



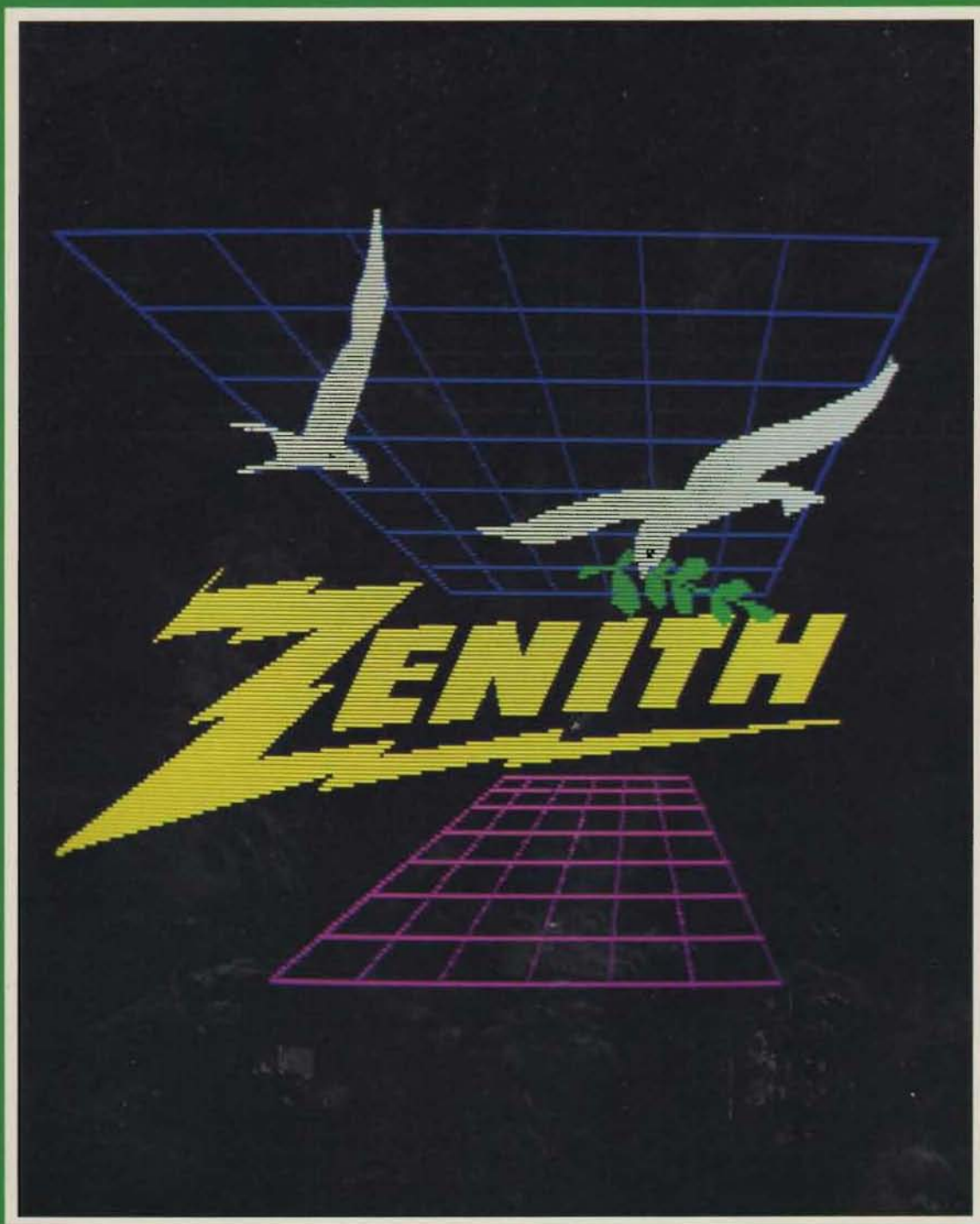
REMark®

\$2.50

Volume 6, Issue 4 • April 1985

P/N 885-2063 Issue 63

Official magazine for users of  computer equipment.



ACCESS™

The State of the Art in Communications

- Talk to bulletin boards, information utilities, main frames or other micros
- Send and receive any kind of file—text, binary, .EXE, .COM, etc.
- Transfer files using a variety of protocols, including XMODEM protocol
- Use ACCESS to turn your computer into a communications robot
- Unattended auto-answer mode configures to caller's baud rate, and
- Depending on caller's password, grants unlimited, limited, or no access
- Logs password, duration, date and time of calls you receive
- Logs calls you place—recording number dialed, duration, date & time
- Compatible with all modems, even USR S100 and other board modems

Note: Certain features not included in 8 bit versions.

Start Communicating Right Away!

You can start communicating as soon as you pull ACCESS out of its package. We don't make you learn a strange new language of names and symbols the way most programs do. Our customers proclaim it—ACCESS is the easiest and fastest to use!

Your Own Communications Robot

ACCESS can dial, log on, send or receive, answer prompts, even make decisions. Sit back and relax while ACCESS does your work for you.

While you eat dinner or watch TV, have ACCESS call the local bulletin boards, pick up messages to you, drop off messages to others, and collect the classified ads for you to go through later.

Have ACCESS reduce your telephone and timesharing bills by making calls for you late at night, when rates are lowest. When you get up in the morning, your computer can be chock full of fresh information, just like your morning paper.

Instant Links with Other Micros

ACCESS gives you instant phone links with any other 8-or 16-bit microcomputer for fast, efficient file transfers or just to chat. Or you can direct-cable your computer to others.

And ACCESS answers calls and lets people with one of your passwords have either limited or unrestricted use of your system . . . including freedom to run programs other than ACCESS!

Make Those Tough Connections

Again and again our customers have thanked us for the power to talk with main frames, minis, and specialized networks, with emulators, microcontrollers, and development boards. And you'll thank us too, when ACCESS gets you into systems that defy people with other programs.

Don't Let Your Computer Act Dumb!

Other programs only let your computer act like a dumb terminal while you're on line, but ACCESS leaves the full power of your computer in your hands. Not only can you view your directories and files, you can run your other software—without leaving ACCESS or hanging up the phone.

What Makes ACCESS So Powerful?

To make ACCESS so easy to use, yet fast and flexible, took state-of-the-art programming and years of evolution. To achieve compact power, ACCESS is written in C, the language used to write the famous operating system UNIX. And ACCESS is interrupt-driven for ultimate speed with reliability.

How You Can Get ACCESS

Buy ACCESS from your local Heath or ZDS dealer. If there isn't a dealer near you, order it direct from Hilgraeve Inc.

ACCESS, H-8 or H/Z-89 with CP/M (64K req'd) \$49.95

ACCESS, H/Z-110 or 120 with ZDOS, CP/M-85, or -86 \$59.95

ACCESS, H/Z-150 or 160 with MSDOS or CP/M-86 \$69.95

Add \$3.00 shipping; MI residents add 4%

Hilgraeve Inc. P.O. Box 941 Monroe, MI 48161 (313) 243-0576



SPECTACULAR SAVINGS ON H-89 COMPUTER SYSTEMS

Close out prices on all H-89 Computer Systems now provides Heath User's Group members with a unique opportunity to obtain an outstanding performer at an all-time low price by using their HUG discount.

Nine different system options offer you a choice of kit or assembled computers, anti-glare white or green video displays, hard- or soft-sectored data storage, and type and number of 5.25" floppy disk drives for varying amounts of total disk storage capability.

Along with big price cuts on computer systems, we are offering FREE—your choice of either H-DOS™ or CP/M* operating systems— with any kit or assembled H-89 Computer. And in addition to our 50% discount on any H-89 software, HUG members can add in their HUG discount.

Also, if you buy an H-89 Computer System you can purchase at the same time an H-125 High Speed Dot Matrix Line Printer kit at an all-time low price of \$375.00. (Sorry, at this low price HUG discounts are not applicable.) Be sure to take

advantage of this tremendous buy and get quality 150 characters per inch printing capability for your business or personal use.

Visit your nearest Heathkit/Zenith Computers & Electronics Center* or look through our new Spring Heathkit Catalog. Total up the price of an H-89 Computer System that you want and the software (at 50% off) you desire, then deduct your HUG discount. Then add a quality high-speed printer for just \$375.00. See how much you can save as a member of the Heath User's Group.

*CP/M is a registered trademark of Digital Research Inc. Heathkit/Zenith Computers & Electronics Centers are units of Veritechnology Electronics Corporation.



Heathkit®

Staff

Manager Bob Ellerton
(616) 982-3867

Software Engineer Pat Swayne
(616) 982-3463

Bulletin Board and
Software Developer Jim Buszkiewicz
(616) 982-3463

Software Coordinator Nancy Strunk
(616) 982-3838

Secretary Margaret Bacon
(616) 982-3463

REMark

Editor Walt Gillespie
(616) 982-3789

Editorial and Advertising
Assistant Lori Latham
(616) 982-3794

Printers Imperial Printing
St. Joseph, MI

	U.S. Domestic	APO/FPO & All Others
Initial	\$20.00	\$35.00*
Renewal	\$17.00	\$30.00*

* U.S. Funds

Limited back issues are available at \$2.50, plus 10% shipping and handling — minimum \$1.00 charge. Check HUG Product List for availability of bound volumes of past issues. Requests for magazines mailed to foreign countries should specify mailing method and appropriate added cost.

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

Although it is a policy to check material placed in REMark for accuracy, HUG offers no warranty, either expressed or implied, and is not responsible for any losses due to the use of any material in this magazine.

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

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

REMark is a registered trademark of the Heath/Zenith Users' Group, St. Joseph, Michigan.

Copyright © 1984, Heath/Zenith Users' Group

Buggin' HUG

..... 7

Sequential Files — Part 2

David E. Warnick 10

From Heath To Endburg

Crawford MacKeand 13

Programming Video Modes

Mark Foster 18

Speed Mods For The Z-100

Pat Swayne 20

The FORTRAN Formula-3

Dick Stanley 22

Expanding The Heath/Zenith PC With The AST Research SixPak

Dale Grundon 28

Local HUG Clubs

..... 32

HUG Price List

..... 33

HUG New Products 34

Need More Speed? Try A Compiler
C. Nicholas Pryor 39

TURBO-Charging Your Z
Frank Dreano, Jr. 43

Share Your H-89 Peripherals With Brand "X" Computers
Peter Ruber 47

Two Useful "C" Functions
Charles R. Winchester 50

Patch Page
Pat Swayne 54

Shorten: A Program To Save Memory In BASIC Programs
Patrick Brans 55

COBOL Corner XII
H.W. Bauman 58

Heath/Zenith Related Products
Tom Huber 64

Index of Advertisers

This index is provided as an additional service. The publisher does not assume any liability for errors or omissions.

Abe Dweck Programs 26
Advanced Software Technologies 17
Analytical Products 57
CDR Systems, Inc. 27,53
Computer Graphics Center, Inc. 38
First Capitol Computer 6
Floppy Disk Services 30,31
Generic Computer Products, Inc. 27
Graphic Design Systems, Inc. 66
Guardian Data Systems 66
HUG Conference 21,29
Headware 46
Heath Company 3
Hilgraeve, Inc. 2
Jay Gold Software 26
MDG & Associates 68
M.E. Pittman 63
Paul F. Herman 38
Redwood Development 46
S&K Technology, Inc. 63
Secured Computer Systems 19
Software Toolworks 57
Wizard Software House 12

On The Cover: Shown are the Doves of Peace overlooking the Zenith logo. Made with a Zenith 110 and Hypad DT 11 Digitizer, then photographed with a Rolleiflex SL 66. This was produced by Henry Tournour, a Graphic Illustrator from Brussels, Belgium.



High Powered Ammunition

The pros at First Capitol Computer know what you want—high powered ammunition for your Zenith. So we've engineered a series of the highest quality hard drives available for your Zenith 150 PC. If you're a power PC user, and only have a floppy based system, you're probably wishing for the power of a fast hard disk. Or maybe you have the standard 11 Mb drive and want something larger. Like 36 Mb. Use the full capabilities of powerful programs like Ashton Tate's D-Base III™ and Framework™; Lotus 1-2-3™ and Symphony™, and Micropro Wordstar™.

A Full Range Of Calibres

Our drive kits come in a size for every need; the standard 11 Mb, a serious 20 Mb, and a full bore 36 Mb.*

Our drives will allow autoboot from the hard disk, and require no permanent modifications to your existing hardware, and no changes to your operating system software—we've got everything needed in ROM on the controller board, customized for the Zenith PC.

Technical Wizardry

Our drive kits are engineered by Software Wizardry, the Zenith experts the experts consult, and are available exclusively through First Capitol Computer. You can count on leading-edge performance; quiet, smooth, and fast operation that will speed you through your most serious software.

All Hard Disks Are Not Created Equal

A lot of people are selling hard disks at a lot of prices—know what you are getting! Rapidly improving technology has resulted in the dumping of obsolete drives on the market. Not at First Capitol Computer!

Our drive kits use the latest state-of-the-art drives, with the newest media, head designs, fast access time, and low power consumption. You can be assured of the highest quality, equal to Zenith's original manufacturing standards.

Firepower

Load A High Calibre Winchester In Your Zenith

Value Priced

Our prices are some of the most competitive around, and when you consider our latest state-of-the-art components, First Capitol's hard disk upgrades are the best value on the market for your Z-150 PC series.

All You Need To Know

You don't have to be a technical wizard to install our systems—our wizards have done that for you! You get the drive, controller card, all necessary cables, and complete instructions.

Available Models:	List Price
SWI-WIN150-11 11 megabyte* internal hard disk kit	\$ 895.00
SWI-WIN150-20 20 megabyte* internal hard disk kit	\$1195.00
SWI-WIN150-36 36 megabyte* internal hard disk kit	\$1595.00

Available direct from First Capitol Computer. Please add \$2 minimum (or 2%, whichever is greater) for shipping and handling. If shipped to a Missouri address, please add appropriate sales tax.



First
Capitol
Computer

*drive sizes are unformatted capacities

First Capitol Computer is a division of Software Wizardry, Inc.

1106 First Capitol Drive
St. Charles, MO 63301

Express Order Line (orders only please)
1-(800)-TO-BUY-IT (1-800-862-8948)

(314) 946-1968 (technical info)

BUGGIN' HUG

Why No H/Z Graphics Program?

Dear HUG,

We own an H89 with a Zenith 25AA printer. Why hasn't Heath and/or Zenith come up with a graphics program for these? Did get the H25/Z25 Graphics Generator from a different source. Then found, with our printer, we cannot do reverse video. Is someone working on this?

Can anyone who also has this graphics program explain to me why when printing 3 screens vertically ours will only print a total of 56 lines?

Sincerely,

Jane S. Melberg
St. Rt. 1, Box 135
Webb Lake, WI 54830

Hardware Reset for Zenith 150

Dear HUG,

As you know, the Zenith 150 computer is highly compatible with the IBM PC series of desktop computers. This compatibility includes some of the PC's short comings, such as the lack of a true hardware CPU reset.

You can add true hardware reset to the Z-150 with a small modification of the CPU board. Simply add a push button switch to the CPU's clock generator circuit to momentarily ground the CPU's reset input. You cut no traces. The connections can be removed and your CPU board returned to its original condition should service be required.

Begin by unplugging the computer and removing the CPU board from the computer. Locate capacitor C7. This capacitor is a large yellow 22 ufd. electrolytic capacitor located near the top center of the CPU board. Solder a stranded wire to each capacitor lead. The wires should be long enough to run to the back of the case near the power supply.

Attach a normally open push button switch to the wires. Drill a hole in the back panel of the cabinet and mount the switch. Replace the CPU board, power up the computer and test your modification. On power up, the computer should behave in the normal fashion. Once the computer is powered up and DOS loaded, push the new reset button. Your computer should reset itself, begin the power up sequence, and reload DOS.

Although you might be tempted to drill a hole in the CPU board's mounting bracket and mount your reset switch there, don't. You will alter the board's appearance and might have difficulties should you wish to replace the CPU board in the future.

Sincerely,

Jerome F. Jankura
Nela Park
Cleveland, OH 44112

Canada Customs

Dear HUG,

RE: Canada Customs

You might wish to pass on some information to your readers who export software to Canada, or those in Canada who import it. It might save the ex/importer a few dollars.

Canada Customs divides software into the categories of operating system software and applications software. Operating system software is considered to be an integral part of the computer and is, therefore, considered as hardware, which attracts both duty and federal sales tax. This duty and sales tax is paid on the total purchase price of the operating system software.

No duty is paid on applications software. Federal sales tax is payable, but ONLY on the value of the physical medium on which the program is recorded.

Of course, to take advantage of this distinction, you must provide sufficient information on the shipping invoice, and probably bring the Canada Customs officer's attention to his department's policy in this regard, which is set out in Revenue Canada Memorandum D13-11-2.

Most invoices that I have seen, or have received, have merely shown, for example, information such as: "1 MyChess \$25.00." What the invoice should also show is that "this is applications software, recorded on 5-1/4" (or 8") diskette of approximately \$2.00, the medium itself having a value of \$X".

The practical effect of this, of course, is that instead of paying 10% federal sales tax on the sum of \$25.00, (after making appropriate adjustments for the rate of exchange on the Canadian dollar), you pay only on the value of a diskette of approximately \$2.00. The amount of federal sales tax on \$2.00 is so small that the Customs officer is likely to hand the item over to you without requiring you to pay anything.

The same situation applies if the applications software is recorded on a "chip", as for example the Watzman/HUG ROM HUG part number 885-4600). The local customs house attempted to treat this ROM as hardware and would have had me pay both duty and federal sales tax on the total purchase price. The Minister of Revenue Canada has confirmed to me that Watzman/HUG ROM should, in fact, have been treated as any other piece of applications software.

I have found the Minister to be a very understanding person. If you have had, or should you have any Canada Customs problems, do not hesitate to communicate with him. His name and address are:

Honourable Perrin Beatty
Minister, Revenue Canada - Customs and Excise
Connaught Building, McKenzie Avenue
Ottawa, Ontario, K1A 0L5

Incidentally, disk drives, disk drive power supplies and cabinets, imported into Canada by themselves, and not as part of a computer, attract federal sales tax, but NOT duty, which is generally 3.9% for computer equipment that is subject to duty.

Yours truly,

Jim Shantora
Barrister, Solicitor, Notary Public
109 Silversted Drive
Agincourt, Ontario M1S 3G4

Screen Dump Doesn't Work With Some Printers

Dear HUG,

When I started up my MS-DOS (H100) system for the first time, I was pleased to find that it included a screen dump utility, (Psc). Unfortunately, I don't have one of the supported printers and I had to resort to the "plain vanilla" version of the program, PSC.COM. What is worse, it didn't work! The whole screen was printed on one line. My prowriter went nuts.

With help from DEBUG, I discovered that PSC.COM was sending Line Feeds to the printer prior to the Returns. That does not work with the Prowriter and some other printers. The codes need to be sent in the other order. The following DEBUG session will solve the problem.

```
A>DEBUG PSC.COM <Return>
-L <Return>
-ECS:034F <Return>
1243:034F 0A.0D <Return> You add the 0D followed by Return
-ECS:035A <Return>
1243:035A 0D.0A <Return> You add the 0A followed by Return
-W <Return>
Writing 0299 bytes
-Q <Return>
```

I hope this will help somebody else out there.

GOD Bless,

Bill Locke

Which Is Better . . .

Dear HUG,

I work in a department that now has three Zenith computers, one Z-100 and two Z-150s. After using both systems, I have become more and more convinced that the Z-100 is the way to go. I'd like to present a couple of points that I feel should be made in light of the Z-100 PC article by Mark Foster that appeared in the January 1985 REMark.

I will not argue with the fact that the Z-150 is a better IBM-compatible than most others, but then I am not clearly convinced of the need for IBM-compatibility, if you have to sacrifice hardware capabilities or ease of use. I would suggest that Mr. Foster was actually quite wise in not mentioning the Z-100, since several very nice features of the '100 are missing from the '150.

Why was "the design model for the keyboard naturally . . . that used on the IBM-PC" when all good Z-100 users know that the '100 has a keyboard that is far closer in both layout and feel to a typewriter than either an IBM-PC or the "new and improved" Z-150's? I strongly suspect that if I had not encountered a '100 prior to the '150, I would not know what I was really missing and would not mind the fewer function keys in an awkward place, the Alt key below the Shift, the cursor keys among the numbers on the keypad, the much longer reach to the "Delete" key, or the very different (and to me annoying) "speed-up" auto repeat.

Are the graphics capabilities of the '150 improved over the '100? In some areas, such as the smooth scroll, I'd say yes. But if they were in all areas, one would not need to purchase a 319 board to emulate '100 graphics in the '150.

Another beef that I have is that the Z-150 was designed with the IBM idea of "everyone has a parallel printer" in mind. Connecting a serial printer is far easier on a '100 than on a '150, since if my information is correct, the '150 requires the serial printer to be connected with a

null modem cable. Why? The '100 is not that fussy. If you are installing a '150 in the midst of a land full of serial printers, that is a hidden fact that should be more publicized.

Can't resist noting there is no mention in the article of the fact that the Z-150's Winchester can only support 4 partitions, while one in the Z-100 can support 16. Which is better? I'd take the 16.

What's the bottom line? If you are interested in IBM-compatibility, the '150 becomes an obvious possibility. Just don't sit in front of a Z-100 first.

Sincerely,

Richard K. Rabeler
Dept. of Botany & Plant Pathology
Michigan State University
East Lansing, MI 48824-1312

From January 29th To January 1st!?!?

Dear Walt:

Several users have reported a minor bug in the clockread software which is provided with the standard P-SST distribution disk. This particular bug only shows up on January 29, in non-leap years (which is why we did not catch it before this year!)

On January 29, the clockread will set the date back to January 1. The quick fix is to simply reset to the correct date, and everything will be fine. We will have the fix included in a software update that will be available sometime before January 29, 1986! For those who do not want to wait, and wish to correct it themselves, the following patch will correct things:

```
A>DEBUG CLOCK.COM
-E519
xxxx:0519 01.7
-W
WRITING xxxx BYTES
-Q
A>
```

Thank you for getting this information to your readers.

Sincerely yours,

Craig S. Ware
Director of Marketing
Software Wizardry, Inc.
122 Yankee Drive
St. Charles, MO 63301

Improving Z-150 MS-DOS Version 2

Dear Walt:

Additional programs and documentation have been developed to improve IBM compatibility and provide additional features for the Z-150 MS-DOS version 2 operating system (OS-63-50). The programs are:

NODEBUG — Temporarily disables Zenith extended debugging features which conflict with some third party software.

BOOTF — Temporarily disables the winchester and boots the floppy to run some third party software.

MODE — An IBM compatible program similar to CONFIGUR.

COMP — An IBM compatible program similar to FC.

TREE — An IBM compatible program similar to SEARCH.

MAP — A program to change drive names similar to IBM ASSIGN.

ANSI.SYS — Documentation on the ANSI console device driver.

MDISK.DVD — A memory disk device driver.

Documentation on these additional programs is included for incorporating into the current manual. These programs and documentation are being included with all current production and are available as updates for a small charge to current owners of the product (part number 840-43, price \$17.35 plus \$1.74 shipping and handling) by contacting the Heath Company parts department. The phone number is (616) 982-3571.

Frank T. Clark
ZENITH DATA SYSTEMS

Needs Help In Hong Kong

Dear HUG,

I am a HUG member for nearly two years. My occupation is an Electronic Engineer and also a freelance programmer. In Hong Kong, there are a certain number of Heath/Zenith computer systems installed in various kinds of companies. Therefore, I decided to write some application software for them. I need to have the following documentation for the development job.

- A) HKS — Complete source listings for Cassette O.S.
- B) HOS-1-SL — HDOS V.20 source listing

The sole agents of Heath in H.K. told me that they got no reply from Heath for my order. Therefore, my final chance is to ask for help from HUG.

Please try your best to find the documentation for me, even a photocopy will be beneficial to me. I have both an H-89 and H8 systems on hand and willing to pay any expense for the documentation.

Your effort will be greatly appreciated.

Yours faithfully,

Ving Hong Ho
Room 1815, Shek Kuk House
Shik Wai Kok Est.
Tsuen Wan, Hong Kong

Better Late Than Never

Dear HUG,

My thanks to Pat Swayne for his article on RCLOCK for Hero in the July '83 issue of REMark. (Just a little slow getting around to trying it.) In order for the program to work, I had to add one statement in the RDCLK routine. The statement was ANDA #0FH to clear the high order bits from CLKDAT. The statement was placed just before the RTS statement. Otherwise, the ten's digit for hours and minutes were invalid, which created all sorts of problems. RCLOCK works fine and am thinking about extending it to speak the current time when any key, except reset, is pressed.

Sincerely,

J.P. Weichert, Jr.
14506 Kolbe Spur
Cypress, TX 77429

Two Part Puzzle

Dear HUG:

Some of your readers may be interested in a problem which I recently encountered while attempting to write a simple "Submit" program under CP/M-80 on my H-89. This program, written using "Fast Eddy" (HUG part number 885-8018), would try to set up but would return to the system prompt without running. After several days of writing a variety of test "Submit" programs, I managed to get one to work. A printout of this appeared to be the same as a supposedly identical program that did not work.

A dump of these two showed that they were indeed the same, except that the working one was terminated by one line-feed, while the non-working one ended with two line-feeds. Apparently, the "Submit" program looks for a command after each line-feed and gives up if it encounters another line-feed in quick succession. I have never seen any reference to this in anything I have read about CP/M.

The second part of this puzzle was that I had not entered two line-feeds (or carriage returns) at the end of the program. It appears (in Fast Eddy) that when you go from the "Edit" mode to the "Command" mode by hitting the HOME key (as is necessary in order to exit the program) a line-feed is automatically added. This may be a desirable feature for normal editing, but must be avoided in this case. Thus, in order to write a working "Submit" program, it is necessary to hit the HOME key at the end of the last line of the program without a CARRIAGE RETURN.

I obtained one of the first H-8 kits and later one of the first H-89 kits for use in number-crunching problems in engineering and as a learning tool. Since then I have retired (for the third time) so use as a learning tool is my main justification for having a computer. The collection of excellent programs I have purchased will keep me busy for a long time. I do not feel a need for a fancier system and thus hope that all our H-8 and H-89 users and programmers will continue to support these fine machines.

Sincerely,

Robert W. Tripp
1055 Peppertree Drive
Sarasota, FL 34242

Help Wanted

Dear HUG:

Help wanted. Would like to correspond with anyone who has successfully interfaced the ET100 to the S100 buss.

Sincerely,

Donald E. Risher
908 Charlotte Place
Charleston, WV 25314

Pertaining To Wheby's Letter

Dear HUG,

This letter pertains to Frank Wheby's letter in the November '84 issue of REMark.

I suggest Mr. Wheby try using the TTY.BAS communications program in Appendix E, ZBASIC, via the J2 port.

I recently had occasion to transfer programs between my Z-120

Vectored to Page 66

Sequential Files — Part 2

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

Last month we looked at what makes up a sequential file, and we created records in the proper format to be entered into sequential files. This month we'll create and use those sequential files. As stated last month, this information is copyrighted and is presented for your personal non-commercial use. Any programs you develop with the knowledge gained here, however, may be used any way you like. All the programs presented in this series are available on a disk from the author. Details are in last month's article.

As with any type of file, in any of the programming languages (MBASIC, C, FORTRAN, COBOL, etc.) you can't do a thing to the file until you OPEN it. If you followed our series on random files, you know that a file, once opened, could be written to or read from. It's different with sequential files. You can only open the file to read from it, OR to write to it. The same file may be opened and written to, then closed and reopened for reading. As we go along, you'll see that you can't open an existing file and add data to it. As soon as you open a sequential file for writing, anything which was in the file is lost. Fear not, you can add to an existing sequential file, however, by using a temporary file. It's not at all complicated, but you must beware.

NEVER OPEN AN EXISTING SEQUENTIAL FILE FOR WRITING IF YOU WANT TO RETAIN THE DATA THAT'S ALREADY THERE!

We'll use the program GETINFO.BAS, which we wrote last month, and put its data into a file called SEQFILE.DAT (SEQuential FILE DATA). To do that we'll have to OPEN our file. We'll do this by using the OPEN command. In reading the MBASIC manual, we see that a mode must be assigned. The permissible modes are:

O = open a sequential file for Output
I = open a sequential file for Input
R = open a Random file

In the above descriptions, output means OUTPUT TO THE FILE, and input means INPUT FROM THE FILE. We want the "O" option so that we can output to, or write to, our file. We must also assign a number to the file so that our operating system can set aside a buffer. With the CP/M operating system, three files may be opened at one time (unless the /F: option is used). We'll worry about that option at another time. For now, we'll assign our file as file number 1. Finally, we've got to pick a file name. We've already done this. When you assign your own file names later, remember that the operating system sets the convention for permissible names. In both HDOS and CP/M the file name may be from one to eight characters, and may have an optional extension of one to three characters. We chose "SEQFILE.DAT". The

MBASIC statement to open our file must be placed before the routine we wrote last month, so let's begin our work this month as follows.

```
2 ' **** WRITESEQ BAS
4 ' **** DAVID E. WARNICK
6 ' **** COPYRIGHT 1984
4900 'OPEN SEQUENTIAL FILE FOR OUTPUT
4910 OPEN "O",#1,"SEQFILE.DAT"
```

When we worked with our program last month, we printed each record on the CRT as we created it. We'll continue to do that so you can see what's going on in the system. However, this month we'll also send the data to our sequential file so that it can be stored on the disk. We'll begin at line 6000. We do this with the PRINT command. Normally, PRINT sends its information to the CRT. However, we can redirect its output to any file (buffer) we like by specifying that file's (buffer's) number. The command must print the contents of our record which we assigned to R\$ last month into the file SEQFILE.DAT which we opened as file number 1. The command line looks like this:

```
6000 PRINT #1,R$ 'PUT THE RECORD INTO THE FILE
```

With this record written, we can go back to the beginning of the program for another. We'll use a GOTO statement.

```
6010 GOTO 5000 'GO BACK AND GET ANOTHER RECORD
```

When we combine our two programs, we'll remove the old line 5250. After we've written all our information, we'll need a way to get out of the program. We can do that simply by entering just a carriage return for the last name and testing for it. The last name entry was made in last month's program on line 5030, so we'll make our test on line 5035.

```
5032 'TEST INPUT FOR JUST A CARRIAGE RETURN
5035 IF AS="" GOTO 6500
```

We'll add our exit routine at line 6500. As with any file we open, we've got to close all open files prior to program termination. This assures that anything in the buffers is taken care of and the operating system has the correct status. We continue our program with the shutdown procedure.

```
6500 'CLOSE ALL FILES BEFORE ENDING
6510 CLOSE #1
6520 END
```

Reread what we just did, then turn on your computer, call up MBASIC, and enter each of the numbered program lines we've presented here. Then type the following commands.


```

SAVE "WRITESEQ",A
MERGE "GETINFO",A
5240
5250
2 ' **** MAKESEQ BAS
SAVE "MAKESEQ",A

```

For this to work, last month's file GETINFO.BAS had to be on our disk. The MERGE command put both files together. The lines 5240 and 5250 were a cheap and dirty way to remove the unwanted GOTO which put us in an endless loop. Line 2 is entered with our new file name and showed how we can save the result of merging two or more files without losing any of the original files. Simply save the result of the MERGE as a new file name.

This would be a good time to turn on the printer and type LLIST to list a hard copy of the program. Verify your typing and run your new program by typing:

```
RUN "MAKESEQ"
```

When asked for inputs, enter data just as you did last month. This time, as you begin to add more and more records, you'll hear your disk drive start and stop. You've done it. You're creating a sequential file full of names and phone numbers. When you get tired of typing, type the return key when asked for a last name. Program execution should stop. The file you created is saved on the disk. DON'T RERUN THE PROGRAM AT THIS TIME!!

Return to your operating system by typing

```
SYSTEM
```

Then type DIR (CAT for HDOS users). The file SEQFILE.DAT should be shown in the directory listing. You can view the contents of the file by typing

```
TYPE SEQFILE.DAT
```

or list it on the printer by typing

```
LIST SEQFILE.DAT
```

if you have the utility LIST.COM on your disk. The data you entered has been saved. You have created a sequential file. So far everything works. It's time for a very important lesson on the care and handling of sequential files.

In the second paragraph of this article, I told you that you must not open an existing sequential file and attempt to write additional data to it. If you do, all the old data will be lost and only the new additions will be saved. To prove this, we'll rerun our program and enter new data. Then we'll look at the contents of our file.

Call up MBASIC and type RUN "MAKESEQ. This will reopen the file SEQFILE.DAT to be written to. Again, enter data when asked for it, but be sure it is different than the data you entered the first time. After you've entered a few records, enter just a return for the last name. Return to the operating system and use the DIR command to see that the file still exists. Finally, review the contents of the file by using the TYPE or the LIST command. Notice that the newly added data is in the file, but that the original data has been lost forever. Let this be both a very important lesson and a warning that an existing sequential file loses all its data if it's opened for writing. We'll look at how to add data to an existing file in next month's article.

Thus far, we've been able to put data into a file. We could look at the file by using the TYPE and the LIST commands of the operating system, but this isn't a very useful thing to us. Let's look at how

MBASIC permits us to extract individual records from a sequential file, and how to break those records down into their component fields.

The first step will be to open the file. This time, we'll open it for output from the file (reading). We can open a sequential file as often as we like for reading and no data will be lost. Let's start our program.

```

2 ' **** READSEQ.BAS
4 ' **** DAVID E. WARNICK
6 ' **** COPYRIGHT 1984
5000 OPEN "I",#1,"SEQFILE.DAT

```

This time the option "I" is used to show input from the file. We had to number our buffer as before, and to specify the name of the file to be opened. Just as we use the INPUT command to get data from the terminal, we can use it to get data from the buffer assigned to the sequential file. The command used with files is INPUT # and does not echo a question mark "?" as the INPUT command does.

