR.MAC       A VERY SIMPLE ASSEMBLY PROGRAM
;                       FOR M80. ALLEN GILCHRIST - JAN,
;                       1984
;**********EQUATES**********
ESC     EQU     027
BDOS    EQU     05
;**********MAIN PROGRAM**********
START:  LXI     H,0             ; LOAD H-L PAIR WITH 0
        DAD     SP              ; ADD STACK POINTER TO
                                ;  H-L
        LXI     SP,STACK        ; SETUP LOCAL STACK
        PUSH    H               ; CP/M SP ON LOCAL STACK
        MVI     A,ESC           ; PUT ESCAPE IN REGISTER
                                ;  A
        CALL    PCHAR           ; OUTPUT IT TO THE
                                ;  SCREEN
        MVI     A,'E'           ; PUT AN 'E' IN REGISTER
                                ;  A
        CALL    PCHAR           ; OUTPUT IT TO THE
                                ;  SCREEN
        POP     H               ; GET CP/M SP FROM STACK
        SPHL                    ; SWAP SP WITH H-L
        RET                     ; AND RETURN TO CP/M
;**********SUBROUTINE**********
PCHAR:  PUSH    B               ; SAVE REGISTERS ON
                                ;  STACK
        PUSH    D
        PUSH    H
        MVI     C,2             ; SET REGISTERS FOR CP/M
        MOV     E,A             ; CONSOLE OUTPUT,
        CALL    BDOS            ; AND CALL BDOS
        POP     H               ; RESTORE REGISTERS
        POP     D
        POP     B
        RET                     ; RETURN TO MAIN PROGRAM
;**********DEFINE STORAGE**********
        DS      20              ; SPACE FOR 10 PUSHES
STACK:  DS      0               ; IN LOCAL STACK
        END     START
```

Notice that the ORG directive has been left out of the M80 source program, and that all labels, except for EQU statements, have colons. The Microsoft Macro 80 Reference manual indicates the need for the colons, but it does not say anything about not having them for certain pseudo-opcodes or that the ORG statement may cause problems. I learned this by trial and error. After discovering these things, I found that the information is included at the beginning of chapter three of Alan Miller's book, "Mastering CP/M". This book appears to be a good source for those wishing to learn something about macro-assemblers in general. Apparently, any label which corresponds to a memory address must be followed by a colon.

Now that we can use M80, the next step is to write a subroutine. The simplest call is one with no parameters passed to the subroutine. Consider the following BASIC source listing.

```
1 '*****************************************************
2 ' HELLO.BAS - Allen Gilchrist - January, 1984        *
3 '*****************************************************
4 '
10 PRINT "HELLO FROM BASIC BEFORE CALL"
20 CALL HIMAC
30 PRINT "WELCOME BACK TO BASIC"
40 END
```

This program prints an initial message on the screen, calls the subroutine, HIMAC, and then prints another message on the screen after control returns to the main program. An appropriately simple assembly subroutine follows.

```
; HIMAC.MAC - A very simple subroutine
; ALLEN GILCHRIST - JANUARY, 1984
;******* Equates and definitions *******
TYPE    EQU     9
BDOS    EQU     005
HELLO:  DB      'Greetings from the M80
```

```
        DB      ' subroutine',10,13,'$'
;******* subroutine *******
        NAME    ('HIMAC'); module name (optional)
        ENTRY   HIMAC    ; or PUBLIC   HIMAC defines
                         ; entry point for the module.
HIMAC:  PUSH    PSW      ; save registers
        PUSH    B
        PUSH    D
        PUSH    H
        LXI     D,HELLO  ; point to "HELLO" string
        MVI     C,TYPE   ; setup for BDOS call
        CALL    BDOS     ; and call BDOS
        POP     H        ; restore registers
        POP     D
        POP     B
        POP     PSW
        RET              ; return to calling program
        END              ; No start label with end statement
```

This is quite similar to the PCHAR routine in the CLEAR program above, except that a different BDOS function is called. The PUSH PSW and POP PSW statements save and restore the A and FLAG registers. A greeting is printed on the screen and then control is returned to the calling program. An ENTRY or PUBLIC statement or a double colon (HIMAC::) following the label must be used to define the entry point. In this example, "HIMAC:" is the entry point. The END statement must not have a start label specified. The start address of the subroutine is taken care of by the L80 loader.

The BASIC main program, HELLO.BAS, can be created with any editor and then compiled with the command,

```
A>BASCOM =HELLO <cr>.
```

If all is well, a relocatable file, HELLO.REL, will be produced on the default drive. Similarly, the subroutine, HIMAC.MAC, can be created and then assembled with the command,

```
A>M80 =HIMAC <cr>.
```

Again, if everything is alright, another relocatable file, HIMAC.REL, will be generated on the disk. These can be linked and loaded using Microsoft's L80.

```
A>L80 HELLO,HIMAC,HELLO/N/E <cr>.
```

This will produce an executable file, HELLO.COM, which can be run in the usual way. This little example may not be very practical, but it serves to illustrate how an assembly language subroutine can be written and called from a BASIC main program.

Now back to our original problem, the BASIC terminal program. We will need to pass parameters to and from the subroutines. A subroutine may be called with a CALL statement or a USR function call. These are discussed in appendix E of the Microsoft BASIC Compiler manual. The USR function call is normally used with interpreted BASIC, and the CALL statement is recommended for compiled programs. Fred Hochschild's article in the September, 1983 issue of REMark deals with calling assembly language subroutines from BASIC under HDOS. Fred describes the USR function call with parameters passed in the FAC (floating point accumulator).

The CALL statement passes parameter addresses in the registers. If only one parameter is needed the address is passed in the HL register pair, if two are passed the DE pair is also used, and when three are needed the BC register pair is pressed into service. According to the manual, if more than three parameters are to be passed, the BC register pair stores the beginning address of a continuous block of memory storing the remaining values. The terminal program will only need to pass one parameter.

The following source listing is the BASIC terminal program with subroutine calls.

```
1 '*********************************************************
2 ' TERM1200.BAS - 1200 baud modem program - Jan. 5, 1984 *
3 '*********************************************************
5 DEFINT A-Z
7 PORT=216: STAT=PORT+5: ONE = 1
9 ' Set the USART FOR 1200 BAUD
10 OUT PORT+3,131
20 OUT PORT+4,3
30 OUT PORT,96
35 OUT PORT+1,0
50 OUT PORT+3,3
80 ' Main loop
90 CH$=INKEY$: 'Check keyboard
100 IF CH$<CHR$(1) THEN 140: 'If no character go to 140
120 CH=ASC(CH$): 'convert to an integer
130 CALL PRTOUT(CH): 'Output to port
140 CALL PORTIN(PC): 'check port
150 IF PC<ONE THEN  90: 'if no data at port go to  90
160 PRINT CHR$(PC);: 'print received character on screen
170 GOTO  90
180 END
```

Port status checking and I/O are handled in two subroutines, PRTOUT and PORTIN. Notice how much simpler this BASIC main program is. The source code for the subroutines PRTOUT and PORTIN is given below. Parameters are passed as integers.

```
; TERMSUBS.MAC - Allen Gilchrist - Jan., 1984
        STAT    EQU     221     ; Status address
        PORT    EQU     216     ; Port address
PRTOUT::IN      STAT            ; Get status byte
        ANI     32              ; If bit 5 is on
                                ; Port is ready
        RZ                      ; Return if port not
                                ;  ready
        MOV     A,M             ; Move character to A,
        OUT     PORT            ; and output it to Port.
        MVI     A,0             ; Put a zero into
        MOV     M,A             ; memory and
        RET                     ; return.
PORTIN::IN      STAT            ; Get status byte
        ANI     01              ; If bit 1 is on
                                ; Port has input
        JNZ     READY           ; Jump to label READY
        MOV     M,A             ; Put the zero into
        RET                     ; memory, and return.
READY:  IN      PORT            ; Get data from the port
        MOV     M,A             ; Move the data to
        RET                     ; memory and return.
        END
```

Both subroutines are included in the same source file with their respective entry points designated by double colons. Since the address of the integer parameter in each case is passed in the HL register pair, no special effort is needed to get the values as long as the HL pair is not altered before accessing memory. Since no BIOS functions are called, there is no need to save and restore the registers with PUSH and POP statements. These routines are not much more difficult than their BASIC language equivalents, and they work much faster. This little program will work nicely at 1200 baud. There are many fine smart terminal emulator programs available commercially or in the public domain. Occasionally there is a need for a completely dumb terminal program which does nothing but pass keystrokes to the port and print port data on the screen. This program has no exit other than SHIFT - RESET.

Assembly language subroutines can be used with compiled BASIC main programs to your advantage for many applications. It is not necessary to be an experienced hacker, but you must be somewhat familiar with assembly language, and you must be able to use the Microsoft M80 macro assembler. Perhaps this introduction to the use of M80 will help someone get started. Who knows, it might be fun!

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.

885-8025-37 ZDOS FAST EDDY Text Editor and BIG EDDY $20.00

Introduction: FAST EDDY is a text file screen editor that was written for everybody. It was written using the basic commands and keypad keys, so that anyone, even with no experience with an editor, can learn to use it while reading the instructions.

For those files that are too large for your computer's memory, BIG EDDY will handle the breaking up of the text for editing with FAST EDDY.

Requirements: This disk requires the ZDOS operating system on an H/Z-100 computer. A printer is not required, but both FAST EDDY and BIG EDDY have printer options. Only one disk drive is required.

BIG EDDY can be used with large files. A second drive (or high density drives) may be required to break up large files which cannot fit into memory. The original file is not changed or deleted.

The following files are included on the HUG P/N 885-8029-37 ZDOS FAST EDDY Text Editor and BIG EDDY File Handling Utility:

EDITOR	.COM
BIGED	.COM
BIGED	.DOC
INSTRUCT	.DOC
TUTOR1	.DOC
TUTOR2	.DOC
TUTOR3	.DOC
RETURN	.COM

Author: Hubert L. Reeder

FAST EDDY -- This text file screen editor and its documentation have been designed for anyone not familiar with using an editor. The program uses commands and keys that are easy to remember and use.

The editor contains a limited number of commands, however, the commands are designed to provide a useful, easy-to-use editor. It does not have complex options that require time and effort to use.

The editor contains a command mode and edit mode. The following are a brief list of the options:

COMMAND MODE

Typed Commands:

LOAD filename.ext	(load file)
SAVE filename.ext	(save file)
SAVE XX filename.ext	(save XX number of lines)
MERGE filename.ext	(merge two files)
PRINT	(print enter file, NN lines per page)
PRINT NN	(print double spaced)
FIND anyword	(find the first occurence of a word)
MARGIN nn xx	(set left margin, nn, right margin, xx)
CMPRESS	(replace spaces with tabs in new text)
EXPAND	(cancel the CMPRES command)
BYE	(exit to CP/M)

Key Commands:

Up arrow -- enter EDIT mode at first line of text
Down arrow -- enter EDIT mode at last line of text
HOME -- enter EDIT mode at pointer (last cursor location)
DELETE -- cancel partial commands or stop printer
F0 -- erase all text

EDIT MODE

Key Commands:

Up arrow -- move cursor up one line
Down arrow -- move cursor down one line
Right arrow -- move cursor to the right one character
Left arrow -- move cursor to the left one character
HOME -- return to COMMAND mode
IL -- insert line
DL -- delete line
IC -- insert character
DC -- delete character
f0 -- block erase
f1 -- align paragraph within left and right margins
f2 -- justify right
f3 -- indent on/off
f4 -- margin off
f5 -- split line
f6 -- find next occurence of word, after FIND of COMMAND mode
f7 -- move backward in text
f8 -- move forward in text
f9 -- center line
f10 -- tab set/release
f11 -- jump left
f12 -- jump right

These are most of the basic commands of FAST EDDY. Please note that it has the ability to align paragraphs to new margin settings and then the option of right justifying the paragraph text.

Details of how to use these options are contained in the documentation. The TUTOR1, TUTOR2, and TUTOR3 documentation files are included with the disk to give the user experience in using FAST EDDY while reading the doc files.

BIG EDDY -- This program is a utility to work with text files which are too large to be edited by FAST EDDY directly because of memory limitations. BIG EDDY can be used to browse a file of any size of which the user can break the large file into smaller parts for editing with FAST EDDY.

BIG EDDY asks for the input filename and an output filename. It keeps track of the subfiles and names them accordingly.

BIG EDDY has some useful options to aid the user in preparing the text for smaller files. The BROWSE mode is similiar to the EDIT mode of FAST EDDY, except that no editing can be done to the file.

The following are a list of the commands of BIG EDDY:

SAVEALL -- save the entire text in memory to the disk
SAVEPART -- save part of the text in memory to disk
NOSAVE -- discard part of text
PRINT -- same as FAST EDDY's print commands
BYE -- exit to CP/M

With the SAVEPART command, the user can save the text by subject or modules of his choice. Using the CP/M PIP program, the subfiles can assemble the files into any order.

Comments: This version of FAST EDDY, with the editing features, e.g. align paragraph and right justify, allow formatting features that make it a powerful, easy-to-use editor.

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

IT'S IMPORTANT!!

THAT YOU SIGN UP NOW FOR THE

International Heath/Zenith Users' Group Conference
Heath/Zenith Users' Group
Saint Charles, Illinois • July 27, 28 & 29, 1984

Heath/Zenith Users' Group

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	Decription 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
885-5002-37	CP/M-86 HUG Editor	$ 20.00	5-5
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
885-3011-37	ZDOS ZBASIC Games Disk	$ 20.00	5-5
885-3012-37	ZDOS HUG Editor	$ 20.00	5-5
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.

Connecting the H/Z-100 to a Gemini Printer

Jerome H. Horwitz
14413 Ansted Road
Silver Spring, MD 20904

The following describes how to connect an H/Z-100 computer to a Star Micronics Gemini printer with parallel input. This has been tested on a Gemini 10X printer, but should also work with a 10 or 15, since with one minor exception (which is not used), the manuals indicate that the connections are identical.

The parallel-interface Gemini printers are Centronics "compatible", and have an Amphenol "Blue-Ribbon" type connector. The mating connector for the printer-end of the interconnecting cable should be an Amphenol type 57-30360 or equivalent. This has 36 pins and an overall shape similar to that of the familiar DB25 D-subminiature type, but it is larger and the internal construction is different.

Standard configured H/Z-100 computers have three interface ports on their rear panel, all of which use DB25 D-subminiature 25-position connectors. J1 (female) and J2 (male) are serial ports. J3 (female) is the parallel port connector to be used with a parallel interface printer. The connector needed for the computer-end of the interconnecting cable must be male.

Both of the connector types needed should be widely available, although the author had more difficulty locating the Centronics type connector than the DB25. The Centronics type connector complete with metal back shell is about $9.00. The DB25 with back shell (which is sold separately) is about $5.20 at Radio Shack (catalog numbers 276-1547 and 276-1549).

The interconnecting cable should be of good quality and shielded. Thirteen signal conductors are required, not counting signal returns, grounds, or shield. In order to provide a separate insulated conductor for the signal ground and at least one for the designated returns, a shielded cable with at least 15 conductors is needed. Any additional conductors in the cable used can be left as spares or connected to separate the various return lines. As many as 22 conductors may be used, but only 15 are necessary. A ten-foot length of good quality cable bought new in small quantity will cost about $8.00 to $12.00, so the cost for cable and connectors will total about $25 or so.

Before proceeding further, it is worth stating that a cable is a critical element in any system. A poorly made cable will cause no end of problems. These can include excessive radio frequency interference caused by radiation through inadequate or poorly connected shields, intermittent operation or no operation at all, or, even worse, equipment damage. Therefore, one should not attempt to make his own cable unless he is capable and experienced at soldering, has the patience to do the fine and tedious work required, and knows what he is doing! Connections must be well made, adequately separated and insulated, and properly supported by the connector shell in order to last.

The connections needed between the Gemini printers and the H/Z-100 are listed in Table I. The wire colors are those the author used, and were determined by separating the wires in a sequence that provided the best physical fit to the arrangement of the connector pins to be wired. Figures 1 and 2 depict the layouts of the connectors used at each end of the cable.

Care and skill are required to make a good cable, but a few hints may improve the results. Knives or razor blades used to cut and trim insulation should be sharp. Use a small soldering iron (one suitable for printed circuit work) as there is not much room around each pin, and keep it clean and tinned. Do not overheat joints, as the wire insulation or the connector can be damaged. Strip the cable outer jacket carefully (do not cut the braid or foil shield) so that the exposed wires are just long enough to reach from the cable entry (usually in the center of the connector shell) to the farthest corner pin, with no extra. Other wires should be cut shorter, so that the cable jacket can end up well inside the shell after assembly. Each wire should have a short (about 1/2 inch) piece of sleeving slipped over it before soldering. This sleeving is pushed back down over the exposed part of the connection after soldering to prevent shorts between adjacent connections. It is also good practice to slip a two- or three-inch long piece of sleeving over the cable jacket for strain relief. This should be slid down inside the shell prior to fastening the cable clamps. And be sure that the shell and sleeving are on the cable prior to assembling the connector!

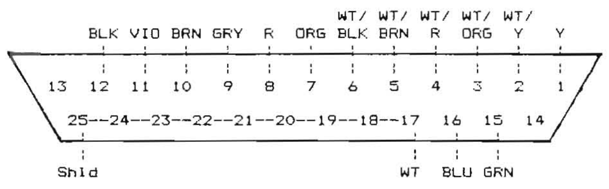


Figure 1. Layout of pins for DB25 connector at computer end of cable.

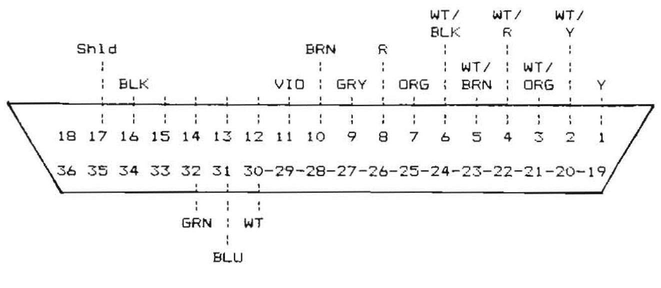


Figure 2. Layout of pins for Centronics compatible connector at printer end of cable.

Table I. Connections between H/Z-100 series computers and Gemini printers with parallel-interface.

SIGNAL NAME*	PRINTER PIN	PRINTER IN/OUT	H/Z-100 PIN	DESCRIPTION OF CABLE CONNECTION
-STROBE	1	IN	1	YELLOW
DATA 1	2	IN	2	WHITE/YELLOW
DATA 2	3	IN	3	WHITE/ORANGE
DATA 3	4	IN	4	WHITE/RED
DATA 4	5	IN	5	WHITE/BROWN
DATA 5	6	IN	6	WHITE/BLACK
DATA 6	7	IN	7	ORANGE
DATA 7	8	IN	8	RED
DATA 8	9	IN	9	GRAY
-ACK	10	OUT	10	BROWN
BUSY	11	OUT	11	VIOLET
PAPER END	12	OUT	---	(Not used)
SELECTED	13	OUT	---	(Not used)
----------	14**	OUT	---	(Not used)
----------	15	OUT	---	(Not used)
SIGNAL GROUND	16	---	12	BLACK
CHASSIS GROUND	17	---	17-25—|	FOIL SHIELD +
			|	DRAIN WIRE
+5 VOLTS DC	18	OUT	--- |	(Not used)
RETURN PIN 1	19----|	---	17----|	WHITE--Pins 19-30
RETURN PIN 2	20----|	---	18----|	at the printer
RETURN PIN 3	21----|	---	19----|	connector are
RETURN PIN 4	22----|	---	20----|	bussed
RETURN PIN 5	23----|	---	21----|	together;
RETURN PIN 6	24----|	---	22----|	pins 17-25
RETURN PIN 7	25----|	---	23----|	at the
RETURN PIN 8	26----|	---	24----|	computer
RETURN PIN 9	27----|	---	25----|	connector are
RETURN PIN 10	28----|	---	---	bussed
RETURN PIN 11	29----|	---	---	together.
RETURN PIN 12	30----|	---	---	
-INPUT PRIME	31	IN	16	BLUE (H/Z-100 name is "INIT")
-ERROR	32	OUT	15	GREEN
EXTERNAL GROUND	33	---	---	(Not used)
----------	34	---	---	(Not used)
----------	35	---	---	(Not used)
----------	36	---	---	(Not used)

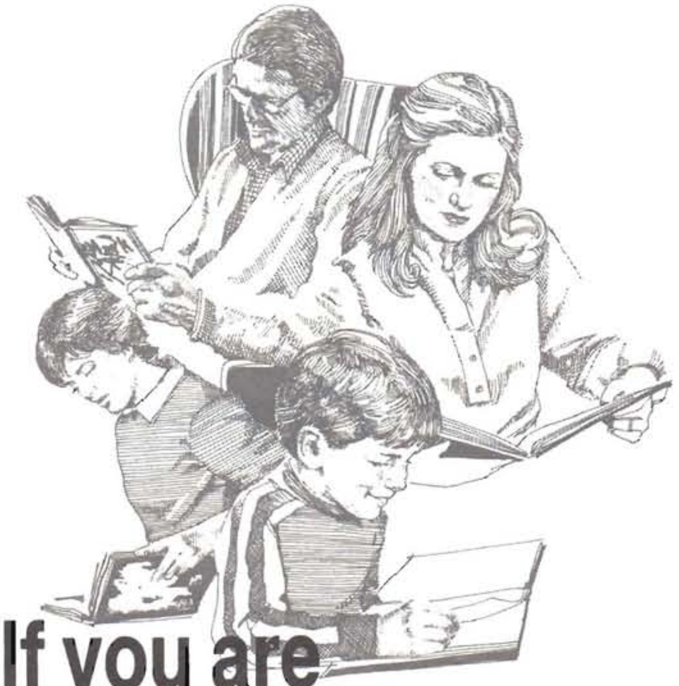
* A "-" before a signal name denotes the complement (in lieu of overscore).

** The manual for the Gemini 10 and 15 printers lists pin 14 as "-AUTOFEEDXT", which is described as an optional automatic line feed. This pin is listed as unused in the preliminary manual for the Gemini 10X.

IT'S IMPORTANT!!

THAT YOU SIGN UP NOW FOR THE INTERNATIONAL HUG CONFERENCE

see registration on page 5 of this issue



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

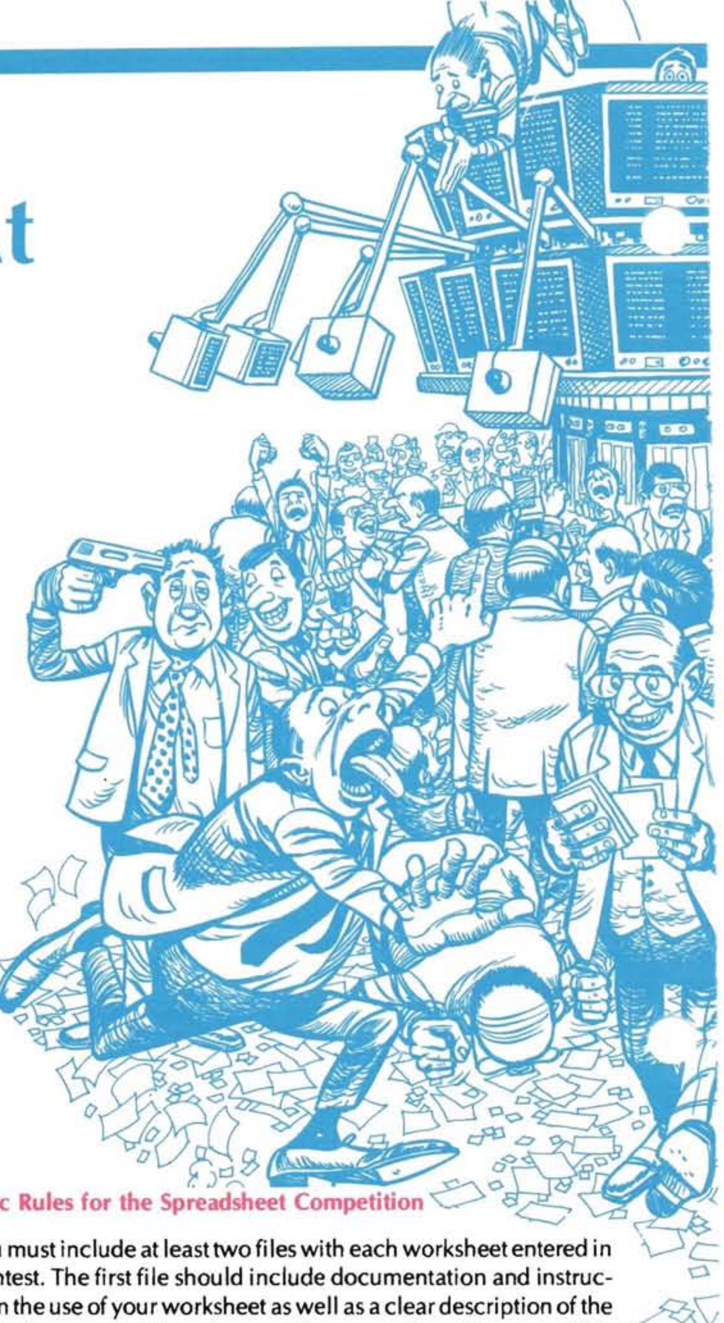
maybe now is the time to join the National Heath/Zenith Users' Group. You will receive:

- *a copy of **REMark** filled with new and exciting articles and programs each month*
- *access to the HUG library filled with a large variety of programs*
- *discounts on a variety of Heath/Zenith computer products (see REMark January, 1984 issue for more details)*

And remember, your local HUG is an excellent source of information, support and comradery. A membership package from the National Heath-/Zenith Users' Group contains a list of current local HUG clubs as well as other interesting information.

Heath/Zenith Users' Group

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

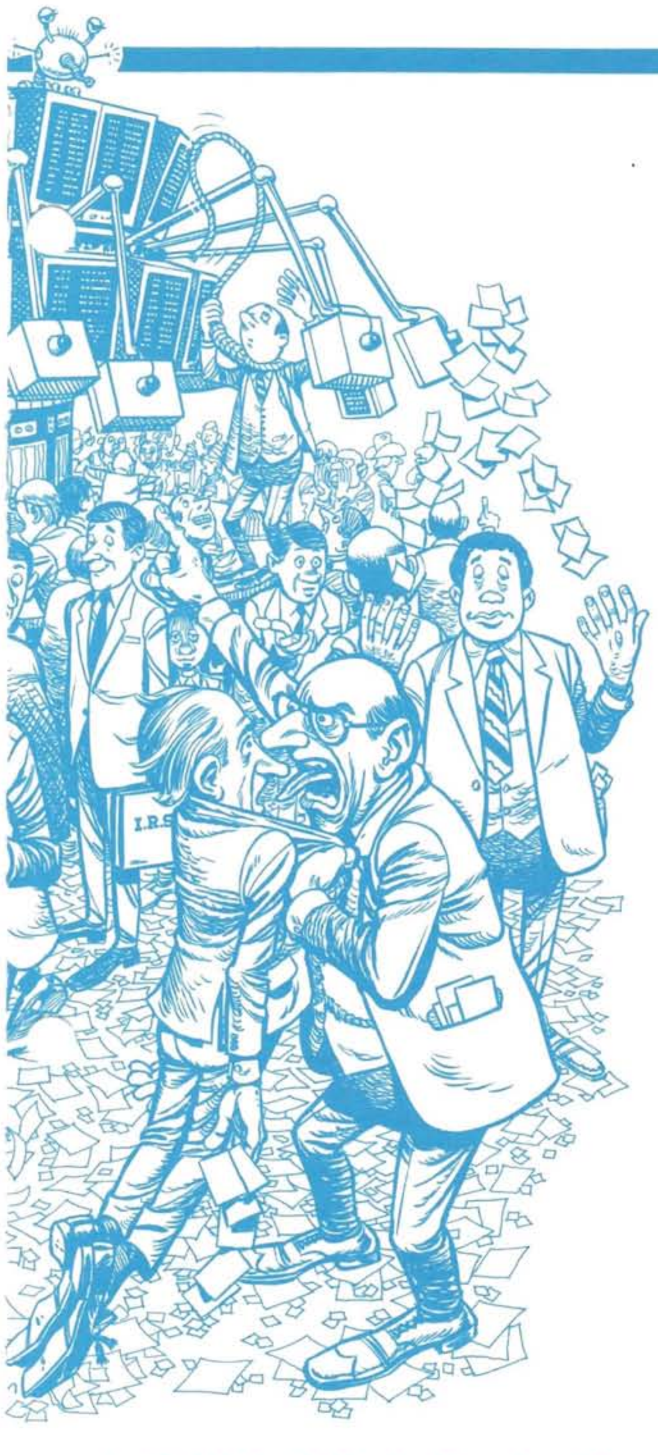
1. You must include at least two files with each worksheet entered in the contest. The first file should include documentation and instructions on the use of your worksheet as well as a clear description of the results to be expected from the use of your creation. The second file should be the worksheet 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.

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

Using A Speech Synthesizer On An H/Z-89

Bill Boyd
3617 Beechollow Drive
Memphis, TN 38128

I had seen the Type-'N-Talk Text-to-Speech Synthesizer in the Heath Catalog, but was scared away by the note: Not recommended for novices at this time. But at the second National HUG Conference, I saw the new model, the VOTRAX Personal Speech System, and fell in love with it. Plugged into the H/Z-89 in place of a printer, this unit will speak whatever text is thrown its way. It contains its own speaker and power supply, parallel as well as serial ports, and a row of setup switches on the back to match it to your computer. It also plays music and has a real time clock and various alarms. I bought one. Even though it is simple and easy to use, some tips might be helpful. Here is a beginner's guide for CP/M users.

Getting Started

The best way to start is to plug the Speech System into port 320Q, the middle one of the three serial ports in back of the H/Z- 89. This is normally the TTY port and is simplest because no "handshaking" is required. Put all the Speech System's configuration switches down. Now push up numbers 1 and 3 to set the unit to 300 BAUD and number 6 to cause the unit to speak a message when powered up. Insert a bootable disk into your computer and run CONFIGUR. When CONFIGUR asks