There are some rules to follow when reading from a file. If a quote mark, ", is encountered, the INPUT# function looks for a second quote and accepts the data between the quotes. So, DON'T EMBED QUOTES IN THE DATA IN SEQUENTIAL FILES. You can read several variables at a time, and MBASIC will consider a comma, a carriage return, or a line feed to be a delimiter. We let MBASIC insert a carriage return and a line feed between records when we wrote our file. This permits our program to read a whole record at a time, a method I prefer as it parallels random file operation. We then have our program parse (break into pieces) the record by using the delimiter we chose. In our case, it was the backslash "\". We'll use R\$ as the record we read from the file we just opened.

```

5100 'READ A RECORD
5110 INPUT #1,R$

```

This line will read a record from the file #1. It will see the record delimiter supplied by MBASIC and will stop at the end of the record. R\$ will look like this:

```
Lastname\Firstname\MiddleInitial\Phone Number
```

We need to separate this line into individual fields. We could read each character and add it to our variable unless it's a backslash. However, MBASIC has provided a very convenient function called INSTR(). This function searches a variable until it finds a character we specify, then stops. We can give it the name of the variable, the character to look for, and at what point in the variable we want it to start looking. The format of the command is:

```
INSTR(N,X$,Y$)
```

where:

N = the number of character in the variable to start looking
X\$ = the name of the variable
Y\$ = the character to search for

Let's look at the variable R\$=

```
Warnick\David\E\555-5555
```

If we use the command

```
A=INSTR(1,R$,"")
```

it will begin on the first character of R\$ and look for a backslash. Because the backslash is the eighth character of the string, A would be set equal to 8. Our next command would be

```
B=INSTR(A+1,R$,"\"")
```

This would begin a search at the character after the first backslash and look for another. It would set B equal to 14. And so we would continue until all the field delimiters had been found. Let's write a program segment to do this for us.

```
5200 'PARSE THE VARIABLE R$
5210 A=INSTR(1,R$,"\"")
5220 B=INSTR(A+1,R$,"\"")
5230 C=INSTR(B+1,R$,"\"")
```

There is now a variable which tells us where every backslash exists in our record R\$. These variables permit us to extract the information we want by using three more MBASIC functions:

```
LEFT$
MID$
RIGHT$
```

Each of these functions permits us to assign part of a string to a variable. While LEFT\$ and RIGHT\$ will get the left or right end of the string, MID\$ will take characters from the middle. With LEFT\$ and RIGHT\$, we must tell them the name of the variable and how many characters we want. With MID\$, we must also include the starting point in the variable. From our example above, the command

```
LN$=LEFT$(R$,7)
```

would make LN\$ equal to "Warnick". Note that the number of characters we want is one less than the position of the backslash. Our program lines would read:

```
5300 'BREAK THE RECORD INTO FIELDS
5310 LN$=LEFT$(R$,A-1) 'LAST NAME
```

The next fields begin after a backslash and end before another backslash.

```
5320 NM$=MID$(R$,A+1,B-A-1) 'FIRST NAME
5330 MI$=MID$(R$,B+1,C-B-1) 'MIDDLE INITIAL
```

We don't know how many characters are in the phone number, however. That means there's no way to write the last line. MBASIC comes to the rescue again. The function LEN() returns the length of a string. The program line

```
5340 D=LEN(R$)
```

sets D to the number of characters in the record, R\$. Now we can add the line

```
5350 PN$=RIGHT$(R$,D-C)
```

To test whether we've done it right, we'll add the print statements:

```
5400 'PRINT THE OUTPUT
5410 PRINT "Last name is ";LN$
5420 PRINT "First name is ";NM$
5430 PRINT "Middle initial is ";MI$
5440 PRINT "Phone number is ";PN$
5450 PRINT 'SKIP A LINE
5460 GOTO 5100 'DO IT AGAIN
```

Now we're in an endless loop and will run into trouble when we get to the end of our file and are out of data. MBASIC provides a

test to check for the end-of-file condition in sequential files. It's called, appropriately, EOF() and the test is true if we're at the end of our sequential file. We'll test for end-of-file before we try to input data by adding the program line

```
5105 IF EOF(1) GOTO 5500 'TEST FOR END OF FILE
```

Finally, we'll add two program lines to close our file and end execution.

```
5500 CLOSE #1 'ALWAYS CLOSE FILES BEFORE ENDING
5510 END
```

Now go back to program line 2 which has the title READSEQ.BAS and type all the numbered program lines. Then SAVE and RUN READSEQ. You should see all your data from the phone number sequential file scroll up your screen. We couldn't jump into the middle of the file to get a specific name, but rather had to start at the beginning of the file and read each record until we found the information we wanted. Had this been a name and address file, we could have made it print mailing labels.

Next month we'll begin by showing you how to change the data in an existing file without losing anything. See you then. ✕

**ILLUSTRATOR
GRAPHICS DESIGN PACKAGE**
by Wizard Software House
79 MARSHALL STREET
PROVIDENCE, RI 02909
(401) 331-5034 (617) 881-2543

<p>Box & Box Fill Diamond & Diamond Fill Triangles & Triangle Fill 16 x 16 Font Generator</p>	<p>Circle & Circle Fill Ellipse & Ellipse Fill Turtle Mode 26 User Defined Macros</p>
---	---

Get a World of Graphics Power



Relocatable Graphic Images
Support for Most Dot Printers
Easy Interface to User Programs

Unlimited Colors
Rubber Band Mode
User Defined Line Styles
Irregular Area Fills
Easy Single Key Commands
Full Interface Operation
640 x (225, 400, 480)

Be a HERO with your Computer Buy One NOW!



\$89.95 for ZDOS or HDOS • Imaginator

Name _____
Address _____
City/State _____
Zip _____

Zdos Hdos

From

Heath



To Endburg

*A Story by
Crawford MacKeand
115 South Spring Valley Road
Greenville, DE 19807*

The Background

From time to time wild boar were to be found in the hills between the land of Heath and the City-State of Endburg. But little else, for the border lands were, for the most part, inhospitable trackless wastes, supporting no agriculture, with no roads or trails, barren with bogs and marshes, unmapped and known only to the very few who had journeyed there. The King of Heath had become anxious to establish trade with Endburg, and his Ministers discussed and debated endlessly about the ways and means which the Heathens should use to establish routes through the waste. Some favored one faction, some another, and others yet a third and a fourth, until Heath was thoroughly taken up with the debate. Some members of the Assembly thought theirs' to be the only way, while others favored a more BASIC approach, and the third major group, led by M. Pascal, favored what they called a Structured method.

The King was a wise man and he listened to his advisors, and when they had quite debated themselves hoarse he laid his plans. I'm not going to tell you just yet what they were, because the King also took good care of his money and as you will see, that is as in all well regulated families, a very important piece of the action. However, in essence, the Assemblymen said "We should travel light, for in this trackless land why would we wish to suffer from carrying much tiresome baggage?" And the men from the BASIC constituency said "But we have invented a good and flexible engine for these lands," while M. Pascal and his supporters said "You must map your way and have adequate support, for how else can you establish trade across the waste to Endburg?" And the King had to listen. He was a good King and a good King has to be a good listener. He was a wise King, and believing he might well need the support of all his countrymen, he thought well about all that he heard. And, as I have told you, he laid his plans, but especially he remembered the question of the money.

First Scene — The Lone Traveler

It is a dark night, with a cool mist turning to light rain. The people of Endburg are going to bed, banking the fires in the hearths, checking on the children, locking the doors if they are of the timorous persuasion, although Endburg is a friendly place with little crime and an able Master, and finally walking the dog. Dog

is not always of the same mind as his master and the dogs of Endburg knew that someone went by, but they barked and then said no more as Special Agent Z80 went through the streets. Unseen and unheralded Z80 was a hand-picked man from the ranks of the Assembly. He was one of the very few who knew the wastes like the back of his hand. Indeed, he had known them for years, first as a teenager when he loved to fish and play games and later when the King had selected him for more serious business. His fishing trips and long training in the rugged sports of this forbidding terrain, beautiful indeed to those who knew it, had made it very easy for him to pass through to the King's contact in Endburg. Certainly there were brooks and rills to jump, but he had learned well to JMP and to ROTATE and SHIFT. He carried little equipment, and that was in his pockets; he carried no fire arms, though his knife was keen, and in short, he was fast and unencumbered. His strengths for the King's mission were his detailed knowledge of the country, his powerful mastery of the commands at his disposal, and though he was but one man, Z80 carried messages in the deepest code which would in time move mountains.

The King knew all this, and entrusted this fundamental task to Z80. But there was more than this. Special Agent Z80 had also been given many years of training in Assembly, and the King knew that this was expensive. (I told you the King had a good eye for money.) And there were few with his skills. The King knew that if many of his subjects were to make the journey to Endburg, then Z80's route would not be their route. But this was to be the building block on which his policy would go forward. The coded messages which Z80 bore from Heath to the Master of Endburg were to be the starting point for the rest of the story.

Second Scene — The Expeditions

A number of messages had passed between Endburg and Heath, and the time had come for the King to take the next step. There were yet no maps, only such sketches as Z80 and his like could provide, and these were of little help for a larger party, even with the best equipment. But with the knowledge that his people had found that the way was possible, and Z80 was one of his people, the King sent out his first trading expedition with confidence. I call this a trading expedition because the King had skeptics in his country. He had to convince them that this journey could be

made by ordinary people, even though they might be the more adventurous souls at first. Truth to tell, there had been little competition for Z80's arduous journey, but now an altogether different picture came to view. Business people were interested. Profit was to be made, and folk pricked up their ears.

The merchants encouraged the King to take a BASIC approach, and they said "Our men have proved this in trade to other countries". The academics and the military men were for once in agreement, to the wonder of the people, for the academicians said "Structure!" and the soldiers said "Discipline!" and ordinary citizens cringed. Then the King issued his proclamation which was to take effect immediately. A small expedition, much smaller than the military wanted, was to set out over the hills, valleys, rivers, and swamps to Endburg. It was to be guided by John BASIC, a well traveled merchant adventurer, who had crossed the seas and had made a great deal of money in the process. The Professors and the Generals fumed, but the King was adamant and they had to hold their tongues. (There were certain advantages to living in those ages.)

John was very much concerned with the profitability of the expedition, and yet he was not averse to taking a proportionate risk. (He and the King were both money men as you will see). So he consulted the sages and he talked to the soldiers, to the academics, and he talked to the assemblymen. He was even allowed to talk to Z80, but gained little from the exercise. So he took his BASIC & Sons cross country traveling engine, which today we would call a "SYSTEM", and to move it forward he took a motley crew from his company (they really were very motley) and set off with the King's blessing and a payload of golden goose eggs, which both he and the King expected to bring considerable reward. He went quietly without fanfare and headed North into the wilderness.

They were a wild company, always wanting to GOTO that place or GOSUB to the other place, but John was a good leader and they had little trouble. Of course, there came a time when a long thin man called Stringy fell into a morass, and they had to pull him out with a rope of GOTOs. But day-by-day they moved forward, testing the ground under them at each step, moving carefully and slowly through the marshy places, and down the steep places, but always interpreting the many signs to be seen by those whose life has been spent on such. And there came another time when the whole expedition came to a halt. A small, but deep, river crossed their path and the traveling engine could not pass beyond. Even with all the GOTO ropes they had brought, the few men in the party were not strong enough to carry it forward. The leader sent scouts up the hills to look for other routes as they had done successfully before, but now it was misty in the hills. He thought briefly of the Structures that the Pascal party had lovingly described to him and the picture of Z80 fleeing by on a chance floating point to make the crossing entered his mind. But this was not John's way. He had his team (I nearly said he had his tea, and that would have been a good idea, but they were traveling light and the tea had gotten soaked in the marshes the night before)... he had his team build a raft with some GOSUBs lashed together with GOTO ropes, and for flotation he used some ONGOTOs he had in his pack. Altogether a most untidy contrivance, but the engine made it safely across the river.

So they traveled on their way, having many adventures, oftentimes being lost, and once even going in a complete circle. Eventually, after wandering on a rather erratic path across the

wastelands, for they had no maps at all, and John was not a map reader by nature, one of John's scouts on one of his frequent trips to the top of some hill or brow to try and find out where they might be, reported smoke on the horizon. Other such expeditions had done no less, and had found the wrong burg and been imprisoned or lost or ransomed, but John was well chosen and soon thereafter they came into the outskirts of Endburg, passed through the Master's Customs post, known there as the Department of Ins and Outs, or I/O for short, and were welcomed well as traders often were in those days.

Over the next few years, John and other traders instructed by him, made many such journeys to their profit and that of the King, and everyone was pleased, except of course, the Professors and the Generals. But the King was not an unwise man, and one day there was a surprise in store for those people in Heath who took an interest in such things.

Third Scene — Building A Highway

The King, who had been counting his wealth much increasing with the steady trade with Endburg, called an audience with his principal advisors. There were members of the Assembly, and there were Generals and Merchants of the highest reknown, and there were the brightest and best from the groves of Academe. He said to them "Loyal subjects" (they talked like that in those days), "I have considered long and hard on the subject of the trade with Endburg, and I now decree that a highway shall be built to supersede the present trade route from Heath to Endburg. Further," he said "the said highway shall be well provided with structures, and to this end I command the Professors and the Military Engineers," (you may remember that in those far off days before Civil Engineers were invented, they were all Military), "to prepare adequate designs and structures for the new highway, and the Army must send mapping expeditions into the hills and the traders must give their best information to the Mapmakers. In this way, the latest traveling engines will carry yet further forward our most desirable trade program with Endburg."

And this was the King's decree. And nobody was very truly happy. But the Professors were fairly happy because the King had taken their advice, but they realized that they would now have to think up structures that would work. And the Generals were fairly happy because the Army would have work to do, but they would have to leave their snug Officer's Club for the field. And the engineers were happy because now they could build their bridges, but they were not very happy because they were after all engineers like me and possibly like you. And the merchants were of divided counsel: first, because they saw control of the trade slipping away, but against that they saw the chance of a much greater trade in more and bigger and heavier and more profitable goods. But then they also saw more regulation, and then again they were merchants and so divided their counsel most naturally. John BASIC kept his peace for he saw the logic of the King's position.

Despite the fears of the other parties, the Structure teams went ahead with a will. The engineers built their prefabricated bridges, the survey teams made splendid progress with their maps, and the army officers contributed their wisdom to the routing of the highway, for the King was mindful of the defense of his realm too. And as the field construction people came upon obstacles, they found that they had been foreseen by the planners and by the engineers. The structures were well crafted, and much of this was credited to the now aging M. Pascal, whose stern admonitions on typing of the parts was now seen by the

constructors to have much value. And nobody had to go without his tea, for this was a well considered venture, and cookhouses and all the other overheads so necessary for such grand works, were provided at the King's order. The quiet undercurrent of the merchants was still heard in the land. They did not dare to speak too loudly against the King's decree, but he heard and let it go, for he knew that they had real and honest concerns. Nowadays, we would worry about capital and taxation and other like things, but the merchants saw that much of the Kingdom's wealth was going into the new highway, and this was not their accustomed manner of business. This the King understood.

As I have told you, the King was a wise man, and again he had chosen his men well. (We have not said much of the women of Heath, but in the manner of the times they were working quietly in real time. It was the fashion then for the deepest decisions to be made by the joint heads of families, and so Queen Ada had introduced some very fundamental concepts to the King, which together they quietly threaded into the fabric of their Royal project. But this must be another story.)

As the highway was completed in the fullness of time, but on schedule and within budget, (yes, even in those far-off days) there was a great opening ceremony. And there were present the Master of Endburg, of whose help we have not spoken, although it was great, and the King and Queen Ada, the Army, the Burgers of Endburg, the Professors, the Merchants, the machine makers from the BASIC & Sons foundries, the Assemblymen, and "in-cognito", of course, Special Agent Z80. The King ensured that he was well looked after, but everyone had tea and cake and ale and marveled at the new road. And it was well used and the fortunes of both Heath and of Endburg increased most wonderfully until they were the envy of the rest of the known world. And, of course, the King found great employment for the talents of all his good kindly people as the trade increased, and they eventually forgot their arguments and labored happily and well at the things they each did best.

How The King Explained

In our more enlightened days, it is well accepted that a success is not to be rejoiced at, but rather to be used as a goad so that further of the sort may bring profits. And so, instead of the dry analyses of our times, it was customary for the Kings and Queens to explain to their people why they had succeeded, or even why they had failed so that all could improve their positions in the world and improve the standing of the Kingdom withal. This was the cause also of much feasting and merrymaking, for the King also knew that in such a way those of different opinions could come together and respect and even understand the motives and reasoning of their opponents.

It was at just such a merrymaking, but after the tea and before the ale, that the King spoke to his people in this manner. "We have just completed the new highway from Heath to Endburg, and all my people may take pride in it, as indeed may the Master and the people of Endburg. But it is the Heathen efforts which I will explain today." This was, of course, the signal for a long speech and all the children groaned and were shushed, but I will just give you the salient points of the King's address.

"I know that you know that we have built a highway, and most of you know that before that time we made trade with Endburg through the hills and wastes, and a few of you know that we communicated with Endburg before the first BASIC expedition. And I am sure that most of you know of the very successful first and

subsequent expeditions to Endburg through the then very much unknown wilderness. These were well supported, strong and profitable expeditions, sometimes as much of discovery as of trade, for we all learned much from them. And you all have heard today of the great highway, which has now been built through these same lands with the assistance and direction of M. Pascal and his academic friends and our own Royal Army in a cooperative effort, which has been as pleasing to me as it was surprising to you." The King, as you may see, kept his eyes and ears open, and as he was King he could say what he thought, and quite often-times he did just that.

So, when he had said the things that people thought he would say, then he went on, to the great disgust of all the children, as you might have guessed, as follows. "I also know that many of you were disappointed and even surprised by my decisions. I would like you to imagine a few years ago when the first BASIC expedition was setting out, and contemplate the scene if my decisions had been otherwise."

"The first expedition is leaving for Endburg, and is well provided with cunningly crafted structures, all carefully typed. There are also fast traveling engines, able to carry heavy loads of merchandise. The structures are strong, for they may be used to cross swift rivers and they require good foundations, for they are built to last. At the first river, the leader sets to with a will, and his team of one hundred builds a very adequate base. And soon they have brought the first members from the base camp to the river bank. But now they need a lifting procedure, and one is brought from the base. After a great effort the bridge structure is built and with relief they move on. Soon a marsh is encountered and an argument ensues. Some say go this way, others say go that way, and yet others say let us use the proper structures and build another of those special bridges we have designed for these purposes. On this occasion they GOTO around the marsh, and soon come to yet another small stream. This is easily bridged using the now standard procedure and so the expedition forges on. But now they come to a misty forest area where there is real indecision. The leader sees something more than the physical problem. That he can handle for he has good people on his team. However, he starts to see that in this condition where the maps are not drawn, where the plans are yet not well defined, he may be building a good road into a defile from which his procedures provide no exit. He talks through the problem with his team, and again they are of two minds. One group says let us go look, let us grope our way through this forest and see what lies beyond. We may be close to Endburg! The other group says let us bring up some appropriate and strong procedures and some well designed machinery, and so we will bring our road through this pesky wood. The leader compromises and they build a small bridge to a promising looking dry land area. But then they need yet another procedure, for big rocks lie in their path. They start to lose time, for the lifting routine is not designed for their rough trails. But eventually the road is cleared and the party moves on. But you see, each obstacle is more of a problem than the last, not in its intrinsic difficulty, but as each one is more and more inaccessible. Also, many of the fast new traveling engines now lie wrecked along the road, which though much better than the original track of the very first BASIC expedition, is still far from being a major highway."

"As these realities become known to the merchants who support the venture, so is that support withdrawn. At length, after a valiant effort struggling in the mists and fogs without maps they return disillusioned and discouraged from their period in the

wilderness to report that the road to Endburg is beyond their capabilities. And worse, they are sure that all the work they have done has gone for naught, in that they still do not know where is Endburg. And they roundly berate the good M. Pascal and his friends, and are lost to the interest of trade."

"Now if you will calculate, John BASIC and his very motley crew of five have taken a hundred days to Endburg. I have paid them each two Dahlers each day, and the worth of the golden goose eggs in trade is D.10,000. at an initial cost of D.3,000. So my profit is D.3,000 and the share of John BASIC is D.3,000 and so we continued to trade. We have covered the risk that some expeditions will fail and as long as we were satisfied with this trade, all was well. On the other hand, if I had at that point set out to use the methods of M. Pascal, you can see that the team of one hundred working for one hundred days in the wild would have cost us D.20,000. There would very likely have been a decent country road to nowhere, and much confidence lost. A poor investment for me and for my people." At this time there was a loud round of applause, although an impartial observer would have seen that it came mainly from one side of the King's audience. But M. Pascal was also a wise man. He did not applaud, but paid the greater attention. And the King knew this and went on.

"For the last part of my explanation" (and there was an audible sigh of relief from all the children present, and they were again quickly shushed by their parents and given small toys to keep them quiet), the King said, "I would like you to remember a more recent time, when other expeditions came in quick succession after that of J. BASIC. Let us now follow a small army which we will imagine I have commissioned from the ranks of the merchants to build a highway to Endburg. Again we come to that first bridge, and with a will the teams of the merchant adventurers set to, and soon a structure is seen across the stream. It is enough to carry the small cross country engines and a steady stream of men, and so using the ad hoc plans of the more gifted members of the force the road moves on. But the road is a rough one, and the bridges need frequent repair for here a wooden procedure has slipped in with a company of steel, and there an integer remains to trip the unwary traveler in a real road. A wider river is still crossed by a raft, now named a ferry to be sure, but the road does reach to Endburg and the trade continues."

"This might in some trades be a very good solution. But in this case, people in Endburg would still be waiting for the faster engines, nor would the trade be so well carried forward. Neither would we have good maps of the country and so we would not know our new road to be the best road, and we would not see that the wilderness could add to our enjoyment and wealth in other ways. Have not farmers and herders moved in since the highway was built, and have not other trades opened up with other cities beyond Endburg? Let us also consider the cost of this work. My Army sent ten men into the wild for five months making maps and plans. This was followed by a great labor at our Royal Factory building all the procedures and typing all our materials with great care, and in this work there were two hundred men for ten months. And finally the construction force of a thousand men for ten months which has cost you all greatly in taxes." The King expected a low moan of pain to rise from the crowd. He was not disappointed.

"This I think you may understand, has added up to no less than four million Dahlers, by the time you have added in consultation with my academics and other costs, while the total cost of the twenty earlier expeditions was approximately D.120,000. A great difference. But now consider this. The trade in golden goose

eggs to Endburg is now as great as our plants can produce them, and my advisors tell me that for reasons of trade (one of my famous ancestors called it Killing the Golden Goose) we should now look to other trades." The Queen nudged him and whispered "That's the speech for the Rotary next week, dear." He went on "and this trade is very good. We have reduced the price of goose eggs to D.5000 and cut the competition to pieces. And selling three hundred crates of our best quality golden goose eggs each year to Endburg, we have been able to sustain a return of over fifteen percent on our capital. But now we have a shortage of engineers and" The rest of the King's speech is unfortunately lost as it is reputed that here Queen Ada nudged him much harder and told the Royal stewards to serve the ale, but quickly. Much cheering is also recorded by the Royal scribe, but it is not certain whether this was for the speech, or for the fifteen percent or for the ale. I am sure that the children were just cheering the end of the speech.

The Royal Rules

After it was all over the King graciously, and of course, with his well known eye for business, was pleased to issue the Royal rules. Translated into English, they go something like this.

First Rule. If you have a small expedition (nowadays we call them programs) to set up, and you are not very sure where you are going, neither are you sure of the risks, nor of the rewards, then use very flexible, but certainly interpreted methods. (We would say languages). For you will otherwise carry much heavy baggage into wild countries where speed counts more than precision, as when chased by fierce animals. (I think the King must have meant clients or customers, but the manuscript is torn at this spot).

Second Rule. If you are intending to fit out a major expedition, then look to your maps, do your groundwork, be sure that you are well acquainted with all the good procedures and practices, choose your compilers well, but above all, take time to establish exactly where you are going. Otherwise, you will almost certainly return with your objectives still not accomplished, and even worse, you will have lost confidence in the power of the structured methods.

Third Rule. (The handwriting is different for this rule, and the evidence is that it was written in by the Queen) If you don't know the size of the expedition, and you are not sure of your starting place or your destination, then you must surely be working in the real time area, and your only salvation is to work even harder on your structures and your procedures, for now they must be strong enough and flexible enough to carry an expedition forward with a good road through territory where the detail maps may never be drawn.

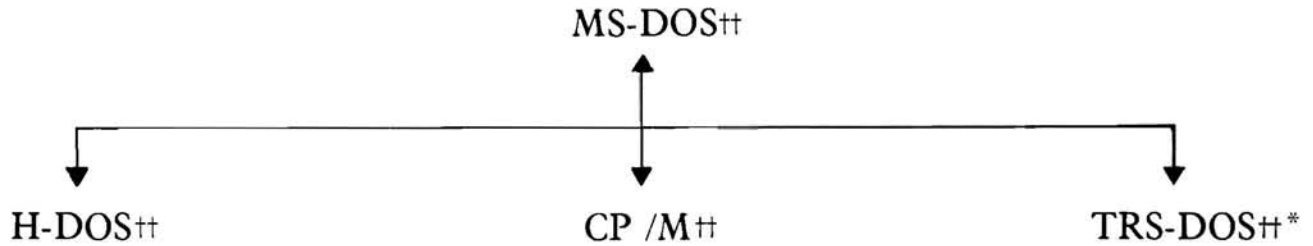
As everybody knows, the Royal Rules have been followed most faithfully ever since, and nowadays nobody ever falls into the trap of using Pascal where he should have used BASIC, or BASIC where he should have used Pascal, or (perish the thought), Fortran or BASIC where he should have used Modula-2 or Ada.

THE END



micro/VERSAL™

- DISK TO DISK FILE TRANSFER UTILITY -
FOR IBM-PC†† OR ZENITH Z100 ††



If you:

- Have or use more than one computer
- Want to Access Public Domain Software
- Have clients with various Systems
- Want to upgrade your present system

Then you can:

- Move files between your Systems
- Transfer from Public Domain format to your own
- Create disks for use on their system
- Move all your source, WP, and data files to your new your new system

Use your IBM-PC/compatible or Z-100 to READ, WRITE, or FORMAT disks for the following systems:

- | | | | |
|--|---|---------------------------------------|--------------------------------|
| • 8 inch standard (SSSD)* | • Heath With Magnolia CP/M 2.24 (DSDD) | • Radio Shack TRS-80 I DOS (SSDD)*† | • NEC PC (8031) (DSDD) |
| • Avatar TC 1 (SSDD) | • Heath With Magnolia CP/M 2.24 (SSSD)* | • Radio Shack TRS-80 DOS 1.3 (SSDD)*† | • Sanyo (SSDD) |
| • Avatar TC 1 (DSDD) | • IBM-PC/CPM-86 (SSDD) | • Radio Shack TRS-80 DOS 2 (SSDD)*† | • Sanyo (DSDD) |
| • Cromemco CDOS - I (SSSD)* | • IBM-PC/CPM-86 (DSDD) | • Radio Shack TRS-80 I DOS (SSSD)*† | • Superbrain 2* |
| • Cromemco CDOS - II (SSDD) | • Kaypro II (SSDD) | • Radio Shack TRS-80 NEWDOS (SSSD)*† | • Televideo (SSDD) |
| • Cromemco CDOS - II (DSDD) | • Osborne (SSSD)* | • Lobo MAX-80 (SSDD) | • TI Professional (SSDD) |
| • Datavue (SSDD) | • Osborne (SSDD) | • Morrow MD2 (SSDD) | • Xerox 820-I (SSSD)* |
| • Datavue (DSDD) | • Osborne (DSDD) | • Morrow MD3 (DSDD) | • Xerox 820 (SSDD) |
| • Epson QX-10 (SSDD) | • Osborne (Ozmosis SSDD) | • Morrow MD11 (DSDD) | • Xerox 820 (DSDD) |
| • Epson QX-10 (DSDD) | • Radio Shack TRS-80 CP/M (SSDD) | • NEC PC (8001) (SSDD) | • Zenith Z-100/CPM-85 (SSDD) |
| • HDOS (SSSD)*† | • Radio Shack TRS-80 CP/M (DSDD) | • NEC PC (8031) (SSDD) | • Zenith Z-100/CPM-85 (DSDD) |
| • HDOS (SSDD) | • Radio Shack TRS-80 CP/M (SSSD)* | | • Zenith Z-90 (SSDD) |
| • HDOS (DSDD)† | | | • Zenith Z-90 (DSDD) |
| • Heath With Magnolia (SSDD) | | | • Zenith Z-90 CP/M 2.24 (SSDD) |
| • Heath With Magnolia (DSDD) | | | • Zenith Z-90 CP/M 2.24 (DSDD) |
| • Heath With Magnolia (SSSD)* | | | • Zenith Z-89 (SSSD)* |
| • Heath With Magnolia CP/M 2.24 (SSDD) | | | |

*Not available on PC version
†Formatting option not supported

(48 PTI SOFT-SECTORED OR 8 INCH*)

micro/VERSAL™ is easy to use – menu driven, and requires only 64K and 2 floppy drives or 1 floppy and a hard disk; runs under both DOS and DOS2.

Also: Define your own CP/M formats with our ANYCPM™ program for disk formats not supported by **micro/VERSAL™** directly. So why settle for anything less -- **ORDER NOW** and get our AST/Utilities package (a \$49.99 value) **FREE**** to celebrate **micro/VERSAL™**'s 1st anniversary. AST Utilities includes a disk sector editor, a file recovery program, a file/disk dump program and more!

**This is our 1st Anniversary Special good until July 1, 1985.

Current **micro/VERSAL™** customers, send \$15.00 (sorry no credit cards) to cover shipping and handling and your original disk to receive latest version and utilities.

micro/VERSAL™ runs on: IBM-PC, IBM-PC/XT, IBM-PC/AT, Zenith Z-100, Zenith Z-150, Chamleon, Columbia, COMPAQ, Corona, Eagle, Panasonic, Otorona, NCR, Sanyo MPC (no formatting on Sanyo), Televideo 1605 and others.

\$79.99

plus \$4.00 shipping & handling

Send check, money order, M/C, or VISA numbers to:



**ADVANCED
SOFTWARE
TECHNOLOGIES**

452 W 47TH STREET, SUITE 1B
NEW YORK, NY 10036
212-247-0150

†† MS-DOS, CP/M, H-DOS, and TRS-DOS are trademarks of Microsoft, Digital Research, Zenith, and Tandy Corps, respectively.



Programming Video Modes

Mark J. Foster
Senior Systems Engineer
Systems Software Engineering
Zenith Data Systems Corporation

Hello again! For the past two months, I've been promising a discussion of programming video modes on the Z-100 PCs, so this column will be devoted to just that. The prime focus will be on using assembly language to change the video and scroll modes.

Getting Started

To begin with, the next few columns will be largely devoted to assembly language programs, so if you would like to try out some of the ideas shown here, you'll need access to an assembler. If you plan on doing a lot of assembly language work, you ought to order Zenith's MS-DOS Version 2 Programmer's Utility Package (PUP), which includes the Microsoft Macro-Assembler (MASM), along with a full-screen editor for both the Z-100 and the Z-150/160 (BSE). In addition, this package contains a number of useful programming utilities like binary-to-hex converters, directory utilities, etc. On top of that, the package includes documentation on the software interface to both the Z-100 PCs (the Z-150 and Z-160), as well as information on programming with MS-DOS. This package was created after many of the engineers at Zenith decided they needed better software tools, which they then created for themselves. After these tools started to get used more often, Zenith decided to sell them as a package so that end users could take advantage of them, as well.

While the PUP is really very useful, you may just want a simple tool to play with assembly in order to get the feel of development work. If so, you can use the DEBUG debugger which is supplied with the Z-150 and Z-160. It contains a line-by-line assembler which can be used to create simple programs. To learn more about DEBUG, read the section on DEBUG in the MS-DOS 2 manual which came with your machine.

Setting the Video and Scroll Modes

Prior installments of this column described the video modes which are available in the Z-150 and Z-160, so I won't elaborate on them all over again. The following table summarizes the video modes which are available:

Modes 0 & 1:	40 Character by 25 Line Text
Modes 2 & 3:	80 Character by 25 Line Text
Modes 4 & 5:	40 Character by 25 Line Graphics

Mode 6:	80 Character by 25 Line Graphics
Mode 7:	80 Character by 25 Line Display on the IBM Monochrome Display Adapter.

To change the video mode, only three instructions are needed. The first two instructions are used to tell the ROM (Read-Only-Memory) in the computer what video mode you want to set, and the third actually calls the ROM to perform the operation. This "transfer of control" to the ROM is performed by an "INT 10H" instruction. In general, virtually all of the facilities provided by the ROM are accessed via "software interrupts" (which just means that the computer executes an INT instruction). Prior to executing this instruction, your program will load the CPU's registers with information related to the operation you want to perform. After the software interrupt has been executed, any information passed back from the ROM to your program will have been placed in the CPU registers by the ROM. Therefore, the CPU registers are used to communicate information back and forth between programs you write, and the ROM's Input/Output drivers.

In this case, we want to tell the computer to change video modes under program control (previous installments of this column discussed changing the video mode from the keyboard). In order to set the video mode, the ROM actually needs to know two different things: first, that you want to set the video mode (since the INT 10 software interrupt actually performs multiple functions), and the second is what video mode you want to change to. As it happens, the function code to set the video mode is 0, and the ROM accepts the function code in CPU register AH. Therefore, the assembly language instruction "MOV AH,0" will satisfy this requirement. The second piece of information needed by the ROM is the video mode you want to set up, which is passed in CPU register AL.

As an example, say you want to set Video Mode 6, high-resolution graphics mode. To do this, the video mode number is 6, so the instruction needed will be "MOV AL,6". Simply combine the instructions, and you've got a program to change the video mode:

```
MOV AH,0 ;Set the function code to 'Set-Video-Mode'
MOV AL,6 ;Select video mode 6 — high-res graphics
```

```
INT 10H ;Execute the video code
RET ;Finished with the program
```

The RET instruction at the end of the listing is used to transfer control from your program back to the operating system (or DEBUG) once it is finished. If you save this program as a .COM file, (for example, HIGHRES.COM), then you can execute it like any other program by typing HIGHRES at the operating system prompt. Feel free to experiment with other video modes — to do so, just replace the “6” with any number from 0-6.

Note that if you use DEBUG to type this program in, don't type “10H” — just type “10” instead, since DEBUG automatically assumes that all numbers are entered in hexadecimal. The process of making a .COM file from DEBUG is fairly straightforward. First, enter DEBUG (by typing DEBUG). Then type “A100” to begin assembling a program at location 100H. Then type the instructions you want to assemble, and then enter a blank line to tell DEBUG that you've entered all of the instructions. After you are done, change register CX to match the length of the program you have assembled. Huh? As you enter each assembly language instruction, DEBUG will show you the start address at which the object code for the instruction will be placed. In this case, the blank line will have a “0107” displayed at the start of the line (following the colon — “:” — ignore the numbers before the colon). The 0107 represents the length of the program plus the starting address, which is 100H. So, subtract 100H to get the actual length of the program, which is: 0107H-0100H = 7H, or just plain 7. To set the length in CX, type “RCX”, and then enter a 7. Next enter the name of the program with the Name command, as in “N HIGHRES.COM”, and then “W” to write the file. To exit back to the operating system, type “Q” to quit.

Setting the SCROLL Modes

The Z-150 and Z-160 also allow you to define the scroll mode, as mentioned before. The scroll modes which are available are:

- Scroll Mode 0 - Software (compatible) scrolling
- Scroll Mode 1 - Hardware (fast) scrolling (usable in all video modes except 0 and 1)
- Scroll Mode 2 - Smooth scrolling (usable only in graphics video modes 4, 5 and 6)


To set a scroll mode, AH is set to 100 decimal, or 64H (without the “H” from DEBUG), and AL is set to the scrolling mode. Thus, you can set high-resolution graphics mode with smooth scrolling with the following program:

```
MOV AH,0 ;Set the function code to 'Set-Video-Mode'
MOV AL,6 ;Select video mode 6 — high-res graphics
INT 10H ;Set the video mode
MOV AH,64H ;Set the function code to 'Set-Scroll-Mode'
MOV AL,2 ;Select scroll mode 2 — smooth scrolling
INT 10H ;Set smooth-scroll mode
RET ;Finished with the program
```

Next Month

As many of you have noticed, I'm starting with the basics of using the machine. As this column progresses, I'll move on to more advanced areas; eventually, I'd like to cover some advanced concepts, such as device drivers for intercepting ROM Input/Output calls, etc. If you have a particular area you would like to see covered, or if you have some comments and suggestions, please drop me a line in care of HUG. Till next month, Have Fun!





Secured Computer Systems

HEATH/ZENITH 88, 89, 90 PERIPHERALS

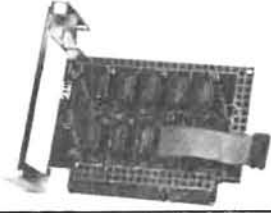
16K RAM EXPANSION CARD

Expand your H/Z 88, 89 RAM Memory to a FULL 64K and begin using larger and more powerful programs with our 16K RAM card.

Fully compatible with: Magnolia Microsystems and CDR CP/M and disk I/O interface cards

Featuring: Complete installation instructions • Mounting bracket • 90 day warranty
Field reliability record exceeding 3 years

Only \$65.00 Shipping & Handling \$5.00



PORT SERIAL CARD I/O

2 PORT PARALLEL

not your typical vanilla-flavored serial and parallel interface

Your H/Z 88, 89, 90 can now directly connect and operate EPSON 10S, ANADEX, GEMINI, SILVER REED, NEC, SUPER 5, PROWRITER, OKIDATA, and many more line printers using CENTRONICS style parallel interface with our 2/3rds, 2 port serial, 3 port parallel interface card. Or you may use all 24 digital lines in various configurations of input, output or bidirectional modes for industrial control or data sampling

Features:

- 2 Serial Ports Supporting Ring Input and External Clock
- 3 Parallel Ports
- 24 Total Digital Input/Output Lines
- Fully Compatible with All Models of H/Z 88, 89, 90 using Heath/Zenith CP/M or HDOS
- Now Supporting CP/M Version 2.2.04
- Choice of Centronics Line Printer Support Software for the CP/M or HDOS Operating Systems
- Reduced Computer Bus Loading and Chip Independent Design

Complete with installation instructions, documentation, 90 Day Warranty, Two Serial Cables and a Parallel Cable internal to the Computer

Price \$199.00 Second Operating System Driver \$25.00
Shipping & Handling \$10.00

REAL TIME CLOCK

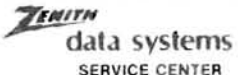
You will be able to perform time and date stamping for point of sales software, and bulletin board software or perform time studies as well as real time data sampling with our REAL TIME CLOCK. This peripheral card is a perfect companion to our 2/3rds card for industrial control and data sampling. STOP WATCH time study and alarm demonstration software is included for either the CP/M or HDOS operating systems. You will be able to view the current date and time on screen continuously or simply listen to an audible beep every fifteen minutes and the hour chimed or disable the clock entirely at your option.

Features:

- True I/O Addressing, Not Memory Mapped
- User Selectable Address
- Rechargeable Battery Backup Using Commonly Available Batteries
- Installable on the Left or Right Side of the Computer (Left side operation requires our I/O expansion module)
- Month, Day, Year, Hours, Minutes, Seconds, 1/10's, 1/100's and 1/1000's of Second Accuracy
- Interrupt Capability Based on Tenths of Seconds, Seconds, Minutes, Hour, Day, Week or a Specific Date and Time
- Choice of CP/M or HDOS Operating System Software Driver and Demonstration Programs

Price \$105.00 with Batteries \$89.00 without Batteries
\$ 25.00 Software for Second Operating System
Shipping & Handling \$5.00

HDOS is a reg. trademark of the Heath Co. CP/M is a reg. trademark of Digital Research
 PRICES ARE LESS SHIPPING AND TAX IF RESIDENT OF CALIFORNIA
 MAIL ORDER 12011 ACLARE ST. 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 — VISA AND MASTER CARD GLADLY ACCEPTED



Speed Mods For The Z-100



Pat Swayne
HUG Software Engineer

This article presents two options for making your Z-100 computer run faster. One is a commercial modification that you can purchase and install very easily, and the other is a "do it yourself" approach.

The CDR Z-100 Speed Module

The ZS100 Speed Module, by Controlled Data Recording Systems, is a small printed circuit board that you can easily install in your computer. It allows you to operate your Z-100 computer at either the standard speed of 5 MHz or at 7.5 MHz, a 50 percent increase. It does this by taking advantage of the ability of the 8284 clock generator chip used in the Z-100 to accept input either from its own built-in oscillator or an external oscillator. Normally, the internal oscillator is used. It runs at 15 MHz (hence, the 15 MHz crystal near the chip, which is U236), and is divided in the chip to produce the 5 MHz clock. If pin 13 on the chip, which is normally grounded, is brought high (5 volts), the 8284 will take input from 14 instead of the internal oscillator. The ZS100 mod provides a switch to change the state of pin 13, and a 22.5 MHz oscillator that is connected to pin 14 after the mod is installed. That oscillator, like the internal one, is divided by 3 by the 8284, and a 7.5 MHz clock results.

The ZS100 is installed by removing the 8284 from its socket at U236 and plugging it into the ZS100 board. Then the board itself is plugged into the U236 socket. The switch is mounted on a small metal plate, and is connected by a twin lead wire that is long enough to permit it to be mounted in one of the unused DB-25 connector holes in the back of the computer.

The manual provided with the ZS100 provides step-by-step installation instructions for both the low profile and all-in-one Z-100 models. In the event that the computer cannot run at 7.5 MHz (they say that only a few can't), a trouble shooting section suggests chips to change that could be too slow.

CDR was curious about whether the ZS100 would work in an ET-100 trainer (which, when fully expanded, works like a Z-100, but without the 8-bit side). They provided one for me to try in my ET-100, and I found that it could be installed if I plugged two empty low profile sockets into the 8284 socket (which is U201 on the ET-100) before plugging in the ZS100. However, I could not get my ET-100 to operate at 7.5 MHz. I tried some of the things suggested in their manual, but was unsuccessful. Later, at the Western Regional HUG Conference, the CDR people told me that if the 8088 chip in my ET-100 was made by NEC, that was probably the trouble. It was, so I replaced the chip with an Intel 8088, and that nearly did the trick. The ET-100 now ran at 7.5 MHz, but it would occasionally reset itself, as if I had pressed CTRL-RESET. I suspect

that the trouble might have had something to do with the processor in the keyboard, because the symptom was not covered in the CDR manual, and because the processor in the detached keyboard of the ET-100 is a factor that would not affect a Z-100.

I decided to try a compromise to get my ET-100 running faster. A 19.7676 MHz oscillator is available from the Heath parts dept. as part no. 150-139, and is the same type of oscillator as the 22.5 MHz in the ZS100. When you divide that by 3, you get a 6.5892 MHz clock. I installed the new oscillator in the ZS100 and it worked. Although it provides only a modest increase in clock speed, the result is still noticeable. On a Z-100 or ET-100, any increase in clock speed not only affects processing time, it also affects screen output time, since screen output is handled by the processor. The speed increase is noticeable in just about everything you run.

Since I now had my ET-100 working faster, I could not stand to have my office Z-100 poking along at the original speed, so I decided to do my own home made version of the speed mod to it. Figure 1 is a drawing of the modification I made. You can perform this modification yourself, but I would recommend it only for those who are capable of doing delicate electronic work.

To make the mod, you will need a 4700 ohm resistor, a small switch, and an oscillator, part no. 150-139. Remove all components of your computer that are necessary to access the main PC board. The oscillator can be mounted upside down near U236 with double stick tape, or you can just allow it to be held in place by the wires that will be attached to it. You should put some regular tape on the back to insulate it if you do not use double stick tape. Remove the 8284 at U236 and carefully bend pins 13 and 14 out so that they will not be in the socket when the IC is reinstalled. Cut 3 two-inch lengths and 2 6-inch lengths of #28 or #30 wire-wrap wire with 1/8-inch of insulation removed from each end. Solder one end of each of the two-inch lengths to pins 9, 14, and 18 of the 8284 chip. The wires connected to pins 9 and 18 should be carefully soldered to the top of the pins near the bend in them, so that the connections do not inhibit reinstallation of the IC. Solder one end of one 6-inch wire to pin 13, then replace the IC.

With the oscillator in place as shown in the drawing, connect the wire from pin 14 to oscillator pin 3, the wire from pin 18 to pin 4, and the wire from pin 9 to pin 2. Prepare the 4700 ohm resistor as the drawing shows, and connect it from pin 13 of the 8284 to pin 4 of the oscillator. Route the 6-inch wire from pin 13 through a ventilation slot in the computer case and connect it to one of the

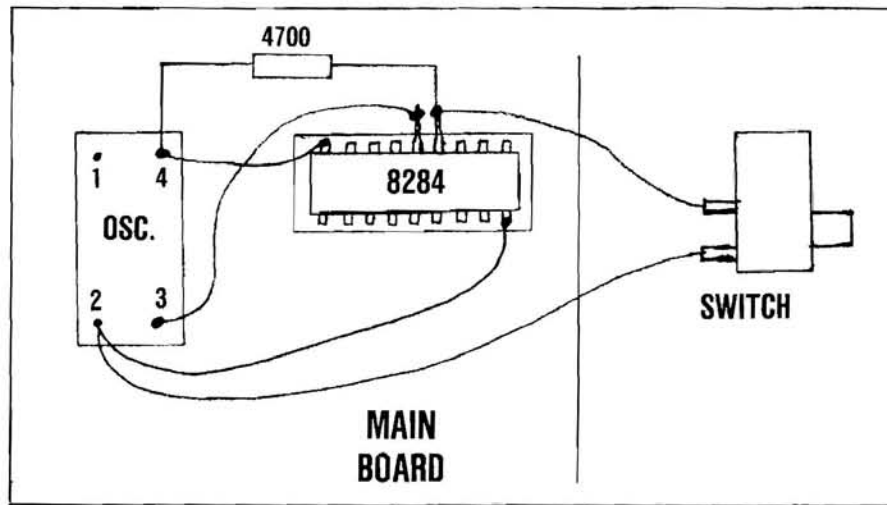
switch pins. Connect the other 6-inch wire from oscillator pin 2, through the ventilation slot, to the other switch pin. If you want your switch to be somewhere else besides hanging out the side of the computer, you may want to use longer wires in place of the 6-inch ones, and plan a route for them before cutting and installing them.

Now, you can put your computer back together and try the mod. You should try it at 5 MHz first (switch closed), to make sure that everything is OK. Then open the switch and reset the computer. If the hand prompt comes on, then everything else should be OK. If you have version 2.3 or 2.5 of the monitor in your computer, you can use the C (Color bar) command to get an immediate indication of the effect of the speed change. Hold down the

C key so that the bars are redrawn repeatedly, and you can see that they are drawn faster at the higher speed.

When you change speeds, you should only do it at the monitor level (reset the computer afterwards) or when the computer is off. Do not try to change speeds while you are running a program.

If you don't feel up to doing the homemade modification and want to try the faster speed, get the ZS100 from CDR Systems, 7210 Clairemont Mesa Blvd., San Diego, CA 92111, ph. (619) 560-1272. It sells for \$49.95, and is warranted for two years. CDR also sells faster versions of the parts most likely to be the problem if your computer will not run at the faster speed.



International HUG Conference

*** **IV** ***

August 9, 10 and 11, 1985

O'Hare Hyatt Regency
Chicago, Illinois

It's time to plan your visit with other members of the Heath/Zenith Users' Group. This year, we are returning to the newly remodeled O'Hare Hyatt Regency, Chicago for HUGCON IV.

*More exhibits, more speakers and **more room** highlight the Conference activities for 1985. The Official Conference Registration form will appear in next month's issue of **REMark**, the official magazine for users of Heath/Zenith computer products.*

Plan to attend now! We sure hope to see you there!

The FORTRAN Formula-3

Dick Stanley
P.O. Box 9512
Alexandria, VA 22304

So far in this series, we have learned how to talk to the computer using FORTRAN, and how to use the compiler and linker to produce working programs. In this article, we will actually produce a program to read in stock market data and verify the input!

There is a two to three month lag between when these articles are written to the time they appear in print, so I can only now acknowledge with thanks the many letters received from readers. Your suggestions and comments are appreciated; many of them will find their way into the series. One comment deserves immediate mention: I use a CP/M system, and so refer in the series to COM files. For those of you with MS-DOS systems, substitute EXE for COM, and everything will be correct.

Stock Market Technical Basics

To help us build our stock market analysis I/O system, let's take a look at what we want the computer to figure out for us. Let me emphasize at the outset that this is not a "get rich quick" article; the thrust of this project is to learn FORTRAN, and the vehicle for doing that is a technical analysis system. There are as many opinions on how to play the stock market as there are players; all I can promise about this analysis system is that it will accurately calculate and display the things we ask it for.

Technical stock analysts are those who believe a stock's future performance can best be predicted by analyzing its previous performance. Said another way, the present price of the stock represents the value of everything known about the stock at the moment. They discount the importance of financial fundamentals such as profit ratios, liquidity, etc. on the grounds that the effects of changes in these things are already reflected in the price of the stock. A market index is an average of individual stock prices, so technical analysis can be applied to the stock market as a whole, as well as to individual stocks. I am not arguing the merits of either approach to the stock market — I have chosen the technical approach for this project because the data needed as inputs are more readily available, being typically found in the business pages of the daily paper.

There are several measurements that technicians usually use as tools to evaluate the market. Because it is the largest, we will be working with data from the New York Stock Exchange. Some routinely reported market statistics are listed below. In parentheses after each one is listed the variable name that will be used to refer to it in our FORTRAN program.

Market Index: Indexes measure share prices. If you could buy a share of the market index, this is how much you would pay for it. There are several, the most widely known of which is the Dow Jones Industrial Average. The Dow Industrials have been shown

to be representative of the trend of the overall market in the long run, so we will use this as our market price indicator. (DJIA)

Market Volume: The total number of shares traded that day on the NYSE. (VOLUME)

Advances: The number of stock issues on the NYSE whose price closed higher than yesterday. (ADVNC)

Declines: The number of stock issues on the NYSE whose price closed lower than yesterday. (DECLN)

Issues Unchanged: The number of stock issues on the NYSE whose price closed at the same level as yesterday. (UNCHNG)

Advancing Volume: The number of shares traded in issues whose price rose during the day. (UPVOL)

Declining Volume: The number of shares traded in issues whose price fell during the day. (DNVOL)

New highs: The number of stocks reaching a higher price today than at any time in the past 52 weeks. (HIGHS)

New lows: The number of stocks reaching a lower price today than at any time in the past 52 weeks. (LOWS)

The premise of technical analysis is that these figures are indicative of forces at work in the market that can be determined by applying these values in analytical relationships. The primary study tools that we shall use are described below, together with the variable name given to each, where applicable. When we are done, we will derive a single index, the Stock Traders' Index (STI), from the conditions that these analyses reveal. This approach is adapted from one by Eleanor Hess in "The Stock Traders' Index", published by Investors Intelligence, Larchmont, NY 10538 in their "Encyclopedia of Stock Market Techniques."

Moving Averages: A moving average is an average comprised of the most recent values of the value being measured. A 5-day moving average is formed by adding the values for today and the four days preceding (total, 5 days) and dividing by five. Tomorrow, we add the value for tomorrow and the 4 days before it, and divide by 5. A twenty-day moving average would involve the values for 20 days, and so on. A moving average has the effect of "smoothing" data. The longer the duration of the moving average, the less the effect of any one day's value on the average, and the more it reflects long-term trends. In stock market analysis, it is customary to use moving averages of 5 days, 20 days, and 200 days, which relate to a trading week, a month, and ten months, respectively. We shall use several moving averages in our model, and they will be introduced as they are required.

The Advance/Decline line. Technicians believe the health and breadth of the market is measured by how many stocks are going up versus how many are going down. This is usually measured as a cumulative total over time which begins at any arbitrary value. The absolute value is unimportant, the trend is all-important. Any two advance/decline lines will coincide on any given date if one merely adjusts for the difference in their starting values. For simplicity, we will begin our line at zero. The advance-decline line today [ADLINE(TODAY)] is computed as follows:

$$ADLINE(TODAY) = ADLINE(YESTERDAY) + ADVNC - DECLN$$

A market trend indicator can be constructed by measuring the daily percentage change in the Dow Jones Industrial Average (DJIPCT) and in the Advance/Decline line (ADPCT). These are calculated as follows (we have not multiplied the result by 100 to form a "true" percentage, as the 100 is merely a scale factor that is unneeded here):

$$DJIPCT = \frac{DJIA(TODAY) - DJIA(YESTERDAY)}{DJIA(YESTERDAY)}$$

$$ADPCT = \frac{ADVNC - DECLN}{ADVNC + DECLN + UNCHNG}$$

Measures of buying and selling pressure are derived from the volume figures. It is presumed that in a perfect market, as much volume would take place among rising stocks as in falling stocks. In practice, this rarely happens. When the volume on advancing issues moves from above to below half of the total daily volume, this is seen as a "sell" signal, as the sellers are apparently turning into buyers. A "buy" signal would be registered when the volume on declining issues, having been above half the total volume, drops to less than half of the total volume. The advancing and declining volumes (ADVVOL and DECVOL) are calculated as follows:

$$ADVVOL = UPVOL - (VOLUME/2)$$

$$DECVOL = DNVOL - (VOLUME/2)$$

The Short-Term Trader's Index is another favorite measure. It measures the relative volume behind up and down stocks. Named TRIN (TRader's INdex), it is computed using the following formula:

$$TRIN = \frac{ADVNC / DECLN}{UPVOL / DNVOL}$$

When TRIN is greater than 1.00, the average declining stock is getting more volume than the average advancing stock. If TRIN is less than 1.00, the average advancing stock is getting more volume than the average declining stock. In theory, the market should move in the direction of the volume (i.e. it should advance if $TRIN < 1$ and decline if $TRIN > 1$). Obviously, if it were that simple, Wall Street would be filled with millionaires armed only with pocket calculators. Nonetheless, TRIN and its trends as revealed by moving average analysis can be useful for sensing the market.

Finally, we have the daily difference between the number of new highs and the number of new lows. This is seen by many as a sensitive indicator of market momentum, i.e. whether a move is running out of steam. The high-low differential (HLDIF) is calculated by:

$$HLDIF = HIGHS - LOWS$$

None of the above calculations is particularly complex. However, when each is involved in one or more moving averages of varying durations, and then charts must be plotted to display the results so they can be analyzed, you are talking about a good bit of work and not a little time to do it. There are many calculations to be done repetitively, and that is why speed in processing is important. Let's let our Heath/Zenith number crunchers take some of the sting out of that work with FORTRAN.

Analyzing The Variables

The first step in any program design project is to identify the necessary inputs and outputs. We have just done that in the paragraphs above. The next step is to analyze these quantities to ensure that we allow enough space for them in the I/O formats, and that we operate on them correctly in the program. Things we need to know include: maximum and minimum magnitude, sign, type (integer, real, logical, etc.), precision (how many decimal places and significant digits do we want), and — for FORTRAN — what FORMAT specification shall we use? I find that a good way to do this analysis is with a matrix, so that things don't get forgotten. A matrix of the input variables for our program might look like this:

Variable	Max	Min	Sign	Type	Sig		FORMAT
					Dec	Digits	
DJIA	9999.99	0	+	Real	2	6	F8.2
VOLUME	999,999,000	0	+	Integer	0	6	I10 *
ADVNC	2500	0	+	Integer	0	4	I5
DECLN	2500	0	+	Integer	0	4	I5
UNCHNG	2500	0	+	Integer	0	4	I5
UPVOL	999,999,000	0	+	Integer	0	6	I10 *
DNVOL	999,999,000	0	+	Integer	0	6	I10 *
HIGHS	2500	0	+	Integer	0	4	I5
LOWS	2500	0	+	Integer	0	4	I5

* commas are not entered as part of the data, so they don't take up spaces in the FORMAT specification.

The Fine Points of FORMAT

In the previous article, we discussed Real and Integer numbers. Now would probably be a good time to turn back to that discussion to refresh your memory if you aren't sure which is which. If you scan the FORMAT specifications above, you will notice that we have used the minimum size field possible, remembering that the sign of the number, even if it is not supplied specifically, occupies a space. While we have correctly specified the size of the FORMAT fields needed, however, there remains a problem. If we merely write our program using the data as shown, it will not run.

Because of the way in which integers are represented internally, they have only 5 significant digits, and range from -32768 to +32767 in value. Clearly, the volume quantities for which we have chosen the I10 format lie outside both those restrictions. There are two ways around this: we can choose another representation for the volume data, such as Real, or we can find a way to show them as integers in their true range. As they really are integers (you cannot buy or sell a fractional share of stock), we should represent them as such, if possible. Is there a way?

Yes, there is. FORTRAN has a variable type called Extended Integer, which allows representation of integers having 9+ significant digits within the range of values from -2,147,483,648 to +2,147,483,647. Clearly, our data does fit within these limits. To

inform the program that these variables are of the Extended Integer type, we must declare them so early in the source listing. We will do this by writing the statement:

```
INTEGER*4 VOLUME, UPVOL, DNVOL
```

Are there other precision and size problems? As it happens, there are not. FORTRAN allows Real variables with precision up to 7 significant digits within the range of $10^{**(-38)}$ to 10^{**38} ($**$ in FORTRAN represents exponentiation, rather than the XX used in BASIC. The range shown is read "Ten to the -38th power to ten to the 38th power."). Had we needed more significant digits, there is also a variable type called Double Precision which permits Real variables with up to 16 significant digits. A review of the matrix shows that we do not need such precision in this application.

Dealing With Dates

We are almost ready to start coding, but there is a critical variable we have not discussed: the date. Without tagging each line of data with the date to which it applies, it becomes just so much gibberish. That sounds logical, but is a date Real or Integer?

We could write the date as either Real or Integer, in fact, but we are accustomed to seeing it written as a string in a form we readily understand, like 12/25/84, or 7/4/76, or maybe 6 May 82. Rather than getting involved in coding to make the date look like something it is not, let's invoke Stanley's First Principle of Data Processing: "Any computer output that must be read with a ruler, red pencil, and decoding sheet is no darn good!" Let's make the date meaningful and useful to us — it's our project!

I admit to prejudice in favor of a format like 15 Aug 77, because it is fixed in length and thus palatable to the computer, and because everyone understands which is the month, day and year. It is not uncommon to find a date written as 7/5/80 being interpreted as July 5, 1980 in much of the U.S., and as May 7, 1980 elsewhere. However, it turns out that this is not a good format to use for an introductory project in FORTRAN, because the handling of string data in FORTRAN is not one of its strong points. Also, sorting is facilitated if all the parts of the date are represented by numerals, and if each component (day, month, year) is kept separate, or is separable, by the program. To do this, we will enter dates to this program in the format MM/DD/YY, where

MM = number of the month

DD = day of month

YY = last two digits of the year

We are all familiar with this format, and it is widely used in other programs, such as dBASE II. This is also the format used by most databanks, such as Dow Jones News Retrieval. It makes sense and

is consistent, so we will be all right.

OK, but how do we tell the FORTRAN compiler about our date? The FORTRAN FORMAT statement is a very powerful tool; it has the ability to ignore things as it scans the input. If we use the "nX" descriptor in the FORMAT specification, it will cause the computer to ignore the n characters referred to by the specification. How is this useful?

We indicated that it would be nice to keep month, day, and year separate to facilitate sorting and reference. Suppose we are interested in the date October 17, 1981. Using our format for input, we will write it 10/17/81. Now let's scan it for input with the FORMAT specification shown below.

```
      Date: 10/17/81
10  FORMAT (I2,1X,I2,1X,I2)
      READ (1,10,MO,DAY,YR)
```

The result? We will find that MO=10, DAY=17, and YR=81. We must be a little careful with single-digit inputs. If we enter 7/5/80, we will find that MO=70, DAY=5, and YR=80, clearly not what we meant!

We are almost there. There is one other feature of FORTRAN that now becomes significant: typing of variables. You may remember in BASIC that you could declare a variable to be of Integer type by appending a % to the variable name. Thus, in BASIC, MO is a Real variable, and MO% is an Integer. FORTRAN uses a somewhat different system. Variable names beginning with the letters I, J, K, L, M, or N are Integer variables; all the rest are Real variables. While this is convenient in classical mathematics (where typical integers are represented by the letters I through M) it can be a nuisance in our applications. In the example above, for instance, DAY and YR are Real, but MO is Integer. We could change the month's variable name to one that looked like a Real variable name, but then it wouldn't look like "month." We can also, however, explicitly declare variables to be of a certain type. We do this by naming the type and then the variables. In the example of the date's components, all are of type Integer. We tell the program this by putting in the statement:

```
INTEGER MO, DAY, YR
```

We have redundantly declared MO to be Integer (it already was because its name begins with M), but that does no harm. We have also defined DAY and YR as Integers with this statement. We have not redefined anything else; if there were a variable named YR1, it would be of type Real.

Does it work? Try the program named DATE, found at Figure 1. A copy of the screen made during its compilation and running is at Figure 2. True enough, the above scheme really does work.

```
C      Program DATE
C      This program demonstrates the X format descriptor and the
C      explicit declaration of variables
C
10     FORMAT(' Enter the date as MM/DD/YY:  ')
11     FORMAT(I2,1X,I2,1X,I2)
20     FORMAT(' The date is: ',I2,'/',I2,'/',I2, ' (American Format)')
21     FORMAT(' The date is: ',I2,'/',I2,'/',I2, ' (European Format)')
      INTEGER MO, DAY, YR
      WRITE (1,10)
      READ (1,11) MO, DAY, YR
      WRITE (1,20) MO, DAY, YR
      WRITE (1,21) DAY, MO, YR
      END
```

Figure 1. Source listing of Program DATE.

```

A>F80 =B:DATE
$MAIN

A>L80 B:DATE,B:DATE/N/G

Link-80 3 4 01-Dec-80 Copyright 1979,80 (C) Microsoft

Data 0103 1A9C < 6553>

36899 Bytes Free
[0116 1A9C 26]
[Begin execution]

Enter the date as MM/DD/YY: 11/26/84

The date is: 11/26/84 (American Format)
The date is: 26/11/84 (European Format)

```

Figure 2. Copy of screen during compilation and running of Program DATE on an H89.

Let's DO A Little FORTRAN

It's programming time again! We are going to prepare a program segment to read these data into the computer so that in future articles we can construct the technical indicators discussed above. As you can see from the above program, the screen gets cluttered. It would be helpful to start with a clean screen and it would look more professional. A simple, "brute-force" way to clear the screen is to print 24 blank lines, and the easiest way to do that would be to use a loop.

FORTRAN does indeed have a loop capability. It takes the form

```
DO k i = m1, m2, m3
```

where:

k is the number of the last statement in the loop. It must be executable (e.g. not a FORMAT statement), and may not be another DO statement. (There are other restrictions, but they do not affect us now).

i is an integer variable that controls the loop, called the loop index. It must be positive and cannot be modified by any statement within the loop. It is available for use within and without the loop (that is, you can use the value of **i** in equations).

m1 is an integer which gives the initial value of **i**.

m2 is an integer which gives the final value of **i**.

m3 is an integer which specifies the counting interval. If it is 1, it may be omitted.

The DO loop functions just like the FOR..NEXT loop in BASIC. A loop to print 24 blank lines, and thereby clear our screen, looks like this:

```
100 FORMAT (1X)
    DO 200 J=1,24
200 WRITE (1,100)
```

When a DO loop is completed, program control passes to the statement following the last statement in the loop, just as with a FOR..NEXT loop. While in the loop, the value of the index is available, but the index value cannot be changed. You could insert a statement in the loop above, for instance, that reads $M = J + 3$, but not one that reads $J = M + 3$.

Putting It All Together

To conclude this session, let's use what we have learned to write

the program segment needed to read in stock market data. To be certain that the machine has gotten the correct data, we'll echo back what was input on the right-hand side of the screen, so you can verify it. This is called data validation, and is an important step in making certain you have given the system the proper information with which to work. We'll call our input routine STOXIN; it will make use of the DATE program and the screen clearing routine above. Enter the source code with your text editor (Non-document mode in WordStar, remember), and then compile and run the result. A copy of my session is at Figure 4 for comparison.

```

C      PROGRAM STOXIN
C      This program inputs stock market data for analysis
C
C      Set up FORMAT specifications for I/O
100  FORMAT(1X)
110  FORMAT(19X, 'Stock Market Technical Analysis System')
111  FORMAT(' Enter the date as MM/DD/YY:  ')
112  FORMAT(I2,1X,I2,1X,I2)
113  FORMAT(' Enter the data indicated:')
114  FORMAT(' Closing Dow Jones Industrial Average:  ')
115  FORMAT(F8.2)
116  FORMAT(' Volume of shares:  ')
117  FORMAT(I10)
118  FORMAT(' Advancing Volume:  ')
119  FORMAT(' Declining Volume:  ')
120  FORMAT(' Number of Issues Advancing:  ')
121  FORMAT(I5)
122  FORMAT(' Number of Issues Declining:  ')
123  FORMAT(' Number of Issues Unchanged:  ')
124  FORMAT(' Number of New Highs:  ')
125  FORMAT(' Number of New Lows:  ')
126  FORMAT('+',65X,I2,'/',',I2,'/',',I2)
127  FORMAT('+',65X,F8.2)
128  FORMAT('+',63X,I10)
129  FORMAT('+',68X,I5)
C
C      Declare variables
INTEGER  MO, DAY, YR, ADVNC, DECLN, UNCHNG, HIGHS, LOWS
INTEGER*4 VOLUME, UPVOL, DNVOL
C
C      Clear screen
DO 200 J=1,24
200  WRITE(1,100)
C
C      Get data, print value input at right side of screen
WRITE(1,110)
WRITE(1,111)
READ(1,112) MO, DAY, YR
WRITE(1,126) MO, DAY, YR
WRITE(1,113)
WRITE(1,114)
READ(1,115) DJIA
WRITE(1,127) DJIA
WRITE(1,116)
READ(1,117) VOLUME
WRITE(1,128) VOLUME
WRITE(1,118)
READ(1,117) UPVOL
WRITE(1,128) UPVOL
WRITE(1,119)
READ(1,117) DNVOL
WRITE(1,128) DNVOL
WRITE(1,120)
READ(1,121) ADVNC
WRITE(1,129) ADVNC
WRITE(1,122)
READ(1,121) DECLN
WRITE(1,129) DECLN
WRITE(1,123)
READ(1,121) UNCHNG
WRITE(1,129) UNCHNG
WRITE(1,124)
READ(1,121) HIGHS
WRITE(1,129) HIGHS

```



```

WRITE(1,125)
READ(1,121) LOWS
WRITE(1,129) LOWS
END

```

Figure 3. Source code listing for program STOXIN.

```

A>F80 =B:STOXIN
$MAIN

A>L80 B:STOXIN,B:STOXIN/N/E

Link-80 3.4 01-Dec-80 Copyright 1979,80 (C) Microsoft

Data 0103 1E0D < 7434>

36018 Bytes Free
[012E 1E0D 30]

A>B:STOXIN

[The screen cleared here. The 24 blank lines are omitted.]

      Stock Market Technical Analysis System
Enter the date as MM/DD/YY: 12/12/84

      12/12/84

Enter the data indicated:
Closing Dow Jones Industrial Average: 1175.13

      1175.13

Volume of shares: 79408000

      79408000

Advancing Volume: 30000000

      30000000

Declining Volume: 49408000

      49408000

Number of Issues Advancing: 652

      652

Number of Issues Declining: 804

      804

Number of Issues Unchanged: 543

      543

Number of New Highs: 3

      3

Number of New Lows: 27

      27

```

Figure 4. Screen copy of compilation and run of STOXIN. The data are fanciful — some of the figures are correct, but the rest are merely test values chosen to show that it works.