```
Standard system (Y or N)? <Y>:
```

press the Y key. When the system returns to CP/M, reset the computer and do a cold boot. This last step is crucial. Not all of the CONFIGUR commands implement properly without saving to disk and rebooting. In this case without rebooting, the Speech System will speak "error seven" and nothing else. A call to Studio Computers, who sold me my system, brought me this tip and ended a long night of frustration.

Now let's run a test. Type

```
PIP TTY:=CON:
```

and press the ENTER key. PIP will then channel all input from the CONsole to the TTY port, where we have connected the Speech System. Anything typed on the keyboard will be spoken by the Speech System. Type in your name and press RETURN; type in anything your wish: Votrax will speak it. Your system works! Note that you are connected directly to the Speech System and can try out all of its commands. For example, set the time to 7:30 a.m. by typing "[escape]T073000". Have it tell you the time by typing "[escape]T".

When you are tired of letting your fingers do the talking, type a CONTROL-Z (hold the control key down while you type a Z). This will return you to CP/M.

Getting Started In MBASIC

Now that you know the Speech Synthesizer works, let's see what you need to do to use it in BASIC. MBASIC has the LPRINT command for sending output to a printer. You could reCONFIGUR your H/Z-89 so that this output goes to the Speech System instead. However, this means you couldn't use the printer. What is needed is a switch allowing changes from printer to Speech System and back. Try the following simple program, SWITCH.BAS

```
10 'SWITCH.BAS SWITCHES LPRINT FROM PRINTER TO SPEECH
   SYSTEM AND BACK
20 POKE 3,41            'SWITCH TO VOTRAX AT 320Q
30 LPRINT "TESTING THE VOTRAX PERSONAL SPEECH SYSTEM"
40 POKE 3, 169          'SWITCH BACK TO PRINTER AT 340Q
50 LPRINT "TESTING THE PRINTER"
60 END
```

Run this program with the Speech System connected to the TTY port (port 320Q, the middle serial port of the H/Z-89) and the printer connected to the LPT port (port 340Q, the outermost port). The message in line 30 should be spoken by the Speech System and the message in line 50 should be printed by the printer. Now you have it: to switch to the Speech System at port 320Q, write the decimal number 41 to memory location 3. And to switch back to the printer at port 340Q, write the decimal number 169 to the same memory location.

As simple as this is to do, understanding it is almost as simple. CP/M reserves memory location 3, the so-called IOBYTE, as a table to direct output from what it calls "logical devices" to "physical devices". CONFIGURed to the Standard Heath System, this IOBYTE has binary value 10101001, which is decimal 169. The leftmost two bits assign the logical device LST to port 340Q when they are set to 10 and to port 320Q when set to 00. The statement POKE 3,41 changes the value of IOBYTE to 00101001, thus sending the LST output to port 320Q. The statement POKE 3,169 changes the IOBYTE back to its original value, thus sending the LST output to port 340Q again.

Using the Speech System In BASIC

Now that you have the Speech System interfaced to the H/Z-89 and are able to switch back and forth between it and the printer, try the program in Listing 1. A simple program for adding two numbers and printing the result, it prompts the user with a combination of verbal and on-screen commands. Near the beginning is a routine for asking the user whether he wants to play again or quit. At line 130 is the main routine for inputting the numbers to be added and speaking the result, while simultaneously printing it on the terminal screen. At line 250 is a routine for passing time while waiting for the Speech System to finish talking. And, of course, lines 20 and 290 switch the LPRINT command to the Speech System and then back to the printer.

Let us look more carefully at the time delay routine. In line 40, the variable PHRASE$ is set equal to the expression to be spoken. LPRINT is then used to send this expression to the Speech System, and the routine at line 250 is called. First the routine sets L% equal to the number of characters in PHRASE$. It then goes through a loop 70*L% times to create a delay long enough for the Speech System to

```
10 ' ADDUM.BAS - ADDS TWO NUMBERS USING VOICE PROMPTS
20 POKE 3,41                        'SWITCH LPRINT TO THE VOTRAX PORT
30 ' BEGINNING - PLAY OR QUIT
40 PHRASE$="DO YOU WANT ME TO PLAY CALCULATOR WITH YOU?":LPRINT PHRASE$
50 GOSUB 250                        'DELAY FOR SPEECH
60 PRINT "ANSWER Y FOR 'YES' OR N FOR 'NO'"
70 Q$=INPUT$(1)
80 IF Q$=CHR$(0) THEN 70            'CYCLE BACK WAITING FOR
                                     RESPONSE
90 IF Q$="Y" OR Q$="y" THEN 130
100 PHRASE$ = "I'M SORRY YOU DON'T WANT TO PLAY.  HAVE A GOOD DAY. BYE NOW."
110 LPRINT PHRASE$:GOSUB 250
120 GOTO 290                        'QUIT
130 ' MAIN ROUTINE - INPUT TWO NUMBERS AND ADD THEM
140 PRINT
150 PHRASE$ = "ENTER YOUR FIRST NUMBER":LPRINT PHRASE$:GOSUB 250
160 PRINT "FIRST NUMBER IS  ";:INPUT X1%
170 PHRASE$ = "ENTER YOUR SECOND NUMBER":LPRINT PHRASE$:GOSUB 250
180 PRINT "SECOND NUMBER IS ";:INPUT X2%
190 S%=X1%+X2%
200 PRINT "   THE SUM IS     ";S%
210 PHRASE$ = "THE SUM IS ":LPRINT PHRASE$;S%:GOSUB 250
220 PRINT:PRINT:PRINT:L%=15:GOSUB 270
230 PHRASE$ = "DO YOU WANT TO DO ANOTHER?":LPRINT PHRASE$:GOSUB 250
240 GOTO 60
250 ' THIS ROUTINE CREATES ADEQUATE DELAY FOR THE SPEECH UNIT TO FINISH
260 L%=LEN(PHRASE$)                 'DETERMINE LENGTH OF SPOKEN PHRASE
270 FOR I% = 1 TO 70*L%:NEXT I%
280 RETURN
290 POKE 3,169                      'SWITCH LPRINT BACK TO PRINTER PORT E0
300 END
```

Listing 1

```
LIST:   LDA     IOBYTE
        (lines omitted for brevity)
        CALL    INDXIT
        DW      LPT2OUT         ;0: NEW VOTRAX PSS (replaces TTYOUT)
        DW      CRTOUT          ;1: CRT
        DW      LPTOUT          ;2: LPT
        DW      DBD             ;3: DIABLO

PUNCH:  LDA     IOBYTE
        (lines omitted for brevity)
        CALL    GOTOIT
        DW      TTYOUT          ;0: TTY
        DW      LPT2OUT         ;1: NEW VOTRAX PSS (replaces DMYOUT)
        DW      MDOUT           ;2: UP1 MODEM PORT OUTPUT
        DW      CRTOUT

LPT2OUT PUSH    A               ;SAVE REGISTER A ON STACK
        LDA     MODE            ;REVERSE BIT 2
        XRI     04H             ; OF MODE
        STA     MODE            ; BYTE
        MVI     A,0D0H          ;CHANGE PORT FROM E0
        STA     H84PT3          ; TO D0
        POP     A               ;RECOVER REGISTER A
        CALL    LPTOUT          ;CALL LINEPRINTER OUTPUT ROUTINE
        PUSH    A               ;SAVE REGISTER A
        LDA     MODE            ;RESTORE BIT 2
        XRI     04H             ; OF MODE BYTE TO
        STA     MODE            ; ITS ORIGINAL VALUE
        MVI     A,0E0H          ;RESTORE PORT TO
        STA     H84PT3          ; E0
        POP     A               ;RECOVER REGISTER A
        RET                     ;RETURN
```

Listing 2

finish saying the phrase before returning to line 60 to continue executing the program. Without this delay, the computer would get to the next step before the Speech System finished talking.

A delay is also thrown in at line 220 to give the user a moment to admire the answer before asking if he would like to do another problem. Note that in this case L% is given a value and only the time-delay loop part of the routine is called.

The Buffer Problem

If your needs are simple, you can now write programs using the printer and the Personal Speech System. CONFIGUR may be used to change from 300 BAUD to any other baud rate obtainable with the Speech System's configuration switches and it may be used to move the Speech System to a different port. To change one of these, answer the question about the Standard System 'no' and go to A or to C on the main menu. By choosing D on the menu and then setting item A on the next menu to false, you can eliminate running CONFIGUR automatically on power-up.

If you want to send a file out to the Speech System, just use PIP. For example, if you have the Gettysburg Address in a disk file named GETTYSBG.ADD, type

```
PIP TTY:=GETTYSBG.ADD
```

and the Speech System will recite it.

A problem can arise here. The VOTRAX Personal Speech System has 3K of RAM buffer. If a long enough file is sent to it, this buffer will fill up and part of the file will be lost. This problem can be overcome by choosing main menu item A of CONFIGUR, then setting the BAUD rate and port assignment of LST to match the Speech System and setting the Printer Ready Signal Polarity to HIGH. As long as its buffer is not full, the Speech System will hold the RTS line (pin 4 of the serial connector at the back of the computer) high, but will change it to low as the buffer approaches full. CP/M will then hold further transmission of data until this line is again high. In this way, the buffer will not be overrun and data lost.

The problem with this approach is that one cannot use the printer again without rerunning CONFIGUR, so that one cannot switch in the middle of a program. To fix this problem, we must turn to the CP/M BIOS, which contains the code for sending data to the output devices.

Changing the Bios

We need to change the BIOS to support a second "printer", the Speech System, at port 320Q (hex D0) while not disturbing the one at port 340Q (hex E0). I have an H-14 printer which poses an additional problem. Its ready signal is RTS low as opposed to the Personal Speech System's RTS high. Because CONFIGUR sets all ports for a high signal or all ports for a low signal, one cannot set the Speech System's port for an RTS high and the H-14's port for an RTS low. The changes that we will make to the BIOS solves both problems.

Listing 2 shows the pertinent sections of the new BIOS. One routine, LPT2OUT, has been added and two lines changed in LIST and PUNCH. In LIST the call to TTYOUT has been replaced by a call to the new routine LPT2OUT, and in PUNCH a call to DMYOUT (an empty routine containing just a return) is replaced by a call to LPT2OUT. Thus, if LST is CONFIGURed for TTY (bits 7,6 of IOBYTE set to 00, or binary 0), a call to LST will call the new routine. Furthermore, if PUN is CONFIGURed for PTP (bits 5,4 of IOBYTE set to 01, or binary 1), a call to PUN will also call the new routine.

What does the new routine do? LPT2OUT reverses bit 2 of the MODE byte, changes the printer port from E0 (340Q) to D0 (320Q) and then calls LPTOUT, the usual printer routine. Upon return from LPTOUT these are changed back to their original values. The effect of changing the printer port is to allow LPTOUT to send its output to the Speech System rather than the printer. Thus, a call to LPT2OUT is the same as a printer call except that the output goes to the Speech System.

Bit 2 of MODE indicates the polarity of the LPT ready signal. If bit 2 is 0, the LPTOUT routine interprets an RTS low signal as indicating the printer is ready to receive more data. But, if bit 2 is 1, the LPTOUT routine interprets an RTS high signal as indicating the printer is ready to receive more data. Thus, LPT2OUT has the effect of reversing the RTS signal. The combination of switching to the Speech System port and reversing the RTS signal allows the H-14 and the Speech System to be run using the same CONFIGURation.

If your printer uses an RTS high signal to indicate that the printer is ready, then you need to leave the three lines following each of the PUSH A statements out.

Because the print routine LPTOUT checks the RTS line to be sure the Speech System is ready each time it outputs a character, the Speech System's buffer cannot be overfilled as it could when using the TTY output routine to send to it. With this change we now have full use of the VOTRAX Personal Speech System and the H- 14 printer.

About the Author:

Bill Boyd is Operations Manager of COM PRO, Inc., a manufacturer of SxS telephone equipment, and a Visiting Associate Professor of Operations Management at the University of Arkansas. His present interests include speech synthesis and recognition, the H-89 and HERO I.

The "Getting Started with Assembly Language" Contest Winners

Pat Swayne
HUG Software Engineer

I have selected the winners of the contest I announced in my last "Getting Started with Assembly Language" article (in the February 1984 issue). The HDOS winner is Stephen Liddle of Pleasant Grove, Utah, and the CP/M winner is John F. Smith of Horsham, Pennsylvania. Congratulations to both of them for producing programs that passed difficult operational tests. In fact, the two winning programs were the only two of all submitted that passed all tests! In next month's issue, I will start up the 'Getting Started' series again, and explain the tests I applied to the contest programs. I will also include the winning source codes, and explain how they work. Don't miss it!

Microsoft COBOL-86 (Z-DOS)

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

Introduction

Heath and Zenith Data Systems are providing the H/Z-100 owners with a "great" selection of software. The "NEW" Microsoft COBOL-86 Compiler is an excellent example that makes full use of the H/Z-100. It is priced at $395.00 in the 1983 Christmas Catalog No. 863R, Catalog No. MS-463-3. It requires two (2) high-capacity disk drives and the Z-DOS Operating System. We should be thankful for this attractive price! This is about one-half the IB and "something" company price of $700.00.

COBOL offers a portable, standardized business-oriented language that is ideal for processing business data. COBOL provides the powerful use of disk drives; CRT screen-handling; English self-documenting, long variable names; and readable-structured programs. It is used almost completely by government agencies and many large companies. COBOL-86 offers the H/Z-100 owner the capability to work on main-frame computer problems at home or at his small business. It also offers the owner the opportunity to learn this language with his own computer and possibly improve his employability.

The timing could not have been better for any present or just coming aboard H/Z-100 owners. HUG has just started the publication of the instructional COBOL Corner Series in REMark. The first article brought inquiries of how can COBOL be run on the H/Z-100! COBOL-86 solves this problem.

COBOL-86

COBOL-86 is nearly some of the same as COBOL-80. I will explain the main differences. Prior reading of the first COBOL Corner-I article might be helpful in understanding this article.

1. One main difference is that COBOL-86 uses the Z-DOS Operating System. COBOL-80 uses CP/M-80 (also available for HDOS). This means that disk handling commands and Error Messages will be different.

2. COBOL-86 Hardware and Software requirements are:

A) Two (2) soft-sectored 5.25 inch drives or one (1) 5.25 inch and a Winchester disk drive. You could also use one or two 5.25 inch and one or two 8 inch disk drives.
B) Printer capable of 132 print positions per line.
C) Z-DOS Operating System Software.
D) Editor or Z-DOS line text editor, EDLIN. I use PeachText (also known as MagicWand) in the program mode. We will further discuss this later in this article. I am still looking for a better Editor that is more to my liking!
E) COBOL-86 can make use of colored video if you have it with your H/Z-100.

3. COBOL-86 is already configured for the H/Z-100. You do not need to configure the CRT display as required for COBOL-80.

4. COBOL-86 Screen Section provides for color controls if your H/Z-100 is equipped for color video display. There are two (2) controls. Foreground-color controls the color of the characters. The color is chosen by a value of 0-7 with white as the default. The color definitions are as follows:

0	BLACK	4	RED
1	BLUE	5	MAGENTA
2	GREEN	6	YELLOW
3	CYAN	7	WHITE

The other, background-color, controls the color of the screen field and this color is also chosen by the range of 0-7 with black as default. These colors would provide an excellent way to show negative values (RED) and positive values (GREEN) for example. I do not have color available, so I was not able to experiment with the color facilities.

5. COBOL-86 has an Interactive Debug Facility that allows the programmer to control the execution of a program and to examine or change data items in the program! When the program is compiled, a "debug information file" is created along with the object file. This information file contains the line numbers and data-names from the program. The extensive list of debug commands available can use the numbers and names to affect data items and program execution in a number of ways to enable the programmer to find the program problems. This is a COBOL subject that must be covered in detail with a sample program. If there is sufficient interest, I will prepare an article on this topic for COBOL Corner when we get to advanced programs that are hard to debug with the other usual ways. Please let me have your input! COBOL-80 has dynamic debugging statements - READY TRACE/RESET TRACE and EXHIBIT. These are just a part of COBOL-86 debugging commands. The COBOL-80 and COBOL-86 debugging commands are extensions to the ANSI-74 COBOL standards.

6. If the COBOL-86 programmer uses TAB stops in his source program, he must use the stops defined below:

8, 12, 20, 28, 36, 44, 52, 60, 68, 73

The COBOL-80 programmer must use the following TAB stops:

7, 17, 25, 33, 41, 49, 57, 65, 73

These stops can be changed by patching the internal TAB table in the COBOL Compiler. Both manuals explain how to do this if you are handy with "patches". The COBOL-86 selection of TAB stops is the closest to any kind of standard. Some Editors provide for adjustable TABS but you must determine if the COBOL compiler will read the Editor's symbols by trial and error. I do not use TABS because I want my CODE to be portable!

7. If you are now using COBOL-80 and you wish to change to COBOL-86, you must be aware of the TAB stops we have just discussed. However, you MUST be aware of a "DISTURBING" change (This really upsets me!). COBOL-86 has changed COMPUTATIONAL USAGE (COMP) to COMPUTATIONAL-0 (COMP-0)! Now, this is a step backwards! COMP-0 is not in the ANSI-74 Standard! This means that all references to COMPUTATIONAL (COMP) must be changed to COMPUTATIONAL-0 (COMP-0) to work with COBOL-86. Even worse, if you program with COBOL-86 and wish to have your program portable with 99% of the COBOL Compilers I know about, all the COMP-0's in the program must be changed to COMP! As you can surely tell, I am not happy with this. I hope that the ones who made this decision have a GOOD REASON!!!

8. While I am dealing with negatives about COBOL-86, I must object to the sample programs included with the package! Don't get me wrong, I believe in example programs to help the user to understand the software, but they should be prepared in up-to-date structured COBOL. The main objection is the use of 77-levels. The 77-levels are going to be eliminated in the next ANSI COBOL standard. I am also put out that the file format was changed between versions 4.01 and 4.06 of COBOL-80 without notice and without giving the users a chance to update! This is really serious to programmers that have thousands of lines of COBOL code that they now want to convert to COBOL-86.

9. Another change between COBOL-80 and COBOL-86 is the subject of Error Messages. Here, I am not negative. I think that the COBOL-86, Appendix E does an excellent job of listing the error messages that you can encounter while compiling and executing a COBOL program. The explanations are brief but to the point. This is a big improvement over COBOL-80.

10. The COBOL-86 Manual is the really big improvement!!! The index could be improved and additional examples of the use of COBOL verbs would make the manual better. However, I would accept it as a big step in the right direction as compared with a lot of software manuals.

11. I have kept my major complaint about COBOL-86 for the last. COBOL-80 did not include an internal SORT verb, but you could purchase the M/SORT software from Microsoft separately. It did a good job. COBOL-86 does not say one word about the COBOL SORT verb in the manual. This is a serious omission! I have no objection to it being optional, but it should be available. There are many, many COBOL programs requiring the SORT verb. I discussed this with the manager of my local Heathkit store. He checked into this problem and advised me that my objection was not the first. Microsoft is working on the problem and hopefully will have a SORT utility software package on the market in the next few months. I hope that this is true!

COBOL-86 Software

COBOL-86 comes with two (2) 40 track soft-sector distribution disks. They are organized as follows:

I - Disk I has the compiler and run-time systems:

A) COBOL.COM - Main compiler program.
B) COBOL1.OVR - Overlay 1.
C) COBOL2.OVR - Overlay 2.
D) COBOL3.OVR - Overlay 3.
E) COBOL4.OVR - Overlay 4.
F) COBOL1.LIB - Run-time library of optional routines.
G) COBOL2.LIB - Run-time library of routines required to load COBRUN.EXE.
H) COBRUN.EXE - Common run-time executor.
I) COBDBG.OBJ - Interactive Debug Facility.
J) REBUILD.EXE - Utility for recovering damaged indexed files.

II - Disk II has the test and demonstration programs:

A) CTEST.COB -- Test program for color display systems.
B) CTEST.EVE - Executable version of CTEST.COB.
C) CRTEST.COB - Test program for terminal interface module.
D) CRTEST.EXE - Executable version of CRTEST.COB.
E) CENTER.COB - Test program for compiler and runtime system.
F) CENTER.EXE - Executable version of CENTER.COB.
G) DEMO.COB - Program to demonstrate the screen section, to CALL subprogram BUILD, and CHAIN to program UPDATE.
H) DEMO_01.OVL - Overlay file generated by linking DEMO.
I) DEMO.EXE - Executable version of DEMO already linked to BUILD.
J) BUILD.COB - Subprogram to create am indexed (ISAM) file of names, addresses, and telephone numbers.
K) UPDATE.COB - Program to list or update the ISAM file created by BUILD.
L) UPDATE.EXE - Executable version of UPDATE.
M) README.DOC - Contains current release information.

Note: If you will either Print or Type the README.DOC file on your COBOL-86 Distribution Disk you will find that Heath/Zenith has provided five (5) additional example programs. These additional programs demonstrate, by example, the different type of file organizations that are supported by COBOL-86 System. They are described below:

EX-I1.COB	Index file demo program.
EX-R1.COB	Relative file demo program.
EX-S1.COB	Sequential file demo program.
EX-L1.COB	Line sequential demo program.
EX-P1.COB	Demo program that generates printer output.

You will not understand these programs at this time unless you are experienced with COBOL. Do not worry, we will explore all of these in future COBOL Corner articles (In my mind, they are not good examples!).

Even though I have expressed some negative thoughts about COBOL-86, I would still highly recommend COBOL-86 for use with the H/Z- 100. It is an ideal software program for the H/Z-100 because COBOL programs use a lot of RAM and a lot of disk storage. The 16-bit 8088 side of the H/Z-100 provides this memory and disk storage.

Major COBOL-86 Process Steps

There are three (3) major steps required to compile and execute a COBOL-86 program:

1 -- Compiling. The COBOL-86 compiler consists of the main module (COBOL.COM) and four overlays (COBOL1.OVR thru COBOL4.OVR). The routines contained in the compiler analyze the COBOL program and produce the object code file with the filename extension (.OBJ). The compilation requires two passes. The first pass creates the Intermediate Version of the program that is stored in a binary work file called COBIBF.TMP. The second pass then creates the object code and then erases the work file.

2 -- Linking. The Object Linker produces the machine executable code from the object code which is placed in a file with the extension (.EXE). The Object Linker (LINKER) performs the following tasks:

a) Combines separately-produced object code files, if any.

b) Searches the library files for definitions of unresolved external references.
c) Resolves external cross-references.
d) Produces a printable listing of symbols.

3 -- Loading and Executing. The run-time system (COBRUN.EXE) is loaded and executes the executable program.

Disk Backup

Before you start using COBOL-86, you MUST make working copies of your two original MASTER DISKS that came with the system! To perform this backup you must know how to use Format, Configur, and Copy utilities that are supplied with the Z-DOS Operating System. (If you have not made backup copies of the Z-DOS disks, this should be done first! Refer to your Z-DOS Manual for instructions.) I am going to assume that you know the Z-DOS system. If not, this should be done first!

The backup copies of COBOL-86 should include the "operating system files" and the necessary COBOL files. To do this, follow these steps:

1. Place Z-DOS Disk I in drive A and a blank DSDD (double-sided, double-density) soft-sector disk in drive B.
2. Enter by keyboard, FORMAT B:/S/V
3. When (2) is complete, enter COPY LINK.EXE B:
4. When (3) is complete, replace the Z-DOS disk in drive A with COBOL-86 Distribution Disk I.
5. Enter by keyboard, COPY *.* B:
6. When (5) is complete, the disk in drive B will contain a bootable COBOL-86 system.
7. Remove your working COBOL-86 Disk from drive B and label it "COBOL-86 SYSTEM DISK-A".
8. Now mount this disk in drive A and use the CONFIGUR utility and prepare the system to match your printer.

Now, you can create a bootable disk using another blank disk, following the above procedures. Skip step 3, and substitute the COBOL-86 Distribution Disk II in step 4. Also, using a similar procedure copy your Editor that you will use to code your source program to this disk. When complete, label this disk "COBOL-86 Program Disk-B".

Verify COBOL-86 Working Disks

We now want to verify our working disks that we made above. We will do that by compiling, linking, and executing the test program "CENTER.COB".

1. Put your COBOL-86 System Disk in drive A and the COBOL-86 Program Disk in drive B.
2. Use "DIR" to check the directory for the following files:

Program Disk	**System Disk**
"EDITOR".COM (YOUR CHOICE)	COBOL.COM
CTEST.COB	COBOL1.OVR
CRTEST.COB	COBOL2.OVR
CENTER.COB	COBOL3.OVR
DEMO.COB	COBOL4.OVR
BUILD.COB	COBDBG.OBJ
UPDATE.COB	LINK.EXE
Plus Z-DOS utility & boot	COBOL1.LIB
files you will need. Erase	COBOL2.LIB
all other Z-DOS & COBOL-86	COBRUN.EXE
files.	REBUILD.EXE
	Plus Z-DOS utility & boot
	files you will need. Erase
	all others.

You can see that COBOL-86 will require two working disks. The COBOL-86 System Disk will always be in drive A and the COBOL-86 Program Disk will be in drive B. This arrangement will simplify access to the program files and will always place them on the same disk. (For you readers using the 8 inch disks or the Winchester drive, I am going to assume that you have the experience to know how to set up your system.)

Test Program "CENTER.COB"

Let's try a test run! Please follow this procedure:

1. Boot the system and select B as the default drive by typing B: and return.
2. Compile CENTER.COB so that an object file named CENTER.OBJ is produced by typing one of the following commands after the drive prompt B:.