If you examine some of the FORMAT statements in STOXIN.FOR, you will find that the first character specification of each is '+', and yet this does not print on the screen. The plus sign is used for "carriage control", to avoid printing excess carriage return/line feed combinations on the screen. Next time, we will go into that in detail. For now, take it on faith.

Looking at the compiler and linker commands, you will notice that both those programs support command lines. If you have only one thing to do, put it on the command line just as you would at the "*" prompt, and it will be done. The /E switch on the linker causes it to write the compiled program to disk as a .COM file and then exit to the operating system, rather than write the program and then immediately execute it. I did this here simply to demonstrate the ability to use this switch.

What Have We Learned?

We have learned in this article about basics of stock market technical analysis, and a great deal more than last time about the FORMAT specification. Representation of numbers for high precision has been discussed and demonstrated, and we have begun to work with loop constructs — known to FORTRAN

aficionados as "DO loops."

Next Time

The next article will pick up where this one leaves off. Now that we know how to input and output data, we will learn how to calculate in FORTRAN, and how to present the results of our calculations in a form that is useful and meaningful to people. Until then, experiment with STOXIN. Notice what happens when you overflow a FORMAT field, or when you enter dates with single digits at the left of the field, rather than the right. You are becoming more proficient and familiar with this new tool, and the more you learn about it, the easier it will be to use. Remember, no matter what you do with the program, even if it "blows up", no smoke will rise from your machine!



PIP—Parts Inventory Program A Fully Menu-Driven Parts Inventory Program

Automatic:

Purchase Orders + Posting of Items Received + Register
MORE! • Updates + Reports + Searches • MORE!

\$49.95

ABE DWECK PROGRAMS

PO Box 207, Carmel, New York 10512

Heath/Zenith CP/M Formats • NY residents add 6% sales tax.

Now for
CP/M too!

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 130 page users manual with 5 program disks (5-1/4") and a sample data disk.

Hardware: H8/HZ89 (64K) or HZ100 with 2 disk drives. Any Heath*, Zenith* or other printer.

Software: HDOS 2.0 and MBASIC 4.82 for HDOS, or CP/M or CP/M-85/86 (Ver. 2.2) and MBASIC 5.21 for CP/M.

Order: Complete System \$89† (specify hard or soft sector 5 1/4", HDOS or CP/M, HZ89 or HZ100). Manual alone \$21.†

Master Card/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



**Controlled
Data Recording
Systems Inc.**

**AVAILABLE NOW 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 attached to the FDC-H8 without hardware additions.



**PRICE
\$495**

Contact:

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

ANNOUNCING THE MODIFIER

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

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

MODIFY 89 is set for the following disk types:

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

* = Double sided 5.25" drive required
S.S. = single sided, D.S. = double sided, S.D. = single density, D.D. = double density

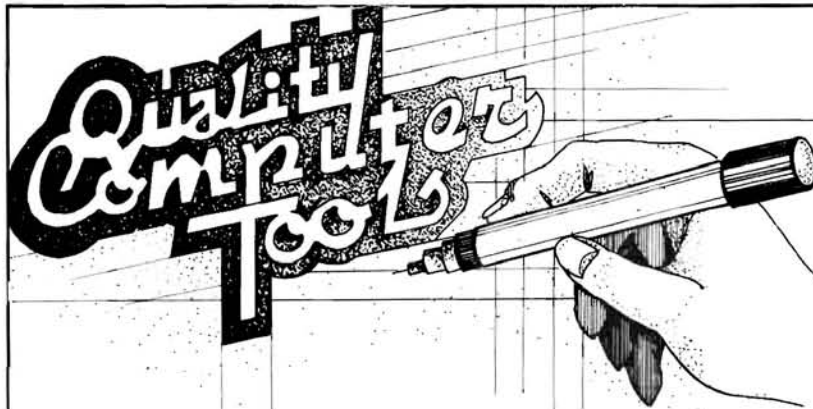
Limitations: MODIFY 89 is not a disk duplicate program. It is currently available for use with an H/Z89 or H/Z90 computer that has an FDC-880H double density 8" and 5.25" controller, using C.D.R.'s BIOS V.2.9 or with an H8 computer using the FDC-H8 by C.D.R. Systems, Inc.

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

Contact:



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



**... FOR YOUR
HEATH/ZENITH COMPUTER**

Call or Write for more information
or a **FREE** catalog of products.

- BUSINESS/FINANCIAL SOFTWARE
- UTILITIES
- ENTERTAINMENT PACKAGES
- DATABASE SOFTWARE
- BLANK DISKS & OTHER SUPPLIES

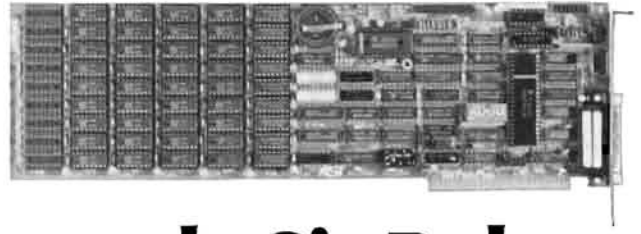
PLUS Turbo PASCAL Products for Your Computer at 15% off List!!

Our products expand with your needs....



P.O. BOX 790 DEPT. 485R
Marquette, Michigan 49855
Phone: **906-249-9801**
10 AM - 5 PM EST

Expanding The Heath/Zenith PC With The AST Research SixPak



Dale Grundon
11456 Links Drive
Reston, VA 22090

"Happiness" is never having to type the date and time at system boot. This is just one of the features you can gain by adding the AST Research, Inc. SixPakPlus to your Heath/Zenith model 150-PC.

This expansion kit appropriately follows the implications of its name. It provides six additional features to your system. Starting with a memory expansion to allow the full compliment of user-addressable memory; the board also has an RS-232C serial port, a parallel port, a real-time calendar/clock, an optional expansion for a joy-stick port, and the SuperPak software necessary to accent all of these additional features.

Although the SixPak was initially designed as an expansion board for the IBM-PC, there is no problem in adding it to the Heath/Zenith PC system; thanks to the near total compatibility retained by Zenith. Installation is quick and simple and can be done in about one hour that includes checking out some of the SuperPak software. The unit uses one of the 150's expansion slots for the main SixPak board; and if you chose to implement the parallel output, a second rear panel connector position is consumed. If you were to purchase the optional joy-stick or game port upgrade, one more of the rear connector slots would be needed.

It would be no surprise if it were announced that the staff of AST Research were former employees of Heathkit. The documentation that is provided with the kit is exceptional. Information for the immediate support of the supplied hardware and software is thorough, along with a considerable amount of information relating to interface of the system for special applications. There are two separate manuals provided. One contains information relating to the SixPak expansion board hardware and the other has extensive detail for running the support software. They even provide an audio cassette that describes the step-by-step process of hardware installation along with a 15 minute "how to" on implementing the support software at a basic level. The audio cassette assumes an IBM installation, so it has limited use when it describes some of the computer segments.

The SixPak board is supplied with 64K of RAM chips, already installed, and there are empty sockets for the addition of chips to allow 384K on the board. Since the Zenith PC can only address 620K, you would probably not add that last 64K to the SixPak. Incidentally, there is a one page sheet of comments with the Zenith supplied board that provides a few hints on installing it. One of the main points that Zenith indicates is that the Z-150

must have the main memory fully populated before adding additional memory.

Installation of the SixPak for memory expansion purposes requires the setting of dip switches to account for three factors. First, is the settings for the starting address of the SixPak memory. This can be designated at any of the 64K increments starting at 64K. With this provision it would be possible to have less than a fully populated memory on a computer. The second on-board setting of dip switches is to define the amount of memory installed on the SixPak. All of these are set on an 8 position dip switch with the last position reserved for selection of parity checking. This third option allows you to enable full parity error checking to ensure the highest possible data integrity. AST Research recommends that this feature be enabled.

You will have to remove your Z-150 CPU board during installation to configure the system to recognize the additional memory. Don't be too quick to reinstall the CPU because there is one other system configuration that you will have to make and it is the one that really puts all of that new additional memory into a practical application. The SuperPak support software provides you with a RAM disk application program, so you will have to set your system to recognize the additional memory-disks that you will want to implement.

The AST Research program, SuperDrive, uses the additional memory to simulate additional disk drives within the random access memory. Numerous options can be designated by the user when this program is initialized. The options include naming the drive, the number of sides to the drive, the number of sectors per track on the simulated drive, and system memory allocation parameters. For example, establishing a two sided drive can provide up to 360K on the simulated disk.

Using this RAM drive increases the speed of the operation of many software packages, particularly when many disk accesses are required. Using a MultiPlan spreadsheet that would take 32 seconds to load, I found the load time dropped to around 12 seconds. One important thing to remember is that the SuperDrive that you create in memory will be gone when you power down the system. So you must be sure to copy all critical data back to a real disk when ending your operating session.

The memory allocation provisions with SuperDrive allow selection of a specific quantity of memory to reserve strictly for the use of the simulated disk drive. Additional switch type command

parameters will set memory space for the application program and prevent SuperDrive from using that segment. Should your mental arithmetic be slightly in error when you load SuperDrive, the program presents a display that indicates exactly how much memory is actually set aside for your application program, the number of bytes available on the simulated disk, and the amount of memory that the memory disk has reserved for itself. Error messages are included in the program to indicate several potential error conditions.

Another program that is supplied with the SuperPak software is SuperSpool. This file spooling program frees the microprocessor for your next application program while the printer is running at its slower pace in the background. Like the SuperDrive program, command switches can be specified when the program is initialized to allocate the amount of memory you desire in the print spool buffer. Unlike many public domain print spooling programs that I have seen, output from SuperSpool can be stopped, restarted from the last page or the previous page, or aborted.

Printing this article on a 1200 baud printer would normally take around four minutes, but with a SuperSpool buffer of 16K the system was available for other functions after just 30 seconds. I tried several tests with the SuperSpool running while I loaded MultiPlan spreadsheets. I could find no decrease in the load time of the sheets because of the action of the simultaneous spooling operation.

The calendar/clock on the SixPak is a 24-hour clock with a 4-year calendar. There is no provision for a leap year. A battery back-up is installed that keeps the clock running when the computer is turned off or completely disconnected from the A/C power source. Two disk programs support the calendar/clock feature. One program is used to set the date and time of the SixPak microprocessor chip. The second program reads the date and time from the SixPak, displays both, and enters both into the operating system clock. Using an AUTOEXEC.BAT file at initial boot-up to run this program will eliminate the need to manually enter the time and date. In keeping with the quality of the documentation, the hardware manual provides all of the I/O addressing that a programmer may want to implement within a program that may need clock type of timing information.

The accuracy of this clock is well within reason. During the first two months of operation it only drifted behind by one second. Although I could find no reference in the documentation, there is a trimmer resistor that you could vary to effect a slight positive or negative change in clock speed.

Depending on the age of your Z-150, the serial port may be a valuable addition to your computer, since the more recent model 150's only have one serial port available. If you own one of the older Z-150's with two serial ports, fear not as the SixPak has on-board jumpers that permit disabling the port and eliminating any conflict. Since the SixPak would ultimately be used with a variety of devices attached to this serial port, the designers included provisions for configuring the port to meet the occasion. An RS-232C interface jumper block with adjustable jumpers allows the CTS, DSR, or DCD lines to be forced to a "true" condition, if necessary.

Unlike the serial port, the parallel port feature does not have its interface connector on the rear of the SixPak board. The rear connector is fastened to a plate that replaces a blank rear plate on the Z-150. The external rear connector is interconnected by a supplied ribbon cable. This means that it uses up one of the com-

puter's expansion slots. Again, an on-board shorting plug position is provided, should you desire to disable the parallel port. You could choose to not install the rear connector if you did not want to lose an expansion slot.

Similar to expanding the memory of the SixPak from the supplied 64K, the game port or joy-stick feature of the SixPak is a user upgrade option. The major portions of the circuitry are already on-board, but you will have to purchase and install two additional IC's and a cable. AST Research does not manufacture a joy-stick to use with this feature; however, they do provide some information essential to selecting a joy-stick device and implementing its use. Again, as with the other features of the SixPak, jumper provisions permit disabling this segment of the board.

For the dollar spent to upgrade your Z-150, the SixPak is certainly one of the better selections. Particularly from the aspect of memory expansion with the RAM drive and print spooling that AST Research provides. The built-in calendar/clock is a nice feature as a bonus. The serial and parallel ports may end up as an unnecessary tag-along on some systems, since Zenith continued their earlier practice of including a serial and parallel port in the basic system. Comments on the HUG Bulletin Board indicated that Zenith had dropped support of the second serial port on the Z-150, because of incompatibility with some of the optional second source boards that they are selling for the system. If you were caught in this decision, then you will certainly appreciate the second serial port that the SixPak provides.

Manufacturer:

AST Research, Inc
Irvine, CA
(714) 863-1333

Purchase Information:

AST SixPakPlus multi-function card - \$299.00
AST Game Port Option - \$ 39.95
Optional memory chips (64K set) - \$ 50.00

Purchase Source:

Heath/Zenith Computers and Electronics, and other computer accessory stores

Note:

SixPakPlus, SuperPak, SuperDrive and SuperSpool are trademarks of AST Research, Inc. ✖

Plan your summer vacation
NOW!
Attend the 4th Annual
HUG International Conference
Chicago, O'Hare Hyatt Regency
August 9, 10, 11

GREAT DRIVES!

The H89 TWOET systems Just Got Hard to beat!

We've got a great idea for your H-88, 89 or 90. It's a dual internal half height drive system. Two of our half height 5 1/4" drives can replace your built-in disk drive, doubling your information storage capacity.

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

We've thoroughly tested the TWOET/Heath set-up. And now by popular demand, the new 10 megabyte hard disk system! We are now able to offer you the storage you want and need for a fair price! The hard disk system comes with all software needed to run with the 17 or 37 controller and/or the Magnolia Double Density controller.

NEW!

Model TWOET 455

2 Shugart SA-455 half height
48 tpi double sided
All hardware
Metal, shielded mounting plates
Data cable with chassis connector
Power "Y" connector
Complete documentation
Price..... \$355.00

Model TWOET 55F

Two Teac 55F half height
96 tpi double sided
All hardware
Metal, shielded mounting plates
Data cable with chassis connector
Power "Y" connector
Complete documentation
Price..... \$385.00

**HARD DISK
for
H-89
\$995.00
Internal**

SA-860 DS/DD half hgt 8"	505.00 2@ 495.00 ea.
SA-455 DS 48tpi 1/2 hgt 5.25"	140.00
FDD-100-8 SS/DD 8"	125.00
Teac 55F 96tpi DS/DD half height	155.00
FDD-200-8 DS/DD 8"	Sale 185.00
Shugart 10 MB half height hard disk	725.00
PGS, HX-12 RGB color monitor	525.00
Data connectors of all types	CALL
Power connectors for all drives	CALL
Controllers	CALL
Hard disk Z150/160	Internal 750.00

Wondering what to do with your internal drive if you buy this system?

Here's the solution. If you purchase a dual half height system for your Heath computer from Floppy Disk Services, just include an extra \$60.00 plus shipping and receive a single 5 1/4" case with power supply and data cable ready to receive your SIEMENS internal drive! the case with data cable is normally a \$80.00 item. And the cable that comes with your TWOET system includes the external chassis-disk I/O connector.

Due to production deadlines, this ad is 2 months old at time of printing. We encourage you to call for our latest pricing and catalog requests.

Dealer inquiries invited.

We accept cash, MasterCharge, Visa, personal checks (allow 10 days to clear the bank), wire transfers, money orders, purchase orders from government agencies, and approved businesses. Call for info.

Floppy Disk Services inc. has been supplying custom enclosures and disk drives to the hobby and professional market for 6 years now! We have specifically supported the Heath/Zenith community with a high degree of success. We care about you and your system integrity. That's why we have been a success in a time when many others are going out of business. We are proud to be a part of the Heath/Zenith community and will continue our support with new products and ideas.

GREAT PRICES!

SA-455

Shugart SA-455 half height 5 1/4 is the standard for excellence in the disk drive community. Backed by a one year warranty and manufactured by the leader of drive technology, Shugart Corp. Ready to ship at \$140.00 ea. Compatible with ALL of the Heath/Zenith computers.



Z-150/160 Upgrade

We can upgrade your Z-150/160 with half height floppies, hard disks, 8 inch drive systems, clock modules, memory and much more! Glance over our ad and call toll free outside NJ for our latest catalog. Or call and ask our sales/technical experts for their upgrade suggestions...



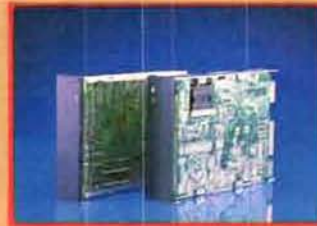
Maynard and DTC

We can supply PC compatible upgrade boards for your Z-150, Z160 system. Hard disk controllers, memory upgrade, and space saving Maynard modular boards. As an example, you can install a Maynard Modular 5 & 8 controller that will run 5 1/4, 8 inch and optional boards such as a clock/calendar module, game port, serial port or parallel port. Call for details on the slick way to upgrade and save a slot at the same time...



SA-712 & MicroScience

Half height 10 megabyte (20mb available soon) hard disk drives. The MicroScience has a plated media and is available with a full size bezel option. The Shugart has an on board low current option and is backed by a one year warranty. It also has a full size bezel option. Both drives are available for the H/Z-89, H/Z-150, H/Z-160. \$550.00 each.



Irwin-110

Data loss! Its a terrible thought. Minimize your chance of lost programs and data files with the new 110 tape drive from Floppy Disk Services inc. It works off the standard 5 1/4 controller which means you save money over standard tape backups that you must buy the controller for. The data cartridge holds 10mb of data and can retrieve in streaming or individual file form. Software included. Available in many configurations. For the IBM-PC, Z-150 or Z-160 only. (CP/M software is being worked on). Tape unit \$395 ea., tape cartridge \$29.95 each...



Connectors

Floppy Disk Services inc. can supply custom data and power cables for your needs. We use only the best! UL rated or listed components, and all power connectors are crimped by our AMP electric certified crimper. No hand crimping to vary the degree of accuracy. Only the best! Floppy Disk Services inc. can make 1 or 1000 connectors to your specs, fast! Call and ask for details...



Special Sale!

Because of our enormous buying power in the industry we have made an excellent purchase of disk drives for your Heath/Zenith computer. Two Teac 55F 80 track double sided drives, in case with power supply and shielded data cable, your price \$395.00. Single drive system, \$245.00.



Enclosures!

We manufacture and stock enclosures for every disk drive on this page and for almost any other drive you are likely to find. Available assembled and tested with state-of-the-art power supplies, these enclosures house a wide combination of half- and full-height drives to complement most available computers.



Toll Free Order Line:
(800) 223-0306

Tech Help or Info:
(609) 799-4440

39 Everett Drive Bldg. D, Lawrenceville, New Jersey 08648

**FLOPPY
DISK
SERVICES™**
INC.

LOCAL HUG CLUBS

Those clubs or members who wish to obtain the Public Domain Library disks in CANADA may do so through THUG (Toronto and area HUG). Please follow directions for obtaining these disks as described in the January 1985 Issue of REMark, page 14. The address for Canadian orders is as follows:

THUG

c/o Stephen Dugas, President
54 Camrose Crescent
Scarborough, Ontario
CANADA M1L 2B7

The following information is either new or updated since the last listing of clubs in the January 1985 Issue of REMark.

New Clubs:

San Diego Computer Society H/Z Special Interest Group

PO Box 81444
San Diego, CA 92138
(619) 698-2945 6-9 pm Pacific Time
Group size: 127
Contact person: Jon Melby

Meet the first Saturday each month at 10:00 am at the La Mesa Heathkit Center. Club carries CP/M Users' Group Library and SIGM Library. Have 24 hr RC System at (619) 461-5117. They are one of 38 SIG's making up the San Diego Computer Society of over 1800.

Salem HUG

PO Box 13434
Salem, OR 97309
(503) 393-0786
Group size: 21
Contact person: Ken Hiigel

Meet the 2nd Tuesday each month. Contact Ken Hiigel for location.

Ames HUG

5006 Todd Drive
Ames, IA 50010
(515) 292-1231
Group size: 40+

Contact person: George F. Covert

Meet the first Wednesday each month 7:00 pm in room 204 Engineering Annex Bldg at ISU College. They produce a monthly Newsletter and are willing to exchange.

H/Z SIG

(A part of the Mid-Michigan
microcomputer group [M3G])
2283 Knob Hill Drive #12
Okemos, MI 48864
(517) 349-9657
Group size: 10-20

Contact person: Bill Goodwin

Meet the second Wednesday each month at 7:30 pm at All Saints Episcopal church, East Lansing. Must be a member of M3G to join SIG. The SIG is a part of an area wide group that publishes a common Newsletter, ENERGY.

HUGOE

(Heath/Zenith Users of Edmonton)
17314 - 106 Avenue
Edmonton, Alberta
CANADA T5S 1H9
(403) 483-4656
Group Size: 12

Contact Person: Edward Hrdlicka

Meet first Wednesday at 7:30 pm, meeting place varies. Have BB (403) 454-6093.

Updated Information

The new contact person for **HUG-Georgia** is Tom Campbell and the new phone number is (404) 449-3328.

HUG of New Jersey has a new address: c/o AMBI-TECH, 319 Knickerbocker Avenue, Hillsdale, NJ 07642. New phone number: (201) 666-0504. New contact person: Matt Baum.

New meeting time for the **Buffalo Users' Group** is the third Tuesday each month at 7:00 pm at the local Heathkit Center.

New president and contact person for the **Pittsburgh HUG** is Bill Pridemore at (412) 882-5932. They no longer have a bulletin board available.

The **El Cerrito, CA HUG** now meets the 4th Tuesday each month.

New contact person for the **Central Wisconsin HUG** is John E. Allman, president. New phone number (715) 359-4156. They now meet the 2nd Tuesday in John's basement at 1006 Foothill, Scofield, WI.

A SQR HUG (Ann Arbor, MI) now meets the 3rd Thursday from 7-9:30 pm Jun-Aug at the Ann Arbor Public Library.

New contact person for the **PENSAHUG** (Pensacola, FL) is Gerry Harris and new phone number is (904) 438-7805. They now meet the second Sunday each month at 2:00 pm at 221 E. Government, Pensacola, FL. Their membership is now 15.

The new contact person for the **FWHUG** (Ft. Worth, TX) is Kent Young. Their membership is now 60.

HUG Hawaii now has a membership of 107.

New contact person for **MUG** (Mission, KS) is Jerry Royle and the new phone number is (913) 381-5470.

New contact person for **LUVAHUG** (Woodland Hills, CA) is Rick Gaitley and the new phone number is (818) 906-7425. Their membership is now 80.

New address for **Tally HUG** (Tallahassee, FL) is c/o LPLC, PO Box 4276, Tallahassee, FL 32315-4276. They now meet the 3rd Tuesday at 7:30 pm at Leon Co. Resource Center in the lower level of Northwood Mall.

New address for **Mystic ZDS/HUG** (Mystic, CT) is PO Box 279 Mystic, CT 06355. Their membership is now 45.

New contact person for **ANAHUG** (Anaheim, CA) is Al Solomon, president at (714) 529-7535.

New address for **SHUG** (Sacramento, CA) is 7607 Maran Avenue, Sacramento, CA 95820. New phone number (916) 739-8074. New contact person is Delven Hamric, president. Send mail Attn: Delven Hamric. Their membership is now 55.

New phone number for the **THUG** (Toronto and area HUG) is (416) 755-0853.



HUG Price List

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

Part Number	Description of Product	Selling Price	Vol. Issue	Part Number	Description of Product	Selling Price	Vol. Issue	Part Number	Description of Product	Selling Price	Vol. Issue
HDOS HARDCOPY SOFTWARE				885-1082	Programs for Printers H8/H89	20.00		885-1059	FOCAL-8 H8/H89 Disk	25.00	13
885-1008	Volume I Documentation	9.00		885-1083-[37]	Disk XVI Misc H8/H89	20.00	11	885-1078-[37]	HDOS Z80 Assembler	25.00	21
885-1013	Volume II Documentation	12.00		885-1089-[37]	Disk XVII Misc H8/H89	20.00	20	885-1085	PILOT Documentation	9.00	
885-1015	Volume III Documentation	9.00		885-1090-[37]	Disk XIX Utilities H8/H89	20.00	22	885-1086-[37]	Tiny HDOS PASCAL H8/H89	20.00	13
885-1037	Volume IV Documentation	12.00	8	885-1092-[37]	Relocating Debug Tool H8/H89	30.00	14	885-1094	HDOS Fig-Forth H8/H89	40.00	18
885-1058	Volume V Documentation	12.00		885-1098	H8 Color Graphics ASM	20.00	19	885-1132-[37]	HDOS Tiny BASIC Compiler	20.00	59
MISCELLANEOUS HDOS COLLECTIONS				885-1099	H8 Color Graphics Tiny PASCAL	20.00	19	CP/M			
885-1032	Disk V H8/H89	18.00	8	885-1105	HDOS Device Drivers H8/H89	20.00	24	885-1208-[37]	CP/M Fig-Forth H8/H89 2 Disks	40.00	18
885-1044 [37]	Disk VI H8/H89	18.00		885-1116	HDOS Z80 Debugging Tool	20.00	27	885-1215-[37]	CP/M BASIC-E	20.00	26
885-1064-[37]	Disk IX H8/H89 Disk	18.00		885-1119-[37]	BHBASIC Support	20.00	29	BUSINESS, FINANCE AND EDUCATION			
885-1066-[37]	Disk X H8/H89	18.00	10	885-1120-[37]	HDOS "WHEW" Utilities	20.00	33	HDOS			
885-1069	Disk XIII Misc H8/H89	18.00		885-1121	HDOS Hard Sec. Sup. Pkg 2 Disks	30.00	37	885-1047	Stocks H8/H89 Disk	18.00	
GAMES				885-1123	XMET Robot & Cross Assembler	20.00	40	885-1048	Personal Account H8/H89 Disk	18.00	
885-1010	Adventure Disk H8/H89	10.00	4	885-1126	HDOS Utilities by PS	20.00	42	885-1049	Income Tax Records H8/H89 Disk	18.00	
885-1029-[37]	Disk II Games 1 H8/H89	18.00	8	885-1127-[37]	HDOS Soft Sector Support Pkg	30.00	45	885-1055-[37]	MBASIC Inventory Disk H8/H89	30.00	
885-1030-[37]	Disk III Games 2 H8/H89	18.00	8	885-1128-[37]	HDOS DISKVIEW	16.00	46	885-1056	MBASIC Mail List	30.00	
885-1031	Disk IV MUSIC H8 Only	20.00	25	885-1129-[37]	HDOS CVT Color Video Terminal	20.00	46	885-1070	Disk XIV Home Fin H8/H89	18.00	
885-1067-[37]	Disk XI H8/H19/H89 Games	18.00	12	885-8001	SE (Screen Editor)	25.00	28	885-1071-[37]	MBASIC SmbusPk H8/H19/H89	75.00	17
885-1068	Disk XII MBASIC Graphic Games	18.00	10	885-8003	BHTOMB	25.00	28	885-1091-[37]	Grade/Score Keeping H8/H89	30.00	14
885-1088-[37]	Disk XVII MBASIC Graph. Games	20.00	14	885-8004	UDUMP	35.00	28	885-1097-[37]	MBASIC Quiz Disk H8/H89	20.00	18
885-1093-[37]	D&D H8/H89 Disk	20.00	16	885-8006	HDOS SUBMIT	20.00	31	885-1118-[37]	MBASIC Payroll	60.00	30
885-1096-[37]	MBASIC Action Games H8/H89	20.00	18	885-8007	EZITRANS	30.00	30	885-1131-[37]	HDOS CHEAPCALC	20.00	47
885-1103	Sea Battle HDOS H19/H8/H89	20.00	20	885-8015	HDOS TEXTSET Formatter	30.00	42	885-8010	HDOS CHECKOFF	25.00	32
885-1111-[37]	HDOS MBASIC Games H8/H89	20.00	23	885-8017	HDOS Programmers Helper	16.00	42	885-8021	HDOS Student's Statistics Pkg	20.00	44
885-1112-[37]	HDOS Graphic Games H8/H89	20.00	23	885-8024	HDOS BHBASIC Utilities Disk	16.00	46	885-8027	HDOS SCICALC	20.00	50
885-1113-[37]	HDOS Action Games H8/H89	20.00	23	CP/M				CP/M			
885-1114	H8 Color Raiders & Goop	20.00	23	885-1210-[37]	CP/M ED (same as 885-1022)	20.00	20	885-1218-[37]	CP/M MBASIC Payroll	60.00	31
885-1124	HUGMAN & Movie Animation Pkg	20.00	41	885-1212-[37]	CP/M Utilities H8/H89	20.00	21	885-1233-[37]	CP/M CHEAPCALC	20.00	47
885-1125	MAZEMADNESS	20.00	41	885-1213-[37]	CP/M Disk Utilities H8/H89	20.00	22	885-1239-[37]	Spread Sht. Contest Disk I	20.00	
885-1130	Star Battle	20.00	47	885-1217-[37]	HUG Disk Duplication Utilities	20.00	26	885-1240-[37]	Spread Sht. Contest Disk II	20.00	
885-1133-[37]	HDOS Games Collection I	20.00	59	885-1223-[37]	HRUN HDOS Emulator 3 Disks	40.00	37	885-1241-[37]	Spread Sht. Contest Disk III	20.00	
885-8009-[37]	HDOS & CP/M Galactic Warrior	20.00	32	885-1225-[37]	CP/M Disk Dump & Edit Utility	30.00	40	885-1242-[37]	Spread Sht. Contest Disk IV	20.00	
885-8022	HDOS SHAPES	16.00	45	885-1226-[37]	CP/M Utilities by PS	20.00	40	885-1243-[37]	Spread Sht. Contest Disk V	20.00	
885-8026	HDOS Space Drop	16.00	49	885-1229-[37]	XMET Robot & Cross Assembler	20.00	40	885-1244-[37]	Spread Sht. Contest Disk VI	20.00	
885-8032-[37]	HDOS Castle	20.00	59	885-1230-[37]	CP/M Function Key Mapper	20.00	42	885-8011-[37]	CP/M CHECKOFF	25.00	32
CP/M				885-1231-[37]	Cross Ref Utilities for MBASIC	20.00	43	ZDOS			
885-1206-[37]	CP/M Games Disk	20.00	11	885-1232-[37]	CP/M Color Video Terminal	20.00	46	885-3006-37	ZDOS CHEAPCALC	20.00	47
885-1209-[37]	CP/M MBASIC D&D	20.00	19	885-1235-37	CP/M COPYDOS	20.00	54	885-3013-37	ZDOS Checkbook Manager	20.00	54
885-1211-[37]	CP/M Sea Battle	20.00	20	885-1237-[37]	CP/M Utilities	20.00	55	885-3018-37	ZDOS Contest Spreadsheet Disk	25.00	58
885-1220-[37]	CP/M Action Games	20.00	32	885-5001-37	CP/M 86 KEYMAP	20.00	51	885-8028-37	ZDOS SCICALC	20.00	50
885-1222-[37]	CP/M Adventure	10.00	35	885-5002-37	CP/M 86 HUG Editor	20.00	52	885-8030-37	ZDOS MATHFLASH	20.00	55
885-1227-[37]	CP/M Casino Games	20.00	38	885-5003-37	CP/M 86 Utilities by PS	20.00	54	DATA BASE MANAGEMENT SYSTEMS			
885-1228-[37]	CP/M Fast Action Games	20.00	39	885-8018-[37]	CP/M FAST EDDY & BIG EDDY	20.00	43	HDOS			
885-1236-[37]	CP/M Fun Disk I	20.00	55	885-8019-[37]	DOCUMAT and DOCULIST	20.00	43	885-1107-[37]	HDOS Data Base System H8/H89	30.00	23
ZDOS				885-8025-37	CP/M 85/86 FAST EDDY	20.00	49	885-1108-[37]	HDOS MBASIC Data Base Sys.	30.00	23
885-3004-37	ZDOS ZBASIC Graphic Games	20.00	37	ZDOS				885-1109-[37]	HDOS Retriever ASM (3 Disks)	40.00	23
885-3009-37	ZDOS ZBASIC D&D	20.00	50	885-3005-37	ZDOS ETCHDUMP	20.00	39	885-1110	HDOS Autofile (2 Disks)	30.00	23
885-3011-37	ZDOS ZBASIC Games Disk	20.00	52	885-3007-37	ZDOS CP/Emulator	20.00	47	885-1115-[37]	HDOS Navigational Program	20.00	25
885-3017-37	ZDOS Contest Games Disk	25.00	58	885-3008-37	ZDOS Utilities	20.00	47	885-8008	Farm Accounting System	45.00	30
UTILITIES				885-3010-37	ZDOS KEYMAP	20.00	51	CP/M			
885-1022-[37]	HUG Editor (ED) Disk H8/H89	20.00	20	885-8029-37	ZDOS FAST EDDY	20.00	53	885-1219-[37]	CP/M Navigational Program	20.00	31
885-1025	Runoff Disk H8/H89	35.00		H/Z100 ZDOS - H/Z150 MSDOS				AMATEUR RADIO			
885-1060-[37]	Disk VII H8/H89	18.00		885-3012-37 ‡	ZDOS HUG Editor	20.00	52	HDOS			
885-1061	TMI Load H8 ONLY Disk	18.00		885-3014-37 ‡	ZDOS/MSDOS Utilities II	20.00	54	885-8016	Morse Code Transceiver Ver 2.0	20.00	42
885-1062-[37]	Disk VIII H8/H89 (2 Disks)	25.00		885-3016-37 ‡	ZDOS/MSDOS Adventure	10.00	57	CP/M			
885-1063	Floating Point Disk H8/H89	18.00		885-3020-37 ‡	MSDOS HUG Menu System	20.00	62	885-1214-[37]	CP/M MBASIC Log Book (64k)	30.00	23
885-1065	Fix Point Package H8/H89 Disk	18.00	10	‡ All program files run on both ‡‡ Program files run partially on both				885-1234-[37]	CP/M Ham Help	20.00	49
885-1075	HDOS Support Package H8/H89	60.00		PC/IBM COMPATIBLE				885-1238-[37]	CP/M ASCIIITY	20.00	57
885-1077	TXTCON/BASCON H8/H89	18.00		885-6001-37	MSDOS Keymapper	20.00	59	PROGRAMMING LANGUAGES			
885-1079-[37]	HDOS Page Editor	25.00	15	885-6002-37	CP/Emulator II & ZEmulator	20.00	59	HDOS			
885-1080	EDITX H8/H19/H89 Disk	20.00		885-8033-37	MSDOS Fast Edit	20.00	62	885-1214-[37]	CP/M MBASIC Log Book (64k)	30.00	23
HDOS				885-1038-[37]	Wise on Disk H8/H89	18.00		CP/M			
885-1038-[37]	Wise on Disk H8/H89	18.00		885-1042-[37]	PILOT on Disk H8/H89	19.00		CP/M			
CP/M				CP/M				CP/M			

Vectored to Page 67



HUG NEW PRODUCTS

HC80E1	.C	Disk D	
HC80F1	.C	CRUNX	.ASM
HC80G1	.C	RMCMT	.ABS
HC80H1	.C	RMCMT	.C
HC80H2	.C		
HC80I1	.C		

Program Author: David A. Wallace

Program Content: The author of this package derived this compiler from Ron Cain's version 1.1 SMALL-C Compiler, which was published in two issues of Doctor Dobb's Journal in 1980. He transcribed the original code and all published fixes to it, added several enhancements of his own, and retargeted the compiler to produce assembly code compatible with HDOS 2.0's ASM assembler. The runnable compiler is on Disk A, along with the standard I/O definitions file, STDIO.H, and the runtime support library, CRUN.ASM. There's also an explanatory or "getting-started" file, README.DOC, on that disk.

The second disk (Disk B), contains the compiler documentation files. HSCUSE.DOC explains how to run the compiler, SMALLC.DOC is a description of the language itself, CRUN.DOC describes the runtime library, and EXAMPLES.DOC explains the included sample programs.

The last two disks (Disk C and Disk D), contain the files necessary to modify or reconstruct the compiler and runtime support library files. Because the maintenance of the compiler requires an H8/H89 system of unusually large capacity, it may not be possible for all users to reconstruct this compiler. These last two disks were included with this product for those programmers and experimenters who are quite knowledgeable in the 'C' language, as well as assembly language and are not required for the proper operation of the compiler itself.

According to the author, the compiler is entirely compatible at the source code level with the UNIX C compiler; it supports a true subset of UNIX C.

Comments: None

TABLE C Rating: (10)

885-1245-37 CP/M-85 KEYMAP Function Key Mapper \$20.00

Introduction: CP/M-85 KEYMAP is a program that lets you designate the responses produced by your computer's function keys. Up to 20 characters, including control characters and RETURNS can be programmed into a single keystroke. When loaded, KEYMAP becomes part of the "system" so that the expanded key responses are available to any program. This version of KEYMAP provides users of 8-bit CP/M on H/Z-100 computers with the ability to map keys that the original CP/M KEYMAP could not handle.

Requirements: CP/M-85 KEYMAP requires any release of CP/M

HUG P/N 885-1134 HDOS SMALL-C Compiler \$30.00

Introduction: This set of diskettes forms a complete compiler for the SMALL-C language. The compiler converts a file of statements in SMALL-C into a file of statements in HDOS ASM assembly language code. Assembling the generated file, along with the runtime support library code (included) creates an executable ABS file. The SMALL-C language is a true subset of the UNIX C language of sufficient power and complexity to write useful programs, including the SMALL-C compiler in which itself was written. Many example programs and massive documentation is provided with the compiler package. The full source code for this compiler is also included so the compiler can be modified and reassembled.

Requirements: The SMALL-C compiler requires the HDOS operating system (Version 2.0) on an H/Z-89 or H8+H19. The computer should have at least 48k of memory although 56k is preferable. It should also have at least two H-17 type disk drives. This program is available in hard sector format only.

The following program and files are included on the HUG P/N 885-1134 HDOS TINY-C Compiler disks:

Disk A		HELLO	.C
CRUN	.ASM	SEE	.C
HSC	.ABS	SSORT	.C
STDIO	.H	WDCNT	.C
README	.DOC		
Disk B		Disk C	
CRUN	.DOC	HSCMAINT.DOC	
EXAMPLES	.DOC	HC80	.H
HSCUSE	.DOC	HC80A1	.C
SMALLC	.DOC	HC80A2	.C
BUG1	.C	HC80B1	.C
BUG2	.C	HC80B2	.C
CHARCNT	.C	HC80C1	.C
ECHO	.C	HC80D1	.C
		HC80DEFS	.H

2.2 on an H/Z-100 series computer, and sufficient memory to run CP/M.

This disk contains the following files:

README	.DOC	KEYBAS	.DOC
KEYMAP	.DOC	KEYSYS	.COM
KEYMAP	.COM	UNMAP	.COM
KEYCON	.COM	KEYMAP	.ASM
KEYWS	.COM	KEYCON	.ASM
KEYWS	.DOC	UNMAP	.ASM
KEYBAS	.COM		

Author: Patrick Swayne, HUG Software Engineer

KEYMAP — This is the executable KEYMAP program, provided in unconfigured form so that you can set up the keys the way you want to. It allows you to define a response of up to 20 characters for each of the following keys: F0 through F12, SHIFT-F0 through SHIFT-F12, I CHR, D CHR, INS LINE, DEL LINE, HOME, the Arrow keys, and the HELP key. If the keypad is shifted, the 1 through 9 keys can also be defined. You can designate one of the keys as an alternate response key, which gives all of the other keys two responses of up to 20 characters each. A total of 36 different responses can be produced without an alternate response key, or 69 responses with one. In addition to the ability to define keys, CP/M-85 KEYMAP offers these other features:

**** Off/On Toggle.** A control code (normally CTRL-SHIFT-6) is provided to toggle keymap on or off, so that it can be temporarily disabled to allow other programs to control the function keys.

**** Off Line Toggle.** A control code (normally CTRL-\) is provided to allow the terminal section of your computer to be taken "off line" so that you can enter escape sequences to set terminal characteristics, etc. This function duplicates the badly missed OFF LINE key found on earlier Heath/Zenith computers.

**** Coexistence With Other Programs.** Not all of the keys must be configured with KEYMAP. Some can be left "unconfigured" so that KEYMAP can coexist with programs that must use the original key responses. For example, if you configure only the shifted function keys, unshifted function keys will continue to function normally.

KEYCON — This program is used to configure the KEYMAP pro-

gram, and allows you to designate the response of each mappable key.

KEYWS — This is a pre-configured KEYMAP for use with WordStar (tm). The requirement to use hard-to-remember control codes is practically eliminated. Cursor and text movement, indenting, centering, underlining, and many other functions are available at the touch of function or keypad keys.

KEYBAS — This is a pre-configured KEYMAP for use with BASIC. 34 BASIC keywords are "programmed" into your keys to make developing BASIC programs easier.

KEYSYS — This is a pre-configured KEYMAP for use with the operating system. Commands such as DIR, STAT, FORMAT, etc. are available at the press of a key.

UNMAP — A program that disables KEYMAP and removes it from the system. It lets you change from one KEYMAP configuration to another without rebooting.

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

Note: These other versions of KEYMAP are available for use on other operating systems or computers as indicated:

885-1230[-37] — CP/M KEYMAP, for use on H8 (with Heath/Zenith terminal), H/Z-89, or Z-90 computers and CP/M. Also works on CP/M-85, but does not map all Z-100 function keys.

885-3010-37 — Z-DOS KEYMAP, for use on H/Z-100 series computers under Z-DOS or MS-DOS.

885-5001-37 — CP/M-86 KEYMAP, for use on H/Z-100 series computers and the Heath/Zenith version of CP/M-86.

885-6001-37 — Z-150 KEYMAP, for use on H/Z-150 series computers or any IBM compatible and MS-DOS or PC-DOS.

**885-3021-37 Z-DOS/MS-DOS
CARDCAT \$20.00**

Introduction: CARDCAT is a program which permits the user to organize information in a manner similar to a library's card catalog. The entries are stored on disk, and can be edited and

TABLE C Product Rating

- 10 - Very Good
- 9 - Good
- 8 - Average

Rating values 8-10 are based on the ease of use, the programming technique used, and the efficiency of the product.

- 7 - Has hardware limitations (memory, disk storage, etc.)
- 6 - Requires special programming technique
- 5 - Requires additional or special hardware
- 4 - Requires a printer
- 3 - Uses the Special Function Keys (F1, F2, F3, etc.)
- 2 - Program runs in Real Time*
- 1 - Single-keystroke input
- 0 - Uses the H19 (H/Z89) escape codes (graphics, reverse video)

Real Time — a program that does not require interactivity with the user. This term usually refers to games that continue to execute with or without the input of the player, e.g. p/n 885-1103 or 885-1211[-37] SEA BATTLE.

ORDERING INFORMATION

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

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

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.

searched using the program's routines.

Requirements:

- Software: ZDOS (Ver. 1.0) or MS-DOS (Ver. 2.0)
- Hardware: H/Z-100 computer or H/Z-150/160
1 disk drive (2 recommended)
128k of RAM (192k really recommended)
Line printer recommended (132 characters per line capacity)

The following files are included on the CARDCAT disk:

README .DOC	CARDCAT .FOR
CARDCAT .EXE	HELP .FOR
HELP .EXE	CARDCAT .ONE

Program Author: Mark Dershwitz, M.D., Ph.D.

Program Contents:

CARDCAT.EXE — This is the compiled version of the card catalog program. For each "card" entry, the program will accept a title up to three authors, up to two subjects, and a "location", a reference to an alphanumeric code relating to the entry, such as its Dewey Decimal System number. The entries are stored on disk, with each disk having a capacity of 2700 entries. Each entry can be examined for correctness and edited, if necessary. There is no limit to the number of disks that can be used to store data. The disks can be searched for all entries containing a specific title, author, subject, or location, and the output list can be displayed on the screen, sent to a line printer, or saved on disk. For use with MS-DOS version 2.0 or higher.

HELP.EXE — This is the compiled version of the detailed operating instructions for CARDCAT. The instructions can be displayed on the screen and/or sent to a line printer.

CARDCAT.FOR — This is the FORTRAN source code for CARDCAT. This would enable the user who has the MS-FORTRAN compiler to alter the program to his or her own specifications, and reassemble the program.

HELP.FOR — This is the FORTRAN source code for HELP.

CARDCAT.ONE — An alternate version of CARDCAT.EXE for users who only have version 1.0 of ZDOS or MS-DOS. Each file disk has a capacity of 2400 entries.

Comments: CARDCAT is an inexpensive alternative to the commercially available filing programs, and permits the user to keep track of collections of books, records, journal articles, etc. The searching routines are particularly useful for readily locating information.

TABLE C Rating: (9)

The HUG P/N 885-3022-37 Z-DOS/MS-DOS Useful Programs I disks contain the following files:

Disk 1

ATD .ASM	GC1000 .BAS
ATD .COM	GC1000 .DOC
ATD .DOC	GC1000A .COM
ATD-CHRO .DOC	GC1000B .COM
CHRONO .BAS	PHONE .BAS
CHRONO .DOC	PHONE .DOC
CHRONO .EXE	PHONE .EXE
DEFCHR .ASM	ZCAT .DOC
DEFMS .ASM	ZCAT .EXE
GC1000 .ASM	README .DOC

Disk 2

ASMP .BAT	SCDMP .DOC
BSHARP .ASM	SCDMP-LR .SKL
BSHARP .DOC	SCDMP-RL .SKL
CVDT .ASM	SCDMPEPS .ASM
CVTNUMBZ .ASM	SCDMPEPS .COM
DEFASCII .ASM	SCDMPEGEM .ASM
DEFMS .ASM	SCDMPEGEM .COM
DLIST .ASM	SCDMPGMX .ASM
DLIST .DOC	SCDMPGMX .COM
DLIST .EXE	SCDMPNEC .ASM
EXPTAB .ASM	SCDMPNEC .COM
GETARGS .ASM	SCDMPOK1 .ASM
PRINTL .ASM	SCDMPOK1 .COM
PRINTL .COM	SCDMPOK2 .ASM
PRINTL .DOC	SCDMPOK2 .COM
PRINTL .EXE	SCDMPPRO .ASM
SCDMP .BAT	SCDMPPRO .COM

Authors:

- ATD** — Larry D. Wakeford
- CHRONO** — Larry D. Wakeford
- GC1000** — Jim Schuster
- PHONE** — Frank Dreano Jr.
- ZCAT** — Mark C. Morrow
- DLIST** — Carl H. Eaton
- PRINTL** — David A. Wallace
- BSHARP** — David A. Wallace
- SCDMP** — Leslie L. Bordelon

Program Contents:

ATD is an automatic time/date assembly language program that reads the current time and date from a Hayes chronograph clock/calendar attached to serial port B of an H/Z-100 computer. It uses this time and date information to set the ZDOS time and date function. This eliminates the need to manually enter this data and the correct time and date are always available for system and program usage.

CHRONO is a compiled ZBASIC program that allows setting and testing of all the chronograph functions from the H/Z-100 keyboard. The program is menu driven and all functions are selected by pressing a single key. If a syntax error is detected in a command sent to the chronograph or if the write protect switch is enabled, you are informed of these conditions. This program also expects the chronograph to be at serial port B.

GC1000 is a MACRO-86 program that interfaces the Heath GC-1000 Most Accurate Clock to the H/Z-100 computer. There are

**HUG P/N 885-3022-37 Z-DOS/MS-DOS
Useful Programs I \$30.00**

Introduction: This two disk product is a collection of utility and application programs for the H/Z-100 computer system. Even though some of the utilities have been written in ZBASIC, they have been compiled, and the interpreter itself is not needed for their proper execution.

Requirements: These programs require the H/Z-100 computer system running Z-DOS or MS-DOS (Version 1.0 or later). At least one disk drive is required also.

many new features of this version that were not contained in the first version as published in REMark (Vol. 5, Iss. 1, Pgs. 73-75). (1) The ability to set and read the current year in the directory entry, thus ignoring the GC-1000's DIP switch settings. (2) Detection of the transmission of a '9' from the GC-1000 in the 1/10 second digit, and the subsequent zeroing of that digit. (3) Detection and correction of the months-with-30-days error as received from the clock. (4) Transparent port configuration — the selected port is configured as required and restored to its original configuration at program termination. This allows port sharing with other devices such as a MODEM.

PHONE. The following is an excerpt from the author's original program description: "It was my hope that by using a computer to simulate the actions of an everyday item, namely the flip-phone-book present on every desk in the government. I have to admit I find myself using this program quite a bit considering I have D-BASE II and LOTUS in my repertoire. The 'Phonebook' never forgets a name, address, or phone number. It is never illegible or dog-eared from use, and it never gets filled with crossed-out or out-of-date entries. Needless to say it was an immediate success at my workplace." This program draws (in full color), a picture of a 'flip' type phone directory. Access is done using the H/Z-100 keypad, and since it's compiled, comes up immediately without the need of the ZBASIC interpreter.

ZCAT is a diskette file librarian which assists you in keeping track of the locations of all your diskette files. It is extremely fast, being composed of compiled BASIC and assembly code, and it features a rich selection of command options which are of use in a cataloging utility. ZCAT is faster than an interpretive BASIC program and will handle 1000 disks and 1300 disk files. It permits catalog information to be sent to the screen, to a printer, or to a disk file for later use. ZCAT maintains file creation dates and times along with the filenames. If you wish, ZCAT will allow you to only see a portion of a disk directory to the catalog. The program accepts filenames and dates from the keyboard, if you wish to only insert a few specific file locations into the master catalog. It will also allow you to do file deletions or searches in the catalog using "wildcard" formatted filenames. ZCAT provides extensive editing facilities during user data-entry and provides recovery methods for most input errors. ZCAT notifies you if you start to catalog a diskette number which is already in the catalog. You then may tell it to delete the old catalog listings for this disk, or add the current disk contents to the catalog anyway, or to assign a new diskette number to the diskette currently being added to the catalog. ZCAT provides an option to only catalog a given diskette file if it is a later revision than the one already in the catalog. Finally, ZCAT lets you run the program from a hard disk environment. (Note that ZCAT is a diskette cataloger that can be resident and operational from a hard disk.)

DLIST is another program designed for cataloging disks. DLIST continuously asks for disks to be placed into the drive of your choice and it reads all the disk directories into its buffer. Upon termination of this mode, it will sort the directory entries as prescribed by several switching options available to you and then store the entire listing onto disk. It will create a new listing name for each iteration thru the program. The listings utilize a typical system 'DIR' like format with some additional information provided for your convenience.

PRINTL is a program that formats, paginates, and optionally line numbers all (or part) of an ASCII text file and prints the report thus produced on the LST device. It is based on a similar utility by

the author which was written for HDOS. This program provides several capabilities not directly achievable with standard MS-DOS commands. First, it provides pagination; printing to an ASCII data file (as opposed to a ".LST" file created by a translator or the output from a word processor program) with the standard MS-DOS PRINT command, does not provide top and bottom margins to accommodate the perforations of an unburst form. Second, the program provides a page heading, which identifies the report by file name and date created, and which numbers the pages. Third, PRINTL will keep track of the current print position and continue output on the next printed line if the forms width is in danger of being exceeded. Fourth, PRINTL can print out a portion of a file instead of the whole thing; you may specify the portion as a range of lines, or as a range of pages. Fifth, PRINTL can accept wildcards in the file name specifier and will print each file as a separate report. Last, the program will (unless otherwise specified) print a line number to the left of the line of text, making the report useful for later editing and for references.

BSHARP is a macro language containing a series of macro definitions which provide an "almost-but-not-quite-C" language facility (for those who are musically inclined, the term BSHARP now becomes obvious). Of the C-language control-of flow features, only the switch/case statements, the oft-abused goto and the do...while statement are not implemented in these macros. Declarations and argument passing is very primitive (all variables are automatic unless defined with DB, DW, or DD statements and all arguments are 16-bit values). The macro set is assuming the 8080 memory model (as in .COM files).

SCDMP is a utility that allows reproduction of a complete video screen on a dot matrix printer, including both text and graphics, without having to exit the current program. The SCDMP program may be loaded manually (by entering SCDMP<cr>, or automatically (via 'autoexec.bat'), into memory at the beginning of a session where it remains resident until needed. To print a desired stationary screen, simply press the 'SHIFT' key and 'F12' simultaneously, which generates an interrupt-5 and activates the screen dump. The program allows a choice of which color bank of video RAM is dumped (if the user has color RAM in his H/Z-100 and the COLOR switch is set to TRUE during assembly of the program). For the color version, entering a for blue, <R> for red, or <G> for green immediately after the <SHIFT-F12> will select the VRAM bank default. If the COLOR switch is set to false, only the green bank can be dumped. The program also allows multiple density printing for some printers. Entering an <H> immediately after the <SHIFT-F12> or color selection would cause the printer to use a higher density mode for printing. Approximately two seconds is allowed after initiation of SCDMP before the default values are assumed. The default density is normal or standard density. SCDMP can be aborted with the <ESC> key. The printers presently supported on this disk set are: C. Itoh 8510A Prowriter, NEC 8023A, Epson MX Series, Star Micronics Gemini, Star Gemini 10X, Okidata 80, Okidata Micro-line Series, and a Skeleton version for use in building a program for your unique printer or application.

Comments: These programs make up the most diversified utility disk set I feel HUG has ever offered. Each one of these routines alone is worth the price of the entire package!

TABLE C Rating: (0), (1), (3), (4), (6), (10)

**HUG P/N 885-3023-37 Z-DOS/MS-DOS
EZPLOT \$20.00**

Introduction: EZPLOT is a user friendly high resolution graphics function plotting routine.

Requirements: EZPLOT works on an H/Z-100 computer running Z-DOS (Version 1.0 or higher), or MS-DOS (Version 2.0 or higher). One disk drive is needed and the program itself occupies 60k of RAM. To use the internal screendump capability of EZPLOT, one of the following printers is needed: C. Itoh Pro-writer, Epson MX or FX series with Graftrax, NEC 8023A, IDS Prism, or an Okidata 92.

The following files are included on the HUG P/N 885-3023-37 EZPLOT disk:

EZPLOT .COM
EZPLOT .ND
AAM .DAT
FNC .DAT
README .DOC

Program Author: Tom L. Riggs, Jr.

Program Content: Some of EZPLOT's features are as follows:

1. EZPLOT is menu driven. The user can quickly and easily format, view, and modify the plot in a logical step-by-step manner.
2. EZPLOT can plot as many as three functions [ie. $f_1(x)$, $f_2(x)$, $f_3(x)$] on the same set of axes. It can also plot X-Y plots, giving

the user the capability of plotting contours or path functions. Finally, it can plot two independent functions [ie. $f_1(x_1)$, $f_2(x_2)$] overlaid on one another.

3. The plot can easily be viewed at any stage of development.
4. EZPLOT makes it simple to define the title, axis labels, and if plotting multiple functions, the function names to be used in the legend.
5. The user can override the default axis ranges and set more esthetic ranges, if needed.
6. The user has control of the number of hashmarks on each axis by setting the number of axis intervals.
7. The user can plot segments of the curve to enlarge areas of interest.
8. EZPLOT gives the user several methods for generating a graphics screendump to printer.
9. The user does not have to worry about presorting the data. EZPLOT automatically checks to see if the data needs sorting and if so, does it.
10. Data files for EZPLOT can easily be produced using any programming language including: BASIC, FORTRAN, PASCAL, and 'C' or COBOL.

An extensive users' manual is included with EZPLOT.

Comments: This program is quite user friendly and according to the author, "virtually GORILLA proof".

TABLE C Rating: (10)



**JOIN US ON OUR JOURNEY
TO ADA* VALIDATION**

As part of its support to advances in the computer industry, **COMPUTER GRAPHICS CENTER, INC.** has acquired a distributorship from RR Software for their Ada related products. They include the **JANUS/Ada** compiler and a full development system. The present compiler contains many of the features of the full Ada language. Growth to validation is scheduled for the near future.

The IBMpc version of the compiler works on the **Z-100** and the **Z-150** series computers. It is now offered at a truly amazing price of \$99.95. The newly revised development system, which includes a complete tool kit, assembler and other products is priced at \$900.00.

A low cost disk is available for assisting the **Z-100** user in making high resolution graphics. Software update contracts and site licenses are also available. Send for your copy of the compiler or development system today to learn and apply the computer language destined for tomorrow.

COMPUTER GRAPHICS CENTER, INC.

140 UNIVERSITY AVE., SUITE 65
PALO ALTO, CA 94301
PHONE: (415) 851-3414



Add 3% shipping/handling. \$3.00 min. California add 6.5% tax.
*Ada is a trademark of the U.S. Department of Defense.

Welcome to *It's Great!*
DOODLER

Graphics Package

... for the Zenith Z-100 and Z-150/160

- Full feature graphics design package
- Save designs on disk for later use
- Playback mode for error correction
- Extended text capability includes...

User designed character fonts
Italic or backslant styles
All text may be scaled

See DOODLER at your local
Heathkit Electronic Center

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



DATA SYSTEMS CONSULTANT
P. O. Box 535
St. James City, FL 33956

Specification Sheet available on Request

Need More Speed?



Try A Compiler!

*C. Nicholas Pryor
Bleak House
Atlantic Avenue
Newport, RI 02840*

In many applications of small computers, such as word processing, spreadsheets, simple data bases, etc., speed of the computer CPU itself is not really an issue. The speed of performing a function in these applications is limited by peripheral device speed, such as disk operations, or perhaps simply by the speed at which the user can interact with the application program. This is the area where the 8-bit microprocessor is the ideal choice.

Even where small computers are used for scientific or other numerical computations, they are often simple enough to be performed in seconds or minutes using an interpretive language such as BASIC. Again, the limited speed of the 8-bit microprocessors does not limit the productivity of the user. However, every job successfully done leads to the temptation to try one just a little harder. Eventually the point is reached where execution becomes painfully slow. What are the options at this point? One, of course, is to go shopping for a faster computer. This can mean an investment of several thousand dollars, plus perhaps obsoleting a lot of expensive existing software. Another choice, though, is to invest in a compiler and to use it for those big jobs. This can make the existing computer seem several times faster than it was with the interpreter, and probably gain as much in speed as simply buying a faster machine. The purpose of this article is to give a better feel for just how much speed can be gained by a compiler.

First, let's review just what is meant by "interpreter" and "compiler". Every computer has its own native language designed into its hardware, and represented by the "assembly" language of the computer. This represents exactly how the CPU works at its basic instruction level. Unquestionably, the fastest execution times are obtained by programming the application in this assembly language, and carefully tuning the program to the hardware capabilities of the computer. However, this is very tedious, and forces the programmer to work in the natural language of the computer.

An interpreter represents the other extreme. The programmer writes in a language, resembling algebra and logical statements, that is natural to him. Unfortunately, this language means nothing to the computer. In order to make any sense of it, the computer actually runs a program called an interpreter that scans the programmer's statements, figures out what they are asking the computer to do, and then executes a series of computer instructions to carry out that task. This is done step by step as the interpreter scans through the program, and is done repeatedly if the program has any loops which are repeated. The good part of this is that the interpreter makes the computer seem to understand the programmer's language, and the program is always available in this input language to make it easy to read or modify. The bad part is that the interpreter spends much more

time interpreting the program, to figure out what to do, than it does actually carrying out the task. Thus, while programming in the natural high level language is much easier than using assembly language, execution of the resulting program is much slower than if the same job had been done in assembly language.

A compiler is an attempt to have the best of both. Again, the programmer is allowed to write in a natural high level language. However, this program is then fed as input to another program (the compiler). The compiler also scans through this input program, just as the interpreter does. However, instead of trying to execute individual computing tasks as it finds them, it produces a set of assembly language instructions which would cause the computer to perform the desired function. These instructions become the output of the compiler program. The programmer's input to the compiler is known as the source program, and the set of instructions generated by the compiler is known as the object program. This object program is then fed back into the computer to be executed, to do the job called for by the programmer's source program. This has the advantage that the programmer still gets to write in the language that is natural to him, yet the program that is finally executed by the computer has been converted into the assembly language that is natural to the computer. The result is a program that may run nearly as fast as if the programmer had written it in assembly language initially. No compiler produces object code that is quite as good as the program that could be written by a good assembly language programmer. However, the result is often close enough that the extra speed obtainable by programming in assembly language is just not worth the effort.

The compiler does have a disadvantage. That is, that it requires the extra step of "compiling"; passing the source program through the compiler program to produce the object program before it can be run. This object program may also have to be linked with other object programs (subroutines or a run-time library) before it can be run. This task typically requires several minutes. The object code is generally difficult or impossible to read and understand, so if there are any changes needed, it is necessary to make them in the source program and repeat the whole compiling process. Thus, program development and checkout is harder when working with a compiler than it is with an interpreter. An ideal solution to this is to have both an interpreter and a compiler that accept the same high level language. Then program development can be done using the interpreter. Once it is satisfactory, the program can be compiled, and the more efficient output of the compiler used for additional time-consuming runs of the program. BASIC is by far the most popular

language used by interpreters. Happily, there are also compilers that use BASIC as their input language. There are also compilers available for a number of other high level input languages, such as PASCAL and FORTRAN, but generally these do not have companion interpreters using the same language.

While the above shows that programming in assembly language produces the fastest programs, and using an interpreter produces the slowest, with a compiler being somewhere in between, it does not answer the question "How much faster?". As we will see, this depends on the type of program. Programs with a lot of logic and little real computation have the most to gain from using a compiler. Programs that are very heavy on floating point computation and mathematical functions (sin, cos, sqrt, etc.) have the least to gain.

To quantify this, three benchmark programs are shown here, each representing a different balance among logic, fixed point arithmetic, and floating point arithmetic. Each is written in Microsoft BASIC and was run both with the Microsoft interpreter and again using their compiler. The times shown are for execution on a standard Heath H89A, with a Z80 CPU running at 2 MHz. An assembly language version of one of the programs was also tested. Finally, two other compilers, one using PASCAL and one using FORTRAN as the source language, were tested and compared with the results from the Microsoft BASIC compiler.

Logic-Intensive Benchmark (Bubble Sort Routine)

The benchmark used as a logic-intensive program is a sort routine for a list of integer variables, using the familiar bubble sort algorithm. This is not the fastest type of sort algorithm, but it is simple and is well known. The main part of the BASIC routine to perform this sort algorithm is:

```
100 N1=N-1
110 F=1
120 FOR I=1 TO N1
130 IF A(I)>A(I+1) THEN SWAP A(I),A(I+1):F=0
140 NEXT I
150 IF F=0 GOTO 110
```

The routine scans through the list A(I), comparing each entry with its neighbor. If they are in the wrong order, they are swapped. A flag F is also set to 0 when a swap is required. At the end of the list, this flag is tested. If any pair of entries was exchanged during the scan through the list, the list is scanned again. This continues until a scan is completed without swapping any entries, indicating that the list is in order.

Execution time of this algorithm depends on the order of the list at the beginning of the sort. If the list is already in proper order, only one pass is made through the data. The worst case occurs when the list starts exactly in reverse order. Then N passes are required through the list, each requiring N-1 comparisons. The execution time in this case is then nearly proportional to N squared. The solid line labeled Bubble Sort in Figure 1 shows the actual execution time of this routine, using the Microsoft BASIC Interpreter, as a function of the list size N for this worst case. Note that the execution time does increase as N squared, as expected.

The same BASIC source program was then compiled, using the Microsoft BASIC Compiler, and execution times were measured using this compiled version. These are shown by the dashed line of Figure 1. The same N squared dependence is seen, but the times are much faster. For example, for N=200 the interpreter required 500 seconds to perform the sort, while the compiled version of the program required only 13.5 seconds. This is a ratio of 37:1 in speed between the interpreted and compiled version of the same program. This is about as big an advantage as one would ever expect from the compiler, because the program involves a minimum of arithmetic and only deals with integer variables.

As an added test on the efficiency of the compiled code, the same algorithm was written in assembly language and the execution time measured. This assembly language program made use of the index registers in the Z80 CPU, as well as some special Z80 instructions that are not in the 8080 instruction set, and was designed to keep as much as possible in the Z80 registers to avoid unnecessary store and load instructions. It is thus an example of a highly optimized assembly language program. It is listed at the end of this article, for those who like to read other people's assembly language programs. (Actually, it is not the fastest possible assembly language program. A version which should have been about twice as fast was written, using the Z80 stack pointer to manipulate 16 bit data. However, this conflicted with the interrupt service routine built into the Heath H89 monitor, and thus could not be run on the H89 without turning off the interrupts.)

Execution time of this assembly language program is shown by the dotted line in Figure 1. For the case of N=200, execution time was only 4 seconds. Thus the assembly language version was about 3.3 times faster than the compiled BASIC program, and 125 times faster than the interpreted version. Again, this is an extreme case, because of the nature of the benchmark.

Typical Scientific Program (Matrix Inversion)

The second benchmark tested was a matrix inversion routine, a fairly common mathematical step in problems such as solving simultaneous equations. It contains a mixture of logic, floating point computation, and fixed point arithmetic (for computing address offsets in array variables); and thus is probably representative of the "typical" scientific program. The main portion of this program, written in BASIC, is:

```
100 FOR I=1 TO N
110 C=1/A(I,I)
120 FOR J=1 TO N
130 A(I,J)=C*A(I,J)
140 NEXT J
150 FOR K=1 TO N
160 IF K=I GOTO 220
170 D=A(K,I)
180 FOR J=1 TO N
190 A(K,J)=A(K,J)-D*A(I,J)
200 NEXT J
210 A(K,I)=-C*D
220 NEXT K
230 A(I,I)=C
240 NEXT I
```

This is a version of the Gauss Elimination Method, which successively adds a fraction of each row of the matrix to all other rows, to produce zeros in each off-diagonal term of a column. It is designed to operate "in place", so that the inverse matrix is formed in the same storage area occupied by the original matrix. The algorithm shown above is slightly incomplete, in that it does not use "pivoting" techniques to maximize accuracy, nor does it test for singular matrices. These features would normally be included in a matrix inversion subroutine.

Inspection of the above routine shows that lines 180 through 200 are in a triply nested loop. That is, the small loop on the J index is inside a loop on the index K, which is in turn inside a loop on the index I. All loop from 1 to N, where N is the linear dimension of the square matrix. Thus line 190 will be executed N cubed times. For large values of N, this little loop dominates the execution time, and this algorithm is considered to be in the class of "N cubed" problems.

The solid line labeled Matrix Inversion in Figure 1 shows the execution time of this routine as a function of N, using the Microsoft BASIC Interpreter. The matrix entries A, and the temporary values C and D,

were defined as single precision real (floating point) variables, while all indices were defined as integers. The N cubed dependence can be seen for values of N above about 10. Below this, parts of the routine which vary more slowly with N begin to be significant. Again, the dashed line shows the result of compiling the program first, and then running the compiled program. The speed increase in this case is much smaller than it was for the bubble sort benchmark. For example, with N=20, the interpreter required 164 seconds while the compiled version required only 43 seconds. This is a ratio of about 3.8:1; still quite a significant improvement.

No attempt was made to write an assembly language version of this benchmark. However, some experiments showed that at most about a 20% improvement could be made over the speed of the compiled BASIC program, as long as the same subroutines for floating point arithmetic were retained. This improvement would come primarily from using the Z80 index registers to step through the matrix rows, and avoiding the messy address computation each time through the loop.

The matrix inversion benchmark begins to have a problem with accuracy for large N, due to the limited 7-digit accuracy of the single precision floating point calculations. Thus this benchmark was also run with the matrix elements defined as double precision variables, with about a 17 digit accuracy. Use of double precision roughly doubled the execution time for the interpreter, raising the time from 164 to 332 seconds for N=20. It raised the execution time of the compiled version from 43 to 200 seconds.