```
A:COBOL CENTER; <CR>            PRODUCE ONLY OBJECT FILE
```

or

```
A:COBOL CENTER,,PRN <CR>        PRODUCE OBJECT FILE & PRINT
                                PROGRAM LISTING
```

or

```
A:COBOL CENTER,,CENTER <CR>     PRODUCE OBJECT FILE & LIST
                                FILE (CENTER.LST)
```

or

```
A:COBOL CENTER,,: <CR>          SAME AS PREVIOUS EXAMPLE
```

or

```
A:COBOL CENTER,,CON <CR>        PRODUCE OBJECT FILE &
                                DISPLAY LISTING ON TERMINAL
```

When compilation is successfully completed, the message -- NO ERRORS OR WARNINGS -- is displayed, and the compiler exits to the operating system. This confirms the compiler. The files CENTER.OBJ and CENTER.LST (if you specified one) will be saved on the Program Disk in drive B. Practice the various commands.

3. Now, we will Link the program, CENTER, with the following command typed after the drive prompt B:

```
A:LINK,,,A:; <CR>
```

This command links the object file with the run-time system, producing the executable file. The A: at the end of the command tells the linker to look on drive A for the COBOL Libraries (.LIB). The executable file (CENTER.EXE) will be saved on the Program Disk in drive B. If you type DIR, you should now find the following files:

CENTER.COB
CENTER.OBJ
CENTER.EXE
CENTER.LST (if you specified it in step 2)

4. Next, we will load and execute the CENTER program by typing the following command after the drive prompt B:

```
CENTER <CR>
```

This command will cause the system to search the B drive and then the A drive for COBRUN.EXE. COBRUN.EXE and CENTER.EXE will be loaded and CENTER will be executed. If no ERRORS are displayed, you have confirmed the Linker and the Run-time Executor.

5. CENTER is a simple COBOL program that does not use the sophisticated screen handling features of COBOL-86. It will prompt you for an input line of text. CENTER takes that text and centers it or aligns it with the left or right margin.

6. You should now know how to compile, link, and load/execute a COBOL-86 program. Before we try another sample program, we should review the three (3) steps in more detail.

COBOL-86 Compiler Operation Methods

Again, we will put the COBOL-86 System Disk in drive A and the COBOL-86 Program Disk in drive B. We will also select drive B as the default drive. The compiler can be operated in more than one way:

1. Response Method
2. Command String Method
3. Partial Command String Method
4. Command String With Switches Method

One Response Method:

```
Type--------A:COBOL                        (type) <CR>
Response---source filename[.COB]:          (type) CENTER
Response---object filename[CENTER.OBJ]:    (type) <CR>
Response---source listing[NUL.LST]:        (type) CENTER
```

This will produce the object file, CENTER.OBJ, and a listing file, CENTER.LST, on the disk in drive B.

One Command Method:

```
Type--------A:COBOL command string
```

where the command string contains:

```
source filename, object filename, source listing
```

The separator character is the comma and no spaces are allowed before comma. Filename is defined as follows:

```
device filename extension
```

where "device" is the name of a system device; such as, disk drive, terminal, or printer. If device is a disk drive, the filename must be given, unless a default filename is available (more on this under Partial Command Strings). If device is not a disk drive, only a device name is required. The device may be followed by a colon for readability (the colon is required for disk drives). COBOL-86 will recognize the following devices:

NUL	Do not create a file.
CON	Display on terminal.
A: or B:	Disk drive (colon required).
PRN	Printer.
AUX	RS-232 item.

Where "filename" is the name of the file on disk. If filename is specified without a device, the default disk drive is assumed as the device. Maximum length of the filename is 8 characters, where the "extension" is a period followed by a three-character suffix to the filename. If an extension is not specified, the following defaults are assumed:

.COB for the source program file
.OBJ for the object file
.LST for the list file

I believe an example will best make all this a lot clearer! For example, type the following command string after the B drive prompt:

```
A:COBOL CENTER,CENTOBJ,CON  <CR>
```

The source program, CENTER.COB, on drive B will be compiled, the object file, CENTOBJ.OBJ, will be saved on the disk in drive B, and the list file will be displayed on the terminal. No list file, CENTER.LST will be saved in this example.

Partial Command String Method:

Now, note that the default object filename can be specified by entering only the comma that normally follows the filename. Also note that if a comma is entered following the object filename, the source listing filename defaults to the source filename. If you fail to specify one of the filenames, you will be prompted for it. Also, if you enter an incomplete command string followed by a semicolon, default entries will be assumed for any unspecified files. An example will best explain the Partial Command String. Type the following after the B drive prompt:

```
A:COBOL CENTER,,;  <CR>
```

The source program, CENTER.COB, on disk B will be compiled and the object file, CENTER.OBJ, and the list file, CENTER.LST, will both be saved on disk B. (The second comma tells the compiler to use the source filename as the default list filename.)

Command String With Switches:

One or more switches can be added to the compiler command string. A slash (/) indicates a switch to the compiler. The syntax for a command string with switch(es) is:

```
drive:COBOL command string/switch(es)
```

The possible switches are:

1) /C , the compiler will look for the (4) overlay files (COBOL1. OVR-COBOL4.OVR) on the default drive and then on drive A. This switch overrides the default drive. (No colon should be used.)

Example: `A:COBOL CENTER,,/CB`

Now, the compiler will look for the overlay files on drive B.

2) /T, the compiler puts its intermediate file, COBIBF.TMP, on the default drive. If /T is used followed by the desired drive, the intermediate file will be put on the specified drive. This switch is helpful for compiling large programs on systems with both 5 and 8 inch drives (two each).

Example: `A:COBOL CENTER,,/TC`

Now, the intermediate file will be put on drive C.

3) /P, each /P allocates an extra 100 bytes of stack space for the compilers use. If a stack overflow error occurs during compilation, use /P.

Example: `A:COBOL CENTER/P/P/P;`

This will allocate 300 extra bytes. Note: No commas, but the semicolon, to show the other default method.

4) /D, this switch suppresses both the generation of the Debug Intermediate File (.DBG) and the source line numbers. Thus, the Procedure Division code is about 16% smaller. This can be important when working with a large program. However, the system will not be able to note the line number at which an error occurs!

Example: `A:COBOL CENTER,,/D`

5) /Fn, this switch is too complicated to discuss at this time.

Note: There is a "short-cut" way to check the program with a "quick" compilation.

Example: `A:COBOL CENTER,NUL;`

This command will compile CENTER.COB and display a list of errors on the terminal. After the errors are corrected, use one of the other methods of compilation. This is for experienced COBOL programmers that do not require any other error finding procedures.

Also: The way to learn the above methods and to decide the way you want to compile your programs is to practice all of the methods over and over so that you can make a wise decision.

COBOL-86 Linking Methods

Keep the COBOL-86 System Disk in drive A and the COBOL-86 Program Disk in drive B, which is selected as the default drive. The Linker converts the compiled object file into an executable file. It does this by searching the COBOL-86 run-time libraries, COBOL1.LIB and COBOL2.LIB, which are part of the run-time system. COBOL1.LIB has the optional routines that may be required for running your program. COBOL2.LIB contains the routines that are ALWAYS necessary for running your program. The necessary routines are then linked to the object file. The Linker also links separately compiled program modules you may have into the object file. To operate the Linker, use one of the following procedures:

1. Specify files interactively. Enter at keyboard after the B prompt:

```
A:LINK <CR>
```

Remember the A: is necessary because the LINK file is not on the Program Disk in drive B. Now, reply to the following prompts:

A) Object Modules [.OBJ]:
Enter the name of the object file (CENTER). You do not need an extension (".OBJ" will be supplied). If you had multiple object files to be linked, they must be added here separated by a plus (+). Remember! The files to be linked MUST be OBJECT FILES!

B) Run File [object filename.EXE]:
Name the file to contain executable code (CENTER).

C) List File [NUL.MAP]:
Name of list file. Defaults work about the same as with the compiler. The default is no list file, unless the Run File is followed by a comma, then the default list file is the object filename with the extension (.MAP).

D) Libraries [.LIB]:
"Libraries" refers to the run-time routines that COBOL-86 may need to run the program. Normally press RETURN following this prompt. A special case would be where the programmer wishes to specify another library.

2. Use a Command String. Enter at the keyboard after the B prompt:

```
A:LINK command string
```

where the command string contains

```
objectfile,runfile,listfile,libfile
```

as defined above.

The object filename MUST be specified! For the other files, a default filename can be named by entering the comma that would normally follow the filename.

Examples:

A) A:LINK CENTER;
This command string links CENTER.OBJ and puts the run file into CENTER.EXE. The Linker will prompt for the drive on which COBOL1.LIB and COBOL2.LIB are to be found. Type A: in response.

B) A:LINK CENTER,,,A:;
This command operates the same as (A), except that a listing, CENTER.MAP, is produced. The second comma indicates that the object filename will be used as the default list filename. The A: at the end of the command line tells the Linker that the COBOL-86 Libraries are on drive A.

C) We will not review some of the more complicated methods at this time.

Executing COBOL-86 Programs

After the COBOL program has been compiled and linked successfully, the final step is Loading and Execution! These functions are performed by specifying the name of the executable file to the system. The run-time executor (COBRUN.EXE) is loaded automatically at the start of execution. To run the program, enter the name of the run-file, without the extension (.EXE).

Example: `CENTER <CR>`

Execution of CENTER.EXE will start immediately.

CRTEST.COB Test Program

Now for practice you compile, link, and execute CRTEST.COB with the method that you have decided you are going to use. You can use the method that we used for CENTER.COB above. I will not review this. It is done exactly the same way! CRTEST is a test program to check the terminal interface. When it is executed, CRTEST will prompt for input. Just follow instructions.

COBOL-86 Demonstration System

The COBOL-86 Demonstration System consists of three (3) COBOL programs:

DEMO.COB
BUILD.COB
UPDATE.COB

We will use these programs for additional practice. You do not need to understand these programs at this time but COBOL Corner will cover the subjects down the road. DEMO is the executive program of the system. It asks if you would like a demonstration of the COBOL-86 Screen Section, or whether you would like to create or update an indexed (ISAM) file of name, addresses, and phone numbers. This system will provide you with a chance to compile, link, and execute a fairly complicated program. We will still keep our COBOL-86 System Disk in drive A and the COBOL-86 Program Disk in drive B. (This might be a good time to back-up these two disks! Do you know how? If not, look it up in your Z- DOS Manual.) Make drive B the default drive by typing B: and RETURN.

1. Type from the keyboard:

```
A:COBOL DEMO,,CON;  <CR>
```

This compiles DEMO.COB and produces DEMO.OBJ. The use of CON directs the compiler listing to be displayed on the terminal screen allowing you to watch the compilation. The message -- NO ERRORS OR WARNINGS -- should be displayed on the screen at the end of the compilation.

2. Next type:

```
A:COBOL BUILD,,CON;  <CR>
```

This will compile BUILD.COB and produce BUILD.OBJ.

3. When the above compilation process is finished, type:

```
A:COBOL UPDATE,,CON;  <CR>
```

This time UPDATE.COB will be compiled and UPDATE.OBJ will be produced.

4. When you have finished the above compilations, type:

```
DIR *.OBJ  <CR>
```

The directory listing should display the following files:

DEMO.OBJ
BUILD.OBJ
UPDATE.OBJ

5. Link DEMO.OBJ and BUILD.OBJ together by typing:

```
A:LINK DEMO+BUILD,,,A:;  <CR>
```

This will produce DEMO01.OVL in addition to DEMO.EXE.

6. Link UPDATE.OBJ by typing:

```
A:LINK UPDATE,,,A:;  <CR>
```

This will produce UPDATE.EXE.

7. Now we will type:

```
DEMO  <CR>
```

When DEMO has been loaded, it will prompt for input by providing Menus and Information Screens to guide you through the demonstration. Do not worry that you do not understand the program at this time. We will take care of that in due time in our COBOL Corner articles.

Program Development

To write a COBOL-86 program, you will again have the two disks in the same drives that we have been using. Also, select drive B as the default drive by typing B: and return. Now, type the the following command after the B prompt:

```
EDLIN PRGM01.COB
```

if you are using EDLIN as your Editor, or else your Editor's name in place of EDLIN. You are now ready to enter the first COBOL Corner program. When you have finished writing the program, use EDLIN E command, or your Editor's command, to save the file, PRGM01.COB, on the COBOL-86 Program Disk and then exit to the Z- DOS operating system. Now, as we have described above, compile, link, and execute PRGM01!

Closing

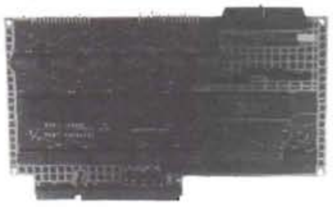
As you can see, COBOL-86 is very similar to COBOL-80, but you must learn the few different commands. A little practice will make it all seem easy. So, we will look for you to join COBOL Corner! Please order the HUG COBOL Corner Disk-I(Z) in place of the HUG COBOL Disk-I!

If you have any questions, suggestions or arguments, send them to me in complete written form along with a stamped, self-addressed business size envelope. We will get a discussion going!

H/Z89 PERIPHERALS from **SECURED COMPUTER SYSTEMS**

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

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



Features:

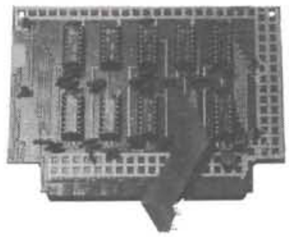
Chip independent design • Reduces computer data buss loading from 3 to 1 • Choice of Centronics or Epson parallel drivers for HDOS or CP/M • Complete documentation and installation instruction.

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

PRICE $199.00
Shipping & Handling $10.00

16K RAM EXPANSION CARD

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



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

NOW INCLUDING SUPPORT MOUNTING BRACKET

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

Now Only $65.00

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

Shipping & Handling $5.00

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

ZENITH data systems SERVICE CENTER

Keyboard Dialing

John H. King
710 Meadowridge Dr.
Warner Robins, GA

Introduction

After reading Mr. Walt Jung's modem tutorials in REMark Issue #40, 42, and 45, most of the questions I had about modems and modem software were answered. After Issue #42 I purchased a small inexpensive modem. This article is about that modem and others like it. I benefited greatly from Mr. Jung's articles, and I would like to share some discoveries I found and expanded upon concerning the modem I purchased.

The appropriate title for this article should be "Keyboard Dialing A Non-Smart Modem". I purchased the modem to take advantage of the after hours use of a Vax 11/780, RCPM and BBS systems. I really wanted a smart modem at 1200 baud, but my wallet said no long before my wife did. So I settled for the cheapest modem that had the most features. I reviewed several modem ads and then decided on the Novation J-CAT, who's features included (1) auto-search, (2) auto-answer, (3) self test diagnostics, (4) audio beeper, (5) LED indicators, (6) keyboard connect mode, (7) break key and EIA-RS232C/TTL Compatible serial interface. The unit sold for $124.00 in July of '83, and now selling for $90.00 at some discount stores.

While reading the owners manual I found an unadvertised feature, the J-CAT had the hardware capability to pulse dial a phone line. Requirements to take advantage of this feature are hardware interfacing with your Heath computer, and writing a program to execute the dialing. I viewed this as a challenge, to have an inexpensive modem with Smart modem-like capabilities. This article concerns the results of my efforts to interface the H-89 to the J-CAT for pulse dialing from the keyboard. For the sake of space, I will skip the problems I encountered for whatever reason, and just explain exactly what is needed to accomplish the keyboard dialing with the J-CAT.

Hardware Requirements

The J-CAT modem comes with a male RS-232 cable which must be changed to a female RS-232 to interface with port 330 on the H-89. Just remove the wires from the male connector and solder them to the same pin numbers on the female connector, they are correct for modem communication. The next two lines are used for pulse dialing and they are not connected from the factory. They are PLS and OHK, the OHK line will connect to pin 4 of the female RS-232, and the PLS line will connect to pin 5 of the RS- 232. See Figure 1 for the signal names and numbers on the J- CAT, H-89 and the RS-232 connector. Also make sure that pin 6 and 8 are jumpered together at the RS-232 connector.

Signal Requirements

The keyboard dialing interface requires a single output bit to execute the dialing, and if you wish to detect a dial tone before dialing, then you will need an input bit. I used pin 4 (RTS) of port 330 for the output bit, and pin 5 (CTS) of port 330 for the input bit. The J-CAT owners Manual has an excellent explanation on what's needed for the interfacing, but for the H- 89 owners I have just outlined all that is needed for the hardware interfacing to the J-CAT. This outline can also be applied to the SIGNALMAN MARK VII modem with some minor differences.

Software

I am submitting the MBASIC program I wrote to implement the keyboard dialing (KEYDIAL.BAS). The J-CAT owners manual explains some basic guidelines needed for writing your program. I will use the steps from those guidelines, and the blocks from the simple block method I used in my MBASIC auto-dial program to show how I accomplished the software requirements. Throughout this software explanation I will cross reference between guideline steps and program blocks. The blocks are shown in Listing 1.

Step 1. [reset modem before using] Procedure: set OHK low (0) for 25 msec, high (1) for 20 msec and then back to (0). Step 1 is implemented in block 16 of the MBASIC program.

Step 2. [set up the off hook condition] Procedure: set OHK high (1). See block 18.

Step 3. [wait for dial tone] Procedure: check that PLS pin 5 stays high (1) for one second. See block 19.

Step 4. [pulse OHK to dial number, according to the specification below]

Pulse duration: 60 msec on-hook (OHK=0)
Pulse interval: 40 msec off-hook (OHK=1)
Digit interval: 700 msec off-hook (OHK=1)
Digit 1 to 9 : 1 to 9 pulses
Digit 0 : 10 pulses

See block 21.

Step 5. [leave line in off-hook condition after dialing] Procedure: set OHK high (1). See block 21.

Step 6. [wait for J-CAT to lock on to remote modem then go to data mode] See block 7.

Step 7. [terminate call] Procedure: repeat step 1. See block 16.

Step 8. [re-dial a number] See block 9.

The remaining blocks are for screen delays, menu, error traps, and error messages. The program is easy to follow and I will explain how it works from top to bottom.

The program starts by displaying a menu, from the menu you have four choices. Capitals or lower case letters will work as valid inputs to the menu prompt. Select D, R, T, and G, selection D will except digits 1 thru 9, zero and spaces as valid numbers. Anything else will cause an error. R will re-dial last number dialed. T will reset modem or terminate call. And G will exit the auto-dial program and return to the CP/M A> prompt. The J- CAT modem requires a reset on start up, so T should be the first selection each time you run the program.

After entering the phone number, the program will check for dial tone. If tone is present, the number will be dialed. The program will check for carrier tone next. If no carrier detection in 10 sec., the program will return to menu. If a carrier tone is detected, then program will exit to CP/M.

Special Notes

Once you return to CP/M through carrier tone detection or through selection G, type in the name of your communication program. I am presently using MODEM7, which should be on the current disk with KEYDIAL.BAS. It will be loaded and executed. At that point you can begin communication with the remote computer. This procedure works OK with some remote systems but proved to be too slow for the VAX 11/780, the Telenet link, and the Tymnet link. The long delay after making contact with the remote system is just long enough to be rejected. The message you get reads "Timed out in log in procedure".

To solve this problem I decided to compile the MBASIC program, then use it in a Submit file under CP/M. This would eliminate the long delay of typing in the program name. The Submit file would respond like a batch process, and execute like one big self contained program, eliminating the need to use MBASIC altogether. I also wrote and compiled a two line routine that resets or disconnects the phone line when communication is terminated. See Listing 2 for the Submit file, and Listing 4 for the two line disconnect program.

Compiling Notes

When compiling the MBASIC program, all of the for/next loop terminal values must be increased because the compiler runs faster than MBASIC. If not changed, the program will pulse dial too fast, and the screen delays are not long enough to read the error messages displayed on the screen. The program lines that should be changed are 800, 890, 900, 1080, 1120, 1200, and 1220. You can use an oscilloscope or the trial and error method to find the correct value. I will not try and list what the values should be because there are too many Heath/Zenith micro computers running at different clock rates, and this program should work with all of them.

I do not own a good oscilloscope, so my for/next loops are ball park figures. Those who do can be more exact according to the J-CAT modem specification. I experimented with slow and fast dialing that worked. I then chose a value at the center of those two limits.

As mentioned earlier, this program will also work with the SIGNALMAN MARK VII with minor changes, which centers around what signal level and duration is needed for resetting and pulse dialing the SIGNALMAN MARK VII. I will list the lines of the MBASIC program that should be changed for keyboard dialing the MARK VII. See Listing 3.

Note: In the MBASIC program, port 330 address is decimal 216.

I hope that others can make use of this program for pulse dialing. And I would like to thank Pete George of MGHUG for his input and for modifying and testing this program on the SIGNALMAN MARK VII.

Figure 1

```
 H-89              FEMALE                 J-CAT
PORT 330           RS-232               CONNECTOR

  ___               ___                 _________
| 2 |----------->| 2 |--------------->| 2 TXD |
| 3 |<-----------| 3 |<---------------| 3 RXD |
| 4 |----------->| 4 |--------------->| 7 OHK |
| 5 |<-----------| 5 |<---------------| 5 PLS |
| 7 |------------| 7 |----------------| 1 GND |
| 6 |<-----------| 6 |---|            |       |
| 8 |<-----------| 8 |<--|------------| 4 CAR |
  ___               ___                 _________
```

Listing 1 (BLOCKS)

Block 1. This block formats the screen with program name.

Block 2. The menu is displayed in this block.

Block 3. This block will display the input prompt.

Block 4. Block 4 validates entries. If entry is not D, R, T, or G, the program will exit to block 5. If true it will jump to block 6.

Block 5. This block will sound console bell and display an error message, then return to beginning of program.

Block 6. If input is equal to G then go to block 7.

Block 7. This block will exit from keyboard dial program back to CP/M.

Block 8. If input is equal to T then jump to block 16.

Block 9. If input is equal to R then go to block 10, if not R then go to block 12.

Block 10. This block checks for previous number dialed. If no number was previously dialed, then exit to block 11. If there is a previous number, then jump to block 15.

Block 11. This block will display an error message, then return to the beginning of prógram.

Block 12a. Your selection has been logically deduced to D, the program will output a message concerning phone number format.

Block 12b. Will give prompt message for inputting number to be dialed.

Block 13, 14, & 15. Will check number to see if it is a valid number, if not then error message. If valid, then go to block 16.

Block 16. Reset modem for selections D, R, and T.

Block 17. If selection was T then return to beginning. If not then 18.

Block 18, 19, & 20. Read input port, check for dial tone. If no dial tone then give error message. If tone is present then go to 21a.

Block 21a. This block will display each number before dialing.

Block 21b. This block will dial phone number.

Block 21c. This block will check for carrier tone. If carrier not present in (10 sec.) then return to menu. Tone detection will exit to CP/M.

Listing 2

```
File name      =  DIAL.SUB

File contents  =  $1 KEYDIAL
                  $2 MODEM7  T TEMP.DAT
                  $3 DISCON
                  $4 DIR
```

Listing 3

Change these lines for the MARK VII:

800, 810, 890, 900, 920, 1080, 1100, and 1120

Listing 4

This program will disconnect phone line by resetting modem.

```
DISCON.BAS

10 FOR I=1 TO XX:OUT 216+4,0:NEXT I:FOR I=1 TO XX:
   OUT 216+4,2:NEXT I:OUT 216+4,0

20 END:SYSTEM
```

Note: XX must be replaced by actual terminal values before compiling.

KEYDIAL.BAS

```
20 '  \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
30 '
40 '                     KEYBOARD AUTO DIALER FOR NON-SMART MODEMS
50 '
60 '    WRITTEN BY- J. KING FOR THE HEATH/ZENITH H/Z-89 USING CP/M & MBASIC
70 '    PROGRAM CAN BE MODIFIED TO RUN ON ANY HEATH MICRO AT ANY CLOCK RATE
80 '  \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
90 '
100 ' PORT 330 PIN 4 (RTS) = OUTPUT AND PIN 5 (CTS) = INPUT FOR THIS PROGRAM
110 '
120 '                           216 IS PORT ADDRESS #
130 '
140 '--------------------------------------------------------------------------
150 ' | BLOCK  (1) OF FLOW CHART |  FORMAT SCREEN
160 '
170 PRINT CHR$(27)+"E"
180 PRINT CHR$(27)+"q":'    TURN OFF REVERSE VIDEO
190 PRINT TAB(20) "  ********************************   "
200 PRINT TAB(22);CHR$(27)+"p";"*      KEYBOARD AUTO DIALER      *";CHR$(27)+"q"
210 PRINT TAB(20) "  ********************************  "
220 PRINT:PRINT
230 '--------------------------------------------------------------------------
240 ' | BLOCK (2) |  PRINT MENU
250 '
260 PRINT TAB(26) "D = DIAL A NUMBER":PRINT
270 PRINT TAB(26) "G = GO TO MODEM PROGRAM "
280 PRINT
290 PRINT TAB(26) "R = RE-DIAL LAST NUMBER"
300 PRINT
310 PRINT TAB(26) "T = TERMINATE CALL ":PRINT:PRINT
320 PRINT TAB(21);"LAST NUMBER DIALED -->   ";PNUM$
330 PRINT
340 '-------------------------------------------------------------------------
350 ' | BLOCK (3) |  GET CHOICE
360 '
370 INPUT "                                CHOICE:";CHOICE$
380 '-------------------------------------------------------------------------
390 ' | BLOCK (4) (5) |  VALIDATE CHOICE/IF NOT GIVE ERROR MESSAGE
400 '
410 C$=CHOICE$
420 IF C$<>"D"AND C$<>"G"AND C$<>"R"AND C$<>"T"THEN 430 ELSE 490
430 IF C$<>"d"AND C$<>"g"AND C$<>"r"AND C$<>"t"THEN 440 ELSE 490
440 PRINT CHR$(7)
450 PRINT TAB(20)"WRONG SELECTION HIT <cr> TO CONTINUE"
460 INPUT "                                          ";Z$:GOTO 170
470 '
480 ' | BLOCK (6) |
490 IF CHOICE$= "G" OR CHOICE$="g"  THEN 1260
500 ' | BLOCK (8) |
510 IF CHOICE$= "T" OR CHOICE$="t"  THEN 800
520 ' | BLOCK (9) |
530 IF CHOICE$= "R" OR CHOICE$="r"  THEN 660
540 '--------------------------------------------------------------------------
550 ' | BLOCK (12a) |  PRINT DIALING INFORMATION
560 '
570 PRINT "               DIAL A NUMBER FROM (0 TO 9) AND <SPACE BAR>"
580 PRINT
590 '
600 ' | BLOCK (12b) |  INPUT PHONE # TO BE DIALED
610 '
620 INPUT "                         ENTER NUMBER:";PNUM$
630 '--------------------------------------------------------------------------
640 ' | BLOCK (10) (11) |  VALIDATE NUMBER/IF ZERO PRINT ERROR MESSAGE
650 '
660 IF PNUM$= "" THEN 670 ELSE 710
670 PRINT CHR$(7);TAB(17);"NO NUMBER OR PREVIOUS NUMBER TO BE DIALED":GOTO 1200
680 '
690 ' | BLOCK (13) (14) (15) |  VALIDATE NUMBER/IF WRONG GIVE ERROR MESSAGE
700 '
710 FOR I = 1 TO LEN (PNUM$)
720 DIGIT$ = MID$ (PNUM$,I,1)
730 IF DIGIT$<>" "AND (DIGIT$<"0" OR DIGIT$>"9")THEN 740 ELSE 760
740 PRINT CHR$(7)
750 PRINT TAB(22)"NUMBER IS INCORRECTLY FORMATED":GOTO 1200
760 NEXT
770 '--------------------------------------------------------------------------
780 ' | BLOCK (16) (17) |  RESET MODEM FOR SELECTION T,D AND R
```

```
790 '
800 FOR I=1 TO 4:OUT 216+4,0:NEXT I:FOR I=1 TO 3:OUT 216+4,2:NEXT I
810 OUT 216+4,0:IF CHOICE$="T" OR CHOICE$="t"  THEN 170
820 '-----------------------------------------------------------------
830 PRINT "               *** DIALING NUMBER - PLEASE STAND BY ***"
840 '
850 ' | BLOCK (18) (19) (20) |  CHECK FOR DIAL TONE BEFORE DIALING
860 '                        |  GIVE ERROR MESSAGE IF NO DIAL TONE
870 '
880 Y = INP(216+6) :'           CLEAR DIAL TONE REGISTER
890 FOR I=1 TO 250:NEXT I:FOR Z=1 TO 130:OUT 216+4,2:NEXT Z
900 FOR J=1 TO 143:IF Y=0 THEN 910 ELSE 920
910 PRINT CHR$(7);TAB(26);"@@@@  NO DIAL TONE  @@@@":GOTO 1200
920 OUT 216+4 ,2
930 NEXT J
940 '-----------------------------------------------------------------
950 ' | BLOCK (21a) |   DISPLAY EACH NUMBER BEFORE DIALING IT
960 '
970 NUM$=PNUM$
980 S = 0
990 PRINT CHR$(27)+"p" :'  TURN ON REVERSE VIDEO
1000 FOR DIGIT = 1 TO LEN (NUM$)
1010 DIGIT$ = MID$(NUM$,DIGIT,1)
1020 S = S+1
1030 PRINT CHR$(27);CHR$(89);CHR$(54);CHR$(62+S);CHR$(27)+"C";DIGIT$;
1040 IF DIGIT$ = "0" THEN DIGIT$ = "10"
1050 '
1060 ' | BLOCK (21b) |  DIAL NUMBER-- THEN WAIT FOR CARRIER TONE
1070 '
1080 FOR PULSE=1 TO VAL(DIGIT$):FOR Y=1 TO 7:OUT 216+4,0:NEXT Y:FOR X= 1 TO 5
1090 '           CREATES THE 60MS LO & 40 MS HI PULES FOR EACH VAL
1100 OUT 216+4,2:NEXT X
1110 NEXT PULSE
1120 FOR U= 1 TO 85:NEXT U:NEXT DIGIT :' CREATE 700MS HI BETWEEN DIGITS
1130 '-----------------------------------------------------------------
1140 ' | BLOCK (21c) |  CHECK FOR CARRIER TONE
1150 I=0:PRINT CHR$(27)+"q"
1160 I=I+1:IF I=500 THEN 1190 ' IF NO CARRIER IN 10 SEC THEN RESET
1170 F=INP(216+6):Y=F AND 128    ' MASK OUT REC.LINE SIGNAL DETECT
1180 IF Y=128 THEN 1210 ELSE 1160
1190 PRINT CHR$(7);TAB(18);"     @@@@   NO CARRIER TONE   @@@@     "
1200 FOR I=1 TO 2000:NEXT I:GOTO 170
1210 PRINT CHR$(7);TAB(20);"  CONNECTION MADE WITH REMOTE MODEM  "
1220 FOR I=1 TO 2000:NEXT I
1230 '-----------------------------------------------------------------
1240 ' | BLOCK (7) |  EXIT TO MODEM COMMUNICATION PROGRAM
1250 '
1260 PRINT CHR$(27)+"E":SYSTEM:'END OF KEYBOARD DIAL/GO TO MODEM7
```

About the Author:

John King *has 15 years of experience as an electronic technician, 10 years of kit building, all Heathkit, and one year with the H-89A. He started in the computer hobby about five years ago with the Radio Shack Model 1 computer. John enjoys learning as much as he can about hardware and software on any computer system, especially the H-89A. He is presently employed by the U. S. Air Force Civil Service as an Electronic Equipment Specialist.*

CORRECTION

There is a minor bug in the HUG TYPING program, P/N 885-3011-37. The counter of correct words and attached words does add up to more than total word counter. The bug would be fixed after adding the following line:

```
2005 IF K=18 THEN RETURN
```

Changing your address?

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

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

Current Local HUG Clubs

*(**NOTE:** This listing is current as of May 1, 1984. If your club is not listed or you are forming a new club and would like to have it included in our list, please send the proper information to: Heath/Zenith Users' Group, Attn: Nancy Strunk, Hilltop Road, St. Joseph, MI 49085.)*

AK, Eagle River
Alaska HUG
206 E Firweed Ln., Suite 208
Anchorage, AK 99503
907 276-5917 Group Size: 15
Contact Person: Roger Pickels or Ben Sevier
RBBS 9pm-9am Pacific time 907 276-5917

AK, Ft. Greely
COLD HUG
C/O Stan Lockhart P. O. Box 229
APO Seattle, WA 98733
907 895-3284 Group Size: 4
Contact Person: Stan Lockhart
Meet Bldg. 856 Apt. B Ft. Greeley
Newsletter, willing to exchange

AL, Birmingham
BEARHUG (Birmingham HUG)
C/O Jack Goertz Rte. 19, Box 248
Birmingham, AL 35244
205 991-5519 Group Size: 20
Contact Person: Jack Goertz
Meeting time and place varies

AL, Huntsville
Huntsville, AL HUG
Rt. 1, Box 427
Lacey's Spring, AL 35754
205 498-2199
Contact Person: Jeff Hamilton
Meet 2nd Thurs. at Intercon Research Corp
Jeff's work # 205 453-2576

AL, Mobile
MOBHUG Mobile HUG
3636 Linden Lane
Mobile, AL 36608
205 344-5065 Group Size: 21
Contact Person: Bud Hobdy
Meet 3rd Sun. at 2 p.m.
Just starting, new members welcome

AL, Montgomery
HUG/M (HUG of Montgomery, AL)
2948 Willow Lane Drive
Montgomery, AL 36109
205 272-6964 Group Size: 30+
Contact Person: Ronald E. Travis
Meet 1st Tue. at 7 p.m.
Meet Air Force Logistics Management Ctr.
ZBBS 208-284-0938 all day Sat.

AR, Little Rock
Little Rock HUG
C/O David Schade P. O. Box 15478
North Little Rock , AR 72231
501 758-4500 Group Size: 32
Contact Person: David Schade
Meet 2nd Sat. 12:00 noon at AR College of Tech.

AZ, Phoenix
PHUG (Phoenix Heath Users' Group)
C/O Will Summers P.O. Box 37783
Phoenix, AZ 85069
Group Size: 75
Contact Person: Will Summers, President
2nd Tues. at 7:00 p.m. at Phoenix HEC
Membership $5 initiation $12/year

AZ, Sierra Vista
HuacHUGa
1964 Viola Dr C/O Gerald King
Sierra Vista, AZ 85635
602 459-2119 Group Size: 25
Contact Person: Gerald King
Meet monthly at homes of members

AZ, Tucson
SUNHUG (Tucson HUG)
7109 E. Broadway
Tucson, AZ 85710
602 325-0096 Group Size: 40
Contact Person: Allan Anderson
Meet even months first Sunday 2:00 p.m. Tucson HEC
Meet odd months first Tues. 7:00

CA, Anaheim
ANAHUG (Anaheim HUG)
330 E. Ball Road
Anaheim, CA 92805
213 330-8118 Group Size: 103
Contact Person: Bob Chamberlain, Sec.
3rd Thursday 7:30 p.m. at HEC
BB 714 774-7860

CA, Campbell
San Jose HUG
2350 S. Bascom Avenue
Campbell, CA 95008
408 377-8472 Group Size: 70
Contact Person: Gerlene York, Sec.
Meet lst and 3rd Wed. 7:00 p.m. HEC Campbell

CA, El Cerrito
ECHUG (El Cerrito HUG)
6000 Potrero Avenue
El Cerrito, CA 94530
415-236-8870
Contact Person: Alan Biocca
4th Wednesday at HEC

CA, El Monte
ETUG (ET/ETA 3400 Users Group)
11231 Oak Street
El Monte, CA 91731
Group Size: 100
Contact Person: Charles Van Dyke
Newsletter 4 times year

CA, Fresno
FresHUG (Fresno HUG)
4833 East Santa Ana
Fresno, CA 93726
209-291-6258 Group Size: 4
Contact Person: Harlen Collins

CA, Glendora
Southern CA H11 Users Group
430 W. Highland Avenue
Redlands, CA 92373
714-886-4766 Group Size: 40
Contact Person: Dr. M. J. DiGirolamo
Meets at 625 E. Palm, Glendora, CA

CA, Los Angeles
Los Angeles HUG
4469 E. Olympic Blvd
Los Angeles, CA 90023
213 248-1580 Group Size: 20
Contact Person: Doug Holser
1st Thursday 7:00 p.m. at HEC
BB 213 749-8442

CA, Los Angeles
LAETUG (Los Angeles ET3400 GP)
2309 S. Flower
Los Angeles, CA 90007
213 749-0261
Contact Person: Gilbert Murillo
Other contact Charlie at 213 443-2237
Contact for meeting time and place

CA, Monterey
SIG/Heath-Zenith
Naval Postgrad Sch. Hobby Computer Club
C/O Dave Smith
376 Bergin Dr. Apt. F
Monterey, CA 93940
408 373-4202 Group Size: 60
Contact Person: Dave Smith, Pres.

CA, Pomona
Pomona HUG
1555 N. Orange Grove Avenue
Pomona, CA 91767
714 985-5303 Group Size: 110
Contact Person: Herb Friedman, President
Meet 4th Thursday each month at 7:30 p.m. at HEC
BB 714 629-1943

CA, Redding
Redding Heath Users' Group
22526 Bridlewood Lane
Palo Cedro, CA 96073
916 547-3461 Group Size: 6
Contact Person: Dave Ballard
Meet monthly various locations

CA, Redwood City
BAHUG Bay Area HUG
2001 Middlefield Road
Redwood City, CA 94063
415-365-4915 Group Size: 219
Contact Person: Bob Bance, Sec.
2nd Tuesday 7:00 p.m. at HEC

CA, Riverside
Tri-HUG
5705 Via Sotelo
Riverside, CA 92506
714-683-2929 Group Size: 20
Contact Person: Kenny Adcock

CA. Sacramento
Sacramento Heathkit Users' Group
1334 Silica Ave. Sp25
Sacramento, CA 95815
916 662-7220 Group Size: 35
Contact Person: Wally Ranch (Librarian)
Meet 2nd Wed. 7:30 p.m. at Sacramento HEC

CA, San Diego
San Diego HUG
12202 Kingford Court
El Cajon, CA 92021
714-561-2540 Group Size: 170
Contact Person: Richard Cobb
1st Wed. 7:00 p.m. at Parkway Jr HS La Mesa

CA, Santa Maria
4168 Glenview Drive
Santa Maria, CA 93455
805 937-6938 Group Size: 18
Contact Person: Raymond S. Isenson
Meet 1st Mon. 7:00 p.m. at Vandenburg Air Force Base

CA, Visalia
Visalia HUG
29924 Road 168
Visalia, CA 93291
209 747-3235 Group Size: 3
Contact Person: Peter Shkabara
Meeting time and place not established yet

CA, Woodland Hills
LUVAHUG
22504 Ventura Blvd.
Woodland Hills, CA 91364
213 883-0531 Group Size: 40
Contact Person: Paul S. Townsend
2nd Thursday 7:00 p.m. at HEC

CO, Colorado Springs
CSHUG (Colorado Springs HUG)
Colorado Springs, CO 80906
303 632-3019 Group Size: 25
Contact Person: Richard Evers
Meet last Thurs. each month 7:00 p.m.
Have 24 hr. BB (303) 634-1158

CO, Denver
DENHUG (Denver HUG)
PO Box 449 Contract Station 22
Denver, CO 80221
303 426-7404 Group Size: 160
Contact Person: Bob Eson Sec/Treas
BB 303 423-3224 24hrs Support newsltr exchange
Meet 2nd Monday 7:00 p.m. at HEC

CO, Ft. Collins
FT.HUG (Fort HUG)
822 E. County Road 30
Ft. Collins, CO 80525
303 669-4116
Contact Person: Ted Benglen, II
Meet once a month at present

CT, Avon
CONNHUG (Connecticut HUG)
395 W. Main Street
Avon, CT 06001
203 589-3824 Group Size: 50+
Contact Person: Bob Conlon, President
1st Wednesday at 7:00 p.m. at HEC
BB 203 674-8915

CT, Mystic
Mystic ZDS/HUG
11 Allen Street
Mystic, CT 06355
203 536-6953
Contact Person: Larry Moxon
Meet last Thurs 7:00 p.m. at 11 Allen, Mystic, CT

FL, Cocoa Beach
Brevard Heath Users' Group
680 Java Road
Cocoa Beach, FL 32931
305 783-6352 Group Size: 25
Contact Person: Gene E. Stillman
Meet last Sun. of each mo. at 7:00 p.m.
Meet at Patrick AFB, Comet Rec. Ctr.

FL, Fort Myers
SWFHUG (Southwest Florida HUG)
P.O. Box 05-37
Tice, FL 33905
813 334-6190 Group Size: 20
Contact Person: Robert Sloat
Meet 2nd & 4th Thurs. 7:30 p.m.
Meet at J. Hamilton Welch Academy

FL, Fort Walton
NWFHUG (NorthWest Florida HUG)
812 Cherokee Road
Eglin AFB, FL 32542
904 651-2108 Group Size: 30
Contact Person: George A Repasy, Pres.
Meetings 2nd Wed. at DATATEC Inc. 7 p.m.

FL, Jacksonville
JUG (Jacksonville Users Group)
8262 Arlington Expressway
Jacksonville, FL 32211
Group Size: 150
Contact Person: Harry Walker
Meet 1st Wed. at 7:00 p.m. at HEC Jacksonville
BB 904 725-4995 24 hrs.

FL, Miami
Miami Amateur Computer Club
4705 W. 16th Avenue
Hialeah, FL 33012
305 823-2280 Group Size: 35
Contact Person: Emileo Crespo
Meet 2nd Thurs. each month 7:00 p.m. at HEC
BB (305) 823-2281

FL, Orlando
HUG of Central FL Computer Sc.
121 Talmeda Trail
Maitland, FL 32751
305 644-6848 Group Size: 11
Contact Person: Joseph Walker, Pres.
4th Wednesday at various locations

FL, Pensacola
221 E. Government
Pensacola, FL 32501
Contact Person: John Causey
Meet 2nd Tue. each month 7:00 p.m. at above address
Meet at Professional Business Sys.

FL, Plantation
PHUG Plantation HUG
7173 W. Broward Blvd.
Plantation, FL 33317
305 791-7300 Group Size: 20
Contact Person: Paul Price
Meet 2nd Tues. 7-9 p.m. at HEC
BB 305 791-7302 24 hrs. on H/Z100

FL, Tallahassee
Tally HUG
C/O TACS P.O. Box 6716
Tallahassee, FL 32314
904 562-1412 Group Size: 14
Contact Person: Bill Hill
Meet 1st Tues. each month 7:30
Meet at Alternative Microcomputing

FL, Tampa
Al Lynch HUG
415 Shore Crest Drive
Tampa, FL 33609
813 253-0093 Group Size: 65
Contact Person: H. Glenn Tanner, Sec.
Meet 1st and 3rd Wed. 7:30 p.m. at Tampa HEC
Dues: $10.00 year

GA, Atlanta
HUG-GA (Georgia HUG)
2775 NE Expressway Apt. 0-3
Atlanta, GA 30345
404 325-8781 Group Size: 30
Contact Person: James Affuso
Meet 2nd Thurs. at 7:00 p.m. at HEC
BB 404 252-4345 24 hrs.

GA, Warner Robins
MGH/ZUG Middle Georgia H/ZUG
P.O. Box 92
Warner Robins, GA 31099
912 922-6470 Group Size: 30
Contact Person: Garth K. Haygood
Meet 3rd Tues. at Nola Brantley Public Library
Forming local CP/M Users' Group

Ga, Augusta
CSRA Computer Club
P. O. Box 284
Augusta, GA 30903
803 648-3603 Group Size: 10
Contact Person: Dave Howard
Meet 4th Wed., location rotates
BB 803 279-5392

HI, Hilo
BIHUG (Big Island HUG)
P. O. Box 4271
Hilo, HI 96720
808 959-8985 Group Size: 10
Contact Person: R.A. Curtis
Meet 1st Thurs. 7:00 p.m. at HELCO Conference Room
BB 808 961-4818

HI, Honolulu
HUGH (HUG Hawaii)
1255 Nuuanu Avenue #1405
Honolulu, HI 96817
808 531-8843 Group Size: 87
Contact Person: Jim Branchaud, Pres.
Meet 3rd Wed. 7:00 p.m. at Pearl City HEC
BB 808 487-8755

IA, Des Moines
DMA HUG (Des Moines Area HUG)
10275 NE 23rd Avenue
Mitchellville, IA 50169
515 266-2382 Group Size: 21
Contact Person: Harold Dykens
Meet third Mon. each month 7:00 p.m.

IL, Champaign
CCCC (Champaign Cty Comp Club)
C/O James Bartlett, J.r 2109 Branch Rd
Champaign, IL 61821
217 398-5956 Group Size: 110
Contact Person: James Bartlett Jr., Pres.
Meet 1st Wed. 7:30 p.m. at Urbana Civic Ctr.
24 hr. BB 217 359-9090

IL, Davis
NI-HUG Northern IL HUG
427 Lockwood Rt. 1
Davis, IL 61019
815 248-2241
Contact Person: Jim Isenhart
Just starting

IL, Downers Grove
I-HUG (Illinois HUG)
6116 Lane
Downers Grove, IL 60516
312 971-1660 Group Size: 15
Contact Person: Len Bateman
3rd Wednesday at various locations

IL, Downers Grove
HUG Metro (Local Chicago)
15 W. 780 Fillmore
Elmhurst, IL 60126
312 985-2381 Group Size: 30
Contact Person: Larry Shipinski, President
Meet 2nd Mon. of each mo. 7:30 p.m. at HEC

IL, Peoria
CIHUG (Central Illinois HUG)
2422 Willow
Pekin, IL 61554
309 347-3366 Group Size: 17
Contact Person: John Cole, Jr.
3rd Sun. at 3 p.m. (Jan, Mar, May, Jul, etc.)

IL, Springfield
217 753-5795
Contact Person: Bobby Wright
Club just forming

IN, Indianapolis
Indiana HUG (IHUG)
11425 Lakeshore Dr. West
Carmel, IN 46032
317 852-3530 Group Size: 75+
Contact Person: Charles C. Hillman, Jr.
Meet 2nd Wed. 7:30 p.m. at HEC

IN, South Bend
MIHUG (Michiana Hug)
52578 US 31-33 North
South Bend, IN 46637
219 255-3923 Group Size: 10
Contact Person: Mark L. Meidel
Meet 3rd Mon. 7:30 p.m.

KS, Mission
MUG (Mission Users' Group)
5960 Lamar Avenue
Mission, KS 66202
913 649-0879 Group Size: 80+
Contact Person: Charles L. Bennett
Meet last Sun. of the mo. 2:00 p.m. at Mission HEC
BB 913 362-9583 and Newsletter

KS, Wichita
Wichita HUG
1909 Siefkin
Wichita, KS 67208
316 681-3456 Group Size: 18
Contact Person: David Horwitz
2nd Sun. of odd months 2:00 p.m. at E. Pike Bldg.
Corner of Webb and Kellog in Wichita

KY, Louisville
LHUG (Louisville HUG)
6802 Crossmoor Lane
Louisville, KY 40222
502 426-9433
Contact Person: Ray Donner
Meet last Sun. at 8:00 p.m. at Louisville HEC

LA, Lafayette
ZUG (Zenith Users' Group)
318 W. St. Mary Blvd.
Lafayette, LA 70506
318 948-7804 Group Size: 40
Contact Person: Tommy Billiodeaux
Meet every other Tues. 6:00 p.m.
Meet at Zenith Computer Depot

LA, New Orleans
NOHUG
1900 Veterans Blvd.
Kenner, LA 70062
504 467-6321 Group Size: 60
Contact Person: Nathan Gifford
1st Wednesday at 7:30 p.m. at HEC

MA, Northampton
Hampshire Computer Club
37 Drewson Drive
Northampton, MA 01060
617 584-6227 Group Size: 80
Contact Person: George Scheurer
2nd Tuesday 7 p.m. at McConnel Hall Smith College
Beginners Group 1st Tuesday

MA, Peabody
HUG North Shore
6 Susan Drive
Saugus, MA 01906
617 233-2941 Group Size: 60
Contact Person: Hal Messinger, Pres.
BB 617-531-9332 24 hours
2nd Wednesday Hilltech Bldg Danvers

MA, Pittsfield
BerCHUG (Berkshire County HUG)
73 Waverly Street
Pittsfield, MA 01201
413 443-1862 Group Size: 12
Contact Person: Paul E. Ouellette, Pres.
Meeting place and time vary

MA, Wellesley
HUG'EM
165 Worcester Ave
Wellesley, MA 02181
617 237-1510 Group Size: 200
Contact Person: Malcolm Partridge, Director
3rd Wed. 7:00 p.m. at HEC
BB 617 237-1511 24 hrs.

MD, Baltimore
Baltimore HUG
6106 Marlora Road
Baltimore, MD 21239
301 323-6093 Group Size: 50
Contact Person: William Frey
2nd Mon. 7:00 p.m. at Park School - Old Court Road

MI, Ann Arbor
A-SQR-HUG
895 Starwick Drive
Ann Arbor, MI 48105
313 769-6052 Group Size: 15+
Contact Person: Leonard E Geisler
Meet last Thurs. 7-9:30 p.m. Jun-Aug Huron High Sch.
Meet Sep.-May at Northside School

MI, Detroit
Metro Detroit Area HUG
7716 Winona
Allen Park, MI 48101
313 928-7423 Group Size: 50
Contact Person: Chuck Dattolo

MI, Kalamazoo
SMHUG (Southwest Michigan HUG)
623 Wildwood Place
Kalamazoo, MI 49008
616 349-3535 Group Size: 50
Contact Person: Al Jacobs, Sec./Treas.
4th Saturday 1 p.m. at Western Michigan Univ.
Moore Hall, Rm. 1034, Newsletter

MI, Saint Joseph
BLHUG (Blossomland HUG)
P.O. Box 414
Saint Joseph, MI 49085
616 983-0161 Group Size: 50
Contact Person: Vance Fisher, Chair Person
1st Tues. 7:00 p.m. at St Joe High Sch. Cmptr. Classrm
$15.00 dues/yr., Monthly Newsletter

MN, St. Paul-Minneapolis
SMUGH
5085 Fern Drive
Loretto, MN 55357
612 479-2127 Group Size: 150
Contact Person: Mary or Gene Hess
Meet last Sun. 2:00 p.m. at Falcon Hgts. Comm. Ctr.
BB 612 778-1213 7 p.m.-8 a.m.

MO, St. Louis
SLHUG (St. Louis HUG)
3794 McKelvey Road
Bridgeton, MO 63044
618 259-8113 Group Size: 120
Contact Person: Brad Pulaski, Treasurer
Meet 2nd Wed. 7:30 p.m. at HEC

NC, Charlotte
HUG Charlotte
2152 Malvern Road
Charlotte, NC 28207
704 375-1581 Group Size: 100
Contact Person: Jim Simpson
All types of computers, H/Z owners comprize 25%
Meet 1st Tues. 7:30 p.m.

NC, Fayetteville
Cape Fear Computer & HUG
2454 Vandemere Avenue
Fayetteville, NC 28304
919 485-4586 Group Size: 25
Contact Person: Jerry Mills, Pres.
Varies, monthly

NC, Glen Alpine
Western Piedmont HUG
Rt. 2, Box 371
Morganton, NC 28655
704 584-3684 Group Size: 10
Contact Person: Bill Poteat
Meeting time and place varies
Just getting started. Will have BB

NC, Hillsborough
HUG-RTP
Rt. 3, Box 39A
Hillsborough, NC 27278
919 73-6678
Contact Person: Joe Williams
Meeting place and time unknown

NE, Omaha
OMAHUG (Omaha HUG)
P. O. Box 777
Bellevue, NE 68005
Group Size: 85
Contact Person: Phil Evans, Pres.
3rd Sun. odd mos. 6:30 Bellevue W HS or Offutt AFB
Meet even mos. Amer. Red Cross 6:30 p.m.

NJ, Fairlawn
HUGNJ (HUG of New Jersey)
124 Mohawk Drive
Cranford, NJ 07016
201 791-6935 Group Size: 155
Contact Person: Mel Beiman
BB 201 791-6936 evenings
3rd Monday 8:00 p.m. at HEC

NJ, Ocean
SHUG (Shore HUG)
1013 State Hwy. 35
Ocean, NJ 07712
201 775-1231 Group Size: 71
Contact Person: James J. Jones, Jr., Sec.
Meet 1st Wed. 7:30 p.m. at Ocean HEC
BB 201 775-8705 24 hrs.

NM, Albuquerque
Albuquerque HUG
7909 Hendrix NE
Albuquerque, NM 87110
505 294-1658 Group Size: 25+
Contact Person: Ken Benson
Meet 3rd Sun. at members homes

NY, APO New York
BAHUG (Bad Aibling HUG)
Unit AA Box 561
APO New York, NY 09098
Group Size: 10
Phone: 08061-4519/6340 West Germany
Contact Person: Louis J. DeMichele

NY, APO New York
BWHUG (Bentwaters HUG)
PSC Box 3703 RAF Bentwaters
APO New York, NY 09755
Contact Person: Sgt. Rodney Jones

NY, Buffalo
BUG (Buffalo Users Group)
223 Clark Road
Kenmore, NY 14223
Group Size: 75
Contact Person: Bob Allen
Meet 3rd Sun. 1:30 p.m. at Amherst HEC

NY, Long Island
Jeri-HUG (Jericho HUG)
5 Helen Place
Glen Cove, NY 11542
516 676-5616 Group Size: 75
Contact Person: Alan Scott Dodge, Sec./Treas.
Meet 2nd Thurs. 8:00 p.m. Jericho Pub. Library
Monthly newsletter, software library

NY, North White Plains
North White Plains HUG
Elliott Ser. Co. 720 White Plns. Rd.
Scarsdale, NY 10583
Group Size: 50
Contact Person: Peter Abramson
Meet 2nd Tues. ea. mo. 7:30 p.m. at HEC

NY, Potsdam
CCT HUG (Clarkston College)
Woodstock Vlg Apt 3B24
Potsdam, NY 13676
315 268-6455 Group Size: 60
Contact Person: Marc A. Rubin
Meet monthly-call for date, time and place
Club just getting started

NY, Rochester
RHUG (Rochester HUG)
937 Jefferson Road
Rochester, NY 14623
716 424-2560 Group Size: 50+
Contact Person: RHUG Editor, Blanche Nail
Meet last Tues. each mo. 7:30 p.m. at HEC
BB 716 424-2576

NY, Schenectady
Schenectady HUG
C/O T. Budge 715 Sanders St.
Scotia, NY 12302
518 377-4273 Group Size: 20
Contact Person: Walter Whipple
Meet 3rd Wed. 7:30 p.m. at above address
BB 518 457-3803

OH, Cincinnati
Cincinnati HUG
10133 Springfield Pike
Woodlawn, OH 45215
513 771-8850 Group Size: 90
Contact Person: President
2nd Tues. 7:00 p.m. at HEC, $10.00 dues/year
Newsletter, 24 hr. BB 513 772-6190

OH, Cleveland
NOHUG (Northeastern Ohio HUG)
4705 Tanglewood Place
Lorain, OH 44053
216 282-4790 Group Size: 70
Contact Person: Art Petkosek
Meet 2nd & 4th Thurs. 7 p.m. at St. Gregorys Church

OH, Cleveland
Cleveland HUG
28100 Chagrin Blvd.
Cleveland, OH 44122
216 291-1612 Group Size: 10
Contact Person: Gerry Ciganko
First Thurs. 7:00 p.m. at HEC
BB 216 292-7553 24 hours

OH, Columbus
Columbus HUG
2500 Morse Road
Columbus, OH 43229
614 475-7200 Group Size: 25
Contact Person: President
Meet 2nd Mon. at HEC
BB 614 475-7201 after store hours

OH, Dayton
Wright-Patterson HUG
4110 Spruce Pine Court
Dayton, OH 45424
513 236-4915 Group Size: 75
Contact Person: Jim Moore, President
Meet 1st Thurs. 4:00 p.m.
Meet Bldg. 640 Rm. 121 W-P AFB

OH, Toledo
THUG (Toledo HUG)
48 S. Byrne Road
Toledo, OH 43615
419 729-4621 Group Size: 300
Contact Person: Ryck Zarich
Meet last Sun. of the mo. at 7:00 p.m. at HEC
BB 419 537-1888 24 hrs. also 729-4221

OK, Oklahoma City
OKC TUGS
C/O Bill Cadwallader P. O. Box 1171
Lawton, OK 73502
405 848-7593 Group Size: 40
Contact Person: Bill Cadwallader
Meet 3rd Thurs. 7:30 p.m. at HEC
BBS 405-848-9329 24 hours

PA, Allentown
Lehigh Valley HUG
1425 N. Broad St
Allentown, PA 18104
215 770-5993
Contact Person: James Batug
2nd contact Carol Bloch 215 770-4640
Just getting started

PA, Frazer
FUG (Frazer Users Group)
1641 Princess Anne Drive
Lancaster, PA 17601
717 397-3146 Group Size: 80
Contact Person: Dave Hendrie, Pres.
1st Saturday 4:00 p.m. at Frazer HEC
BB 215 644-7661

PA, Harrisburg
CPaHUG (Cent Pennsylvania HUG)
7540 Mourningstar Dr. %E. Asper
Harrisburg, PA 17112
717 545-2764 Group Size: 7
Contact Person: Ernest E. Asper
Meeting time & place varies
Club just getting started

PA, Philadelphia
Philadelphia Heath Users' Group
6318 Roosevelt Blvd.
Philadelphia, PA 19149
215 288-0180 Group Size: 135
Contact Person: Henry F. Beechhold, Pres.
Meet 2nd Wed. each mo. 7:00 p.m. at HEC 8

PA, Pittsburgh
PittsburgHUG
3482 William Penn Highway
Pittsburgh, PA 15235
412 793-6781 Group Size: 35
Contact Person: John C. Schultz, Pres.
Meet 3rd Tues. at 7:00 p.m. at HEC
BB 412 824-3565 after store hours

RI, Warwick
HUG-'RI' (HUG of Rhode Island)
558 Greenwich Avenue
Warwick, RI 02886
401 738-5150 Group Size: 150
Contact Person: Leo Therrin/Dave Haskell
2nd Wednesday 8 p.m. at HEC

SD, Sioux Falls
Sioux Falls Area HUG
2001 S. Spring Avenue
Sioux Falls, SD 57105
605 336-8629 Group Size: 20
Contact Person: Lorin Dobson
Meet once a month on Sat. Time and place varies
BB 605 336-3935 M-F 3pm-12am

TN, Knoxville
ETCHUG East Tenn Central HUG
7608 Luscombe Dr.
Knoxville, TN 37919
615 690-3864 Group Size: 20
Contact Person: Walter M. Scott III
Meet 3rd Thurs. 7:30 p.m.
Meet at John XXIII Center

TN, Memphis
Memphis HUG
6874 Kirby Brooks Drive
Memphis, TN 38115
901 362-8860 Group Size: 16
Contact Person: Morris Proctor
Meet 2nd Tues. 7:00 p.m. at The Computer Center

TN, Nashville
Mi Te HUG (Middle Tenn HUG)
C/O Radio Ser. Ctr. 116 17th Ave. S
Nashville, TN 37203
615 242-0556
Contact Person: Charlie Wolf
Meet 2nd Mon. 6:30 p.m. at Radio Service Center

TX, Austin
AHUG Austin Heath Users Group
4206 Tamarack Trail
Austin, TX 78759
512 255-0376 Group Size: 40
Contact Person: George Koehler
Meet 1st Thurs. 8:00 p.m. Univ. of Texas
Meet at Robert Lee Moore Hall

TX, Dallas
DFW HUG (Dallas-Fort Worth)
2715 Ross Avenue
Dallas, TX 75201
214 826-4053 Group Size: 70
Contact Person: Henry Gardiner, Pres.
1st Thurs. and 15 days later (Wed.) at 7:30 p.m.
At HEC BB 214-742-1380

TX, El Paso
EPHUG (El Paso HUG)
4554 hercules #63
El Paso, TX 79904
915 755-1728 Group Size: 18
Contact Person: Rick Peterson
Meet 3rd Wed. at 7:00 p.m. at 444 Executive Ctr Blvd.
BB 915-592-1910 7:00 p.m.-7:00 a.m all day Sun.

TX, Ft. Worth
FWHUG
6825A Greenoakes Road
Ft. Worth, TX 76116
817 737-8822 Group Size: 26
Contact Person: John Ike Mitchell
Meet fourth Thurs. 7:30 each month

TX, Houston
HUG-H
7798 Braniff
Houston, TX 77061
713 644-5689 Group Size: 75
Contact Person: Tom McCormick, Pres.

TX, Houston
NHHUG (North Houston HUG)
8110 Tattershall Circle
Humble, TX 77338
713 446-1787 Group Size: 50+
Contact Person: Paul Eustace
Meet 3rd Tues. 7:30 at HEC
2nd contact Mark Shafer 713 583-1163

TX, San Antonio
San Antonio (SAHUG)
7111 Blanco Road
San Antonio, TX 78216
512 341-8876 Group Size: 65
Contact Person: Tom Schneider
First Tuesday at HEC, 7:30 p.m.

TX, Wichita Falls
NORTEX HUG (N. Texas S. Okla)
2413 Kemp Blvd. in Office World
Wichita Falls, TX 76309
817 322-1007 Group Size: 24
Contact Person: Alan D. Martin
Meet third Sat. 9 a.m. at above address

UT, Castle Dale
Castle Mesa Computer Group
670 N. 90 E. Box 123
Castle Dale, UT 84513
801 381-5173 Group Size: 10
Contact Person: Doug Sorensen
Meet 3rd Thurs. 5:30 p.m. above address

UT, Midvale
UHUG (Utah HUG)
58 E. 7200 South
Midvale, UT 84047
801 262-8810 Group Size: 130
Contact Person: Wayne Newland
2nd Wednesday 7:00 p.m. at HEC
BB 801 566-4551

VA, Christiansburg
New River Valley HUG
C/O CCS Data Sta. 8 Roanoke St.
Christiansburg, VA 24073
703 382-4234 Group Size: 35
Contact Person: Ted Fleshman
Meet 1st Thurs. 7:30 p.m. Christiansburg High School

VA, Fairfax
CHUG (Capital HUG)
P. O. Box 2653
Fairfax, VA 22031
703 759-6176 Group Size: 600+
Contact Person: Mike Supley, Pres.
3rd Monday 7:30 p.m. at Fairfax H.S.
Large Software Library (150+ disks)

VA, Richmond
RHUG (Richmond HUG)
4302 Smithdeal Avenue
Richmond, VA 23225
804 231-6759 Group Size: 20+
Contact Person: Carlos Chafin
Meet 3rd Mon. 7:30 p.m.
Meet at Alpha Audio 2049 W. Broad

VA, Virginia Beach
THUG (Tidewater HUG)
1055 Independence Blvd.
Virginia Beach, VA 23455
804 467-4232 Group Size: 115
Contact Person: Skip Kelly
1st & 3rd Tues. 7:30 p.m. at HEC

WA, Bellevue
Pacific Northwest HUG
C/O Barry Dupler P. O. Box 993
Bellevue, WA 98009
206 363-3927 Group Size: 150
Contact Person: Nathan Hall
Meet 2nd Thurs, odd months Tukwila HEC (both 7:00)
Meet 2nd Mon, even months Seattle HEC

WA, Spokane
SPOHUG (Spokane HUG)
S. 3810 Havana
Spokane, WA 99204
509 448-9727 Group Size: 25
Contact Person: Charles Ballinger
Meet last Thurs. 7-9 p.m. at Acme Business Computers
BB 509 927-0367 24 hrs.

WA, Vancouver
Portland-Vancouver HUG
516 SE Chkalov Drive
Vancouver, WA 98663
206 254-4441 Group Size: 30
Contact Person: Dan Heims
1st Thursday at 7:30 p.m. at HEC
Portland OR and Vancouver Area

WA, Walla Walla
HUG/ZUG of Walla Walla
112 N. Division
Walla Walla, WA 99362
509 525-8404 Group Size: 8+
Contact Person: Pat Hanna
Meet 2nd & 4th Tues. 8p.m. at 112 N Division
2nd contact Pete Parcells 509 527-5267

WI, Madison
Madison Area HUG
3519 Tally Ho Lane
Shorewood Hills, WI 53705
608 233-4588 Group Size: 9
Contact Person: Thomas Gans
Meet 1st Wed. 7:30 p.m. at Wisconsin Union South

WI, Madison
UWHUG (Univ. of Wisconsin HUG)
109 N Few
Madison, WI 53703
608 257-0373 Group Size: 30
Contact Person: Walter Burt
Meet 1st Wed. 7:30 p.m. at Univ. WI Union South
Club newly formed 12/83

WI, Milwaukee
MHUG Milwaukee Heath Users Gp.
9040 N. Lake Drive
Milwaukee, WI 53217
414 352-3346 Group Size: 65
Contact Person: Marvin Olson, Treas.
Meet 3rd Sat 2:00 p.m. at Milw. Sch. of Eng. Rm. L-100
BB 414 873-7564 6:00p.m.-6:00a.m.

WI, Mosinee
CWHUG-Central Wisconsin HUG
2294 CTH DB
Mosinee, WI 54455
715 693-3429 Group Size: 10
Contact Person: Edward Ignace Porwit
Meet last Sun. 3:00 p.m. in Ed's livingroom
BB coming soon

CANADA, Calgary, ALBERTA
HUG (Heath Users of Canada)
101 5809 Macleod Trail South
Calgary, Alberta T2H 0J9 CANADA
403-252-2688
Contact Person: Gary Selman

CANADA, Ottawa, ONTARIO
HUG 'O' (HUG Ottawa)
866 Merivale Road
Ottawa, ONTARIO K1Z 5Z6 CANADA
613-728-3731 Group Size: 30
Contact Person: Brian Fultz, Pres.
2nd Wednesday 8:00 p.m. at HEC

CANADA, Toronto, ONTARIO
THUG (Toronto HUG)
1480 Dundas Street E.
Mississauga, ONT. CANADA L4X 2R7
416 273-3797 Group Size: 25
Contact Person: Bill Smith

CANADA, Vancouver BC
VANHUG (Vancouver HUG)
3058 Kingsway Attn. Robert Hudak
Vancouver BC, CANADA V5R 5I7
604 437-7626 Group Size: 50+
Contact Person: Robert J. Hudak
Meet last Tues. 7:30 p.m. at HEC

CANADA, Vancouver, BC
Vancouver Island HUG
2022 Douglas St.
Victoria, BC CANADA V8T 4L1
604 384-4711
Contact Person: Greg Greene, Pres.
Meet each month at Excalibur Systems Ltd.
For further info call above number

FRANCE, Paris
GUFIN (French HUG)
34 Boulevard Saint-Jacques
75014 Paris, FRANCE
1-336-39-68 Group Size: 300
Contact Person: Dr. Bernard Pidoux
Meet weekly on Wed. eve. at club address
CBBS (1) 336-32-02

HOLLAND, Apeldorn
Dutch HUG
Hofstraat 30
7311 KW Apeldorn HOLLAND
Group Size: 70
Contact Person: Evert Jan Stokking

HONG KONG
Compudragon
273 Prince Edward Road
11/C Kowloon, HONG KONG
3-711-8904
Contact Person: K. T. Lee
Club just organizing

NETHERLANDS
Dutch Heath Users' Group
NIEUWE KERKHOF 16
9712 PV Groningen, NETHERLANDS
050-180203 Group Size: 107
Contact Person: Evert Jan Stokking
Meet quarterly at Amersfoort

New Zealand
HUG New Zealand
94 Dowse Dr.
Maungaraki, Lower Hutt, NEW ZEALAND
695-924 Group Size: 1
Contact Person: Mr. R. Siebers
Would appreciate New Zlnd REMark readers contact
Eager to expand group

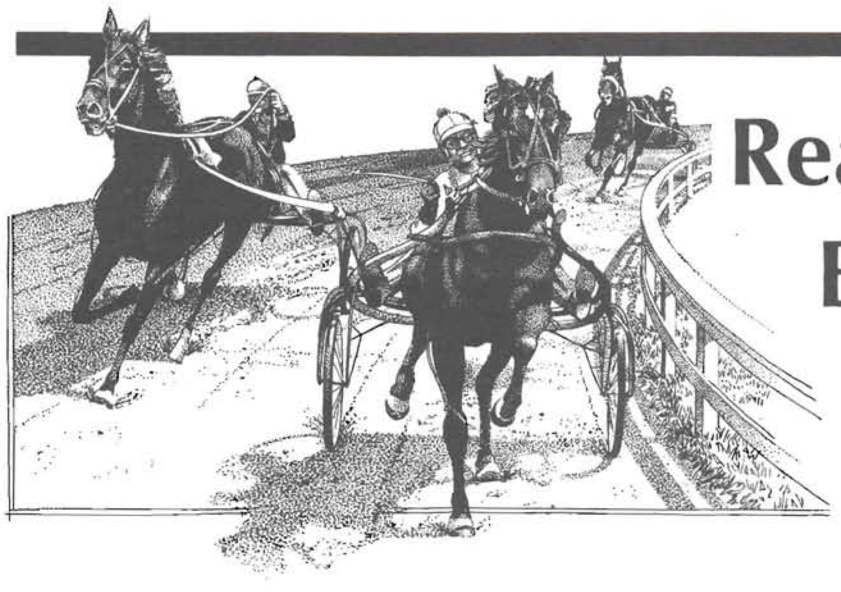
OKINAWA
OKIHUG (Okinawa Users Group)
C/O Carl Eaton Box 376 USAFSO
APO San Francisco, CA 96331
Group Size: 22
Contact Person: Carl H. Eaton
Meet one Friday month at 7:00 p.m.
Meeting place varies

PANAMA CANAL
Canal HUG
P.O. Box 1112
APO Miami, FL 34001
84-4094 Group Size: 6
Contact Person: Michael Gulick, Pres.
1st Tuesday 7:30 p.m. at Howard Air Force Base

PUERTO RICO, Rosario
PRHUG (Puerto Rico HUG)
Calle La Paz #706, Miramar
Santurce, PR 00907
809 725-1612 Group Size: 21
Contact Person: Joseph Gonzalez
Meet 2nd Sunday of odd numbered months

W. GERMANY, Frankfurt
Frankfurt HUG
American Consulate General FRDCO
APO NY, NY 09757
566187 Group Size: 3
Contact Person: Carl Lovett

W. GERMANY, Sprendlingen
HUG-Deutschland
Robert-Bosch-Strasse 32-38
D-6072 Dreieich W. GERMANY
06103-34037 Group Size: 200
Contact Person: Lydia Luguet



Realistic Benchmark Results

Which H/Z System Is Fastest?

Henry Fale
Quikdata Computer Services, Inc.
2618 Penn Circle
Sheboygan, WI 53081

One day when I had nothing to do (HA HA!), I decided to see which of my H/Z computer systems was fastest. I recently converted to CP/M Plus on my H-89 and that seemed to really be a fast system compared to the Magnolia CP/M 2.242 I had been using. When I went to CP/M+, I was telling professionals in this field about the 'apparent' (and I say apparent because I never tested it, it was just instantly noticed sitting down and using the system - when you constantly work with 8000 record databases, there is quite a difference between a 1 hour index and a 6 hour index) speed increase and they all told me it was impossible. That also forced me and my ego into performing the tests. No one could believe the difference in results I claimed to be obtaining. I then got a Z-100 with a winchester and although that seemed fast, it did not seem to match my H-89! How could that be, a 4MHz H-89 beating a 5MHz Z-100 (both with winchesters).

A benchmark was indeed necessary, but I had to develop my own since too many benchmarks we see in all the magazines only tell part of the story - CPU time. In most programs, especially where much disk access is taking place (the bottleneck of the programs), the published benchmarks don't tell the true story. The majority of the business software we use around here depends heavily on disk I/O and CPU speed. We use dBII almost extensively for all our business applications and mailing list maintenance. Whenever you use something like dBII for indexed files, the power is enormous, but the disk must be accessed twice for every record - once for the index and once for the actual record. A SORT or INDEX takes a long time for medium sized files (1000 to 5000) and an unbelievable time for large files (8000 or so). In fact, with something like dBII where sort and index are done on the 'bubble' technique, the time increases almost exponentially with the number of records.

The benchmarks done here were standardized as much as possible, using only H/Z computers with standard available options, if any options were used. Tests were performed on a range of 5" hard sector floppy drives to winchesters, most at both 2 and 4MHz. Please note that all floppy drives were computer start/stop motor drives, thus were not ready to go and rotating upon access, although even these tests kept all drive motors going constantly once the test started. A winchester is always spinning. The 5" floppy spins at 300 RPM, the 8" floppy at 360 RPM, and the winchester at about 3000 RPM!

These tests allow one to see the disk access part vs the CPU speed. For this test I used 836 records, each being a random indexed file of 165 bytes. For us this is a very small database, as our main database consists of over 8000 records. dBII 2.3B was used as the main program under various CP/M operating systems on various H/Z computers at various CPU speeds and using various processors. A dBII command file was made to first pack the records. This involves going through every record and deleting the ones which were previously marked for deletion, which none were. Then the entire file of 836 records was indexed on its 9 byte 'key' character field. This involves reading the entire file and creating a sorted index, or reference pointer, to the main data base file. After this, it then indexed the main database file according to a 25 byte 'name' character field. Naturally, the larger the field, the larger the index will be and the longer it will take to index that part. All this was done using one of our modified modules in QUIKMAIL. Since 'sorting' or indexing involves much CPU power, and it is also very disk I/O bound since dBII builds both its database file and index on disk, I felt it was an excellent test. The 836 records occupied 136K disk storage. The 9 character 'key' field when finished occupied 16K disk storage and the 25 character 'name' field occupied 42K disk storage space. This should be sufficient data for anyone to repeat the experiment on other systems, or verify the data.

The program used very little screen I/O, thus that was not a factor. Some systems were going at 9600 baud, some at 19.2K baud, and the Z-100 seemed to be about 4800 baud. If much screen I/O was going on, the Z-100 would have been slower than tests indicate. There was no printer output either, thus printer speed, another bottleneck, did not enter into the results. Keep in mind that when I mention the H-8, the results should apply equally to the H-89, and likewise when I mention results found on the '89, it should apply to a similarly structured H8 system.

This test is the result of nearly a nervous breakdown from converting so many disk formats, and having the stopwatch batteries constantly run down only to have to repeat the test again (it is a Heath stopwatch and the floppy tests were very time consuming).

I don't want to go on sounding like I'm needlessly babbling, but I felt the data was important since it gives a very thorough test to all H/Z computers with many different accessories and different versions of CP/M operating systems.

The Results

Z-100 with 192K RAM, CP/M-85 on the 8-bit 5MHz processor using an 8" external floppy stepping at 3ms - 19:20 (minutes:seconds). Same Z-100, same 8 bit 5MHz 8085 processor with the built-in winchester - 9:26. Same Z-100 and operating system, only this time on the internal soft sector 5" floppy - 46:42. These results are not surprising. The disk transfer rate of data on an 8" soft sector double density floppy is twice that of the 5". The surprising thing is, more than twice the time difference, perhaps because of buffering differences. In each case, the processor, operating system and CPU chip were identical. The winchester results were not at all surprising, since the winchester is always much faster, sometimes about 10 times faster than the floppies. That's why I always feel any serious business computer must have a winchester on its system, and have the CPU clocked to its highest rate to get efficient results from the machine. After all, time is money! The hobbyists can afford to start a sort or run a G/L for a small customer and come back 7 hours later. The business man cannot - not to mention the reliability of winchesters.

I completed additional tests on the Z-100. This test used the Z- 100 with 192K internal main board memory and 256K Z-205 card. Tests were run on the 16 bit side, under MP/M-86. Three terminals were installed but idle except for one where occasional DIR's were done to be sure the system was multi-user. The MP/M is Barry Watzman's implementation complete with interfacer IV serial I/O card for the extra terminals. A winchester was used. Time for the test was 12:05. This is not bad since the dBII CP/M-86 version (set up for MP/M operation) is actually a translated 8 bit version, thus does not have the 16 bit speed advantages. Also considering MP/M overhead, I don't think it's real bad.

Next is the 64K H8 with the Z-67 winchester running the H/Z 2.2.03 CP/M operating system at 2MHz. The 4MHz was run on the same system, modified by BIOS-80 for 4MHz operation and H-17 support from Livingston Logic Labs. The H8 had the Trionyx Z80 CPU, 4MHz ready memory board from Trionyx and gold mother board, and is fan cooled for superior dependability. Z67 winchester at 2MHz - 50:54. At 4 MHz - 48:29. I was extremely disappointed in these results. The Z-67 was Zenith's best answer for the businessman's super system a few years back, and the Z-100 on 8" drives beats it at either speed - and by a mile!! I'm sorry, but I wouldn't dare run that test on the built-in Z-47 compatible 8" floppy. It would have either torn the heads apart with constant head loading and unloading, or I'd still be waiting for the test to finish. Being fair to the Z-67, it was yesterday's technology released too late after its time and it does make a good heater - but it is very reliable!

H8 using Heath's Z80 CPU card and also Heath's 64K RAM memory, only running at 2MHz on 5" hard sector drives. Keep in mind that a 5" hard sector drive transfers at 1/2 that of soft sector 5" which is only 1/2 that of 8" which is many times slower than winchesters - whew! This test again used H/Z CP/M 2.2.03 modified by Livingston's BIOS-80, since naturally we can't fit that much test data on a 100K disk! The test required minimum of 300 some K, as would most business applications. Our 8000 name 128 byte record client database takes up several megs of winchester storage for the database and indexes. And you should see the index times on dBII! 836 names were all I could tolerate during the tests! Anyway, back to the results, the time was 70 some minutes (sorry, stopwatch batteries burned out again and this one I did by clock and I did not feel like repeating it!)! This is perfectly understandable considering the slow speed of 5" hard sector floppies. I don't care if it is less expensive, any business man using 5" hard sector floppies (or soft for that matter, especially if he uses his computer a lot) is out of his head! It did surprise me that the hard sector 5" floppy was really not much slower than the Z67 winchester! Totally blew my mind!

H8 using the Heath Z80 CPU and Heath's 64K RAM running at 2MHz with 48 TPI DS DD drive on the soft sector H-37 controller under Heath CP/M 2.2.03. Time was 37:52. 5" soft sector floppy beating the Z-67 winchester? Wow!

What about our Quikstor winchester? Using the H8 with Trionyx Z80 CPU and 64K RAM card, the winchester was interfaced to the H8 using the WH8-37 card. Again, H/Z 2.2.03 CP/M was used and the tests were run at both 2 and 4MHz. Time at 2MHz - 12:57. Time at 4MHz - 8:57. Again, a bit surprised that our standard CP/M H8 with a winchester and CPU zipping at 4MHz did indeed beat the 5MHz Z-100 with winchester! I don't quite understand that, but am very disappointed with my Z-100 and pleased with my H8/ Quikstor combo.

Next test was with an H-89 running Magnolia CP/M 2.2.42, 64K RAM with our 15 meg winchester at 2MHz - 12:21. Not bad compared to the Z-100 5MHz with winchester. Consider this when thinking about trade vs update. Since I no longer use 2.2.24 CP/M and it was a real pain getting this back up on the winchester, no other tests were done on this CP/M with Magnolia DD board and SASI interface.

I am presently using, and have been for about 6 months, Magnolia Microsystem's CPM+ complete with 128K RAM card. I am operating at 4MHz CPU with 19.2K baud CRT rate and 15 meg Quikstor winchester. This system never ceases to amaze me. In addition, I also have on this system a set of 5" hard sector H17 drives, 5" soft sector H37/MMS/Z-100 compatible drives, 8" CP/M compatible, Z-47 and MMS compatible drives via MMS 77316 double density controller, running both HDOS and CP/M and CP/M+! (Guess which system I use for all my disk conversions?) This system beats the Z-100 very shamefully, and needless to say, it is my main business system and has been even before CP/M Plus came around. Here are the astounding results: 8" floppy - 26:30 at 2MHz and 24:10 at 4MHz! Not as much difference as I expected, but then the programs are quite I/O bound and that accounts for much of the processor's time. The 15 meg winchester had unbelievable times of 4:42 at 4MHz and 7:32 at 2MHz - beating the Z-100 to pieces even when running the H-89 at 2MHz! There's much more to the story than meets the eye, however, since CP/M Plus uses more and better disk buffering and uses hash techniques for directory R/W - this makes for a faster operating system, especially concerning directory lookup. Also, with CP/M+ CCP is always in banked memory. Some of the banked memory is also used for disk buffering and other efficient functions, thus there is none for the user in the Magnolia implementation. But fast it is.

Brad Gjerding of Magnolia also ran the tests for me in their Magnolia File Server being used in a single user computer (that's actually what a file server is, a 4MHz Z80 computer). He was running CP/M+ at 4MHz. The 8" drive completed the test in 14:00, while the 15 meg winchester took 4:56. The winchester time was very close to the results I obtained using the H-89 at 4MHz with CP/M+, but the 8" floppy was faster for some unknown reason. Brad next ran the test in the network mode (via MP/M and CP/NET) using E29 terminal requester to access the file server. This test completed in 20:47, which is about what was expected since CPU power is used to handle the network.

There you have it. The results show some interesting things about system efficiency using different CPU speeds and various assortments of disk drives. The most important differences here and the greatest speeds were obtained in every case (except Z-67) using winchester and CP/M+.

Quick Summary

I'll itemize the findings in descending time order in a table for easy comparison. Because of column width I'll use the following abbreviations: DS = double sided, W = winchester drive, 2M = 2MHz, 4M = 4MHz, MMS = Magnolia 2.2.04 CP/M, MC+ = Magnolia CP/M+, HC = Heath CP/M 2.2.03, HLC = Heath CP/M 2.2.03 modified by Livingston's BIOS-80, QSW = Quikdata/Heath Quikstor Winchester 2/4 MHz BIOS, C5 = Zenith's CP/M-85 = Z-100 8 bit CP/M, QSW = Quikstor 15 meg winchester, FS = MMS File Server, MPM = MP/M multi user operating system. All computers have maximum memory, 64K for H8 and H-89, 192K Z-100.

H8	H17	HLC	2M	70:00
H-89	8"	MMS	2M	61:02
H8	Z67W	HLC	2M	50:54
H8	Z67W	HLC	2M	48:29
Z-100	5"	C5	5M	46:42
H8	Z37	HC	2M	37:52
H-89	8"	MC+	2M	26:30
H-89	8"	MC+	4M	24:10
FS	W	MPM	4M	20:47
Z-100	8"	C5	5M	19:20
FS	W	MC+	4M	14:00
H8	QSW	QHC	2M	12:57
H-89	MMS	QSW	2M	12:21
Z-100	W	MPM	5M	12:05
Z-100	W	C5	5M	09:26
H8	QSW	QHW	4M	08:57
H-89	QSW	MC+	2M	07:32
FS	W"	MC+	4M	04:56
H-89	QSW	MC+	4M	04:42

PAUL F. HERMAN

DATA SYSTEMS CONSULTANT
P.O. Box 535
St. James City, FL 33956

NEW - PERSONAL DATA MANAGEMENT SERIES
....including....

- Appointment Calendar
- Mailing List
- Household Inventory
- Periodical Index
- Record Index
- Audio Tape Index
- Vehicle Log
- Recipe List
- Coupon Organizer
- Library Index
- Video Tape Index
- Check Register

All on ONE disc for ONE price. . . $ 59.95

DOODLER Graphics Package $79.95

Z100 SOFTWARE

Prepaid orders sent post-paid. Florida residents please include 5 percent sales tax.

Send SASE for additional Info on Programs.

We are an authorized ZENITH Data Systems Dealer.

This entire ad was printed actual size using DOODLER, a Z-100, and a GEMINI-10x printer.

Dealer Inquiries contact - YELLOW ROSE SOFTWARE
10208 N. 30th Street, Tampa, FL 33612

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

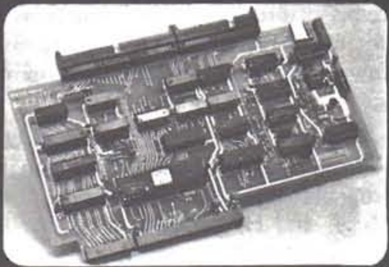
Limitations: MODIFY 89 is not a disk duplicate program. It is currently available for use with an H/Z89 or H/Z90 computer that has an FDC-880H double density 8" and 5.25" controller, using C.D.R.'s BIOS V.2.9 or with an H8 computer using the FDC-H8 by C.D.R. Systems, Inc.

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

Discovering Pascal With a Financial Calculator

Karl L. Remmler
13090 LaVista Dr.
Saratoga, CA 95070

Pascal was originated by Niklaus Wirth (reference 1) to teach structured programming. It has become one of the predominant programming languages and a favorite tool for implementing software on microcomputers. Other famous languages such as C and ADA have their basis in Pascal. One of Pascal's strongest points is its readability. This factor alone contributes heavily to many other desirable attributes. It is easy to learn and easy to use. Software development is not nearly as difficult or as messy in comparison with other languages such as FORTRAN and BASIC. Pascal is well suited as a program design tool. Program organization and implementation plans are very effective when prepared in a Pascal-like pseudo language. It eliminates the need for flow charting. Modular packaging of the software code in terms of **procedures** and **functions** invokes the divide and conquer approach greatly simplifying a program development task.

The example of a financial calculator is chosen to demonstrate some of the advantages of programming in Pascal and how H/Z-89, H/Z-19, and H/Z-100 features are implemented with this language. It also provides a basic library of procedures and functions for application in a variety of financial programs.

This code was written for Lucidata Pascal, version 3.J, from Polybytes, 325 19th St. S.E., Cedar Rapids, IA 52403. Tutorial Pascal articles in previous issues of REMark (reference 2) used this implementation. It has many excellent features, all of which can hardly be touched upon in this simple demonstration. Three of them, however, should be pointed out. For the sake of portability it adheres rather closely to the ISO standard; the few departures are well chosen, essential for serious programming, and are clearly identified. The cost is very reasonable and finally, I have not knowingly encountered any nasty bugs.

There are numerous books available for people who wish to learn and write programs in Pascal. I have selected three of these books. Reference (1) is a good reference for the standard. Reference (3) is a good book to read for learning Pascal. Reference (4) is an excellent book to consult for writing programs. The Lucidata Pascal Manual is also a good reference source. The distribution disk for Lucidata Pascal includes very exceptional demonstration programs. Each of these demonstration programs treats a specific topic of the Pascal programming language.

To avoid typing errors and to apply this example as a learning tool, I would recommend that you attack each procedure and function independently. Transform each one into a separate program, adding non-local definitions and declarations so that it is a self contained package. Compile and run the separate packages, testing each one. When you are finally satisfied, perform the inverse transformation with your editor and copy the file to your library disk.

Main Program

As can be seen in Figure 1, the main program is only a few lines of simple code. It's purpose is to perform the duties of a top level executive, commanding and controlling as required for specific actions of the program.

Data descriptions are always provided as the first part of every Pascal program. It is good practice to use standard identifiers, such as BELL, BS, DEL, ESC, and SP to define the global constants. If your compiler will allow it, another helpful practice is to reserve all upper case for symbolic names and constants. I have elected to only use upper case for constants in this example.

The constants ERR0 and ERR1 are used for error checking. Since this is a relatively short program, it seemed best to declare them as global constants. The GOTO with labels is used only for the data error protection. The variable flg is set to 1 for no data error and -1 if there is a data error. I have attempted to attain a reasonable amount of numerical precision and accuracy over a realistic range of solutions without adding too much extra code. Your compiler may require limits on such functions as exp() and ln() than those used in this example.

An array type, **string** having components of the standard type char, is defined for handling data I/O. All of the financial equation variables (n, nai, i, pmt, fv) and the two frequency parameters (pf and cf) are typed real. The menu selection variable ch is typed char to accomodate functions keys as well as number items in the menu. The equation parameter continuous is TRUE for continuous compounding and FALSE for discrete compounding. Parameter x is either 0 for ordinary annuity or 1 for annuity due; it is typed integer. The financial equation, its variables, and parameters are discussed in the sequel.

Pragmats are included in the program following the variable declarations. The Lucidata manual defines pragmats as sequences of characters embedded in a program text that do not form part of the formal language sequences. They are used here to direct the compiler to insert Pascal procedures and functions from the named file. (There are a number of other uses included in the Lucidata implementation, however, it is not necessary to go into that now.) I normally work with my programming tools (i.e. language processor, editor, utilities, etc.) on drive A and library of source codes on drive B. Therefore, b: is included with filenames for procedures and functions being used from my Pascal library.

The statement part of this program appears rather short and simple. With exception of the single **while** statement, it consists solely of procedure calls. Variables and parameters are first assigned their default values. Then the menu is painted on the screen with variables printed at the proper locations by the **procedure newvals**. Finally, function key assignments are shown by painting the 25th line with appropriate labels in reverse video.

The while loop implements user interaction with the program. Inkey reads the keyboard character. Then **procedure entry** enables the selection from the menu. If a number is selected, the user is prompted for a new value for that financial value. If a function key is selected, a new value is calculated for that variable holding the other four variables fixed. I have noticed that sometimes a user gets confused and presses a number key when he really wanted to press a function key. If this is a problem, code can be added to calculate a new value whenever a "C" is pressed at the data entry prompt.

Whenever the red key (F7 on the H/Z-100) is pressed, the while loop condition will cause an exit from the program to the operating system.

Screen Control

Three procedures (menu, line25, and locate) demonstrate essential programming methods for achieving screen displays needed for user interaction. The code for **procedure menu** was generated using Ed-A-Sketch and PIE by The Software Toolworks. MBASIC source code generated using Ed-A-Sketch is easily edited with the recording feature in PIE. Simply change the PRINT to write, replace double quotations with single quotations, replace semi-colons with commas, add the parenthesis, and terminate each write statement (except the last) with a semi-colon. Alternately, you can code the menu manually. Don't forget to add the begin-end. ESCape sequences and graphic symbols employed here are adequately described in your Heath/Zenith hardware manuals.

The first write statement in **procedure line25** is included to first clear the 25th line and turn off the cursor. (This is really not necessary, but a recommended practice to ensure a clean display). The second write statement re-enables the 25th line and positions the cursor on the 25th line. Subsequent write statements paint labels, in reverse video, for the appropriate function keys. The last two write statements provide ESCape sequences to position the cursor on the prompt line of the menu, turn the cursor on, and enable the block cursor.

The **procedure locate** is really two procedures built into one. It will provide a blank line in reverse video, with the cursor on the leading edge. The leading character position of this line is specified by the line and column variables, while its length is specified by the variable spaces. If you wish to locate the cursor without the blank line, assign spaces the value of 0 or less. You will find a very good description of the ESCape sequences for direct cursor addressing on page 5-8 of the H89 Operating Manual, a not-as-good description on page 10.42 of the H100 Technical Manual, and a rather poor description on page B.14 of the Z-100 User's Manual.

Function Keys

Escape sequences provided by the function keys are defined on pages 11-16 and B.19 of the H89 and H100 operating manuals, respectively. The repeat-until construct in the **procedure inkey** returns a one character string from the terminal. The until condition used here allows the ESCape character to be passed through and the ch variable only takes on the printable character. The write statement cleans up the display by backspacing and deleting the printable character.

Function key decoding is handled by the last six of the twelve selectable actions in the case construct of the **procedure entry**. Function keys F1 through F5 are used to call the respective routine for calculating the desired financial variable. The red key (F7 on the H/Z-100) satisfies the while condition in the main program, resulting in program termination and exit to the operating system.

I/O Methods

The remaining procedures and functions (inchr, inchr1, inchr2, newvals, and ctor) handle all the input and output data manipulations. Inchr, inchr1, and inchr2 are very similar and could very well be packaged into one unified procedure. For the sake of clarity, however, I have elected to treat them separately. These procedures are examples of what can be accomplished with Pascal to achieve user-friendly and bullet- proof programs. Inchr2 allows the user to enter numerical data in a character format, with the option of including commas, decimal point, sign and dollar sign. Bullet proofing is accomplished by ringing the bell, backspacing and erasing the character not included in the allowable set. Data entry is automatically terminated on the following conditions: (1) when the number of characters reaches a limit set ty the const MAXSTR, (2) when the number of digits after the decimal point are equal to the const POINTS, and (3) when eoln is TRUE. Inchr and inchr1 differ from inchr2 only in the string size permitted, specific characters allowed in the set, and conditions for terminating data entry. Rewriting these procedures as one unified and more general procedure would be a useful exercise. This can be accomplished by simply passing variables from the calling statement for the non- nominal allowable characters, MAXSTR and POINTS.

The Financial Equation

Five basic variables are required to describe most time-valued financial transactions: time period, interest rate, payments, present value and future value. A unique equation describing the relationship of these five variables is:

```
          (pv + c)*a + pv + fv = 0

where a   = (1 + i)↑n - 1
      b   = (1 +i*x)/i
      c   = pmt*b
      fv  = future value
      i   = effective interest
                      rate per period
      n   = number of payments
      pmt = periodic payment
      pv  = present value
      x   = 0 for ordinary annuity
        (payment at end of each period)
      x = 1 for annuity due (payment at
              beginning of each period).
```

fv,pv, and pmt take on positive values when money is received and negative values when money is paid out. For example, if pv is positive for amount received and fv is zero, then pmt must be given a negative value.

Financial Algorithms, Procedures, and Functions

The relationship between effective interest rate per period and the nominal annual interest rate is described by either of the following two relationships:

```
nai + cf*(1 -(1 + i)↑(pf/cf)) = 0
                  for discrete compounding
```

and

```
nai - ln((1 + i)↑pf) = 0  for continuous
                                 compounding,

where
    nai = nominal annual interest rate
      cf = compounding frequency per year
      pf = payment frequency per year.
```

A conversion to i from nai, as supplied by the user, is accomplished by **function effint**. An inverse conversion is performed as the final step in the procedure for calculating interest; this might have been coded as a separate procedure in order to build a more universal library.

When the payment is zero, the financial equation yields a solution for the effective interest,

```
            i = (fv/pv)↑(1/n) - 1
```

Otherwise, interest must be calculated by iteration using an initial guess. Newton's method is applied in **procedure interest** where convergence criteria is given by declaring tolerance as a constant. The initial guess is calculated by **procedure guess**. There exist other algorithms for computing an initial guess, however, this one is simple and reliable.

The remaining procedures numperds, presval, futrval, and payment are solutions to the financial equation for calculating the number of periods, present value, future value, and periodic payments, respectively. Standard Pascal does not provide for exponents and therefore it is necessary to implement the **function power**. Numeric problems that need to be avoided are ln(ERR0), divide by ERR1, exp(ERR2).

As usual, the mathematical algorithms and "number crunching" part of the program do not require many lines of code. There are, however, a number of different calculations to be made. These same calculations are required for a wide range of different financial applications and analysis procedures. To name a few: amortization schedules, various types of lending and leasing, residential home buying analysis, rent or buy residential analysis, income property analysis, mortgage pricing, etc. The virtue of Pascal programming structure is that one may implement a library of these functions and procedures and then, using the same procedures, it is a rather easy task to prepare various programs for different analyses. The approach is similar with BASIC, using subroutines and GOTO's. The line numbering requirement, however, makes it much more difficult and, to say the least, it sure can become rather messy.

Using the Financial Calculator

FINANCE.COM is a machine code program that will execute directly from the CP/M command line.

When the financial menu-table is first displayed on your screen, default values will be displayed for each of the financial variables. Any or all of these values may be changed by pressing the respective number on the keypad, and then entering the desired value on the prompt line.

When entering data, you may include commas, $-sign, %-sign, and decimal point where it is appropriate. Corrections can be made at any time while the cursor is still on the prompt line; use either the backspace or delete key. The backspace key will not clear characters from the screen as the cursor is moved back; they will be removed as you type over them. The delete key will clear characters as it backs up.

The new value will be entered into the finance table when the return is pressed. It will not be necessary, however, to press the return if you enter the maximum number of characters. In this case, when you enter the last character, the computer will ring her bell and put the number into the proper pigeon hole.

When you are satisfied with four of the five financial values in the table, press the appropriate function key and obtain the solution for the fifth variable. The computer will then display her solution in the respective pigeon hole. WARNING!! If you do not obey the sign convention, she will display a DATA ERROR message. She may also display this error message for unrealistic transactions which would otherwise result in numeric error conditions.

If a value is too large for the display format, a line of asterisks will be printed on the screen. This has not been a problem for us poor folk.

Default values are defined in the source code of procedure data. These values cannot be changed without either this code and a compatible compiler or a suitable patch. Sorry, I have not determined where the patch should be made in the program.

This program is for use by HUG members and readers of REMark. If you find any bugs or have any suggestions, I would appreciate hearing from you.

Conclustion

This is a useful and practical program in its present form. There are several refinements and enhancements, however, that may be added as useful exercises. The procedures inchr, inchr1, and inchr2 can be consoli-

Figure 1

```
program finance;
{ EIGENWARE TECHNOLOGIES, Karl L. Remmler, Prop.
  13090 Saratoga Drive, Saratoga, CA 95070
  (408) 867-1184
  For reader's of REMark }
const
  BELL = chr(7)   ;
  BS   = chr(8)   ;
  DEL  = chr(127);
  ERR0 = 0        ;
  ERR1 = 1E-6     ;
  ERR2 = 87       ;
  ESC  = chr(27)  ;
  SP   = chr(32)  ;
type
  string = array[-1..13] of char;
var
  n,i,nai,pv,pmt,fv,pf,cf  : real    ;
  ch                       : char    ;
  s                        : string  ;
  continuous               : boolean;
  x,flg                    : integer;
                              (*$I b:cls      *)
                              (*$I b:data     *)
                              (*$I b:menu     *)
                              (*$I b:line25   *)
                              (*$I b:ctor     *)
                              (*$I b:inchr2   *)
                              (*$I b:inchr1   *)
                              (*$I b:inchr    *)
                              (*$I b:inkey    *)
                              (*$I b:locate   *)
                              (*$I b:newvals  *)
                              (*$I b:power    *)
                              (*$I b:effint   *)
                              (*$I b:numperds *)
                              (*$I b:guess    *)
                              (*$I b:interest *)
                              (*$I b:presval  *)
                              (*$I b:payment  *)
                              (*$I b:futrval  *)
                              (*$I b:entry    *)
begin
   flg := 1;
   cls;data;menu;newvals;line25;
   while (ch<>'Q') do begin inkey;entry;end;
   cls
end.
```

dated into a single unified procedure. An option for calculating financial variables by pressing the C-key after selecting a menu number can be added. Finally, you may wish to add the "Do you really want to QUIT!!" second chance.

A menu procedure should be added so that the user can redefine the parameters (x, continuous, pf, cf) without having to recompile the program. Also default values can be adjusted by the program for consistency with parameters selected.

Numerical precision and accuracy might be improved somewhat. There are other iteration methods and convergence accelerators that you may wish to experiment with. When running this program, you will find some cases that may take an abnormal long time to converge.

One other useful addition, would be a procedure to print out an amortization schedule for selected solutions. This would be handled by adding the appropriate function key code to procedure entry.

Another useful exercise would be to write a Pascal program to convert the Ed-A-Sketch generated MBASIC to Pascal source code. Reference (4) can be consulted for some neat methods to accomplish this.

I have one final comment on the Lucidata Pascal. Since it is sufficiently standardized with respect to many other implementations, I find the conversion to other systems rather trivial. My experience with JRT Pascal is not as easy.

References

(1) Jensen, K. and Wirth, N., "User Manual and Report", Springer-Verlag, Berlin, 1974.

(2) Fale, H.E., "Pascal Corner - Part V", REMark, Issue 29, St. Joseph, MI, June 1982.

(3) Grogono, P., "Programming in Pascal", Addison-Wesley Publishing Co., Inc., Reading, Massachusetts, 1980.

(4) Kernighan, B.W., and Plauger, P.J., "Software Tools in Pascal", Addison-Wesley Publishing Co., Ltd., New York, New York, 1981.

Figure 2

```
procedure cls;
{ clears screen including 25th line
and resets cursor }
begin
   write(ESC,'y5',ESC,'y4');
   write(ESC,'y1',ESC,'E ')
end; { cls }


procedure data
{ This procedure contains all the parameter and variable
default values.  It could be replaced by a menu procedure
for configuration by the user, writing and
reading the data from a disk file. }
begin
  { initialize }
     ch          := chr(0) ; {preclude an initial character of Q }
  { parameters }
     continuous := false   ; {continuous or discrete compounding}
     x           := 0       ; {ordinary or annuity due}
     cf          := 12      ; {annual compounding frequency}
     pf          := 12      ; {annual payment frequency}
  { defaults }
     fv          := 1322.346; {future value}
     nai         := 0.1     ; {nominal annual interest rate, decimal}
     n           := 24      ; {number of payments}
     pmt         := -50     ; {periodic payment}
     pv          := 0.0     ; {present value}
end; { data }


procedure menu;
{ Created using Ed-A-Sketch and PIE (Products of Software
Toolworks).  See text for recommended procedure. }
begin
  write(ESC,'E',ESC,'x5');
  write(ESC,'w',ESC,'H',ESC,'J',ESC,'G');
  write(ESC,'q',ESC,'F',ESC,'Y',CHR(34),'<faaaaaaaa');
  write('aaaaaaaaaaaaac',ESC,'Y#<',ESC,'p',ESC,'CFIN');
  write('ANCIAL CALCULATOR',ESC,'q ',ESC,'Y$<eaaaaaaaaaaa');
  write('aaaaaaaaaad',ESC,'Y%4faaaaaaaaaaaaaaaaaaaaaaaaaaasaaaa');
  write('aaaaaaac',ESC,'Y&4<1> NUMBER OF PERIODS',ESC,'Y');
  write('&0',ESC,'Y&T   ',ESC,'Y&[',ESC,'Y',CHR(39),'4vaaaaaa');
  write('aaaaaaaaaaaaaaaaaaabaaaaaaaaaaat',ESC,'Y(4');
  write(ESC,'C<2> ANNUAL INTEREST RATE',ESC,'C');
  write(ESC,'Y(S       ',ESC,'Y([',ESC,'Y)4vaaaaaaaaa');
  write('aaaaaaaaaaaaaaaabaaaaaaaaaaat',ESC,'Y*4',ESC,'C');
  write('<3> PRESENT VALUE',ESC,'Y*0',ESC,'C');
  write(ESC,'Y+4vaaaaaaaaaaaaaaaaaaaaaaaaaaabaaaaaaaaaaat');
  write(ESC,'Y,4',ESC,'C<4> PERIODIC PAYMENT',ESC,'Y');
  write(',0',ESC,'Y,[',ESC,'Y-4vaaaaaaaaaaaaaaaaaaaaaaaa');
  write('aabaaaaaaaaaaat',ESC,'Y.4',ESC,'C<5> FUTURE VALU');
  write('E',ESC,'Y.0',ESC,'Y.[',ESC,'Y/4eaaaaaaaaaa');
  write('aaaaaaaaaaaaaaaauaaaaaaaaaaad',ESC,'Y0<',CHR(94),' TO');
  write(' ENTER VALUES ---> ',ESC,'pw',ESC,'G',ESC,'q');
end;  { menu }


procedure line25;
{ paint function key labels on 25th line }
begin
  write(ESC,'x5',ESC,'y1');
  write(ESC,'x1',ESC,'Y8 ');
  write('      ');
  write(ESC,'p','  F1-N  ',ESC,'q',' ');
  write(ESC,'p','  F2-I  ',ESC,'q',' ');
  write(ESC,'p','  F3-PV  ',ESC,'q',' ');
  write(ESC,'p',' F4-PMT ',ESC,'q',' ');
  write(ESC,'p','  F5-FV  ',ESC,'q',' ');
  write('           ',ESC,'p',' RED - QUIT ',ESC,'q');
  write(ESC,'K');             { RED => F7 for H100 }
  write(ESC,'Y',chr(48),chr(83));
  write(ESC,'y5',ESC,'x4')
end; { line25 }
```

☞

```
function ctor(var s : string) : real;
{ Converts a type char representation of a number to a type real.
The string may contain characters such as $,%, and commas. }
const
  MAXSTR = 13;
var
  r          : real   ;
  i, k, sign : integer;
begin
  r := 0; i := 1; k := -1; sign := 1;
  while (i < MAXSTR + 1) do begin
    if (s[i] = '-') then sign := -1;
    if (s[i] = '.') then k := 0;
    if (s[i] in ['0'..'9']) then begin
      r := 10 * r + ord(s[i]) - 48;
      if (k > -1) then k := k + 1
    end;
    i := i + 1
  end;
  if (k > 0) then for i := 1 to k do r := r/10;
  ctor := r * sign
end; { ctor }


procedure inchr2(var s : string);
{ Keyboard input for payment, future value and
present value. Allows input to contain $ and commas. }
const
  MAXSTR = 13      ;
  POINTS = 2       ;
var
  i, k : integer;
begin
  for i:=-1 to 13 do s[i]:=SP;
  i:=0;k:=0;
  repeat
    i:=i+1;
    while i<1 do begin
       write(SP,BELL);i:=i+1
    end;
    read(s[i]);
    if (s[i]=BS) then begin
       s[i]:=SP; i:=i-1; k:=k-1; s[i]:=SP; i:=i-1; k:=k-1
    end;
    if (s[i]=DEL) then begin
       write(BS,SP,BS);
       s[i]:=SP; i:=i-1; k:=k-1; s[i]:=SP; i:=i-1; k:=k-1
    end;
    if not (s[i] in [',','$','.','-',BS,DEL,SP,'0'..'9']) then begin
       write(BELL,BS,SP,BS);
       s[i]:=SP; i:=i-1; k:=k-1
    end;
    if (s[i]='.') then k:=1;
    if (k>0) then k:=k+1
  until (eoln) or (i>=MAXSTR) or (k>POINTS +1)
end; { inchr2 }


procedure inchr1(var s : string);
{ Keyboard input for interest rate. }
```

```
procedure inkey;
{ Returns just one character from the terminal,
allowing the ESCape character to pass through. }
begin
   repeat
      begin
         read(ch)
      end
   until (ch<>ESC);
   write(BS,DEL)
end; { inkey }


procedure locate(line,column,spaces:integer);
{ This procedure provides a blank line in reverse video, with the cursor
located on the leading position specified by the line and column vari-
ables.  Its length is given by the variable spaces.  If spaces is given
a value of zero or less, the cursor is still located at the specified
line and column, but without the blank line in reverse video. }
var
   l,c:char   ;
   i  :integer;
begin
   l:=chr(31+line)  ;
   c:=chr(31+column);
   if (spaces>0) then begin
      write(ESC,'Y',l,c,ESC,'p')      ;
      for i:=1 to spaces do write(' ');
   end;
   write(ESC,'Y',l,c,ESC,'q')
end;    { locate }


procedure newvals;
{ Prints new values in pigeon holes of menu-table. }
var id:real;
begin
  if (flg>0) then begin
    id := nai * 100;
    locate(7,53,0);write(  n:3:0) ;
    locate( 9,52,0);write( id:5:2);
    locate(11,50,0);write( pv:9:2);
    locate(13,50,0);write(pmt:9:2);
    locate(15,50,0);write( fv:9:2);
    locate(17,52,0);write(esc,'F','w',ESC,'G',ESC,'D') end
  else begin
    locate(19,1,0);
    write(ESC,'p');
    write('DATA ERROR - MAKE CORRECTIONS');
    write(ESC,'q',ESC,'J',BELL);
    flg := 1
  end
end;  { newvals }


function power(y,x:real):real;
{ raise y to power of x }
label
   1,2;
var
```

```
const
   MAXSTR = 6        ;
   POINTS = 3        ;
var
   i, k : integer;
begin
   for i:=-1 to 13 do s[i]:=SP;
   i:=0;k:=0;
   repeat
      i:=i+1;
      while i<1 do begin
         write(SP,BELL);i:=i+1
      end;
      read(s[i]);
      if (s[i]=BS) then begin
         s[i]:=SP; i:=i-1; k:=k-1; s[i]:=SP; i:=i-1; k:=k-1
      end;
      if (s[i]=DEL) then begin
         write(BS,SP,BS);
         s[i]:=SP; i:=i-1; k:=k-1; s[i]:=SP; i:=i-1; k:=k-1
      end;
      if not (s[i] in ['%','.',BS,DEL,SP,'0'..'9']) then begin
         write(BELL,BS,SP,BS);
         s[i]:=SP; i:=i-1; k:=k-1
      end;
      if (s[i]='.') then k:=1;
      if (k>0) then k:=k+1
   until (eoln) or (i>=MAXSTR) or (k>POINTS + 1)
end; { inchr1 }


procedure inchr(var s : string);
{ Keyboard input for number of payments. }
const
   MAXSTR = 3        ;
var
   i : integer;
begin
   for i:=-1 to 13 do s[i]:=SP;
   i:=0;
   repeat
      i:=i+1;
      while i<1 do begin
         write(SP,BELL);i:=i+1
      end;
      read(s[i]);
      if (s[i]=BS) then begin
         s[i]:=SP; i:=i-1; s[i]:=SP; i:=i-1
      end;
      if (s[i]=DEL) then begin
         write(BS,SP,BS);
         s[i]:=SP; i:=i-1; s[i]:=SP; i:=i-1
      end;
      if not (s[i] in [BS,DEL,SP,'0'..'9']) then begin
         write(BELL,BS,SP,BS);
         s[i]:=SP; i:=i-1
      end;
   until (eoln) or (i>=MAXSTR)
end; { inchr }
```

```
   arg0 : real;
begin
  if (y>ERR0) then arg0 := x*ln(y) else goto 1;
  if (arg0<=87) then power := exp(arg0) else goto 1;
  goto 2;
  1: flg := -1;
  2:
end; { power }


function effint(var nai:real):real;
{ Converts nominal annual interest to effective interest }
label 1;
var
   temp : real;
begin
   if (flg < 0) then goto 1;
   if (continuous) then temp := exp(nai/pf)-1 else
                  temp := power((1+nai/cf),cf/pf)-1;
   if (temp>ERR0) then effint := temp else begin
          effint := 1  ;
          flg := -1 end;
   1 :
end; { effint }


procedure numperds;
{ Number of payments from finance eq'n. }
label 1,2;
var
   temp,b, c, e : real;
begin
   if (flg < 0) then goto 2             ;
   temp := effint(nai)                  ;
   if (temp>ERR1) then begin
         i := temp                      ;
         b := (1+i*x)/i                 ;
         c := pmt*b                     ;
         e := (c-fv)/(c+pv)             ;
         if (e>ERR1) then
            temp := ln(e)/ln(1+i);
         if (temp>ERR0) then n := temp else goto 1 end
   else goto 1;
   goto 2;
   1:
      flg := -1;
   2:
end;  { numperds }


procedure guess;
{ Initial guess for iterative solution of effective interest rate }
var
   tv:real;
begin
   tv:=abs(pv)+abs(fv);
   i:=abs(pmt/tv)+abs(tv/(pmt*n*sqr(n)));
end; { guess }
```

```
procedure interest;
{ Calculates interest from financial eq'n. }
label 1,2;
const
  TOLERANCE=0.00001;
var
   temp,tv,a,b,c,f1,f2:real;
begin
     if (flg < 0) then goto 2;
     if pmt=0 then i:=power(fv/pv,1/n)-1 else
       begin
         guess;
         repeat
           temp := power((1+i),n)-1                  ;
           if (flg>0) then a := temp else goto 1;
           if (abs(i)>ERR1) then
                       b := (1+i*x)/i else goto 1;
           c := pmt*b                                ;
           f1:= a*(pv+c)+pv+fv                       ;
           f2:= (n*(pv+c)*(a+1)/(1+i))-(a*c)/i       ;
           i := i-f1/f2                              ;
         until (abs(f1/f2)<TOLERANCE)
      end;
    if (i>ERR1) then begin
        { Convert i to nai. }
        if (continuous) then
           begin
              tv  := power(1+i,pf);
              if (tv>ERR0) then nai := ln(tv) else flg := -1;
           end
           else begin
              tv  := power(1+i,pf/cf);
              nai := cf*(tv-1)
           end
    end else goto 1;
goto 2;
1 :
   flg := -1;
2 :
end; { interest }


procedure presval;
{ Present value from finance eq'n.
See text for sign convention. }
label 1,2;
var
   temp,a,b,c : real;
begin
   if (flg < 0) then goto 2                ;
   temp := effint(nai)                     ;
   if (temp>ERR1) then i := temp else goto 1;
   a := power(1+i,n)-1    ;
   b := (1+i*x)/i         ;
   c := pmt*b             ;
   pv:= -(fv+(a*c))/(a+1);
   goto 2;
   1: flg := -1;
   2:
end; { presval }
```