Computation-Intensive Benchmark (Three-Body Motion)

The third benchmark was designed to be computation intensive, with a minimum of logic or address computation. The problem chosen for this was the three-body motion problem from celestial mechanics, where each body moves under influence of gravity from the other two. The BASIC program for the main loop of this calculation is:

```

100 FOR I=1 TO N
110 R12=((X1-X2)*(X1-X2)+(Y1-Y2)*(Y1-Y2)+(Z1-Z2)*(Z1-Z2))^1.5
120 R13=((X1-X3)*(X1-X3)+(Y1-Y3)*(Y1-Y3)+(Z1-Z3)*(Z1-Z3))^1.5
130 R23=((X2-X3)*(X2-X3)+(Y2-Y3)*(Y2-Y3)+(Z2-Z3)*(Z2-Z3))^1.5
140 VX1=VX1+T*M2*(X2-X1)/R12+T*M3*(X3-X1)/R13
150 VY1=VY1+T*M2*(Y2-Y1)/R12+T*M3*(Y3-Y1)/R13
160 VZ1=VZ1+T*M2*(Z2-Z1)/R12+T*M3*(Z3-Z1)/R13
170 VX2=VX2+T*M1*(X1-X2)/R12+T*M3*(X3-X2)/R23
180 VY2=VY2+T*M1*(Y1-Y2)/R12+T*M3*(Y3-Y2)/R23
190 VZ2=VZ2+T*M1*(Z1-Z2)/R12+T*M3*(Z3-Z2)/R23
200 VX3=VX3+T*M1*(X1-X3)/R13+T*M3*(X2-X3)/R23
210 VY3=VY3+T*M1*(Y1-Y3)/R13+T*M3*(Y2-Y3)/R23
220 VZ3=VZ3+T*M1*(Z1-Z3)/R13+T*M3*(Z2-Z3)/R23
230 X1=X1+T*VX1:Y1=Y1+T*VY1:Z1=Z1+T*VZ1
240 X2=X2+T*VX2:Y2=Y2+T*VY2:Z2=Z2+T*VZ2
250 X3=X3+T*VX3:Y3=Y3+T*VY3:Z3=Z3+T*VZ3
260 NEXT I

```

This is a loop that is repeated N times. Lines 110-130 calculate the cube of the distance between each pair of bodies. The next nine lines use the gravitational forces to update each of the three velocity components of each of the three bodies. The next three lines use the velocities to update the positions. It is all just brute force arithmetic. Each trip through the loop requires 69 floating point adds or subtracts, 54 multiplies, 18 divides, and three exponentiation operations. This is not necessarily the most efficient way of writing this program. Some grouping and precalculation of repeated terms would probably save a little bit of time. Execution time for the interpreter, as a function of the number of times through the loop, is shown by the solid line labeled Three-Body in Figure 1. It is clearly just proportional to the value of N. The dashed line shows the execution time for the compiled version. Here the difference is still

smaller than it was for the matrix inversion benchmark. For N=100, the interpreter required 78 seconds, while the compiled version required 47 seconds. Thus the speed increase with the compiler was only a factor of 1.66. This is because most of the time is spent actually performing the floating point computations in this benchmark, and the compiler does not help this part of the problem.

No attempt was made to do an assembly language program for this benchmark; the speedup over the compiled version would have been negligible. This benchmark was also tried using double precision arithmetic. For N=100, the execution time for the interpreter increased from 78 to 213 seconds, while the execution time for the compiled version increased from 47 to 350 seconds. If you are reading carefully, you will have noticed a strange anomaly. The compiled version actually took longer in double precision than the interpreter. The reason for this is hidden in the Microsoft Compiler manual. The Microsoft Interpreter only performs single precision arithmetic on functions (such as the exponentiation operator), even for double precision variables. However, the run-time library supplied with the compiler performs true double precision functions. The extra time for the compiled version is the result of this much more difficult computation.

Other Compilers

The efficiency of the code produced by a compiler is, of course, dependent on the sophistication of the compiler. Two other compilers were available, providing an opportunity to compare their performance with the Microsoft BASIC compiler. One of these was the TURBO Pascal Compiler, produced by Borland International. The other was the Nevada FORTRAN compiler, sold by Ellis Computing Company. Of course, the benchmarks had to be rewritten into equivalent programs in Pascal and FORTRAN, respectively. The bubble sort benchmarks looked rather different because of the different logical constructs in the languages, even though the algorithms remained the same. For example, the Pascal routine for the bubble sort was:

```

repeat
  NOSWAP:=TRUE;
  for I:=1 to N-1 do
    if A[I]>A[I+1] then begin
      T:=A[I];A[I]:=A[I+1];A[I+1]:=T;
      NOSWAP:=FALSE;
    end;
until NOSWAP;

```

while the FORTRAN version of the routine was:

```

N1=N-1
100 FLAG=.FALSE.
DO 110 I=1,N1
IF (A(I).LE.A(I+1)) GO TO 110
T=A(I)
A(I)=A(I+1)
A(I+1)=T
FLAG=.TRUE.
110 CONTINUE
IF (FLAG) GO TO 100

```

The other benchmarks in Pascal and FORTRAN are more nearly line-for-line equivalents to the BASIC versions.

Execution of the bubble sort program using TURBO Pascal was about 20% slower than with the Microsoft BASIC compiler, requiring 16 seconds for N=200. Execution of the matrix inversion benchmark was about 7% slower than the BASIC compiler, while execution of the three-body dynamics problem was about 2% faster than in BASIC. Thus, overall, TURBO Pascal seems to provide about the same execution speed as the Microsoft BASIC compiler. One additional advantage of TURBO Pascal is that the floating-point real

variables have 11 digit precision, about halfway between the normal and double precision cases for BASIC. Thus, TURBO Pascal not only kept up with the single precision speed of the BASIC compiler, but it provided additional accuracy. By the way, TURBO Pascal is a very nice integrated package, containing its own full screen editor (a subset of Wordstar), the compiler, and run-time library. It is almost as easy to use as a BASIC interpreter, yet provides the speed of a compiler. This is not the case with the other compilers, which require much more fiddling around with separate editors, compilers, loaders, etc.

Execution of the bubble sort with Nevada FORTRAN required 512 seconds for N=200, as compared with 500 seconds for the BASIC interpreter and 13.5 seconds for the BASIC compiler. Execution of the matrix inversion for N=20 required 151 seconds, compared to 164 seconds for the BASIC interpreter and 43 seconds for the compiled BASIC program. The three-body benchmark for N=100 required 110 seconds in Nevada FORTRAN, compared to 78 seconds for the BASIC interpreter and 47 seconds for the compiled BASIC program. Thus the Nevada FORTRAN, in spite of being a compiler, is more in the speed class of an interpreter than of the other compilers. This is not a fault of the FORTRAN language. While it was not tested, I would expect the Microsoft FORTRAN compiler to produce code about as efficient as their BASIC compiler. It is simply that speed was clearly not a goal of the Nevada FORTRAN compiler when it was developed.

Summary

The table below summarizes the relative execution speeds of the BASIC interpreter and the various compilers for each of the three benchmark programs. For each of the benchmarks, the speed is normalized to 1.00 for the BASIC Interpreter in single precision. The speed listed is the inverse of the relative execution time; so higher numbers mean faster execution.

Figure 1

	Bubble Sort	Matrix Invert	3-Body Dynamics
BASIC Interpreter (Single Precision)	1.00	1.00	1.00
(Double Precision)	-	0.49	0.37
BASIC Compiler (Single Precision)	37.00	3.81	1.66
(Double Precision)	-	0.82	0.22
TURBO Pascal Compiler	31.30	3.57	1.70
Nevada FORTRAN Compiler	0.98	1.09	0.71
Assembly Language	125.	4.7	1.8

Clearly, the speed advantage of a compiler varies over a wide range, depending on the type of program involved. The results are fairly consistent with the Microsoft literature which claims compiled programs are "typically 3 to 10" times faster and "up to 30" times faster than interpreted programs. Thus, if your programs have a fair amount of logic or integer arithmetic, or require address computation for array type variables, pretty substantial gains in speed are possible with the compiler. If, on the other hand, the program is heavy on floating point arithmetic and function calls, the improvement may be disappointing. In this case, probably the only solution is a faster computer. The effect of changing to a 16 point machine, and particularly, to one with an 8087 floating point processor, will be explored in a future article.

Assembly Language Version of Bubble Sort

```

L1: LD  IX, _____    point to beginning of list
    LD  BC, _____    contents determined by length of list
    SCF                      clear carry flag
    CCF                      clear carry flag
    LD  E, (IX+0)          load first entry into DE
    LD  D, (IX+1)
L2: LD  L, (IX+2)          load next entry into HL
    LD  H, (IX+3)
    SBC HL, DE              subtract entries
    JR  C, L3              test for carry
    ADD HL, DE              restore value of HL
    EX  HL, DE              exchange entries
    JR  L4
L3: ADD HL, DE              restore value of HL
    CCF                      clear carry flag
    LD  A, FF              set swap flag
L4: LD  (IX+0), L           store contents of HL in list
    LD  (IX+1), H
    INC IX                  increment IX pointer
    INC IX                  by 2
    DJNZ L2                 loop until B=0
    DEC C                   decrement C portion of loop count
    JR  NZ, L2              loop until C=0
    LD  (IX+0), E           put last entry back on list
    LD  (IX+1), D
    BIT  A, 0               test swap flag
    JR  NZ, L1              loop until swap flag is clear

```



What's The HUG B.B. or HUG SIG?

It's HUG members just like yourselves using their computers and modems to exchange ideas and solve problems.

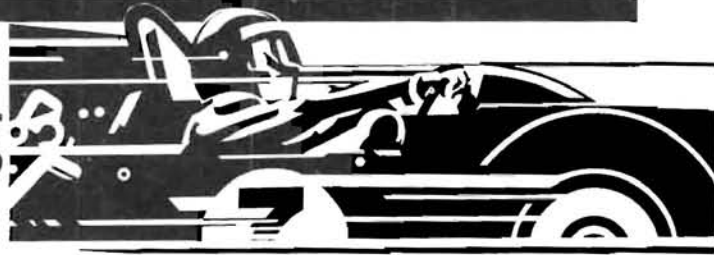
It's also a large Data Base containing hundreds of Public Domain programs that are shared amongst the HUG and SIG members.

If you're not part of the HUG/SIG, then why not get started today. Join HUG's Micronet Connection (PN 885-1122-[37] HDOS; PN 885-1224-[37] CP/M; \$16.00) which gives you your ID number and password and your first hour of connect time **FREE** or call Jim Buszkiewicz at (616) 982-3837 for more information.



TURBO-Charging

Your Z



Frank Dreano, Jr.
209 Thrasher Road
Chesapeake, VA 23320

No, this article is not in the wrong magazine! It doesn't have anything to do with automobiles. This is a brief description of how to use a high-level language to do low-level functions. Specifically, TURBO Pascal by Borland International. This is without qualification one of the fastest, most efficient compilers in the MS-DOS world. It produces 8088 machine code and is thus a true compiler and does not require intermediate interpretation. An optimizing Pascal compiler, such as this, can generate machine code at least as efficient as you can write by hand, if not more so. The source is easier to debug, develop and understand than 8088 assembly language with a minimal trade-off in hardware manipulation. However, many of the "bells and whistles" available in TURBO for the IBM-PC do not operate on the Z-100 series of computers, especially graphic, cursor, color commands, etc. due to differences in design architecture. Thus, you lose much of the power that ZBASIC has via the Color, Line, Draw and similar statements.

I, therefore, decided to rectify this problem by calling the ZBASIC graphics routines in the BASCOM.LIB compiler library from Pascal programs. This proved cumbersome to implement and would require more memory for initializing the library than it was worth. So then I tried to emulate some of the simpler ZBASIC color commands in TURBO Pascal. After spending several hours digging through the Z-100 hardware manuals, I began to develop my own set of graphics procedures. Several of them are included in this article. The point is that a first-class optimizing compiler like TURBO can perform low-level hardware manipulation, operating system calls, etc. with a much lower sweat-to-code ratio than an unfamiliar assembler, and in most cases, just as quickly. I offer as proof of this the fact that the original UNIX operating system was written in the high-level 'C' language and Ada will soon become the government standard for imbedded processor and scientific applications.

A few notes on Borland International's particular implementation of Pascal are in order. They have provided a very easy-to-use 'include' facility in the compiler to enable the development of a library of known-good procedures and functions. Also, the ability to directly manipulate memory, segment registers, ports, etc. is provided via specialized statements. These are NOT standard Pascal statements and are, therefore, not universally acceptable to all compilers.

Description Of Program Modules

GLOBAL.VAR — This is the heart and soul of the program. It defines all Zenith peculiar ports, addresses, etc. It is the normal Pascal fare in all other respects. A user-defined data type is used to create an array of absolute color ram locations, so loops can be used to write into the three color planes.

INIT.PRO — This procedure initializes the graphics variables and insures the program starts with all colors visible and pointing to the green plane of memory. Should only be used once in any program.

COLOR.PRO — This procedure will accept, as input, one of the eight possible colors in a parameter form and set up the proper port data and color plane pointer. All colors are treated the same except black (which means turning memory bits off instead of on). Black sets a flag so that bits will be turned off in ALL planes so nothing shines through!

PIXEL.PRO — This procedure converts screen pixel column (between 1 and 640) and row (between 1 and 225) data into color memory locations in video memory. Row locations are exact, but column locations are to the nearest byte containing the pixel — this is the nature of color memory.

PSET.PRO — This is the procedure that has the hardest job. It turns on (or off if black color is called) the right pixel in the right byte in the right color plane of memory. This gave me the worst time, because I originally tried to use the 'mem[]' statement in TURBO to write to ram and this created inexplicable 'holes' on the screen when large areas were being colored via loops. (Never did isolate this...probably due to the weird addressing scheme for video memory!) I now use the 'fillchar()' command with satisfactory results. The fact that memory is byte-oriented, while graphics are pixel-oriented requires the program to read what is already in a video memory byte, change a given pixel within the byte while preserving all others, and write the whole byte back into video memory.

DRAW.PRO — This routine will accept an input string of up to 80 characters and produce a corresponding figure (just like ZBASIC) in whatever color you choose. You must use capital letters.

SCREEN.PAS — This is the main body of the program and is merely a confidence check of the foregoing procedures. It really doesn't do much, but it represents a lot of work!

I have also developed procedures and functions to determine the position of the cursor in the x and y planes, draw circles and lines and determine the color set at a particular screen location.

About The Author

Frank Dreano currently develops system software for the U.S. Navy's A-6 fighter aircraft flight simulators at the Oceana Naval Air Station in Virginia Beach, Virginia. He is a former electronics technician turned programmer who will receive his B.S. in Computer Science in May of 1985.

Global.Var

```

(* These are the Pascal vars needed to do graphics work. You may add
 * more for a specific application but DO NOT delete anything that is
 * already here !!! Procs using these equates are as follows: INIT.PRO, *
 * CLS.PRO, COLOR.PRO, PIXEL.PRO, PSET.PRO, WHERE.FCN, HEX.PRO, DRAW.PRO *)
const zvideo = $D8; (* CRT-Controller Port *)
    CRT_port_A = $D9; (* CRT Port bit 3 low clears screen *)
    CRT_port_B = $DB; (* CRT Port bit 3 low VRAM=00; hi VRAM=FF *)
    CRT_reg_sel = $DC; (* CRT-C port to select register *)
    CRT_reg = $DD; (* CRT-C reg to write/read to *)
    CRT_latch = $DA; (* CRT-C start address latch port *)

(* Following equates determine which planes of VRAM are displayed *)
read_all = $78;
read_blu = $7B;
read_grn = $7D;
read_red = $7E;

(* Following equates allow the CPU to write to VRAM color segments *)
red_hex = $68; blue_hex = $38; green_hex = $58; (* red OR blue OR green *)
white_hex = $08;
aqua_hex = $18; (* blue/blue/green *)
purple_hex = $28;
yellow_hex = $48; (* red/blue *)
black_hex = $7E; (* red/green *)
(* turn off pixel *)

(* Following equates are ABSOLUTE VRAM color segment addresses in ram *)
blu_seg = $C000;
red_seg = $D000;
grn_seg = $E000;

(* Following equates are
or graphics manipulations of VRAM data *)
type string5 = string[5];
string80 = string[80];
all_plane = (blu_plane, red_plane, grn_plane);

var k : byte;
    vramay : array[blu_plane..grn_plane] of integer; (* ALL color planes *)
    width,height : integer; (* for future proc to draw box of size n *)
    x_pixel : integer; (* pixel location *)
    y_pixel : byte; (* in 24x80 CRT *)
    curr_plane, vram_ptr : integer; (* segment and byte within VRAM *)
    Hex_Addr : string5; (* used to hold result of hex conversion *)
    row, black_flag : boolean; (* flag for turning pixels Off !!! *)
    red,blue,green,white,aqua,purple,yellow,black : byte; (* 8 colors *)

(* ##### *)

```

Pixel.Pro

```

(* ##### *)
Procedure Pixel(var x : integer;
               var y : byte);
(* This routine takes inputs x and y and computes the proper VRAM address *)
(* in any plane. It returns the byte in VRAM via x and the pixel within *)
(* that byte via y. See the PSET.PRO procedure for example of proper use. *)
var col,row,line : integer;
    dot : byte;
begin (* Pixel *)
    col := (x-1) div 8;
    dot := (x - (col * 8)) - 1;
    row := (y-1) div 9;
    line := (y - (row * 9)) - 1;
    row := row shl 4;
    row := row or line;
    row := row shl 7;
    x := row or col;
    y := $80 shr dot;
end; (* Pixel *)
(* ##### *)

```

PSET.Pro

```

(* ##### *)
Procedure Pset(x : integer;
              y : byte);
(* This routine requires two global vars : curr_plane is the color plane *)
(* now active and must have been set up thru the video I/O port and the *)
(* pointer into curr_plane. vram_ptr, will contain the computed pixel addr. *)
(* The routine PIXEL.PRO converts pixels into VRAM addresses. Also the file *)
(* GLOBAL.VAR is needed to provide the global variables used within. *)
var prev_dat : byte;
    pointer : ^byte;
    ramndx : all_plane; (* global data user-defined data type *)
begin (* Pset *)
    Pixel(x,y);
    vram_ptr := x;
    prev_dat := mem[curr_plane:vram_ptr];
    (* if pixel is already as desired leave screen alone ! *)
    if (v and prev_dat = 0) and (black_flag = false) then
        begin (* if *)
            y := y or prev_dat;

```

Init.Pro

```
(* ##### *)
Procedure Init;
(* This procedure initializes the necessary parameters to use the graphics *)
(* among other things. It needs to have the file GLOBAL.VAR in the body *)
(* of the program in order to get the proper variables to work with. *)
begin (* Init *)
  (* set up the color vars *)
  red := red_hex; blue := blue_hex; green := green_hex;
  white := white_hex; aqua := aqua_hex; purple := purple_hex;
  yellow := yellow_hex; black := black_hex;
  vramray[blu_plane] := blu_seg; (* ABSOLUTE locations
  vramray[red_plane] := red_seg; of VRAM
  vramray[grn_plane] := grn_seg; color planes *)
  curr_plane := vramray[grn_plane];
  port[zvideo] := read_all;
end; (* Init *)
(* ##### *)
```

Color.Pro

```
(* ##### *)
Procedure Color(forgrnd : byte);
(* This routine defines the color to use in graphics work. It sets up *)
(* the 6845 port to write the foreground color and points to the pro- *)
(* per plane in VRAM to write to. It may or may not work exactly as *)
(* expected depending on what other colors have been previously used. *)
(* It also requires the global constant zvideo := $D8 for CRT-C mode. *)
(* Procedures GLOBAL.VAR and INIT.PRO will set-up data for this proc. *)
begin (* Color *)
  black_flag := false;
  port[zvideo] := forgrnd;
  case forgrnd of
    red_hex,white_hex : curr_plane := vramray[red_plane];
    purple_hex,yellow_hex : curr_plane := vramray[red_plane];
    blue_hex,aqua_hex : curr_plane := vramray[blu_plane];
    green_hex : curr_plane := vramray[grn_plane];
    else (* black_hex *)
      black_flag := true;
      port[zvideo] := white_hex;
  end; (* case *)
end; (* Color *)
(* ##### *)
```

```
pointer := ptr(curr_plane,vram_ptr);
fillchar(pointer,1,y);
end (* if *)
else
  for ramndx := blu_plane to grn_plane do
    begin (* for *)
      prev_dat := mem[vramray[ramndx]:vram_ptr];
      if (y and prev_dat <> 0) and (black_flag = true) then
        begin (* if *)
          y := not y;
          pointer := ptr(vramray[ramndx],vram_ptr);
          fillchar(pointer,1,y);
          end; (* if *)
        end; (* for *)
    end; (* Pset *)
  (* ##### *)
```

Draw.Pro

```
(* ##### *)
Procedure Draw(graphic : string80);
(* This routine will draw graphic figures defined in a string format. It *)
(* will draw lines in all 8 compass directions using strings 'inside single *)
(* quotes' of a letter indicating direction followed by a number indicating *)
(* length. The letter indicates directions the same as WordStar commands to *)
(* move the cursor e.g. D100 draws a 100 pixel line to the right. It needs *)
(* the files GLOBAL.VAR, PSET.PRO and PIXEL.PRO in order to function. *)
var code,amt,k,m : integer;
    cmd,amount : string5;
    done : boolean;
begin (* Draw *)
  k := 1;
  done := false;
  while k < Length(graphic) do
    begin
      amount := '';
      cmd := Copy(graphic,k,1);
      k := k + 1;
      while (done = false) and (ord(copy(graphic,k,1)) - 48 < 10) do
        begin
          amount := amount + copy(graphic,k,1);
          if k = Length(graphic) then
            done := true
          else
            k := k + 1;
        end; (* while *)
        Val(amount,amt,code);
        for m := 1 to amt do
          begin
            Pset(x_pixel,y_pixel);
          end;
        case cmd of
          'E' : y_pixel := y_pixel - 1;
          'X' : y_pixel := y_pixel + 1;
        end;
      end;
    end;
  done := true;
end; (* Draw *)
(* ##### *)
```

```

'S' : x_pixel := x_pixel - 1;
'D' : x_pixel := x_pixel + 1;
'R' : begin
      x_pixel := x_pixel + 1;
      y_pixel := y_pixel - 1;
    end;
'C' : begin
      x_pixel := x_pixel + 1;
      y_pixel := y_pixel + 1;
    end;
'W' : begin
      x_pixel := x_pixel - 1;
      y_pixel := y_pixel - 1;
    end;
'Z' : begin
      x_pixel := x_pixel - 1;
      y_pixel := y_pixel + 1;
    end;
    end; (* case *)
    end; (* for *)
    end; (* while *)
    end; (* Draw *)
  (* ##### *)

```

```

(* ##### *)

```

Screen.Pas

Program Screen;

```

{$I GLOBAL.VAR}
{$I INIT.PRO}
{$I COLOR.PRO}
{$I PIXEL.PRO}
{$I PSET.PRO}
{$I DRAW.PRO}

```

```

Begin (* main *)
  Init;
  ClrScr;
  Color(red);
  x_pixel := 140;
  y_pixel := 450;
  Draw('D150E50S150X50');
  Color(yellow);
  x_pixel := 150;
  y_pixel := 150;
  Draw('R100D100C100S100W100');
  Color(blue);
  x_pixel := 150;
  x_pixel := 100;
  Draw('C50D100E60');
  Color(black);
  for x_pixel := 475 to 575 do
    Pset(x_pixel,140);
  port[zvideo] := read_all;
end. (* main *)

```



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 54 format checked

ON 1 HARD REGISTERED TO \$17.00
 ON 2 SOFT REGISTERED TO \$17.00
 ON 1 DOUBLE REGISTERED TO \$17.00

HEADWARE
 2865 AKRON STREET
 EAST POINT, GA. 30344

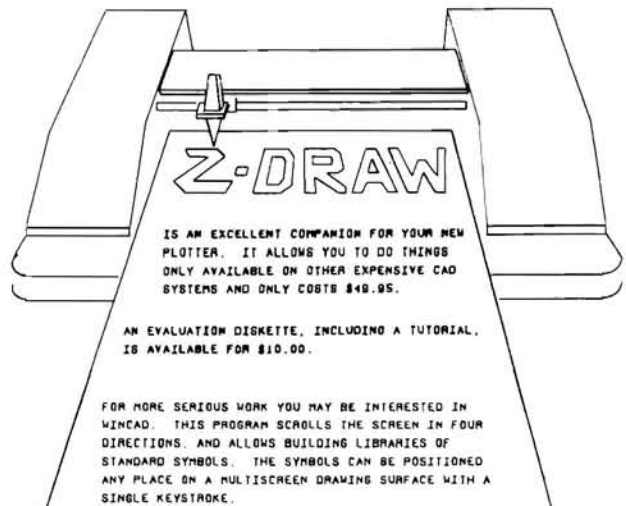
TERMS: NO RISK, MONEY back guarantee. Add \$2 domestic \$4 foreign per order for S&H. Enclose your check or M.O. with your order. Sorry, no charge or phone orders.

Name _____

Address _____

City, State, Zip _____

CP/M Reg. Th Digital Research Corp.



WINCAD IS AVAILABLE FOR \$249.95. VISA AND MASTER CARD ACCEPTED

ASK ABOUT Z-DRAW AND WINCAD AT YOUR LOCAL HEATHKIT ELECTRONIC CENTER.

Redwood
 Development
 7 Redwood Dr.
 Butte, MT 59701
 (406) 484-5610

Z-DRAW AND WINCAD WILL RUN ON Z-100 AND Z-150 COMPUTERS. WHEN ORDERING PLEASE SPECIFY.

Share Your H-89 Peripherals With Brand "X" Computers



Disk Observations & The Power Supply Jinx

Peter Ruber
P.O. Box 502
Oakdale, N.Y. 11769

Like most hobbyists, my greatest pleasure lies in superfluous acquisitions. After all, what is more noble than collecting thousands of first editions, Konica cameras, Gretsch guitars, magazines, electronic parts, ham radio gear and computers? Certainly, stock options and a fat bank account are dull by comparison.

Collecting is in my blood. I've been at it since I was ten; and thirty-odd years later the basement of my house is a veritable "Rat's Nest" of goodies.

Computers have been sneaking into my basement cove since 1981. I have five on a long working desk; my wife has one, as do my three children. And rumor has it that more will arrive at Christmas. I'll use any excuse to buy one.

And herein lies the problem. Sooner or later you'll want more than just the computer and a smattering of software. There is always a pressing need for disk drives and printers.

Unfortunately, certain manufacturers of brand "X" personal computers have done their best to befuddle the computer buying public by establishing proprietary I/O devices so that we are locked into buying that manufacturer's peripherals. Radio Shack and Commodore are two such culprits.

Having invested goodly sums on my H-89 and numerous disk drives, controllers, color cards, serial and parallel cards, expansion boxes and an MX-80 printer, I couldn't justify buying more drives and printers to satisfy the other growing systems. Besides, there are so many H-89 related items that I don't have, that some rational way of financing them must be found. I suspect that when the time comes that I finally possess every H-89 peripheral ever made, I shall probably turn on the switch and get sucked into the CRT.

An \$11.46 MX-80 Interface

My printer is usually mated to the H-89, but every so often I have a need to print out a program listing on my Radio Shack Color Computer and my Commodore-64. This is usually a horrifying procedure of removing the case screws, the paper advance knob, the sound shield, the case cover, the serial cable, the serial board, and a host of other annoyances.

Equally irritating is reassembling the pieces, and I began to dwell on an easier way of switching computers between the MX-80 — one that would only take a few seconds. Fortunately, the serial card I purchased from CNR Engineering for my Color Computer bore a striking resemblance to the 2k Serial Card that came with my MX-80 from Heath, but it terminated in a 4-pin DIN plug at the computer's end.

The Commodore-64 presented a different problem: its serial configuration is only appropriate for Commodore printers. But it will interface with the MX-80 through a Cardco (or similar) parallel card. Things looked promising. I was fond of my MX-80; it was a tireless workhorse that has never exhibited a temperamental act in nearly three years of operation. Thus, I saw no need to buy an overpriced Radio Shack printer that I couldn't use with any other computers; nor buy another one of Commodore's featureless tinker-toy printers. Besides, my wife already had a Commodore 1525 printer for her "64" and it sounds like a Panzer Tank heading for the Battle of the Bulge.

As luck would have it, the MX-80 has a rudimentary Serial Board already built in. It is accessed through a 26-pin connector on which you mount the serial interface card for your computer. That built-in board contains a four position DIP switch and an eight position DIP switch for configuring your word length,

number of stop bits, parity, handshaking, etc. in order to accommodate computers from A(pple) to Z(enith). The baud rate is established by setting the switches on the Serial card that plugs into the printer. Fascinating stuff!

I reasoned that if I created a cable with the appropriate connectors, I could connect many different Serial Interfaces externally, without ever having to open my MX-80 again.

The parts were simple enough: a 26-wire ribbon cable and two (2) 26-pin ribbon header sockets. JDR Microdevices (1224 S. Bascom Avenue, San Jose, CA 95128) carried the ribbon header sockets at \$2.43 each, and a 10-foot length of 26-wire ribbon cable for \$6.60. (If you're addicted to loud suspenders and wish to go first class, you can buy a multi-color ribbon cable of the same length for \$11.60.)

After posting a check with my order, I began to scrounge through my parts boxes for odds and ends that would enable me to build a mock-up cable interface and test my theory. I was in luck. I located a 25-pin single row wire-wrap header, a 25-wire ribbon cable and an old Heath internal disk drive cable with a 34-pin ribbon header socket at one end.

I cut the 25-pin header into three pieces: one with 13 pins, one with 10 pins and one with 2 pins. I glued them together leaving a space where pin #22 would normally be because it wasn't used. Separating the wires from one end of the ribbon cable, I stripped off the insulation and carefully soldered each of the wires to the fabricated header. Then I stripped the wires at the other end.

I clipped the edge card connector from the disk drive cable and then soldered the appropriate 25 wires to the bare end of the ribbon cable. When the operation was complete, I taped off each connection.

I plugged the header into the MX-80, pushed the ribbon socket on to the Serial Interface, and then mounted it to one of the cutout slots in the wall of my Microflash M-89 Expansion Box. I selected a short letter from a working disk and watched the MX-80 spit it out flawlessly.

So far, so good!

But there were a couple of other matters to consider. There is a short wire with a spade lug connector coming out from under the MX-80 circuit board that the documentation instructs you to connect to TP1 on the board if you are using the Parallel connector, and to TP1 on the Serial Card if that is what you have installed. Since the printer seemed to work fine without that wire connected to TP1 on the Serial Card, and since the Radio Shack Color Computer Serial Card by CNR doesn't contain that provision, I drilled a hole in the back of the MX-80 case, installed an ON/OFF switch and wired that cable to it, so that I could make or break the TP1 connection as needed.

Ultimately, however, by experimenting, I discovered that I had performed an unnecessary operation, because the Parallel Interface on my Commodore-64 worked fine with the switch in either position. The only real tricky problem you are likely to encounter when changing serial cards, is the setting of the proper DIP switch positions inside the printer. Usually, though, only one switch on each DIP unit needs to be changed in order to accommodate the other board.

As the DIP switches are relatively close to the back of the printer, it is a simple matter to insert one's finger and, with a little practice, make an instantaneous change. Be certain, though, that the

printer is turned OFF or it might be the last change you make. You can, of course, remove the DIP switches, solder an eight-pin and a sixteen pin socket in place, and run BDIP header sockets to the outside of the case and install the DIP switches at the other end. But that would probably result in Epson refusing to service the printer in the event of a malfunction. Most printer manufacturers delight in making it inconvenient for the user to adjust switches.

One final item should be mentioned. The Parallel Interface will not function as long as you have a Serial Card attached to the MX-80. If you try to print while a Serial Card is installed, you are likely to encounter a DEVICE NOT PRESENT error statement. Merely remove the interface cable you constructed and the MX-80 will work. Conversely, the Serial Cards will work even if you have the Parallel interface attached and the computer connected to it is operating.

Oh, well! You can't have perfection for \$11.46. I might add that the concept behind this project will work with most other printers, such as the Okidata and Olivetti PR-2300 jet ink printer. It will also allow owners of H-89 and Z-110/120 computers to combine several Serial and Parallel interfaces on one printer, without having to resort to expensive printer switch boxes costing hundreds of dollars.

Sharing Your H-89 Disk Drives

The price wars that have been going on quietly in the disk drive market, during the past six to eight months, is a hobbyists dream come true. And H-89 owners have the opportunity of loading their system to a maximum of three 5-1/4" 48 tpi drives on the hard-sector controller, and three 96 tpi DS/DD drives on their soft-sector controller.

With the installation of the HSY Device driver, you can double your storage capacity on the 48 tpi drives, if they're double-sided and you have a whopping total of 2.5 Meg. bytes of storage available.

That's four times the disk storage of an IBM PC and their clones. Besides, who needs a Winchester drive? You can buy a new computer for the price of one of those.

Last Spring, when I attended a large computer show at the Nassau Coliseum, I picked up two of the new Panasonic DS/DD JA-551-2B half-height disk drives for a paltry \$290, including a mounting plate and a Y-cable power connector.

I planned to retire the old Wangco single-sided drive that had come with the H-89, and give my old war horse a modern, contemporary look with two drives installed internally.

I bought Heath's H-89-9 High-Capacity Drive Installation Kit in order to provide the necessary shielding when using these new drives. Quite frankly, I wasn't thrilled when I had finished this project. The new half-height drives are longer than the standard drives and things really get cramped. I had to bend the transistor heat sinks on the power supply board in order to get the drives to fit flush with the front of the cabinet. Worse still, once I installed a new connector cable to the hard-sector controller, there was no room at all, and air circulation came to a standstill.

As I like to take my H-89 apart now and again, I decided to remove the internal drives altogether. Some months earlier, I had purchased a two-drive horizontal cabinet to house two Teac DS/DD half-height 96 tpi drives to work off my soft-sector controller. The cabinet housed two separate power supplies with an

exhaust fan, so I stacked the 96 tpi drives on top of each other on the left, and did the same with the 48 tpi drives on the right.

Since the cabinet contained a master power switch and two additional switches that could be used as write-protect switches (if the installed drives had that provision available), I rewired the power lines so that I could turn each bank of the 48 tpi and 96 tpi drives on and off as I needed them.

Having done this, I decided to add a 34-pin edge card connector near the socket connector on the ribbon cable for the 48 tpi drives that plugged into the back of the H-89, and then to the hard-sector controller. This enabled me to switch the drives between my Radio Shack Color Computer and the H-89, as needed. And as I recently ordered a second H-89 for my wife's use (she has since learned to hate the Commodore-64), she can now use these drives, too, without having to go to additional expense.

Disk Observations

When I purchased my 96 tpi drives from Software Support, Framingham, MA (an early independent supporter of Heath peripherals), I also ordered several boxes of disks that could write in quad density format. I had fully expected to pay about \$45.00 for each box, which was at the time, the going discounted rate for these disks.

Surprisingly enough, I was only billed \$17.95 for the disks, and I suspected there had been a shipping error. There were no manufacturer's labels on the disks, and a note on the invoice said the enclosed disks would do just fine. Being a doubting Thomas when it comes to "super" bargains, I telephoned Software's technical assistance hotline, but they merely confirmed their confidence in the product.

As a precaution, I initialized all the disks under HDOS and spent an entire evening performing one TEST37 after another. I didn't get a single bad sector error message.

Since my business requires a fair amount of correspondence, billing and report writing, I decided to perform an extensive test with one disk. I decided to use one working disk on which I would put everything. After each session, I would back up the data onto separate disks containing letters, invoices, reports and occasional articles.

When my master working disk had filled up all 2400 sectors, I would delete large groups of files and continue filling the disk. Ultimately, there were broken files scattered all over the disk and the drive frequently worked overtime doing its housekeeping chores.

This experiment has been going on for nearly eight months — a long time to use one disk on an almost daily basis without a read/write failure. During this period, as I needed additional disks, I would pick up whatever bargain I came across, including single-sided/single-density disks as low as \$15.00 per box. They formatted quite nicely in both double-density and quad-density format on two sides.

All disks are coated on both sides, and the ability of a drive to read and write a disk rated for single-sided/single-density in a quad-density format is merely an affirmation of the quality of drives, in general, and the durability of the magnetic coating materials used by disk manufacturers.

The Power Supply Jinx

It was comforting to note in a Radio Electronics article last year

that discussed the potential damage a power spike could do to a computer, that the H-89 was probably the very few computers that could withstand a direct lightning hit on its power line. Good news, inasmuch as the south eastern part of Long Island is on the receiving end of violent electrical storms during the summer months — not to mention power surges and voltage drops.

Thus, I would work on the computer during these bleak periods — not because I liked to tempt providence — but because I had confidence in the durability of my equipment.

But I was being plagued with a small problem. My CRT screen would blank out periodically without any apparent cause. And when it did, it came at awkward moments when I had a lot of data in memory. Oddly enough, on each occasion I was able to dump the contents of memory directly to the disk, even though the screen would not accept my keyboard commands.

I spoke with the Heath technician at the Jericho store, and he indicated that a component on the video board might be overheating and shutting down from time to time. He advised me to bring in the H-89 the next time it happened.

The next time was a Sunday afternoon in August, and I was nearing completion on a 30 page report when it happened again. I dumped the report onto the disk just in time, as moments later the entire computer shut down. It was dead. I replaced the fuse and the fan started going, but nothing else would.

I probed the power supply board and the voltage regulator IC's with a VOM. Electrical current was coming into the transformer and coming out, but that's as far as it got. During my probing, I noticed a burnt spot on the Molex plug connected to P101 on the power supply board.

It took a while to pry it off because the plastic had melted and fused the plug to the connector pins. The pins were badly oxidized, as were the metal shoes inside the plug, and that had apparently caused a short circuit. I reasoned that on the previous occasions when the screen blanked out, the connector plug must have been working itself loose during the heat build-up inside the machine.

As I had upgraded my H-89 from an H-19 terminal, I still had the old transformer and a matching Molex plug. I scraped the pins on P101 and did the same to the metal shoes that fit inside the Molex plug. I inserted the wires into the new plug and the computer sprang to life.

But within an hour or so, the same problem occurred. I had probably not cleaned the pins on P101 sufficiently, or the spring action on the metal shoes inside the plug had diminished, so that it caused another heat build-up and another burned out Molex connector.

This time I played it safe. I removed the Molex connector and tack-soldered the seven transformer leads directly to the protruding pins on P101 and taped them off. The result of this surgical transplant is that the H-89 has since performed in a flawless manner.

As an added precaution, however, I reversed the internal fan to blow cold air into the H-89, leaving the top case unclamped on the sides in order to provide an exit for any possible heat build-up. I also mounted another Muffin fan on top of the left rear ventilation holes to provide additional cooling to the high-voltage transformer for the CRT.



Two Useful "C" Functions

Charles R. Winchester
P.O. BOX 5893
Aloha, Oregon 97006

Software Toolworks' C/80 C Compiler is a very good and powerful program. It contains many of the library functions needed to write programs in the C language. Two functions it does not contain, however, are the functions puts.c and gets.c. These two functions provide unformatted I/O to and from the terminal. The function, puts (put string), is similar to the PRINT statement in BASIC, while gets (get string), is similar to BASIC's Line Input statement.

C/80 contains several functions for outputting data, including printf.c. Version 3.0 of C/80 also includes scanf.c which provides formatted input. Since these I/O functions are provided by C/80, you might ask why the need for puts.c and gets.c. One answer is the amount of memory used by the printf.c and scanf.c functions.

As an example, take the simple program for printing "Hello world!" on the terminal:

```
#include "printf.c"
main()
{
    printf("Hello, world!\n");
}
```

This program, when compiled and assembled, uses 2860 bytes of memory. If the program is written as follows, using the function puts.c instead of printf.c, the program uses only 1421 bytes.

```
main()
{
    puts("Hello, world!");
}
#include "puts.c"
```

This is a saving of 1439 bytes of memory. A similar saving will be involved if gets.c is used instead of scanf.c.

Compiling was done with version 3.1 of C/80 and the assembly was done using AS.ABS. AS.ABS is the absolute assembler supplied in the C/80 package.

A smaller, less powerful version of printf.c is included with C/80, but the amount of memory used will still be greater than puts.c uses. For example, if tprintf.c (the smallest version of printf.c) is used, the memory used will be 2322 bytes. This is about 900 bytes larger than the program using puts.c. If fprintf.c (with floats & longs) is used, the program will use 6166 bytes. Why are these programs so big?

The larger functions are more powerful and versatile. They format the data so you can line up the decimal points in a report, as well as right or left justify the data. You can specify the size of a

field and how many digits are to be printed following the decimal point in a number. They do more, and if you need their power and features you should use them. Now, what can puts.c do?

The function, puts.c, can do only one thing. It can output an ASCII character string to the terminal. The string may contain tabs, newlines, and other character constants as required. Also, if you look at the above simple programs, you will notice that the newline character is missing from the version that uses puts.c. When the terminating NULL in the string is encountered by the function, puts.c substitutes a newline for the NULL and outputs it to the terminal. Therefore, the newline is not needed with the standard puts.c function, while it must be added to the string if printf.c is used.

Several of the C textbooks currently on the book shelves make use of puts.c and gets.c functions in their illustrative examples. If you are trying to learn C from one of these texts, it is a real convenience to have these functions available. This was what prompted me to write these two functions in the first place. They provided me with a good exercise in programming and gave me two useful functions for my library.

The function scanf.c is not included in C/80 libraries prior to version 3.0. Therefore, you will find gets.c a very handy string input function if you have one of the earlier versions. I originally wrote these functions using C/80 version 2.0.

Here is the source code, as I wrote it, for the two functions puts.c and gets.c.

```
/* puts.c outputs a string to the terminal. A newline
   is output whenever a NULL is encountered. */

int puts(s)
char *s;
{
    /* Enter a loop to output a string
     * and check for the NULL delimiter.
     */

    while(*s)
    {
        /* See if character is a legal ASCII character */
        if(*s < 0 || *s > '\176')
        {
            putchar('\007'); /* if not legal, ring bell, */
            return(-1); /* and send back error code */
        }
        putchar(*s++); /* Output legal character */
        /* Point to next character */
    } /* Go back for next character */
    putchar('\n'); /* Add newline over NULL */
} /* End of puts.c */
```

```

/* gets.c inputs a string from the terminal. A NULL is
 * added to the end of the string whenever a newline is
 * sent from the terminal. You will need to provide a
 * buffer with MAX+1 locations in which to stash the
 * resulting string. #define MAX in your header file.
 */

char *gets(s)
char *s;
{
    int i;

    for(i=0; (s[i] = getchar()) != '\n' && i <= MAX; ++i)
        ; /* Loop and get characters from terminal */
        /* until a newline or MAX characters input */

    s[i] = 0; /* Add the NULL to end of string */
    return(s); /* Return string */
}

```

The function gets.c “gets” a string from the terminal and substitutes a NULL whenever a newline character is typed. This NULL, written over the newline, provides the proper string terminating character. Now let’s see how we can use gets.c in a program.

What if you want to get a player’s name and age to be used in an interactive game or computer aided instruction program? The function gets.c will do that job for you. You will have to provide a name and set up an array of the proper size, for each string, to hold the characters. This can be done as follows:

```

#include "printf.c"
#define MAX 80 /* Maximum line length */
main()
{
    char name[11]; /* Declare a character array. This */
                  /* array will hold 10 characters */
                  /* and the NULL terminator */
    char age[3]; /* Declare a char array for age */
    char *gets();
    printf("What is your name? "); /* prompt input */
    gets(name); /* then get the name */
    printf("How old are you %s? ", name); /* Use the name */
    gets(age); /* to prompt */

    etc.
}

```

Two character arrays are declared. Each array is made big enough to hold the required number of characters. In this case, 10 characters plus the NULL character (11 total) for the name and 2 digits plus a NULL for the age. (Remember that the NULL character terminates a character string and must be allowed for).

The BASIC statements corresponding to the above program sequence would be as follows:

```

10 DIM N$(10), A$(2): REM SET UP ARRAYS FOR THE NAME AND AGE
100 PRINT "What is your name? ": REM PROMPT INPUT
110 LINE INPUT; N$: REM GET THE NAME
120 PRINT "How old are you ";N$;"?"
130 LINE INPUT; A$: REM GET THE AGE

```

If you enter the source code for the “C” program, above, compile it and then assemble it; the program output will be as follows:

What is your name? **Jack**
How old are you Jack? **12**

The words in bold are what was entered from the terminal after the prompt message appeared.

Notice that printf.c is used in the above example. The printf function makes it easy to insert information, such as a name and/or numerical data in a message. And printf is a powerful program which allows you to convert and format data and insert it directly into a message. That is one reason why it is so big. However, if you are just outputting strings and don’t need to insert data, then “puts” is very useful. It should also be noted that if you DO need printf, don’t use “puts” along with printf, as this will just use more memory. Now let’s see what “puts” CAN do.

The function puts.c can be used to output predefined constants. For example, to clear the screen of my H19 terminal, I simply declare CLR to be the string “\033E”. Then, whenever I want to get a clear screen in a program, I write the program lines:

```
puts(CLR); /* Output clear-screen code */
```

Let’s write the same program using puts.c and also clear the CRT screen:

```

#define CLR "\033E" /* Clear screen code */
#define MAX 80 /* Set line length max */

main()
{
    Char name[11], age[3]; /* Declare the arrays */
    Char *gets();
    puts(CLR); /* Clear the screen */
    puts("What is your name? "); /* Prompt for a name */
    gets(name); /* Now get the name */
    puts("How old are you "); /* Prompt for the age */
    puts(name); /* include the name */
    puts("?"); /* add question mark */
    gets(age); /* Now get the age */
}
#include "puts.c"
#include "gets.c"

```

With the above source code compiled and assembled, the output will be as follows:

What is your name?
Jack
How old are you
Jack
?
12

There are two things wrong with the above approach. First, we have a number of lines outputted instead of a single line per prompt and reply. Second, the age is input and stored as a string. This “age” string will be useless (as stored) if we want to calculate grade level or handicap with the age factor involved. The newline output by the standard puts.c is one of the problems and I will deal with that a little later. First, let’s look at the number problem.

A convenient way to use the gets.c function when inputting numerical data is as follows. (Remember that the gets.c function inputs ASCII characters as a string. The string characters have to be converted to an integer value to be useful numerical data):

```
num_x = atoi(gets(age_a)); /* Input a ASCII number */
                          /* string and convert */
                          /* it to integer num_x. */
```

Let’s see how this program line works. C always starts at the innermost level when nested parentheses are encountered. The order of events will, therefore, be as follows:

1. The function "gets" is called (with the address of the buffer age_a supplied) and the program will pause for an input from the terminal. One or more digits will then be typed on the terminal, followed by a newline. As soon as the newline is typed, "gets" completes its job by writing a NULL over the newline and stashing the characters in the array age_a.

2. Now the function atoi is called. If all the characters in the string age_a are digits, the numerical value of the string is determined and stored as the integer variable num_x.

The function atoi.c (ASCII to integer) is one of the functions supplied with C/80 vers. 3.0. (The source code for atoi.c is also in the book "The C Programming Language" by Kernighan & Ritchie).

The above program line is similar to the following program lines in BASIC.

```
100 LINE INPUT; A$:      REM INPUT ASCII NUMBER STRING
110 X = VAL(A$):        REM CONVERT IT TO A NUMBER
```

Now let's go a little farther. The above source code for puts.c was written to follow the way puts.c is described in the publications I have read. The version I use actually does NOT insert the newline at the end of the string. I just eliminated the line "putchar('\n');" from the source code. Now I can insert names into a string and keep everything on the same line. The newline can be added as I need it.

To be able to use these functions, I have inserted them in the "stdlib.c" library using the "#ifneed" compilation directive provided with C/80 vers. 3.0. This library now contains puts.c and gets.c, along with the atoi.c function used in the following illustration example. All I have to do to use these functions is #include "stdlib.c" at the end of the program.

A rule for using puts.c must now be mentioned. When you place a string constant (such as CLR in the above example) between the parentheses of the function puts(), you are in effect passing the address of the string to the function. Also, if you insert a string inside quotes between the parentheses, puts.c will output the string as written. Newlines may be inserted anywhere in the string between the quotes. However, a newline CANNOT be inserted between the function parentheses with a string constant. See the examples following:

```
puts(name\n);          /* A no-no */
puts(name"\n");       /* Also a no-no */
puts(name);           /* Add the newline */
putchar('\n');        /* this way */
puts("Hello world!\n"); /* This is OK also */
```

Now let's see how all these things can be tied together in a program.

```
/* A program to test puts.c and gets.c. The printf function
 * will be used to show that we input the right material.
 */
#define CLR      "\033E" /* H19 clear screen code string. */
#include "tprintf.c" /* Use the smallest printf function. */
#define MAX 80     /* Define line length for gets.c */
main()
{
    /* Declare arrays for two names and a temporary buffer
     * for stashing the ASCII string which represents the
     * ages of ...e players. The function atoi uses this
```

```
 * string to convert to an integer.
 */
char name1[11], name2[11], a_buf[3], *gets();
int age1, age2; /* declare for ages */

puts(CLR); /* Clear screen */
puts("\n\tPlayer Number 1, ");
puts("What is your name? "); /* Prompt for a name */
gets(name1); /* Get the name */
puts("\n\tHow old are you "); /* Prompt for age */
puts(name1); /* with name. */
puts("? ");
age1 = atoi(gets(a_buf)); /* Get age #1 */
puts("\n\tPlayer Number 2, ");
puts("What is your name? "); /* Prompt for a name */
gets(name2); /* Get the name */
puts("\n\tHow old are you "); /* Prompt for age */
puts(name2); /* with name. */
puts("? ");
age2 = atoi(gets(a_buf)); /* Get age #2 */
puts("\n\n Now ");
puts(name1);
puts(" and ");
puts(name2);
puts(" your ages will be used to establish\n");
puts("the difficulty level");
puts(" for each of you. There will also be a\n");
puts("handicap calculated if your ages ");
puts("differ by more than two years.\n");

/* Prove we got the right information with printf.c */

printf("\n%s is %d and %s is %d\n",
       name1, age1, name2, age2);
}
```

```
#include "stdlib.c"
```

Note: The printf line was inserted only to show that the proper data was input and stashed in memory.

The output will appear like this:

Player number 1, what is your name? **Jack**

How old are you Jack? **12**

Player number 2, what is your name? **Jill**

How old are you Jill? **11**

Now Jack and Jill, your ages will be used to establish the difficulty level for each of you. There will also be a handicap calculated if your ages differ by more than two years.

Jack is 12 and Jill is 11.

The program demonstrates two things:

1. It shows how to couple names into a sentence (or string).
2. It shows that the conversion from a string to an integer, as shown, does work.

You wouldn't need to convert the number string into an integer if it were strictly a conversational question, but, where the number is to be used in computation, the number must be in integer (or float) format.

One more thing. In the above program example, the buffer a_buf only has room for 2 characters plus the NULL. In this case, it is a dedicated buffer. If you are going to use the characters from this buffer in later portions of the program, you would probably do it this way. Just remember to declare a separate buffer for each age. Now let's see how much memory is needed for the fin-

ished program.

If we compile and assemble the above program (after deleting the #include "tprintf.c" directive and the lines using printf), the program size is 2212 bytes.

Now, just for kicks, we will rewrite the program using tprintf.c (the smallest printf.c function) and scanf.c. To make the comparison fair, we need to remove the FLOAT and LONG capabilities from scanf.c. We do this by removing the comment symbol from in front of the "#define NOFLOAT" and "#define NOLONG" declarations at the beginning of scanf.c.

```
#define CLR      "\033E"      /* H19 clear screen code. */
#include "tprintf.c" /* Include small printf function */
#include "scanf.c" /* and scanf function. */

main()
{
    char name1[10], name2[]; /* Declare for two names */
    int age1, age2; /* and for 2 ages. */

    printf(CLR); /* Clear the screen */

    /* Send out prompt and get name for 1st player. */
    printf("\n\tPlayer number 1, what is your name? ");
    scanf("%s", name1);

    /* Get the 1st player's age with name inserted. */
    printf("\n\tHow old are you %s? ", name1);
    scanf("%d", &age1);

    /* Send out prompt and get name for 2nd player. */
    printf("\n\tPlayer number 2, what is your name? ");
    scanf("%s", name2);

    /* Get the 2nd player's age with name inserted. */
    printf("\n\tHow old are you %s? ", name2);
    scanf("%d", &age2);

    /* Now start with the instructions. */

    printf("\n\nNow %s and %s, ", name1, name2);
    printf("your ages will be used to establish\n");
    printf("the difficulty level ");
    printf("for each of you. There will also be a\n");
    printf("handicap calculated if your ages ");
    printf("differ by more than two years.\n");
}
}
```

Compiling and assembling the above source code will generate approximately 4.8K bytes of object code. The scanf.c and tprintf.c functions use about 2.6K bytes more memory than puts.c and gets.c do while both versions output the same message to the terminal.

Of course, the scanf and printf functions provide formatting and conversion capabilities that are lacking in the two smaller functions. Using a big tractor and trailer rig to do a job that only requires a station wagon is rather wasteful. On the other hand, the tractor and trailer will move a house load of furniture more efficiently than will a station wagon.

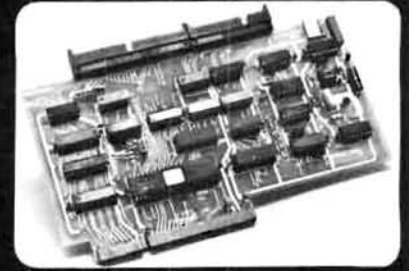
The main message here is that you should use the tools required to do the job. If a function you need is not included in your library, then C/80 provides you with the power to write it. If the function has been previously defined, then you should closely follow that definition so the program you write will be portable. After a function is written, it can be installed in your library for future use.

I hope these program functions will be of use to you. Happy programming!



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

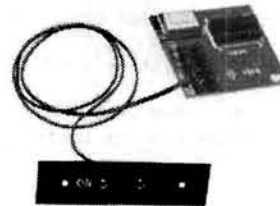
Includes controller board CP/M boot prom, I/O decoder prom, hardware/software manuals BIOS source listing. HDOS driver now available for \$50.00.

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

THE ORIGINAL Z100 SPEED MODULE RUN YOUR Z100 PROGRAMS FASTER



The ZS100 runs the Z100 CPU 50% faster, (7.5 MHz) in 8088 mode.

The ZS100 installs easily with no soldering.

The ZS100 is externally switchable between speed mode and normal.

The ZS100 improves the time performance of applications packages with no software modifications needed.

The ZS100 is for all Heath/Zenith 100, 110 and 120 series computers.

SPEND LESS TIME WAITING FOR YOUR COMPUTER

Contact your local Heath/Zenith dealer to buy this cost effective Z100 enhancement, or contact:

Controlled Data Recording Systems, Inc.
7210 Clairemont Mesa Blvd.
San Diego, Ca. 92111
Telephone (619) 560-1272



Patch Page

Pat Swayne
HUG Software Engineer

This article contains patches for Z-DOS KEYMAP (885-3010-37), as updated to work with MS-DOS 2.13, and the program SEE.COM from disk 885-3014-37.

KEYMAP Patches

When the Z-DOS/MS-DOS KEYMAP program for Z-100 computers was originally released, it would work under Z-DOS, and not under MS-DOS version 2. Later it was updated to run under MS-DOS version 2.13, and re-released. In fact, a new version (KEYMAP2) was made for MS-DOS 2 and included on the disk. However, newer versions of MS-DOS 2 have been released under which KEYMAP2 will not work. If you have KEYMAP, and the KEYMAP2 programs on your disk will not work under your release of MS-DOS version 2, you can modify the programs KEYMAP2.ASM and UNMAP2.ASM as follows. First, format a blank disk and copy KEYMAP2.ASM, UNMAP2.ASM, and ASM2.BAT to it from the KEYMAP distribution disk. Also copy LINK.EXE and EXE2BIN.EXE to it from your MS-DOS distribution disks, and MASM.EXE from either your Z-DOS distribution disks, or from the Programmer's Utility Pack if you have it. If you have only one drive, you will have to copy an editor to the new disk (EDLIN or whatever editor you use).

Load the file KEYMAP2.ASM into your editor and locate the following lines:

```
ADD    BX,3           ;SKIP "MOV SI" INST.
CMP    M,0EAH        ;KEYMAP ALREADY IN?
JZ     ITSIN         ;YES
MOV    M,0EAH        ;PUT FAR JUMP IN
```

Edit the lines so that they now look like this:

```
ADD    BX,3           ;SKIP "MOV SI" INST.
CKIN:  CMP    M,0EAH   ;KEYMAP ALREADY IN?
      JZ     ITSIN    ;YES
      CMP    M,0FAH   ;VERSION 2.18 OR 2.19?
      JNZ    NOT218   ;NO
      ADD    BX,1FH   ;ELSE, CORRECT FOR IT
      JMP    CKIN     ;AND CHECK IF IN AGAIN
NOT218: MOV    M,0EAH   ;PUT FAR JUMP IN
```

Write the edited file to your disk. Load UNMAP2.ASM into your editor and locate these lines:

```
ADD    DI,3           ;SKIP "MOV SI" INST.
CMP    Byte Ptr [DI],0EAH ;TEST FOR FAR JUMP
JNZ    NOTMAP         ;NOT MAPPED, EXIT
INC    DI             ;POINT TO KEYMAP ADDRESS
```

Edit the lines so that they now look like this:

```
ADD    DI,3           ;SKIP "MOV SI" INST.
CKIN:  CMP    Byte Ptr [DI],0EAH ;TEST FOR FAR JUMP
      JNZ    MAPPED   ;KEYMAP IS IN
      CMP    Byte Ptr [DI],0FAH ;MS-DOS 2.18 OR 2.19?
      JZ     NOTMAP   ;NOT MAPPED, EXIT
      ADD    DI,1FH   ;IF SO, CORRECT FOR IT
      JMP    CKIN     ;AND CHECK IF IN AGAIN
MAPPED: INC    DI     ;POINT TO KEYMAP ADDRESS
```

Write the edited file to your disk. Log on to the disk (if you haven't already) and enter:

ASM2 KEYMAP2

You will see a "no stack" error message when the LINK program runs. Disregard that message. When the computer has finished working on KEYMAP2, enter:

ASM2 UNMAP2

When the computer is done, you will have a new KEYMAP2.COM and UNMAP2.COM that you can use to replace the original ones (but not on your back-up copy of the KEYMAP disk!). You will need to make new KEYWS2.COM, KEYBAS2.COM, and/or KEYSYS2.COM files if you use any of those, by using KEYCON to configure the new KEYMAP2.COM. You might want to first run the old KEYWS, etc. through KEYCON and jot down what the key responses are, so that you can configure the new ones correctly.

After you have made these patches, the KEYMAP2 and UNMAP2 programs should work on any release of MS-DOS version 2 for Z-100 computers through 2.19. If you have the original release of KEYMAP that does not have the KEYMAP2 programs on it, you can get an update by sending your original disk and \$5.00 to HUG, with the package marked "Attention Nancy Strunk". Our stock has been updated with the patches for the newer releases of MS-DOS.

SEE.COM Patches

The program SEE.COM on disk 885-3014-37 was designed to work on both Z-100 and Z-150 (Z-100-PC) series computers. The Z-150 code was designed at a time when I only had limited access to a Z-150 computer. As a result, two versions have been shipped that have problems affecting operation on a Z-150. If your SEE.COM is dated before 10-25-84 (directory date), lines longer than 80 characters will wrap on the screen, which should not happen. To fix the problem, use the DEBUG program, and enter the following commands:

```
-NSEE.COM
-L
-ECDE
xxxx:0CDE B2 C3
-W
WRITING 0D45 BYTES
-Q
```

This example assumes that both DEBUG and SEE.COM are on the currently logged disk.

If your SEE.COM is dated 10-25-84, the left and right arrow keys will not work on a Z-150 for sideways scrolling. To fix the problem, enter these commands in DEBUG.

```
-NSEE.COM
-L
-E3D1
xxxx:03D1 43 00
-W
```

Enter this patch only if you use SEE on a Z-150.



Shorten: A Program To Save Memory

In BASIC Programs

In an informative article, "Inside Microsoft Basic", REMark Vol. 5, N8, D.D. Dodgen explained a number of ways to save memory when programming in BASIC. Kenneth Mortimer responded (Buggin' HUG, REMark Vol. 5, N10) that the tips given by Dodgen were contrary to good programming practice. Both "experts" are correct, there being a time and a place for everything. On a single user microcomputer, there is no point in saving memory unless the program won't fit or, in the case of the MBASIC* interpreter, won't manage the string space efficiently. In our case, a number of programs use all available memory and still require expansion. It is, therefore, helpful to use some of Dodgen's ideas. However, because the programs are long and intricate, it is very desirable to maintain some readable code. We think that we have come up with a good compromise: Use the computer to solve the problem!

Next time you write a BASIC program, make your source code as readable as you like. Use REMARKS liberally and leave blanks between all words. Use one statement per line, unless it's necessary to combine statements on a line (ex. IF...THEN, ELSE). In general, increase space for the sake of better readability. If, when you try to run your program, you find that your program is too big, don't panic!

With SHORTEN you can shorten your program. SHORTEN will cut the average BASIC program by about 20%. Chances are, this savings will allow you to run your program. How does SHORTEN save space?

Your BASIC program is input. Output, with a different filename, is the same program, devoid of REMARKS and blanks. Lines are combined wherever possible, saving the space taken up by unnecessary line numbers. The lines of the output file can be as long as a BASIC line can be: 255 characters.

The output file won't be easy to read. That's okay since your original, readable file will still be there. If you forget how your program works or if you need to make changes, you can refer to your original file. When you make changes, edit the original file. Then use SHORTEN again to make a run version.

Requirements For The Input Program

Your original file must be an ASCII file. After writing the program in the interpreter, use the SAVE "Filename",A command. Another approach is to use a text editor, such as PIE**, to write the code. The keywords REM, IF, THEN, ELSE, ON, GOTO, and GOSUB must be written in uppercase letters. This requirement could be changed (see lines 380/440, 640, 1010, and 1020) to make SHORTEN recognize keywords of both upper- and lower-case letters.

No other requirements must be met in ordinary BASIC programs before they can be SHORTENed. For example, before combining two statements, SHORTEN closes any unclosed quotes on the first line.

*Patrick Brans
Southern Regional Research Center
U. S. Department of Agriculture
New Orleans, Louisiana 70179*

Combining Line

Statements are combined whenever possible, creating lines of up to 255 characters. Lines are combined without changing the logic of the original program. Line references and conditional statements are accounted for.

Use the function FRE(0) to find out how many bytes SHORTEN saved. Load your original program in MBASIC. Use the statement PRINT FRE(0) to find out how much space is left. The number returned is the number of available bytes after your program was loaded. Do the same for your shortened version. Compare the two results to find out how much space was saved by SHORTEN. (Note that the amount of memory available will decrease further once the CLEAR and DIM statements are executed, but the savings from SHORTENING will be the same before or after CLEAR and DIM are reached.)

Once shortened, your program may well fit in memory. You can increase string space with the CLEAR statement by the amount of available memory shown by the PRINT FRE(0) statement. You could even use a statement, A=FRE(0), followed by CLEAR A to automatically clear the maximum string space. To retain the convenience of the interpreter while developing programs, use smaller arrays and smaller amounts of string space. When you are ready to use the program, increase the array sizes and the amount of space CLEARED before running SHORTEN.

You should find that execution speed of SHORTENed programs is almost unchanged. Besides making more memory available, SHORTENed programs save disk space, especially when the SHORTENed ASCII version is loaded into MBASIC and saved in tokenized form.

The major drawback to SHORTEN is the time spent running

SHORTEN. A compiled version runs 4 times as fast as the interpreted version. The only change to the interpreted code, before compiling, is the deletion of the CLEAR statement.