```
      locate(11,50,9);
      locate(19,1,0);
      writeln('ENTER PRESENT VALUE AMOUNT: ',ESC,'K');
      write(ESC,'p','Positive for value received and negative for ');
      write('value given out.',ESC,'q',ESC,'K');
      locate(19,29,0);
      inchr2(s); pv := ctor(s)
      end;
   '4' : begin
      locate(13,50,9);
      locate(19,1,0);
      writeln('ENTER PERIODIC PAYMENT AMOUNT: ',ESC,'K');
      write(ESC,'p','Positive for payment received and negative for ');
      write('amount paid out.',ESC,'q',ESC,'K');
      locate(19,32,0);
      inchr2(s); pmt := ctor(s)
      end;
   '5' : begin
      locate(15,50,9);
      locate(19,1,0);
      writeln('ENTER FUTURE VALUE:',ESC,'K');
      write(ESC,'p','Positive for value received and negative for ');
      write('value paid out.',ESC,'q',ESC,'K');
      locate(19,21,0);
      inchr2(s); fv := ctor(s)
      end;
   'S' : numperds;
   'T' : interest;
   'U' : presval ;
   'V' : payment ;
   'W' : futrval ;
   'Q' : write('  ',ESC,'p','QUITTING',ESC,'q')
   end otherwise write(BELL);
   newvals
end; { entry }
```

ZWAR!

Tired of trying to get your latest Pascal compiler to work? Your spreadsheet won't compute? Sick of syntax errors in line 4658? It's time to take a break and indulge your fantasies of global domination with ZWAR.

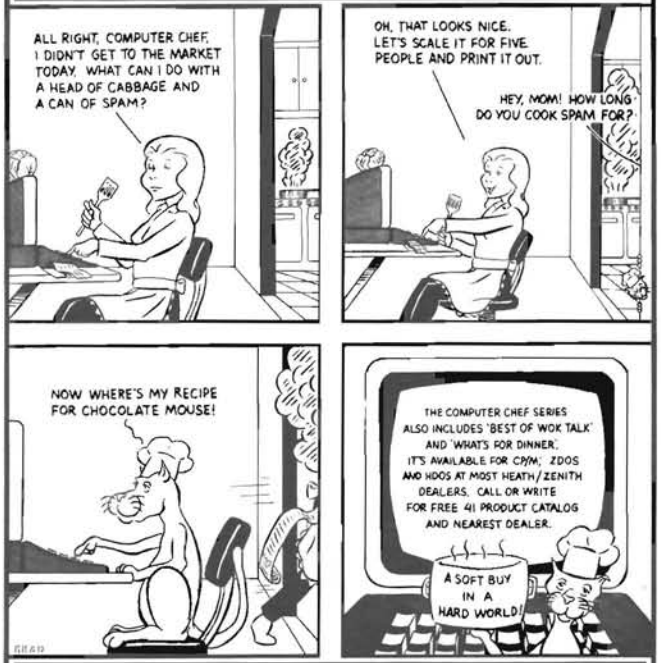
ZWAR is a *two player* strategy game of world conquest. The screen displays a map of the world and the Soviet and American cities. You must decide what mix of the nine available weapons systems you will build to outwit your enemy. You can destroy his cities with ICBM's, or attempt to capture them with an invasion, thus gaining their production facilities for your own use. Each weapon has its strength: carriers and destroyers for sea superiority, ABM's for defense against missiles, jets for quick response, satellites for spying on enemy activity. The possible strategies are nearly endless.

So if you're getting tired of humdrum computing, grab a friend and try ZWAR. It's your chance to make history.

ZWAR, $19.95 + $1.50 shipping. For H8-H19, H/Z89 or H/Z100 systems (color graphics on the Z100), 56k, and HDOS, CP/M, or CP/M-85. Specify hard or soft sector disk. Free catalog available.

APOGEE SOFTWARE
Box 15124
Savannah, GA 31416
(912) 925-3765

The Software Toolworks® presents:
THE COMPUTER CHEF™ SERIES


ALL RIGHT, COMPUTER CHEF, I DIDN'T GET TO THE MARKET TODAY. WHAT CAN I DO WITH A HEAD OF CABBAGE AND A CAN OF SPAM?
OH, THAT LOOKS NICE. LET'S SCALE IT FOR FIVE PEOPLE AND PRINT IT OUT.
HEY, MOM! HOW LONG DO YOU COOK SPAM FOR?
NOW WHERE'S MY RECIPE FOR CHOCOLATE MOUSE!
THE COMPUTER CHEF SERIES ALSO INCLUDES 'BEST OF WOK TALK' AND 'WHAT'S FOR DINNER.' IT'S AVAILABLE FOR CP/M, ZDOS AND HDOS AT MOST HEATH/ZENITH DEALERS. CALL OR WRITE FOR FREE 41 PRODUCT CATALOG AND NEAREST DEALER.
A SOFT BUY IN A HARD WORLD!

The Computer Chef Series includes Computer Chef and Best of Wok Talk, $29.95 each, and What's for Dinner, $19.95. The programs store, scale and help choose from over 300 available recipes or from those you've entered.
The Computer Chef Series from your local retailer or The Software Toolworks, 15233 Ventura Blvd., Suite 1118, Sherman Oaks, CA 91403 (818) 986-4885.

CP/M is a registered trademark of Digital Research.

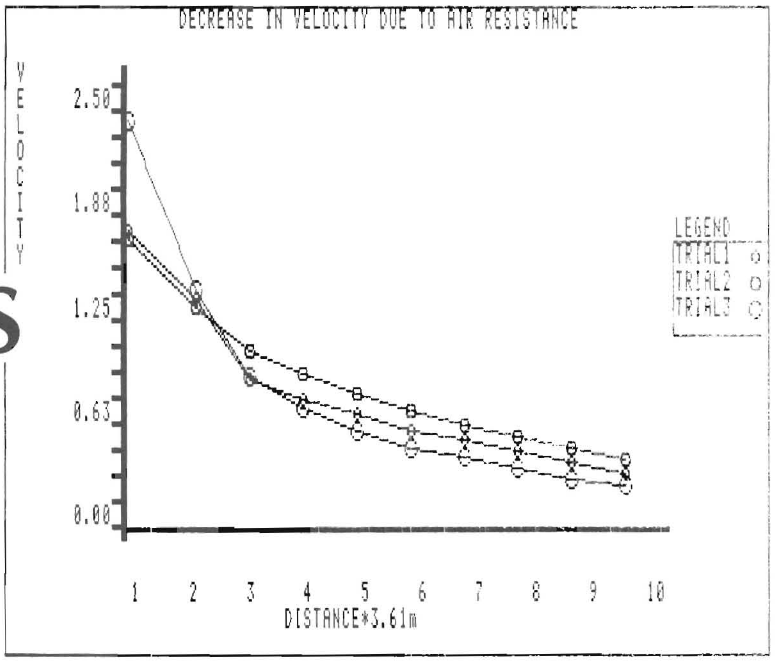
```
procedure payment;
{ Periodic payment from finance eq'n.
See text for sign convention. }
label 1,2;
var
   temp,a,b:real;
begin
   if (flg<0) then goto 2;
   temp :=effint(nai)    ;
   if (temp>ERR1) then i := temp else goto 1;
   a :=power(1+i,n)-1;
   b :=(1+i*x)/i      ;
   pmt:=-(fv+pv*(a+1))/(a*b);
   goto 2;
   1 : flg := -1;
   2 :
end; { payment }


procedure futrval;
{ Future value from finance eq'n.
See text for sign convention. }
label 1,2;
var
   temp,a,b,c:real;
begin
   if (flg<0) then goto 2;
   temp := effint(nai)    ;
   if (temp>ERR1) then i := temp else goto 1;
   a :=power(1+i,n)-1;
   b :=(1+i*x)/i      ;
   c :=pmt*b          ;
   fv:=-(pv+a*(pv+c));
   goto 2;
   1 : flg := -1;
   2 :
end; { futrval }



procedure entry;
{ This procedure is for interactive data entry
of financial values and selection of the desired solution.}
begin
  case ch of
   '1' : begin
      locate(7,53,3);
      locate(19,1,0);
      write('ENTER NUMBER OF PERIODS:  ',ESC,'J');
      locate(19,26,0);
      inchr(s); n := ctor(s)
      end;
   '2' : begin
      locate(9,52,5);
      locate(19,1,0);
      write('ENTER ANNUAL INTEREST RATE: ',ESC,'J');
      locate(19,29,0);
      inchr1(s); nai := ctor(s); nai := nai/100
      end;
   '3' : begin
```

OKIDUMP.BAS



Timothy Ross
1716 S. Soland, Apt. 30
Las Cruces, NM 88001

Oh boy! My first article! I've been reading REMark since October 1983 and have been impressed by the articles and letters I've read. In reading those letters I have noticed that there are at least three people out there, myself included, who run an H/Z-100 with an Okidata 92 dot matrix printer. So I figured that if there's three, there may be four or even five people with a similar setup, people who might have a use for this program.

I'm attending college right now, majoring in Electrical Engineering. I've been exposed to FORTRAN and BASIC and have a working knowledge of the two languages. I imagine that I will be required to use these languages in future courses as well as in my professional career. I anticipate the need to use this computer for graphing functions and, for fun, drawing pretty pictures. Last winter, during semester break, I found myself staring at my "Z", running simple little programs which place shapes all over the screen and wondering how to get a printed copy of these graphics screens. I looked in the HUG software list and, to my dismay, there was no screen dump utility specifically for my Okidata 92. Curses! Multiple colorful four-letter words! I had a bit of time on my hands, video screens filled with neat stuff, a stack of manuals, a silent printer, and a large measure of curiosity. Let's take a look at the monochrome screen, the manuals, and the Okidata 92.

The Screen and the Printer

The screen is made up of dots, thousands of them. The Okidata printer prints dots. There's got to be a way to make them work together. The video screen is 640 horizontal points by 225 vertical points. This yields an aspect ratio of 640/225 or something like three dots horizontal for every dot vertical to display a square. This program will actually print 2 dots in the vertical direction for every horizontal line. In the graphics mode, set at 10 characters per inch, the Okidata 92 will print a line 7 dots vertical by 480 horizontal. To make the printer work it must be fed seven-bit patterns, the least significant bit being the top dot printed, the most significant bit is the bottom dot. In order to get all 640 video points to the printer (remember the Oki is only 480 dots across), it is necessary to print sideways, taking a slice of the screen starting at the upper right corner and scanning down to the bottom right of the screen. This yields a slice 7 dots wide and 225 dots high. This information must be formatted for output to the printer. Then another slice is taken, starting seven dots from the first. This slicing technique is continued all the way across the screen. The last column is tricky because it will be only 3 dots wide, (640 MOD 7=3).

The first problem I encountered in writing this program was how to get the information from the screen into a format suitable for manipulation by a beginning programmer, myself. The Heath-Zenith manuals are loaded with information about the screen. I spent many an hour reading the H/Z-100 technical manual, especially chapter 4, The Video Logic Board. There is a series of formulas on page 4.15 which, upon interpretation and comprehension, allows one to PEEK and POKE the video screen with gusto. Video memory can be accessed 8 bits at a time in 8 bit increments. Since the Okidata requires 7 bit patterns, the video memory must be stepped through in 7 bit increments. POKEing the video screen presents a few problems such as combining the MSB bit of one video byte with 6 bits of the next byte, and 2 MSB bits of this second byte with 5 bits of the third and so on. The problem is approached only with massive amounts of intricate number crunching and brow wrinkling, a bit beyond me at this stage, I'm afraid.

Then there is the POINT command. POINT returns the color attribute of a single pixel at a specified location. I wrote a routine which used the POINT command to read every pixel on the screen, one at a time, convert the returned value to a power of two, and add the values so attained for seven pixels across, corresponding to the seven dot height of the Okidata 92. I then sent this number to the printer and stepped down one line where my program took the next seven pixels, one at a time. And so it went, down and across the screen. The program worked, sort of, but took over two hours to dump a single screen. I realize this is an impressive figure and probably breaks many records, but the program is not for sale, don't ask for a copy. I calmly sat down and calmly screamed, "There's got to be a better way!!"

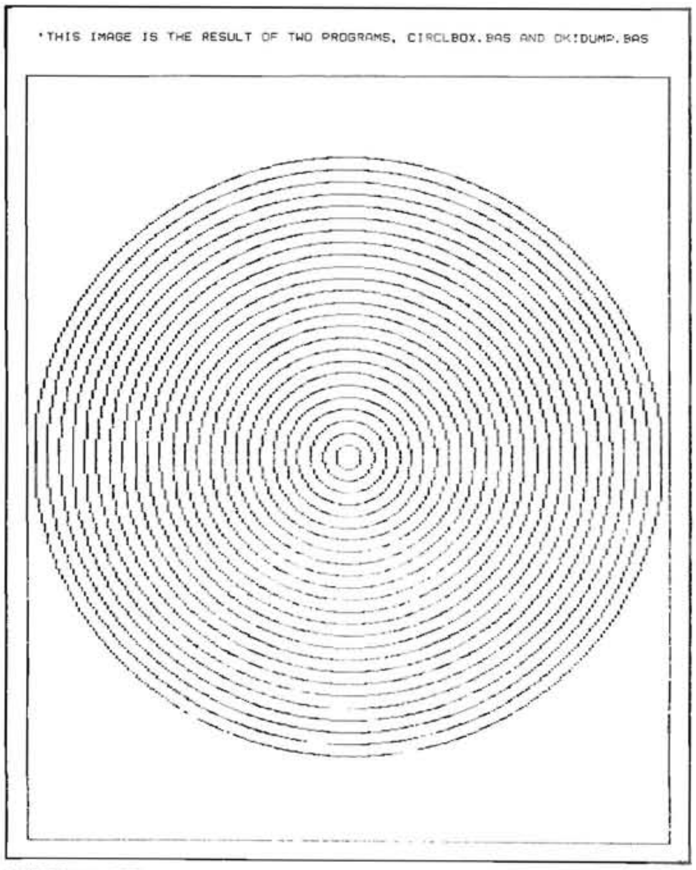
There is a better way, the GET command. The GET command trans-

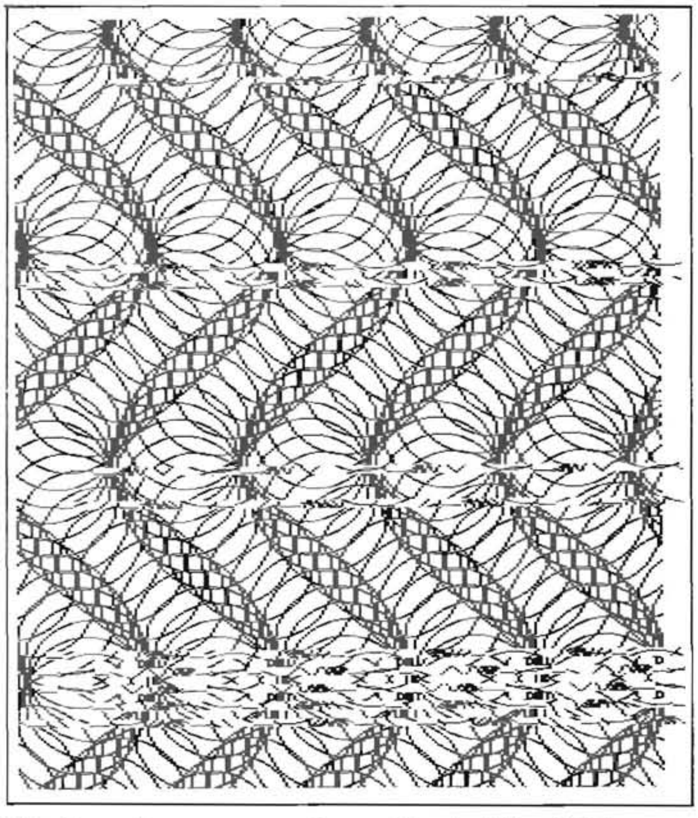
```
10 LPRINT CHR$(3) ;
20 FOR X=0 TO 127
30 LPRINT STRING$(4,X) ;CHR$(3) ;CHR$(2) ;X;"}{";CHR$(3);
40 NEXT X
50 LPRINT
```

```
 0 )(¯ 1 )(¯ 2 )(' 3 )(- 4 )(" 5 )(- 6 )(" 7 )(- 8 )(______ 9 )(= 10 )(" 11 )(- 12 )
(= 13 )(- 14 )(▪ 15 )(- 16 )(= 17 )(= 18 )(= 19 )(= 20 )(= 21 )(= 22 )(▪ 23 )(- 24 )
(= 25 )(= 26 )(= 27 )(▪ 28 )(▪ 29 )(▪ 30 )(▪ 31 )(_ 32 )(= 33 )(= 34 )(= 35 )(= 36 )
(≡ 37 )(= 38 )(▪ 39 )(= 40 )(≡ 41 )(▪ 42 )(▪ 43 )(= 44 )(▪ 45 )(▪ 46 )(▪ 47 )(_ 48 )
(= 49 )(= 50 )(= 51 )(= 52 )(▪ 53 )(▪ 54 )(▪ 55 )(▪ 56 )(▪ 57 )(▪ 58 )(▪ 59 )(▪ 60 )
(▪ 61 )(▪ 62 )(▪ 63 )(_ 64 )(= 65 )(= 66 )(= 67 )(= 68 )(= 69 )(= 70 )(= 71 )(= 72 )
(≡ 73 )(≡ 74 )(≡ 75 )(= 76 )(≡ 77 )(▪ 78 )(▪ 79 )(= 80 )(= 81 )(= 82 )(= 83 )(= 84 )
(≡ 85 )(= 86 )(▪ 87 )(= 88 )(≡ 89 )(▪ 90 )(▪ 91 )(▪ 92 )(▪ 93 )(▪ 94 )(▪ 95 )(_ 96 )
(= 97 )(= 98 )(= 99 )(= 100 )(= 101 )(= 102 )(= 103 )(= 104 )(≡ 105 )(= 106 )(≡ 107
 )(= 108 )(≡ 109 )(= 110 )(▪ 111 )(_ 112 )(= 113 )(= 114 )(= 115 )(▪ 116 )(▪ 117 )(:
 118 )(▪ 119 )(▪ 120 )(▪ 121 )(▪ 122 )(▪ 123 )(▪ 124 )(▪ 125 )(▪ 126 )(▪ 127 )(
```

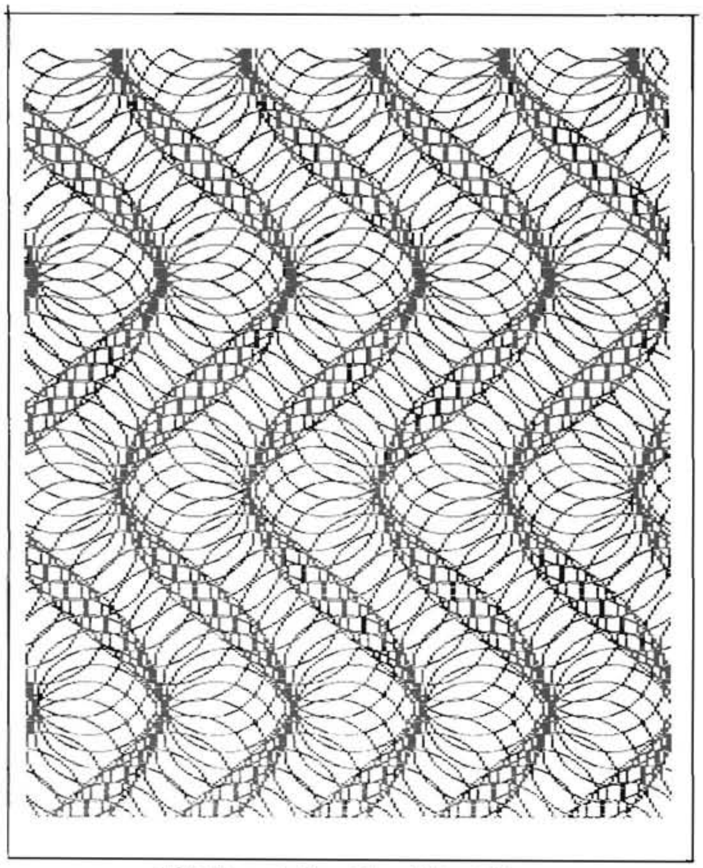
This routine prints all dot patterns from 1 to 127. Notice each character is printed 4 times, except for #3. Also notice #9.

fers the attribute information for each dot in a rectangle contained within the boundaries defined by the specified points and places this information in an array. What a mouthful! The array should be INTEGER. There is a formula on page 8.22 of the Z-BASIC manual which gives an idea of the size array required. It is reproduced here: Y, where X = the number of columns to be stored and Y = the number of rows to be stored. Lets see: X=7, Y=225. That works out to 675 I think. I was a little confused here so I wrote a little program which created an integer array of 1000 elements, DIM A%(1000). I wrote the number 99 into each element of the array. I then filled the screen with reverse video SPACE's so the whole screen was lit. Time for a test. GET (631,0)-(639,224),A%. My program then printed each element of the array. Guess what, only 340 elements (DIM A%(339)) are needed for this array. The first two bytes (the first element, element 0) of the array gives the X dimension in bits, minimum 24, the second two bytes give the Y dimension in lines. The rest of the array gives the data which specifies, in 2's complement numbers, which bits in the individual color planes are active. On page 8.27


'THIS IMAGE IS THE RESULT OF TWO PROGRAMS, CIRCLBOX.BAS AND OKIDUMP.BAS



This figure shows a screen dump without adding 128 to every character.



This figure is fixed by adding 128.

and 8.28 of the Z-BASIC manual is a program which will give an idea of what is happening in the video planes for a single character. I won't reproduce it here but suggest that a FOR/NEXT loop be added after line 260 which would print each element of the array P00%. Seeing the visual representation of the array in conjunction with the array's numerical contents greatly eases the job of interpreting the video screen.

Element 0 of the array specifies how many pixels horizontally in the three color planes, in multiples of 8, have been stored in the array. If there are less than eight bits stored, meaning that if the width of the slice being stored in the array is less than 8 dots, the rest of the byte will be filled with zeroes and left justified. The minimum value of element 0 is 24, which corresponds to 8 bits and three colors. Element 1 specifies the height of the slice. In the case of a single character there would be 9 lines. For this screen dump there are 225 lines, so element 1=225. Things start getting sticky. Element 2 of the array, and every third element from then on contains information about the blue and red planes. Element 3 contains information about the blue and green planes, the least significant byte being the green plane, line 0, if the top of the screen is being referenced. Element 4 holds red and green data, the most significant byte contains the green plane information, line 1. Explaining this is tough stuff, so here's a diagram generated by the program from the manual, with my addition, the array dump.

```
  BLUE        RED        GREEN        ARRAY CONTENTS

11111111    11111111    00000000     P00%(0)=24          3 COLORS, 8 DOTS
11111111    11111111    00100010     P00%(1)=9           9 ROWS  HIGH
11111111    11111111    00100010     P00%(2)=-1          11111111  11111111
11111111    11111111    00100010     P00%(3)=-256        11111111  00000000
11111111    11111111    00111110     P00%(4)=8959        00100010  11111111
11111111    11111111    00100010     P00%(5)=-1          11111111  11111111
11111111    11111111    00100010     P00%(6)=-222        11111111  00100010
11111111    11111111    00100010     P00%(7)=8959        00100010  11111111
11111111    11111111    00000000     P00%(8)=-1          11111111  11111111
                                     P00%(9)=-194        11111111  00111110
                                     P00%(10)=8959       00100010  11111111
character = 'H'                      P00%(11)=-1         11111111  11111111
                                     P00%(12)=-222       11111111  00100010
                                     P00%(13)=8959       00100010  11111111
                                     P00%(14)=-1         11111111  11111111
                                     P00%(15)=0          00000000  00000000
                                                         MS BYTE    LS BYTE
```

The pattern is easy to see. Since I am using a monochrome version of the H/Z-100, only green plane information shows up. In a color version there could be information in any of the planes, depending upon which colors are active. It is apparent that in this case I needed the least significant bytes of elements 3,6,9,12,15, ..., and the most significant bytes for elements 4,7,10,13, ..., of the entire 340 element array. If we AND element 3's contents, -256, with 127 we get 0. Element 6, -222, AND 127 = 34, 00100010 binary. (Element 4, 8959 AND 32512)/256 = 34, 00100010 binary. See this program's listing, lines 470 and 480.

```
             element 3                                  element 4
   decimal          binary                       decimal          binary
      -256 11111111 00000000     |                  8959 00100010 11111111
AND    127 00000000 11111111     |           AND   32512 11111111 00000000
--------------------------       |           ------------------------------
result= 0 00000000 00000000      |     partial, 8704 00100010 00000000
                                 |
...........element 6........     |
      -222 11111111 00100010     |     divide 8704 by 256,  1 00000000
AND    127 00000000 11111111     |     same as shifting right 8 places
--------------------------       |           ------------------------------
result=34 00000000 00100010      |    end result= 34 00000000 00100010
```

Take the least significant byte of these end results and send it to the printer.

Two's complement works like this: find the binary representation of a given number, for this program the number is 16 bits wide. If the most significant bit, the MSB, is 1 then the number is negative. If the MSB = 0 the number is positive. Invert this binary number and add 1 to it. For example:

```
      -222 = 1111111100100010   binary
  inverted = 0000000011011101
     add 1   0000000000000001
             ----------------
   result =  0000000011011110 = 222 and the sign was negative,
                                    from the MSB.
```

This method also works in reverse.

There are a series of special control codes which the Okidata 92 uses to control its operation. To enter the GRAPHICS mode a CHR$(3) is used. The sequence CHR$(3);CHR$(14) is a graphics line feed with carriage return. CHR$(3);CHR$(2) exits the printer from GRAPHICS mode. The Okidata 92 accepts bit patterns from 0 to 127, printing the pattern vertically, LSB at the top. The Okidata recognizes CHR$(3) as a control code so if there is a "3" dot pattern, 0000011, two of these patterns must be output for the printer to accept a "3" as a dot pattern. If only one "3" is sent, it won't be printed and the next number received may cause strange things to happen if it is interpreted as part of a valid control code sequence. See program lines 530 and 720. Eight bit binary numbers may be sent to the printer but it will not sense the most significant bit. For example, the printer will not distinguish between a "9" and a "137" in GRAPHICS mode. The only difference between the numbers is the most significant bit, bit 8, corresponding to 128 decimal or 10000000 binary. This is convenient because it appears that when the Z-BASIC interpreter sees a CHR$(9) being sent to a line printer, it inserts a horizontal tab. And while the Okidata is in the graphics mode the dot pattern corresponding to a 32 is printed a number of times, the number of repetitions of this "32" bit pattern depends on the number immediately following the "9". Sound confusing? It is. The the printed result appeared to be smearing randomly when a "9" was sent to the printer and it took some work to iron this bug out. There is a fix, however. Since the Okidata does not sense the most significant bit of the pattern being sent to it in GRAPHICS mode, this bit can be set HI by adding 128 to every number sent out. The H/Z-100 doesn't seem to mind. See program lines 470 and 480. There is probably a less time consuming solution to this problem than adding 128 to every single character, but it has eluded me.

The final column to be printed presented a new set of problems. Since this last column is only 3 dots wide, an entirely different set of numbers is required to manipulate the array. See program lines 610-750. The final array consists of the information from dot column 7 to 0 and lines 0 to 225. Instead of ANDing element 6 with 127, AND it with 224 and divide the result by 32. For element 3, AND the contents with E000 HEX and divide the result by 8192. Take a look at the diagram below, remembering that only the leftmost three dots of the green plane are of interest.

```
  BLUE        RED        GREEN        ARRAY CONTENTS

11111111    11111111    00000000     P00%(0)=24          3 COLORS, 8 DOTS
11111111    11111111    00100010     P00%(1)=9           9 ROWS  HIGH
11111111    11111111    00100010     P00%(2)=-1          11111111  11111111
11111111    11111111    00100010     P00%(3)=-256        11111111  00000000
11111111    11111111    00111110     P00%(4)=8959        00100010  11111111
11111111    11111111    00100010     P00%(5)=-1          11111111  11111111
11111111    11111111    00100010     P00%(6)=-222        11111111  00100010
11111111    11111111    00100010     P00%(7)=8959        00100010  11111111
11111111    11111111    00000000     P00%(8)=-1          11111111  11111111
                                     P00%(9)=-194        11111111  00111110
                                     P00%(10)=8959       00100010  11111111
character = 'H'                      P00%(11)=-1         11111111  11111111
                                     P00%(12)=-222       11111111  00100010
                                     P00%(13)=8959       00100010  11111111
                                     P00%(14)=-1         11111111  11111111
                                     P00%(15)=0          00000000  00000000
                                                         MS BYTE    LS BYTE
```

```
                        Bit manipulations

          Element 6                                        Element 4
  Decimal        Binary                          Decimal         Binary
     -222   11111111 00100010   |                    8959   00100010 11111111
AND   224   00000000 11100000   |           AND  E000 HEX   11100000 00000000
------------------------------- |           ---------------------------------
result=32   00000000 00100000   |           result = 8192   00100000 00000000
                                |
divide 32 by 32.     00100000   |     divide 8192 by 8192. 00100000 00000000
same as right shift 5 places    |     same as shifting right 13 places
------------------------------- |           ---------------------------------
result = 1  00000000 00000001   |         end result = 1    00000000 00000001
```

In elements 3, 6, 9, etc., I am interested in the three most significant bits of the least significant byte. In elements 4, 7, 10, etc., I am interested in the three most significant bits of the most significant byte. In sending these bit patterns to the printer it is not necessary to worry about the value "9" being present because this value cannot be attained with this routine, therefore it is not necessary to add 128 to every number. It is possible for the results of these manipulations to be negative. If this happens I simply add 8 to that number and continue. The pattern is correct. I leave it to the reader to figure out why it works. This algorithm works for the last column, 3 dots wide, only.

The Program

Line 50 commands that all numerical variables will be of type integer.

Line 70 requests that the DRIVE:FILNAME.EXT of the graphics screen to be printed be entered.

Lines 120-170 set up the screen variables. COLARRAY holds the data taken from the screen with the GET command on lines 410 and 630. LEFTSIDE is set to 0, the value of the leftmost addressable point on the screen. RITESIDE = 639, the rightmost addressable point on the screen. TOPEDGE = 0, the uppermost addressable point on the screen. BOTMEDGE = 224, the lowest addressable point on the screen. COLSTEP = -7, the dimension in dots and the direction that the program will step across the screen. COLWIDTH = 7, the number of dots the line printer will print while in the GRAPHICS mode. LEFTCOLM = (RITESIDE MOD 7). This line determines the addressable dot column at the left side of the screen after taking all those seven dot steps. LEFTCOLM actually equals 2. The command works like this: LEFTCOLM = RITESIDE - INT(RITESIDE / 7) : LEFTCOLM = 639- INT(639 / 7).

Lines 210-250 set up the printer commands and special graphics strings. GRAFCHAR is the integer array which holds the manipulated results of the GET command for output to the printer. GRAFMODE$ is the control code, CHR$(3), sent to the printer in line 310 which puts the Okidata in the GRAPHICS mode. NORMODE$ is the control code, CHR$(3);CHR$(2), used to put the printer back in the draft print mode. A new page command, CHR$(12), is included with this sequence. GRAFCRLF$ is the control code sequence which returns the printhead to the left side and feeds one graphics line feed (a graphics line feed so there will be no gaps from one graphics line to the next). LMARGIN$ is a string of 15 dot spaces. The printer will print 480 dots per 8 1/2 inch line at 10 chr./inch. For a presentable printed output, every dot pattern is printed twice (remember the aspect ratio). There are 225 bit patterns per line, 2 printings of each pattern, with 30 dots left over on the line. 30/2 = 15; the number of spaces in LMARGIN$. Printing 15 dot spaces at the beginning of each line will center the image.

Lines 290-320 initialize the printer. Line 290 OPENs the printer as FILE #1 for output from the program. The command on line 300 allows the printer to print an infinite number of characters on a line. Normally Z-BASIC would insert a carriage return/line feed after 80 characters. This program prints 465 graphics characters per line, or, as far as Z-BASIC is concerned, some 43000 characters are printed on one line. Think about the codes being sent out versus the ASCII character set and note that the majority of the "characters" being printed are greater than 128 in numerical value. Line 310 places the printer in GRAPHICS mode. Line 320 sends a graphics line feed to the printer, mainly to let the programmer know that the program is running properly.

Line 340 loads the screen of your choice into the GREEN plane of video memory.

Lines 400-590 GET the information from the screen, 8 dots at a time, perform bit masking and bit slicing, and output the formatted results to the line printer. The loop counter COL is initialized to the value of RITESIDE, 639, and each time through the loop it is STEPped COLSTEP, or -7. It is used within the loop to define the rectangle or slice being scanned, along with TOPEDGE, BOTMEDGE and (COL-COLWIDTH). Lines 460-500 are interesting. The variable ELMNT, which is used to step through the array GRAFCHAR(ELMNT), is incremented by 2 while the array counter RAYCOUNT is incremented by 3. These increments bypass those elements of COLARRAY which are not wanted (elements 2,5,8,11, etc.) and at the same time place desirable values in contiguous elements of GRAFCHAR. The bit manipulation occurs at lines 470 and 480. At line 350 tests are performed to insure that if a "3+128" is encountered it is sent to the printer four times. Remember, a "3" is a printer control code unless two characters are sent in succession. To print two 3's, four 131's are sent out (131-128=3). The remaining elements of GRAFCHAR are printed twice for symmetry. After element #224 of the array is printed a graphics line feed is sent out and a new line is started. Ninety- one graphics lines are printed in this manner.

Lines 630-750 print the last column in a manner similar to that employed for the previous 91 columns, however the bit manipulations are different.

Line 770 returns the printer to the normal print mode, a page feed is sent and a beep is sounded to annoy the programmer.

Running the Program

Run your graphics routine and when the display is satisfactory, SAVE the screen on disk with the following Z-BASIC sequence placed in a suitable location in your program: DEF SEG=&HE000: BSAVE "DRIVE:FILENAME.EXT",0,&HD000. This will save the green plane in a disk file of some 50K bytes. Load this OKIDUMP program, insuring that your OKIDATA 92 is turned on. RUN. The program will ask you for the screen name to dump to the printer. Enter your file's name and hit return. Now go make a peanut butter sandwich because the program takes about 9 minutes to dump the entire screen to the printer. This is a substantial improvement over the 2-hour program and not bad for an interpreted BASIC screen dump program.

Here is a simple program which should be a reasonable test of this screen dump program. Run it with a scratch floppy in Drive B (the video file requires about 53K bytes) and then run the OKIDUMP program in Drive A. At the prompt enter B:CIRCLSCR <return> and finish that peanut butter sandwich.

```
10 CLS
20 XX=320:YY=112
30 FOR X=10 TO 250 STEP 10
40 CIRCLE (XX,YY),X
50 NEXT X
60 LINE (0,0)-(639,224),,B
70 DEF SEG=&HE000
80 BSAVE "B:CIRCLSCR",0,&HD000
```

It is possible to MERGE this program with your Z-BASIC graphics program. Delete lines 70 and 340, place a RETURN at line 780 in place of the END, RENUM this routine to fit your program, SAVE it in ASCII format, and place it at the end of your program with the MERGE command. Then use a GOSUB at the appropriate point in your program to call this routine.

Possible Problems

Insure that the printer is turned on, the SEL lamp is lit and the print head is at the left side of the carriage before running this program. If you should interrupt printing for any reason prior to the end of the program, make certain that you turn the printer off and then on again to clear the Okidata's internal print buffer of any characters not yet printed. If you should encounter an ILLEGAL FUNCTION CALL AT LINE 410, then try increasing the dimension of COLARRAY like this: 120 DIM COLARRAY (500), or a similar larger figure.

I have made no provision for vertical centering on the page. A few normal line feeds prior to entering the graphics mode should suffice. I will leave this to the individual programmer. The necessary codes are in Okidata's manual, along with page length, dot densities, etc.

Listing 1

```
10 ' *************************************************
20 ' ****   SCREEN DUMP FOR OKIDATA 92 PRINTERS    ****
30 ' ****      AND THE HEATH/ZENITH H/Z-100        ****
40 ' ****              BY TIM ROSS                 ****
50 ' *************************************************
60 DEFINT A-Z 'SET ARRAYS AND VARIABLES TO TYPE INTEGER
70 INPUT "ENTER SCREEN NAME TO DUMP TO THE PRINTER : ";SCRNAME$
80 'PLACE THE FILENAME OF THE DESIRED GRAPHICS SCREEN INTO SCRNAME$
90 '--------------------
100 'SCREEN VARIABLES
110 '--------------------
120 DIM COLARRAY(340) 'HOLDING ARRAY FOR DATA TAKEN FROM SCREEN
130 LEFTSIDE=0:RITESIDE=639 'SCREEN EDGES, LEFT AND RIGHT SIDES
140 TOPEDGE=0:BOTMEDGE= 224 'SCREEN ROW NUMBERS
150 COLSTEP=-7 'STEP THROUGH THE SCREEN 7 DOTS A SLICE FROM RIGHT TO LEFT
160 COLWIDTH=7 'COLWIDTH+1 DOTS ARE TAKEN OFF THE SCREEN IN ONE PASS
170 LEFTCOLM=RITESIDE MOD 7 'SCREEN COLUMN NO. OF LAST PRINTER LINE
180 '-----------------------
190 'PRINTER VARIABLES
200 '-----------------------
210 DIM GRAFCHAR(225)' THE ARRAY HOLDING ONE LINE OF GRAPHICS CHARACTERS
220 GRAFMODE$=CHR$(3) 'CODE USED TO SWITCH OKIDATA TO GRAPHICS MODE
230 NORMODE$=CHR$(3)+CHR$(2)+CHR$(12) 'SWITCH BACK AND ADVANCE TO NEXT PAGE
240 GRAFCRLF$=CHR$(3)+CHR$(14) 'GRAPHICS CARRIAGE RETURN/LINEFEED
250 LMARGIN$=STRING$(15,0) 'LEFT PAPER MARGIN TO CENTER IMAGE
260 '--------------------
270 'INITIALIZING THE PRINTER
280 '--------------------
290 OPEN "LPT1:" FOR OUTPUT AS #1 'LINE PRINTER = #1
300 WIDTH #1,255 'KEEP BASIC FROM ADDING CARRIAGE RETURNS EVERY 80 CHARACTERS
310 PRINT #1, GRAFMODE$; 'SWITCH GRAPHICS MODE ON
320 PRINT #1, GRAFCRLF$; 'SEND A GRAPHICS CARRIAGE RETURN, LINE FEED
330 '--------------------
340 DEF SEG=&HE000:BLOAD  SCRNAME$ 'LOAD THE SPECIFIED IMAGE INTO GREEN PLANE
350 '--------------------
360 'STEP ACROSS THE SCREEN, RIGHT TO LEFT, 7 DOTS AT A TIME.
370 'GET A COLUMN, PERFORM BIT SLICING AND MASKING, AND
380 'SEND THE RESULT TO THE OKIDATA 92
390 '--------------------
400 FOR COL=RITESIDE TO LEFTCOLM+1 STEP  COLSTEP
410    GET (COL, TOPEDGE)-((COL-COLWIDTH), BOTMEDGE),COLARRAY
420    ELMNT=0:PRINT #1,LMARGIN$;
430 '   -------------------------
440 '   BIT SLICING AND MASKING, FORMATTING FOR OKIDATA
450 '   -------------------------
460    FOR RAYCOUNT=3 TO 339 STEP 3
470       GRAFCHAR(ELMNT)=(COLARRAY(RAYCOUNT) AND (127))+128
480       GRAFCHAR(ELMNT+1)=((COLARRAY(RAYCOUNT+1) AND 32512 )/256)+128
490       ELMNT=ELMNT+2
500    NEXT RAYCOUNT
510    FOR ELMNT=0 TO 224 'PRINTING ROUTINE
520      IF GRAFCHAR(ELMNT)=131 THEN PRINT #1,STRING$(4,3);:GOTO 570
530 '     TEST FOR CHARACTER 3, AN OKIDATA COMMAND CHARACTER
540 '     IF A 3 IS PRESENT SEND 2*2 OF THEM TO THE PRINTER, REQUIRED BY OKI
550      PRINT #1,STRING$(2,GRAFCHAR(ELMNT));
560 '     SEND TWO OF EACH CHARACTER TO OKI FOR SYMMETRY
570    NEXT ELMNT
580    PRINT #1,GRAFCRLF$; 'SEND A GRAPHICS LINE FEED TO THE PRINTER
590 NEXT COL
600 ' ---------------------------
610 'LAST COLUMN, THIS ROUTINE IS DIFFERENT FROM PREVIOUS COLUMNS
620 ' ---------------------------
630 GET (7,TOPEDGE)-(LEFTSIDE,BOTMEDGE),COLARRAY 'LAST LINE, WHICH IS NARROWER
640 ELMNT=0:PRINT #1,LMARGIN$; 'CENTER THE IMAGE
```

```
650 FOR RAYCOUNT=3 TO 339 STEP 3
660     GRAFCHAR(ELMNT)=(COLARRAY(RAYCOUNT) AND (224))/32
670     GRAFCHAR(ELMNT+1)=(COLARRAY(RAYCOUNT+1) AND &HE000)/8192
680     ELMNT=ELMNT+2
690 NEXT RAYCOUNT
700 FOR ELMNT=0 TO 224
710     IF GRAFCHAR(ELMNT)<0 THEN GRAFCHAR(ELMNT)=GRAFCHAR(ELMNT)+8:GOTO 730
720     IF GRAFCHAR(ELMNT)=3 THEN PRINT #1,STRING$(4,3);:GOTO 740
730     PRINT #1,STRING$(2,GRAFCHAR(ELMNT));
740 NEXT ELMNT
750 PRINT #1,GRAFCRLF$; 'GRAPHICS CARRIAGE RETURN
760 ' --------------------------------
770 PRINT #1, NORMODE$ 'RETURN THE PRINTER TO NORMAL MODE, PAGE FEED
780 BEEP:END
```

Listing 2

```
10 ' CHRIMAGR, FROM Z-BASIC MANUAL, PAGE 8.27, 8.28
20 CLEAR 100
30 INPUT "CHARACTER :",C$:PRINT C$
35 GOSUB 480
40 'INPUT"COLOR NUMBER <7>:",R
50 IF R<1 OR R>6 THEN R=7
60 'INPUT "POSITIVE OR NEGATIVE <P>:",I$
61 IF I$="" OR I$="P" OR I$="p" THEN I=1 ELSE I=0
65 'INPUT "MASK 0,1,NONE <NONE>",M$
66 IF M$<>"0" AND M$<>"1" THEN M$=""
80 CLS
90 COLOR R
100 DIM P00%(19),P01%(19),P02$(5)
110 PRINT C$
115 COLOR 7
120 GET(7,8)-(0,0),P00%
121 FOR S=0 TO 19:LPRINT S;P00%(S);:NEXT S:LPRINT
130 IF I=0 THEN PUT(0,0),P00%,PRESET:GET(0,0)-(7,8),P00%
140 PRINT
150 PRINT "PIXELS","SCAN"
160 PRINT "ACROSS","LINES"
170 PRINT P00%(0)/3,P00%(1)
180 PRINT
190 PRINT "BLUE","RED","GREEN"
200 PRINT
210 FOR Y=2 TO 16 STEP 3
220 X=Y
230 GOSUB 300
240 GOSUB 380
250 GOSUB 440
260 NEXT Y
270 END
280 'SUBROUTINES
290 '
300 XDEC=P00%(X):GOSUB 570:
        P02$(0)=RIGHT$(STRING$(16,48)+BIN$,8)
310 P02$(1)=LEFT$(RIGHT$(STRING$(16,48)+BIN$,16),8)
320 XDEC=P00%(X+1):GOSUB 570:
        P02$(2)=RIGHT$(STRING$(16,48)+BIN$,8)
330 P02$(3)=LEFT$(RIGHT$(STRING$(16,48)+BIN$,16),8)
340 XDEC=P00%(X+2):GOSUB 570:
        P02$(4)=RIGHT$(STRING$(16,48)+BIN$,8)
350 P02$(5)=LEFT$(RIGHT$(STRING$(16,48)+BIN$,16),8)
360 RETURN
370 '
380 IF M$="" THEN 420
390 K=INSTR(P02$(J),M$)
400 IF K<>0 THEN MID$(P02$(J),K,1)="":GOTO 390
410 NEXT J
420 RETURN
430 '
440 PRINT P02$(0),P02$(1),P02$(2)
445 LPRINT P02$(0),P02$(1),P02$(2)
450 IF X+2=16 THEN 470
460 PRINT P02$(3),P02$(4),P02$(5)
465 LPRINT P02$(3),P02$(4),P02$(5)
470 RETURN
480  B$(0)="000"
490  B$(1)="001"
500  B$(2)="010"
510  B$(3)="011"
520  B$(4)="100"
530  B$(5)="101"
540  B$(6)="110"
550  B$(7)="111"
560 RETURN
570 DPC$=OCT$(XDEC):BIN$="":FOR YCOUNT = 1 TO LEN(DPC$)
580 Q99=VAL(MID$(DPC$,YCOUNT,1)):BIN$=BIN$+B$(Q99):NEXT YCOUNT
590 RETURN
```

Listing 3

```
05 'CIRCLBOX.BAS
10 CLS
20 XX=320:YY=112
30 FOR X=10 TO 250 STEP 10
40 CIRCLE(XX,YY),X
50 NEXT X
60 LINE (0,0)-(639,224),,B
70 DEF SEG=&HE000
80 BSAVE "B:CIRCLSCR",0,&HD000
```

LEARN TO PROGRAM H89/H19 GRAPHICS IN MBASIC

BASIC GRAPHICS PRIMER is a computer based course that will teach you fundamental MBASIC programming techniques to control the H19 terminal or H89 computer graphics. Seven lessons will teach you many forms of graphic plots, data plotting, multiple sequential and random access files used for graphics, basic figure animation techniques, and simple computer graphics game design. Each lesson consists of simple illustrative program segments with complete explanations of the programming techniques involved, questions to reinforce the lesson material, and one or more example programs. Most H19/H89 terminal commands are covered in the lessons, including use of the special function keys. **BASIC GRAPHICS PRIMER** is a "HOW TO" course with plenty of examples. If you want to learn how to take full advantage of the H19/H89 graphics features in your programs, **BASIC GRAPHICS PRIMER** is a must. For CP/M only. Requires a printer.

ONLY $49.95!

CP/M version for H89, H8, H19 requires CP/M and MBASIC

NewLine Software

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

NAME ______________________ Send me ____ BASIC
STREET ______________________ GRAPHICS PRIMER
CITY ______________________ program(s) for CP/M
STATE ______________ ZIP ______ at $49.95 each.

Check one: ☐ payment enclosed ☐ send COD (add $3.00)
Send order to:

NEWLINE SOFTWARE, P.O. BOX 402, LITTLETON, MA 01460
Foreign orders: add $3.00 Airmail, $10.00 for non-U.S. checks

CP/M is a trademark of Digital Research, Inc.
MBASIC is a trademark of Microsoft, Inc.

Vectored from 12

DEFMS.ASM -- Contains the operating system definitions
DEFASCII.ASM -- Contains the ASCII definitions
DEFMTR.ASM -- Contains the monitor ROM routine definitions

Naturally, there are other files you may need for more advanced programming, but these will suffice to begin with. How do you tell what other files you may need? Only by inspection. I printed out all the files on the distribution disk, just to have a reference to what is contained. Usually, when a program is written, it will tell you which files you need to INCLUDE, and you can transfer those to our working disk. It's generally better to have all the INCLUDE files on the same disk as MASM, which means for those of you blessed with two drives, you may want to have the editor and the source file on the second disk. You can also put EXE2BIN on that disk. If you put the library files on the second disk, it's possible to put LINK on that disk, too. Then when you assemble a file, send the output (the .OBJ file) to the second disk for subsequent LINKing and conversion to a .COM file.

Fortunately, with ZDOS and 320K on a disk, you can usually do useful assembly language work with everything on one disk. Only when you get into bigger projects (like reassembling an altered BIOS) do you need to study the MASM and LINK sections of the ZDOS manual to learn the more advanced tricks.

With two drives available, I prefer to have a SYSGENed disk with my editor on drive A: and an unSYSGENed disk with all the assembler-related files on drive B: where I can get at them easily by making B: the default drive. This is done, of course, by typing B: in response to the A: prompt.

The moral of this part of the story is that you should feel free to experiment, and work out your own best distribution of the files you need to work with. You might also pass along what you find to others, since no matter how much assembly language experience one has, there is always something new to learn.

D. C. Shoemaker
HQ US European Command
Box 897
APO NY 09128

From the Author of BASMAPER

Dear Walt,

Concerning the 128K RAM problem with my BASMAPER program in the article which appeared in your February 1984 issue. I have since discovered it may be that the problem is really in the version of ZBASIC being used. If Version 1.0 created on 15-Jul-82 is used, the base load offset is in byte 325,326 instead of 343,344. The changes are in statement numbers 65501 and 65523.

Change

```
I=FNXX(343)
```

to

```
I=FNXX(325)
```

Again, I am sorry for the inconvenience I might have caused, I'll use a different technique next time.

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

Double the Speed of your H/Z89-90

Cut computing time by as much as 50%

Fast Operation

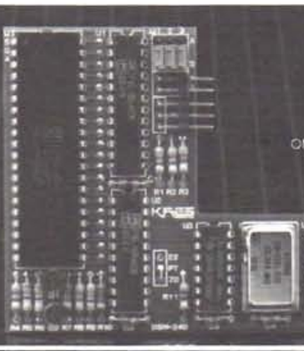
With the Dual Speed Module, software that used to be slow and inefficient now zips right along. Run programs like SuperCalc (in "auto mode") and Spellstar without long waits.

Easy

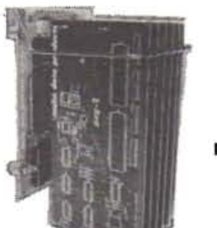
Simply replace your existing microprocessor IC with the Dual Speed Module and stop wasting time. Installation requires no trace cutting or soldering.

Fully Supported

CP/M is a registered trademark of Digital Research



KRES
ENGINEERING
P.O. Box 17328, Irvine, CA 92713
(714) 559-1047
or
(213) 957-6322
Bulletin Board (714) 559-8579



H89/Z90's CAN NOW DEAL WITH A FULL DECK!

THE ORIGINAL ALL-IN-ONE.
ACCESSORY BUS EXPANDER.

MH89+3 doubles expansion capacity. Allow for 6 right-hand type cards instead of the usual 3. Room at last to run those neat accessory boards you've seen advertised!

Piggyback motherboard installs internally with a screwdriver in just minutes – with no modifications! 3 slots exactly duplicate the originals. The 3 added slots occupy unused addresses and eliminate previous conflicts. 100% compatible with all accessory boards!

No overheating problems! Simple design draws little power. Leaves plenty of overhead for the minimal load of most accessories. Full technical information provided.

The best news about this "No-hassle" design is the price – **ONLY $150.** About 1/3 the price of other solutions!

Price includes assembled and tested MH89+3 expander, complete instructions and one (1) year warranty. CA residents add 6% tax. USA include $5 shipping. Foreign add $10. Telephone and COD orders accepted.

mako data products
1441-B N. RED GUM, ANAHEIM, CA 92806
PHONE (714) 632-8583

ZVM-134 and 135
VIDEO MONITORS

DUST COVERS

Now Available from: **KCK ENTERPRISE**
3328 Northwest C Street
Richmond, Indiana 47374

The covers are made of water repellent fabric in basic colors of: blue — brown — green — trimmed in a contrasting color, and are double stitched for DURABILITY and machine washable. **$15.50**
Other sizes and styles available upon request.

Fill out Order Form or call 317/966-4052

Name: ____________________

Street: ____________________

City: __________ State: ________ Zip: ________

CHECK ONE: ☐ 2 VM-134 ☐ 2 VM-135
☐ Blue ☐ Brown ☐ Green

Send me ______ at $15.50 each
Add $2.00 for postage and insurance.

Indiana residents add 5% sales tax.

H/Z-100 AND 150 PC GRAPHICS SOFTWARE

MICROSERVICES continues to expand its support of Heath/Zenith computers.

For H/Z-100s:

ZANIMATE. Make color animation sequences using ZBASIC **$64.95**

ZPALETTE. Draw images in 92 colors using ZBASIC. **Improved** with three (3) painting modes and graphics generator **$59.75**

ZPATTERN. Test your color monitor **$24.95**

Z3D from **COLORWORKS.** Make three-dimensional objects in color **$75.00**

For H/Z-100s and 150 PCs:

SOFTKITS from **KERN INTERNATIONAL.** We are now carrying a full line of this excellent series of books and disks for graphics and business/scientific applications.

We are continuing to introduce new products. Write for our catalog.

VISA

MICROSERVICES
P.O. Box 7093
Menlo Park, CA 94026
Phone: (415) 851-3414

master charge

Include 3% p/h ($3.00 min.). California add 6½% tax.

Here's More On The Z-100 Key Click

Dear HUG,

I found Mr. Richard Hole's letter in the March 84 REMark, "How To Turn Off the Z-100 Key Click", to be very informative. However, I did probe a bit further and expanded on Mr. Hole's idea. I found that you do not need to type in REM in the AUTOEXEC file. Using EDLIN, I created an AUTOEXEC file as follows (note that typing the F8 function key generates a ↑[which is the symbol for ESCape):

```
A:EDLIN AUTOEXEC.BAT <return>
EDLIN Version 1.02
New File
*I <return>
      1*<F8>x2<return>
      2*<CTRL C>
*E <return>
```

This will turn off the key click like Mr. Hole's method, but without using any REM statements.

I expanded Mr. Hole's idea even further. I wanted to control several Z-100 keyboard and video display features any time I was in ZDOS, not only on a cold boot. I created a "key click off file" as follows:

```
A:EDLIN NOBEEP. <return>
EDLIN Version 1.02
New File
*I <return>
      1*<F8>x2<return>
      2*<CTRL C>
*E <return>
```

(Note the period after NOBEEP)

Now if I want to turn off my key click any time in ZDOS, I just type in **TYPE NOBEEP** <return> (you do not need a period after NOBEEP here) and no more key click!

Using EDLIN, I have created several "one-liner" programs which use different ESCape codes to turn on and off different keyboard and video display features. The following is a listing of program titles, descriptions of the features, and the necessary codes to implement them:

Title	Description	Codes
BEEP	Enables key click	<F8>**y2**
BLOCK	Block cursor	<F8>**x4**
NOBLOCK	Underscores cursor	<F8>**y4**
BLINK	Blinking cursor	<F8>**y;**
NOBLINK	Non-blinking cursor	<F8>**x;**
GRAPH	Enter graphics mode	<F8>**F**
NOGRAPH	Exit graphics mode	<F8>**G**
CURSOR	Turn on cursor	<F8>**y5**
NOCURSOR	Turn off cursor	<F8>**x5**
REVERSE	Enter reverse video	<F8>**p**
NOREVERS	Exit reverse video	<F8>**q**
REPEAT	Enable autorepeat	<F8>**y<**
NOREPEAT	Disable autorepeat	<F8>**x<**
NORMAL	Returns everything to normal without resetting the Z-100	<F8>**y2**<F8>**y;** <F8>**G**<F8>**y4** <F8>**y5**<F8>**q** <F8>**y<**

My method allows me to selectively turn on and off certain features and to enable features from a cold boot or anytime while I am in ZDOS. The words are easy to remember, and all those "one-liners" take up only a few bytes (not Kbytes!) on my ZDOS utilities disk. Thank you again, Mr. Hole, for your enlightening letter.

Richard J. Komar
1408H Paegelow
Scott AFB, IL 62225

Correction to "Standards For Terminal Control" (Issue 46)

Dear HUG,

I enjoyed David Warnick's article on Standards For Terminal Control (Issue 46) and especially his sample program on Biorhythms. I think I have found a couple of minor errors though.

Line 3130 should read:

```
3130 D4=D4+0
```

The other error only occurs when the reporting month happens to be February in a leap year (as in 1984). One way to correct the problem is as follows:

```
5130 IF (INT(Y1/4))*4=Y1 THEN GOTO 5135 ELSE 5140
5135 IF D1=30 THEN D1=1:M1=M1+1:
5136 GOTO 5170
```

Doug Bailey
474 Oak Avenue
Aurora, IL 60506

An Amusing Short Program

Dear HUG,

Here's a short program for your amusement. It runs on an H/Z-89 (or H/Z-19) with MBASIC (CP/M or HDOS). Be sure to accurately type the string assignment in line 20, including lower case letters and spaces.

```
10 PRINT CHR$(27)+"F"
20 X$="wi`b:↑   "
30 Y$=SPACE$(79)
40 Y$=MID$(Y$,2,39)+MID$(X$,8*RND(1)+1,1)+MID$(Y$,40,39)
50 PRINT Y$
60 GOTO 40
```

Now, do the patterns appear to move upward, or do they start at the center and move outward?

Bruce C. Trump
7925 N. Soledad Ave.
Tucson, AZ 85741

Beware of Chain Letters

Dear Walt,

Frank and I received an interesting letter from a group in New Mexico this week. In three pages, it detailed a computer- program-listing-by-mail scheme that is a variation of the old chain letter formula. As we are sure that other REMark readers will be receiving similar letters, we wanted to pass on the ruling we received from our local Postal Inspector General:

ANY chain letter scheme in which the primary object is an appeal for money is illegal. It doesn't matter whether the letter asks for recipes, household hints, grocery coupons, or computer programs; if the scheme boils down to "Send $5.00 to the names on list for a XXXXX, remove the top name, add your own, and pass it on", it is illegal.

Even without the legal question, the first guy on the list has used his word processor and mailing list program to hit the immediate world. We think the guy in New Mexico got our name and address from a magazine article we published last year. At that rate, the prospects for receiving a return on your investment make Carlo Ponzi look like a blue-chip stock.

We hope this will save your readers from falling victim to this scheme.

Mary Lynn Hutchison
5327 Edgewater Drive
Ewa Beach, HI 96706

The Demise of the COLD HUG

Dear HUG,

Please place a notice in your magazine noting the demise of the COLD HUG. This is due to my reassignment to Fort Meade, Maryland and the reassignment within the next two months of the remaining members. As of this date, there are no other Heath Computer Users' in the immediate area to carry on the group.

The COLD HUG Bulletin Board will cease operation on May 10, 1984 for the same reason.

Please express my thanks to the suppliers and people that used the board for their support and help.

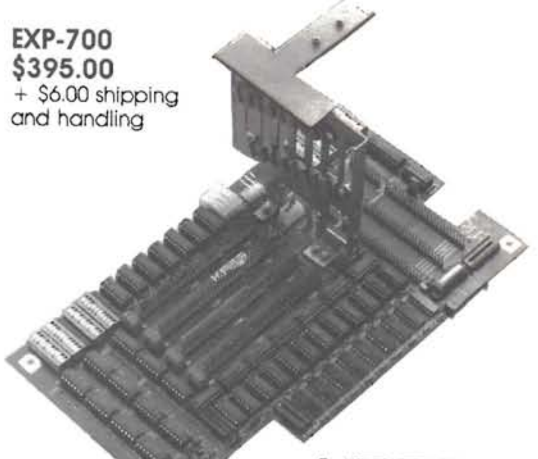
Stan Lockhart
P. O. Box 229
Fort Greely, AK
APO Seattle, WA 98733

Expand your H/Z89-90 from KRES

Make your computer a more useful friend

- Now you can add all of those extra boards you've always wanted (color graphics, speech, robotic control, parallel—to name a few)
- The Kres Expansion enables you to put seven full sized boards inside your H89 cabinet; no external cables or boxes.
- Plug in any boards without any software modification
- Simple to install and use

EXP-700
$395.00
+ $6.00 shipping and handling



KRES

P.O. Box 17328, Irvine, CA 92713
(818) 957-6322 or
(714) 559-1047

Condor™ made Easy! HELP!© Condor™

MENU DRIVEN CONDOR / ON-LINE HELP SCREENS AND CONDOR TUTORIAL

HELP! Condor was designed with the first time computer user in mind. SoftHelp has enhanced the Condor database to make it more "user friendly."

HELP! Condor is the easy way to **enjoy the power of the Condor relational database without learning commands or syntax.**

HELP! Condor is both a front-end generic MENU, as well as on-line help for every Condor command. The expert Condor user may alter the source code of HELP! Condor saving days of programming.

The manual for HELP! Condor includes:

- Merging Condor files and Word Processors
- Merging Condor and Lotus 123
- BASIC programs
- Command Files Explained

HELP!© Condor™
$149.00
HELP! Condor by SoftHelp, Inc.
3527 Oak Lawn Ave., Suite 179
Dallas, Texas 75219
(214) 559-3095
VISA / MASTERCARD

"It is one of the most useful programs of its type we have ever seen, and believe me, we see a **lot** of software."
Don Kenny
Product Evaluation Manager
Micro D
Fountain Valley, CA

"I think this product can make Condor easier to sell, because it makes the users life easier."
Robert Eigler
President
Micro-Age
Houston, TX

"First time Condor users can now be up and going in minutes instead of hours."
Bob Denton
Vice President, Marketing
Seequa Computers

Index of Advertisers

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

CUT ALONG THIS LINE

HUG MEMBERSHIP RENEWAL FORM

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.

Superior Support

D·G

Heath / Zenith
Users'
Group

Hilltop Road
Saint Joseph, Michigan 49085

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

Volume 5, Issue 6

POSTMASTER: If undeliverable, please do not return.

885-2053