Shorten

```

10 REM CLEAR 5000
20 DIM NUM$(300)
30 A%=0 ' A% IS THE INDEX FOR NUM$
40 ES=CHR$(27)+"E"
50 REM*****
60 REM GET FILENAMES
70 REM*****
80 INPUT "ENTER FILE YOU WISH TO SHORTEN ";O$
90 INPUT "ENTER A NAME FOR YOUR RUN VERSION ";N$
100 IF N$<>O$ GOTO 140
110 PRINT
120 PRINT"RUN VERSION CANNOT HAVE THE SAME NAME AS THE
ORIGINAL VERSION!"
130 PRINT:GOTO 90
140 IF (INSTR(O$,".BAS")=0) OR (INSTR(O$,".bas"))
THEN O$=O$+".BAS"
150 IF (INSTR(N$,".BAS")=0) OR (INSTR(N$,".bas"))
THEN N$=N$+".BAS"
160 PRINT:PRINT
170 INPUT"DO YOU WANT YOUR RUN VERSION TO BE DEVOID
OF BLANKS (Y/N) ";A$
180 IF (A$<>"Y") AND (A$<>"N")
THEN PRINT"ENTER 'Y' OR 'N':":GOTO 170
190 B$=(A$="Y")
200 INPUT"DO YOU WANT TO COMBINE LINES (Y/N)";A$
210 IF (A$<>"Y") AND (A$<>"N")
THEN PRINT"ENTER 'Y' OR 'N':":GOTO 200
220 IF A$="N" THEN MAX%=0:GOTO 290
230 PRINT"ENTER THE MAXIMUM LINESIZE FOR "N$:INPUT LS%
240 IF LS%>255 THEN PRINT:
PRINT"ERROR! BASIC LINES CANNOT EXCEED 255 @
CHARACTERS. TRY AGAIN.":PRINT:GOTO 230
250 MAX%=LS%
260 REM*****
270 REM 1'ST PASS
280 REM*****
290 PRINT ES
300 FOR I%=1 TO 10
310 PRINT
320 NEXT
330 PRINT" 1'ST PASS
340 OPEN "I",#1,O$
350 IF EOF(1) GOTO 460
360 LINE INPUT#1,LN$
370 GOSUB 1470 ' DELETE QUOTES
380 R$="GOTO" :GOSUB 1650 ' FIND LINE REFERENCES
390 R$="GOSUB" :GOSUB 1650
400 R$="THEN" :GOSUB 1650
410 R$="ELSE" :GOSUB 1650
420 R$="ERL" :GOSUB 1650
430 R$="RESUME" :GOSUB 1650
440 R$="ERROR" :GOSUB 1650
450 GOTO 350
460 CLOSE #1
470 REM*****
480 REM SECOND PASS
490 REM*****
500 PRINT ES
510 FOR I%=1 TO 10
520 PRINT
530 NEXT
540 PRINT" 2'ND PASS
550 PRINT
560 PRINT"*****
570 PRINT
580 OPEN "I",#1,O$
590 OPEN "O",#2,N$
600 Q%=0:R%=0:S%=NS$ ' INITIALIZE FLAGS AND S$
610 IF EOF(1) GOTO 1820
620 GOSUB 910
630 L1$=LN$

```

```

640 IF (INSTR(L1$,"IF")<>0) OR (INSTR(L1$,"ELSE")<>0)
THEN @ PRINT L1$:PRINT#2,L1$:GOTO 600
650 Q%=0:R%=0:S%=NS$
660 IF EOF(1) THEN PRINT L1$:PRINT#2 ,L1$:GOTO 1820
670 GOSUB 930
680 GOSUB 1330
690 FOR I%=0 TO A%
700 IF NUM$(I%)=NM$ GOTO 730
710 NEXT
720 GOTO 830
730 N2$=NM$:L2$=LN$:LN$=L1$
740 GOSUB 1330
750 IF LEN(LN$)>0 GOTO 810
760 FOR I%=0 TO A%
770 IF NUM$(I%)=NM$ GOTO 790
780 NEXT:GOTO 800
790 PRINT:PRINT:
PRINT" WARNING! LINE "NM$" IS REFERENCED.":
PRINT:@PRINT:PRINT:
PRINT#2 ,NM$"REM THIS LINE IS REFERENCED!"
800 L1$=N2$+L2$:GOTO 640
810 PRINT L1$:PRINT#2 ,L1$
820 L1$=N2$+L2$:GOTO 640
830 IF LEN(LN$)=0 GOTO 650
840 N2$=NM$:L2$=LN$:LN$=L1$
850 GOSUB 1330
860 IF LEN(LN$)=0 THEN L1$=NM$+L2$:GOTO 640
870 GOSUB 1210
880 IF (LEN(L1$)+LEN(L2$)+1)<=MAX% THEN L1$=L1$+"
"+L2$:GOTO 640
890 PRINT L1$:PRINT#2 ,L1$
900 L1$=N2$+L2$:GOTO 640
910 REM*****
920 REM INPUT ROUTINE
930 REM*****
940 LINE INPUT #1,LN$
950 GOSUB 1010
960 IF C$=CHR$(64) GOTO 940
970 RETURN
980 REM*****
990 REM SUBROUTINE TO DELETE BLANKS AND REMARKS
1000 REM*****
1010 RM%=INSTR(LN$,"REM")
1020 CRM%=INSTR(LN$,".REM")
1030 CA%=INSTR(LN$,".")
1040 FOR I%=1 TO LEN(LN$)
1050 R%=( (I%=RM%) OR (I%=CRM%) OR (I%=CA%) )
AND NOT Q%) OR R%
1060 C$=MID$(LN$,I%,1)
1070 R%=( (C$=CHR$(39)) AND NOT Q%) OR R%
' LOOKS FOR " ' "
1080 IF R% THEN GOTO 1130
1090 IF C$=CHR$(34) THEN Q%=NOT Q%
1100 IF C$=" " AND (B% AND NOT Q%) THEN GOTO 1130
1110 IF C$=CHR$(64) GOTO 1130
1120 S$=S$+C$
1130 NEXT
1140 LN$=S$
1150 RETURN
1160 REM*****
1170 REM ROUTINE TO COUNT QUOTES
1180 REM IF THERE ARE AN ODD NUMBER OF QUOTES IN THE
1190 REM LINE, A QUOTE IS ADDED TO THE END.
1200 REM*****
1210 I%=0
1220 NQ%=0
1230 I%=INSTR(I%+1,L1$,CHR$(34))
1240 IF I%>0 THEN NQ%=NQ%+1:GOTO 1230
1250 IF (NQ% MOD 2)=0 THEN RETURN
1260 L1$=L1$+CHR$(34)
1270 RETURN
1280 REM*****
1290 REM NEXT ROUTINE SEPARATES A LINE FROM ITS LINENUMBER.
1300 REM LN$ IS RETURNED AS THE LINE WITHOUT THE LINENUMBER
1310 REM NM$ IS RETURNED AS THE LINENUMBER
1320 REM*****
1330 I%=0:NM$=NS$
1340 I%=I%+1:C$=MID$(LN$,I%,1)

```

```

1350 IF (C$<CHR$(48)) OR (C$>CHR$(57)) GOTO 1380
1360 NMS=NMS + C$
1370 GOTO 1340
1380 LNE$=MID$(LNE$,I%)
1390 IF LNE$="" THEN LNE$=NS$
1400 RETURN
1410 REM*****
1420 REM ROUTINE TO DELETE QUOTES AND REMARKS FROM LINE
    LNE$.
1430 REM LNE$ IS COPIED INTO L2$ AND ALL CHANGES ARE
1440 REM MADE TO L2$. THIS IS DONE SO THAT THE ORIGINAL
1450 REM LINE WON'T BE RUINED.
1460 REM*****
1470 L2$=LNE$
1480 I%=0
1490 J%=INSTR(I%+1,L2$,CHR$(34))
1500 IF J%=0 GOTO 1550
1510 I%=INSTR(J%+1,L2$,CHR$(34))
1520 IF I%=0 THEN T$=NS$ ELSE T$=MID$(L2$,I%+1)
1530 L2$=LEFT$(L2$,J%-1)+T$
1540 GOTO 1480
1550 J%=INSTR(L2$,"REM")
1560 IF J%<0 THEN L2$=LEFT$(L2$,J%-1)
1570 J%=INSTR(L2$,CHR$(39))
1580 IF J%<0 THEN L2$=LEFT$(L2$,J%-1)
1590 RETURN
1600 REM*****
1610 REM      ROUTINE TO FIND LINE REFERENCES
1620 REM ALL REFERENCED LINENUMBERS ARE PUT INTO THE
1630 REM ARRAY NUM$ INDEXED BY A%
1640 REM*****
1650 I%=0
1660 I%=I%+1
1670 I%=INSTR(I%,L2$,R$)
1680 IF I%=0 GOTO 1810
1690 I%=I%+LEN(R$)
1700 C$=MID$(L2$,I%,1)
1710 IF C$=CHR$(32) OR C$=CHR$(61) THEN I%=I%+1:
    GOTO 1700
1720 IF (C$<CHR$(48) OR C$>CHR$(57)) GOTO 1660
1730 A%=A%+1
1740 NUM$(A%)=NUM$(A%)+C$
1750 I%=I%+1
1760 C$=MID$(L2$,I%,1)
1770 IF C$=CHR$(32) THEN I%=I%+1:GOTO 1760
1780 IF C$=CHR$(44) THEN A%=A%+1:GOTO 1750
1790 IF (C$<CHR$(48) OR C$>CHR$(57)) GOTO 1660
1800 GOTO 1740
1810 RETURN
1820 REM*****
1830 END

```



The Software Toolworks presents:
THE FUTURE

WELCOME TO THE FUTURE, FOLKS. THE COMPUTER AND THE SOFTWARE THAT MAKES IT TICK.

THE CHILDREN OF THE FUTURE. PLAYING ADVENTURE, WORD WAGGLE, AND MYCHESS. LEARNIN' COMPUTER OPERATION AND HAVIN' A GOOD TIME.

OH.

THAT'S NOT THE WAY YOU SPELL SILICON.

THE PARENTS OF THE FUTURE. USING MYCALC TO ADJUST THE FAMILY BUDGET AND COMPUTER CHEF TO PLAN TONIGHT'S DINNER.

WE'RE BROKE!

THAT'S OKAY MONEY. HOW DO YOU FEEL ABOUT SPAM AU JUS'?

MOM! DAD! "THE A-TEAM" IS ON TV!

WELL, ALMOST THE FUTURE.

Languages, games, productivity software, computer cookbooks and more are formatted for IBM PC, PCjr, most CP/M systems and Heath/Zenith HDOS. They range in price from \$19.95 to \$59.95. For a free 41 product catalog contact The Software Toolworks, 15233 Ventura Blvd., Sherman Oaks, CA 91103, (818) 986-4885.

EMULATE

A program which allows the H89 to read/write to the following disk formats.

Osborne 1	SSDD	Morrow MD2	SSDD	Cromemco	SSDD
Osborne 1	SSDD	Morrow MD3	DSDD	Cromemco	DSDD
Xerox 820	SSDD	Epson QX-10	DSDD	CDR 40TK	DSXD
Xerox 820	SSDD	Televideo 802	DSDD	CDR 80TK	DSXD
DEC VT180	SSDD	Actrix	SSDD	NEC 8001	SSDD
Ampro	SSDD	TRS80/Omikron	SSDD	Eagle II	SSDD
DEC Rainbow	SSDD	TRS80-4 CP/M	SSDD	Z100 40TK	DSDD

A universal format program will be supplied as a free update. The H37 version requires 64K of RAM and the use of a modified version of CP/M 2.2.03 or .04 BIOS which is included with the program. Allows the use of virtual drives and reading of 40 track disks in an 80 track drive.

Must include your CP/M s/n when ordering.

For H37 with Heath CP/M \$59

**Limited Version For
CDR controller \$39**

Automatic Repeat

Simple plug-in installation of the REP2 gives your H89/H19 keyboard the same auto-repeat function you get with a Z100. Provision for a defeat switch.

A Must For Word Processing!
 Kit **\$32**
 Assembled **\$40**

Real Time Clock

Install the TIM2 in a left expansion slot of your H89 to have date and time keeping with battery backup. Requires soldering 4 wires to the CPU board.

Kit **\$55**
 Assembled **\$65**
Software on Disk \$10
 (Specify Format)

CDR Controllers At Discount!

For H89 FDC880H \$345

For H8 FDCH8 Call For Quote

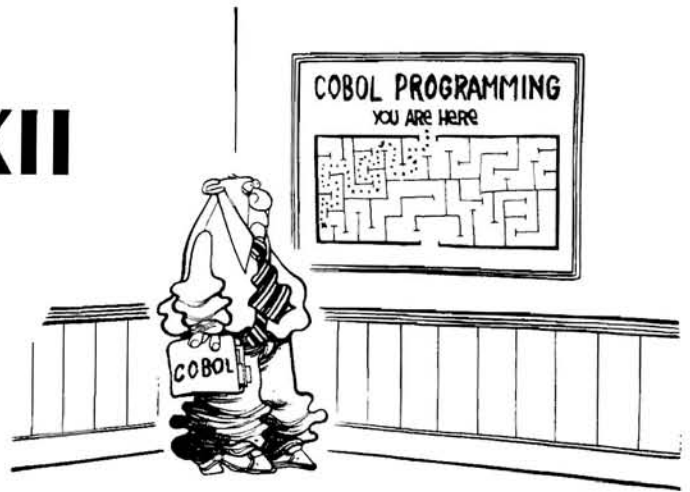
The Software Toolworks® - We Sell It At Discount!
 Other Commercial Software at Discount - Call / Let Us Quote!
 Downloading Service for many CP/M Formats - Only \$5/Disk + \$5/Order.
 Customer Supplies Formatted Destination Disks.

WE PAY POSTAGE! CA Residents Add 6% tax.
 Call or Write For Catalog.

ANALYTICAL PRODUCTS 209/564-3687
 20663 Ave 352 Woodlake, CA 93286

COBOL Corner XII

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



Introduction

I hope that you readers have your "homework" assignment completed. I will assume that you have, so I will proceed with the Program #5, CODE.

Procedure Division

```
000-PRINT-ACCT-REC-REPORT.  
  OPEN INPUT FILE1  
  OUTPUT ACCT-REC-REPORT.  
  PERFORM 100-INITIALIZE-VARIABLE-FIELDS.  
  PERFORM 870-PRINT-REPORT-HEADING.  
  PERFORM 800-READ-ACCT-REC-RECORD.  
  PERFORM 200-PROCESS-ACCT-REC-RECORD  
  UNTIL  
    WS-END-OF-FILE-SW IS EQUAL TO "YES".  
  PERFORM 700-PRINT-PAGE-TOTAL-LINE.  
  PERFORM 750-PRINT-REPORT-TOTAL-LINE.  
  CLOSE FILE1  
  ACCT-REC-REPORT.  
  STOP RUN.
```

Note the changes from Programs #3 and #4. Some of them are as follows:

1. We have added numbered modules. Do they match your Structure Chart?
2. We have a new module to print the report headings. The first thing we want to do when we print a report is put the report headings and column headings on the first report page after we finish the initializing.
3. We next Read the first record (as we have been doing) with the 800 READ module.
4. Then we go into the processing loop with the 200 PERFORM/UNTIL module, as we have done before.
5. After processing all the input records for each page, we use the 700 module to put the page total on the page. We use this module to add the page total to each additional page and on the last report page.
6. The 750 module puts the report total on the last page, as we have done before.

```
100-INITIALIZE-VARIABLE-FIELDS.  
  MOVE "NO " TO WS-END-OF-FILE-SW.  
  MOVE ZEROS TO WS-PAGE-COUNT.  
  MOVE ZEROS TO WS-LINES-USED.  
  MOVE WS-MONTH TO H1-MONTH.  
  MOVE WS-DAY TO H1-DAY.  
  MOVE WS-YEAR TO H1-YEAR.  
  MOVE ZEROS TO WS-TOTAL-ACCUMS.
```

This module has a few changes from previous programs. Note that we must initialize our WORKING-STORAGE AREA counters and accumulators to zeros. Did you CODE for the WS-TOTAL-ACCUMS? Also, note how we have "moved" the date literals that we stored in the WS-DATE-AREA into the first header with the MOVE statements. Is this clear? I believe it will clear up for you as we use it more!

```
200-PROCESS-ACCT-REC-RECORD.  
  MOVE AC-CUST-ACCT-NBR TO DL-CUST-ACCT-NBR.  
  MOVE AC-CUST-NAME TO DL-CUST-NAME.  
  MOVE AC-CUST-ADDRESS TO DL-CUST-ADDRESS.  
  MOVE AC-CUST-CITY TO DL-CUST-CITY.  
  MOVE AC-CUST-STATE TO DL-CUST-STATE.  
  MOVE AC-CUST-ZIP TO DL-CUST-ZIP.  
  MOVE AC-CUST-ACCT-BAL TO DL-CUST-ACCT-BAL.  
  MOVE DL-DETAIL-LINE TO ACCT-REC-LINE.  
  (1) MOVE 1 TO WS-LINE-SPACING.  
  (2) PERFORM 890-WRITE-REPORT-LINE.  
  ADD AC-CUST-ACCT-BAL TO WS-PAGE-TOT-ACCUM.  
  ADD AC-CUST-ACCT-BAL TO WS-RPT-TOT-ACCUM.  
  (3) IF WS-LINES-USED IS GREATER THAN WS-LINES-PER-PAGE  
  PERFORM 700-PRINT-PAGE-TOTAL-LINE  
  PERFORM 870-PRINT-REPORT-HEADING.  
  PERFORM 800-READ-ACCT-REC-RECORD.
```

Notice that we have the following changes in this module as compared to previous programs:

1. We "SET" the report line spacing by statement (1) to single-space.
2. We use the 890 module, line (2), each time we want to write the detail line in the body of the report.
3. We need the two (2) accumulators for SUMMING the Page and Report Totals.
4. Statement (3) uses the "IF" verb to test how many lines we have on the page. If we have +50 or more lines printed, the VALUE specified for the WS-LINES-PER-PAGE will cause the program to go to the 700 module and print the page total on that page and then go to the 870 module to add the report headings on the next page before we Read and Process another record. Do you see how this page count and line control works? Be sure to study and understand this IF statement!

```
700-PRINT-PAGE-TOTAL-LINE.  
  MOVE WS-PAGE-TOT-ACCUM TO PL-PAGE-TOT.  
  MOVE PL-PAGE-TOT TO ACCT-REC-LINE.  
  (1) MOVE ZEROS TO WS-PAGE-TOT-ACCUM.  
  (2) MOVE 2 TO WS-LINE-SPACING.  
  PERFORM 890-WRITE-REPORT-LINE.
```


In the 700 module, we Move the Page accumulator total into the page total and then we Move the entire Page-Total-Line into the report-line. Before we can forget, we initialize the page accumulator back to zeros, so we can use it to SUM the next page total (see statement (1)). Statement (2) "SETS" the line spacing to double-space. Next, we will again use the 890 module to write the page total-line.

```
750-PRINT-REPORT-TOTAL-LINE.
  MOVE WS-RPT-TOT-ACCUM      TO TL-REPORT-TOTAL.
  MOVE TL-REPORT-TOTAL       TO ACCT-REC-LINE.
  MOVE 2                      TO WS-LINE-SPACING.
  PERFORM 890-WRITE-REPORT-LINE.
```

This module is similar to the 700 module. However, we do not need to the report accumulator to zeros, because it is only used once in the report. If we run the program again it will be initialized to zeros in the 100 module as before.

```
800-READ-ACCT-REC-RECORD.
  READ FILE1
  AT END
  MOVE "YES"          TO WS-END-OF-FILE-SW.
```

This 800 module contains the READ statement that we have used in the previous programs. We made a separate module so the program logic would be easier to visualize. Do you agree? You will find that as our programs become more complex that this will be more important!

```
870-PRINT-REPORT-HEADING.
(9) MOVE SPACES          TO ACCT-REC-LINE.
(9) PERFORM 880-WRITE-REPORT-HEADING.
(1) MOVE H1-HEADER       TO ACCT-REC-LINE.
   MOVE 1                TO WS-LINE-SPACING.
(1) PERFORM 890-WRITE-REPORT-LINE.
(2) ADD 1                TO WS-PAGE-COUNT.
(3) MOVE WS-PAGE-COUNT   TO H2-PAGE-NBR.
(4) MOVE H2-HEADER       TO ACCT-REC-LINE.
(4) PERFORM 890-WRITE-REPORT-LINE.
(5) MOVE C1-HEADER       TO ACCT-REC-LINE.
   MOVE 3                TO WS-LINE-SPACING.
   PERFORM 890-WRITE-REPORT-LINE.
(6) MOVE C2-HEADER       TO ACCT-REC-LINE.
   MOVE 1                TO WS-LINE-SPACING.
   PERFORM 890-WRITE-REPORT-LINE.
(7) MOVE SPACES          TO ACCT-REC-LINE.
(7) PERFORM 890-WRITE-REPORT-LINE.
```

This 870 module shows you how we use the same ACCT-REC-LINE to add two (2) report heading lines and two (2) column heading lines to this report. Can you follow the logic of how it works? We move each of the lines to the ACCT-REC-LINE and use 890 module to write them on the report. Statements (1) handle the first heading line. Statement (2) increments the page counter and statement (3) moves the page number into the heading line. The (4) statements write the second heading line. Then the (5) statement writes the first column heading and (6) handles the second heading line. Statements (7) provide a blank line. Note, that we would increment the page counter each time we do the headings because this means that we have a new page. Also, note how we handle line-spacing by using

This 870 module shows you how we use the same ACCT-REC-LINE to add two (2) report heading lines and two (2) column heading lines to this report. Can you follow the logic of how it works? We move each of the lines to the ACCT-REC-LINE and use the 890 module to write them on the report. Statements (1) handle the first heading line. Statement (2) increments the page counter and statement (3) moves the page number into the heading line. The (4) statements write the second heading line. Then the (5) statement writes the first column heading and (6) handles the second heading line. Statements (7) provide a blank line. Note, that we would increment the page counter each time we do the headings because this means that we have a new page. Also, note how we handle line-spacing by using

WS-LINE-SPACING to meet the Program Specifications. Do you understand this? Move 1 to WS-LINE-SPACING for single-space, Move 3 for triple-space, etc. Note statement lines (9) and see the module below.

```
880-WRITE-REPORT-HEADING.
  WRITE ACCT-REC-LINE
  AFTER ADVANCING PAGE.
  WRITE ACCT-REC-LINE
  AFTER ADVANCING 6 LINES.
```

We will use the 880 module when it is necessary to start the report first page or additional new pages. Whenever we go to the 870 module, we will be starting the report's first page or going to the next page of the report because that is the only time we require report headings. The second WRITE moves us one (1) inch (six (6) blank lines) down from the top of the page.

```
890-WRITE-REPORT-LINE.
  WRITE-ACCT-REC-LINE
  AFTER ADVANCING WS-LINE-SPACING.
  ADD WS-LINE-SPACING          TO WS-LINES-USED.
```

This 890 module prints all the heading lines, column headers, detail-lines, page total-line and report total-line. The spacing between each of these lines is controlled by the value moved into the WS-LINE-SPACING. After the WRITE, we ADD the WS-LINE-SPACING value to WS-LINES-USED counter. Of course, when WS-LINES-USED reaches 50 or more, this condition will be checked by the IF statement in the 200 module.

Documentation

With this CODING that I supplied for the Procedure Division, you can now complete the following:

1. Your Structure Chart.
2. Your Flowchart.
3. Your CODING sheets.
4. Key-in the remainder of the source code appended to your "homework" from the last COBOL Corner article.

Now, do a walk-through of your documentation, CODING and source code. Do this carefully, as this is where you want to find your ERRORS! Watch for missing words, misspelled words, typos, etc. Are you SURE you have it right?

If so, using PIP, transfer FILE1.DAT from your HUG COBOL Corner Disk-I to your "A" disk. Now COMPILE your PRGM05.COB source file. Did you get any ERROR messages? If you did get ERRORS, shame on you! You did a poor walk-through! If you did not get any ERRORS, my congratulations! You are ready to go on to BETTER REPORTS!

If you did find ERRORS, do not get discouraged! You will have to do your work better. Use a hard copy of your source listing and compare it with the previous article and this article. Did you find your ERRORS? If so, correct your PRGM05.COB using your Editor. COMPILE again. You should have no Compiler Errors. If you still have ERRORS and you can not find them, prepare a "NEW" disk "A", as we have discussed in previous articles. Use PIP and transfer PRGM05.COB and FILE4.DAT from your HUG COBOL Corner Disk-I to this new "A" disk. COMPILE this disk. Now, if you have the COBOL Compile Procedures correct, you will obtain a "clean", no ERROR Compile. Use a hard copy from this Compile and compare it Line-By-Line with your program listing. Mark the necessary corrections in RED pencil or ink on your hard copy listing. Make the necessary corrections to your program using your Editor. Now, COMPILE again. If you did your work carefully, you MUST have a "clean" Compile!

Now, all readers should LINK/EXECUTE your PRGM05.REL and your print-out of the Accounts Receivable Report should match the Program Specifications. Does it? If it does not, recheck all your documentation. Especially your Print Chart, and then your CODING. Do you now have a GOOD REPORT?

COBOL-80 Version 4.6 Update

I would like to use the rest of this month's COBOL Corner to discuss the COBOL-80 Version 4.6 update that I have just received after waiting four (4) months for it. It is a major update change. Also, as of this month I will be using soft-sector in place of hard-sector drives and disks. This should not cause any problems, once we do a COBOL-80 System Set-Up for whatever hardware we are using! The Update changes fall into three (3) categories:-

- 1--Enhancements (new features).
- 2--Corrections of reported deficiencies by users.
- 3--New Microsoft Linking Loader.

Version 4.6 Enhancements

I--Faster Processing

A--All the runtime routines that generated INDEXED files have been fully revised and now generate files with an entirely new format. A utility program, CVISAM.COM, is included which converts the previous COBOL-80 release INDEXED files to the new format. Note COBOL Corner has not created programs using INDEXED files as yet, so most readers will not be concerned at this time.

B--The multiplication algorithm has been speeded up. So far, our usage of the multiplication COBOL verb has been for simple problems, so you will not notice any great changes. As we get into greater complexity programs you will like the change.

C--The COBOL-80 runtime library has been made a common runtime system. This speeds up the Link procedure, since the interpreter modules re no longer part of the linking process. Larger COBOL programs can now be linked (This sure was needed!). Also, when used, the program chaining facility will be speeded up (We will touch on this in later articles.). The executable COBOL programs will take less disk space because many runtime routines will not be included with every linked program.

II--Easy Debugging (Similar to COBOL-86)

A symbolic interactive debugger has been added to the COBOL-80 Ver. 4.6 runtime routine. It is an optional relocatable object module (DEBUG.REL) that may be linked into the executable program. We will use this when we get to the more complicated COBOL Corner programs.

III--More Power

A--IF statement may now contain Conditional statements, as well as other IF statements (We will discuss in later articles.).

B--COBOL-80 will now create a file, rather than reporting an error, when you perform an OPEN EXTEND on a non-existent SEQUENTIAL or LINE SEQUENTIAL file (We have not used this EXTEND verb yet.).

C--A RELATIVE KEY may now be a numeric field with USAGE DISPLAY (Again, we have not proceeded this far as yet.).

D--Detection of damaged RELATIVE or INDEXED files (Also, later.).

E--Compiler now diagnoses absence of a DELIMITED BY clause in an UNSTRING statement (Again, later.).

F--Compiler now diagnoses the presence of an OPEN EXTEND statement for a RELATIVE or INDEXED file (As above, later.).

G--Compiler now diagnoses the presence of a READ statement that is in a format not compatible with the ACCESS MODE used:

1--READ...AT END is allowed only for SEQUENTIAL ACCESS files (The kind we have been using.)

2--READ...NEXT...AT END is allowed only for DYNAMIC ACCESS files (This will come later.).

3--READ...INVALID KEY is allowed only for RANDOM or DYNAMIC ACCESS (This is also for later articles.).

H--Compiler now enforces the definition of file status items as two byte alphanumeric fields (We are not concerned with this now.).

I--Compiler now diagnoses the presence of AFTER (BEFORE) ADVANCING clause in a WRITE statement to a disk file (Can you find this statement in our program #5?).

IV--Shorter Code

The compiler no longer includes temporary global symbols in the relocatable object file (.REL). This reduces the size of the .REL file and enables larger programs to be linked (This is important to us, because some of our future more complicated programs were going to push the limits.).

V--Easy Customization

New entry points have been added to the CRT drivers. They are \$INCRT and \$OUCRT. This provides greater flexibility for user customization of the drivers (I do not think this will be used by many readers.).

VI--Additional ANSI COBOL 1974 Standards Provided

A--The COBOL-80 DIVIDE statement syntax has been revised to reflect the ANSI standard. This is really worth while! Look them up in your COBOL-80 Version 4.6 Manual and compare them to what we have discussed in previous COBOL Corner articles. I will cover them as we use them again also.

B--GOTO statements which have been ALTERed and which reside in an independent section are reset to the original designations when the independent section is reloaded (I do not expect to use this GOTO as discussed in previous articles.).

C--The statement EXIT PROGRAM is executed as an EXIT statement if it occurs in a main program (More about this when we discuss CALL programs later in the series.).

D--A subprogram (CALL) that does not receive parameters no longer needs an empty USING list in the Procedure Division header (We will discuss this in detail when we get to using subprograms.).

E--The first segment of a program may now be independent (This will mean more to you readers later.).

F--The COPY statement may have source text following it on the same line (We have not had a need for the COPY verb yet.).

G--Syntax has been changed for the SPECIAL NAMES paragraph in the Environment Division (I do not think we will use this. It is treated as a commentary by this compiler.).

H--FIPS flagging can be specified in the command line to the compiler using /F switch and allows the programmer to determine those features within the COBOL program which conform to high-intermediate, high, or extended features of the ANSI 1974 standard (This is useful if you are working with COBOL programs created on higher level COBOL compilers.).

I--The compiler /V switch can be used in the command line to the compiler when compiling programs that MUST conform closely to ANSI standards. In this Version 4.6, the switch changes the meaning of only the USAGE COMP. The limitations in the range of a COMPUTATIONAL data item cause this data type to be interpreted as USAGE DISPLAY. Binary fields can still be declared with USAGE COMPUTATIONAL-0 (COMP-0), which is non-standard (I do not like this, but I do not think it will bother many readers.).

Version 4.6 Code Corrections

NOTE: Most of these corrections will not mean anything to you readers now, but they will as we get to advanced COBOL programming later.

Early Version Deficiency	Version 4.6 Action
I--First few bytes of CRT Driver destroyed when a subprogram was returned	Corrected
II--Could not use INDEXED files in a SORT statement's USING or GIVING list	Corrected
III--MOVE of data item to justified field shorter than the source field destroyed data following destination field	Corrected
IV--COMP-3 subscripts not converted correctly	Corrected
V--Error messages erroneously given when program contained consecutive program sections with same segment number	Corrected
VI--Random value used as a prompt character in the SCREEN SECTION	Corrected
VII--/P switch non-operative in compiler	/P switch now operative, allowing 100 bytes of additional stack space for each switch in compiler command line
VIII--START statement that does not contain the equality relation in the KEY phase return an error when RELATIVE file expected a KEY that was not found in file	Corrected; runtime now searches for existence of a higher-valued KEY
IX--Program abort, under some conditions after an error return from an OPEN statement	Corrected
X--OVERLAY loading destroyed first two bytes of WORKING-STORAGE	Corrected
XI--Could not link large COBOL programs containing overlays	Corrected
XII--Could not correctly MOVE scaled numeric item to an alpha-numeric item	Corrected

Version 4.6 New Distribution Disk Files

I--COBOL Compiler Files

	Early Version	Version 4.6
A--COBOL.COM	30K	32K
B--COBOL1.OVR	12K	14K
C--COBOL2.OVR	14K	14K
D--COBOL3.OVR	18K	18K
E--COBOL4.OVR	8K	8K
Total Disk File Size	82K	86K

II--Runtime System

A--RUNCOB.COM	New File	20K
B--COBLIB.REL	New File	28K
C--COBLIB.REL	36K	52K
D--CRTDRV.REL	2K	2K
E--L80.COM	10K	12K
F--LD80.COM	New File	18K
G--COBLOC.	New File	2K

III--Utility Software

A--LIB.COM or LIB80.COM	6K	6K
B--M80.COM	20K	20K
C--CREF80.COM or CREF.COM	4K	4K
D--DEBUG.REL	New File	6K

IV--Conversion Utilities

A--SEQCVT.COM	6K	6K
B--CVISAM.COM	New File	40K
C--REBUILD.COM	New File	20K

V--Demonstration Programs

A--SQUARO.COB	4K	4K
B--CRTEST.COB	8K	8K

Version 4.6 New File Descriptions

I----The **COBOL.COM** with **COBOL1.OVR**, **COBOL2.OVR**, **COBOL3.OVR** and **COBOL4.OVR** work the same as previous versions of **COBOL-80**. The only change is a slight increase in the size of the **COBOL.COM** and **COBOL1.OVR** (about 4K). The compiler will still fit on a non-boot formatted 40 track single-side, single-density disk.

II----**LD80.COM** is a Disk-to-Memory version of the linker. It eliminates most of the size limitations of the **L80.COM** in-memory linking loader. **LD80.COM** uses disk space rather than memory to build the COBOL program image; thus, it is able to create larger executable program images. To use **LD80.COM**, simply substitute **LD80** in place of **L80**. **LD80.COM** creates three (3) temporary disk files named **<fname>.\$P**, **<fname>.\$FA** and **<fname>.\$FB**; which will be put on the default drive. **<fname>** is the filename of the main module being linked. The three (3) files are automatically deleted from the disk upon termination of the **LD80** compilation.

III--COBLOC. provides the load address of COBOL programs. This file must be resident on the currently logged drive when you link COBOL programs. At runtime the compiler program relocates itself to a high memory address. This is the first address contained in the first line of **COBLOC**. The linker will use this value as the argument for an implied **/P** switch. The second line of **COBLOC** tells the compiled program which drive contains the runtime system. You MUST change the information on this line to suit your configuration! An **A**: on this line indicates that **RUNCOB.COM** will reside on drive **A** at runtime. I use a colon **ONLY (:)** on this line to indicate that the executor, **RUNCOB.COM**, will reside on the currently logged drive! **COBLOC** must **NOT** contain Editor line numbers.

IV--RUNCOB.COM is the Version 4.6 runtime executor. The runtime system for **COBOL-80** can be present in two (2) forms. Normally, you will have it reside in this **RUNCOB.COM** file. When a COBOL program is invoked by typing its name or by the use of the **/G** switch with the linker, **RUNCOB.COM** will not require reloading when "chaining" (We will discuss this in later articles.) from one program to another.

The runtime system can also reside within the compiled COBOL program. This option is selected by the **/X** switch during compilation. This switch produces instructions to the linking loader within the relocatable object file (**.REL**) to combine **COBLIB.REL** with the compiled program. Thus, **COBLIB.REL** contains an alternative version (old method) of the runtime system. Note: Use of this alternative prevents the use of the runtime system contained in **RUNCOB.COM**.

Whichever runtime system used, the linking loader performs three (3) additional functions. First, it resolves relative address references in the compiled program to absolute values. Second, it automatically

binds optional modules from the COBOL library. Without using the /X switch, the optional modules reside alone in the COBLBX.REL library. If subprograms are CALLED (We will discuss this in later articles.) from the main COBOL program, they too will be bound into the executable file by the linking loader. Third, the loader automatically loads the file CRTDRV.REL to include terminal-dependent functions.

V---COBLBX.REL is the runtime executor library (RUNCOB.COM). The runtime library consists, in part, of an executable program that interprets the object code of your COBOL program produced by the compiler. This part of the program is the RUNCOB.COM file. In addition, some optional routines are contained in the COBLBX.REL file. Also, certain extensions for file handling under CP/M-80 are included in COBLBX.REL. These involve the COBOL CALL verb that we will use in future programs. If you want more information now, read Appendix D of your COBOL-80 Version 4.6 Manual.

VI---DEBUG.REL provides runtime debugging support to your COBOL programs. It is a Microsoft COBOL interactive debugger. It is not ANSI standard! It can be linked with a COBOL object program and it is discussed in the COBOL-80 Version 4.6 Manual, if you want more information at this time. We will spend at least one COBOL Corner article on its use in the future.

VII--CVISAM.COM is a special utility program that converts INDEXED files from previous COBOL-80 version's format to this new format. BEFORE you use this program, ALWAYS work with a backup copy. Some real problems can come up with some conversions. To invoke it, enter:

CVISAM and Return

The program will prompt you for the information it needs to perform the conversion. Most COBOL Corner readers will not need it at all.

VIII-REBUILD.COM is a utility program used to recover or restore data in INDEXED files. We will work with INDEXED files sometime in the future.

Version 4.6 COBOL-80 System Set-Up

This might be a good time to go back to COBOL Corner II and re-read it carefully before proceeding with this article. I will not be repeating everything in that article! I DO recommend that you MAKE all-new COBOL Corner Working Disks for Version 4.6. BE SURE to make back-up copies of all the Original Distribution Disks before proceeding any further!

As before, Disk A should contain your operating system (CP/M in my case.), SQUARO.COB, CRTEST.COB and whatever Editor you have chosen. This Disk MUST be a bootable disk for use in drive A (SY0: for HDOS). This disk does not require many of the CP/M files. I use these:

(BIOS.SYS)
ED.COM or your Editor (I use PAGED.COM)
PIP.COM
STAT.COM
DUP.COM
SQUARO.COB } used to verify our
CRTEXT.COB } COBOL-80 Version 4.6 System

Disk B will be the compiler disk and should have the following files:

COBOL.COM
COBOL1.OVR
COBOL2.OVR
COBOL3.OVR

COBOL4.OVR

This disk will be used in drive B (SY1: for HDOS) when you are compiling your COBOL program. It does NOT have to be bootable!

Disk C will be your runtime and linking loader disk and it should have the following files on it:

L80.COM
COBLIB.REL
CRTDRV.REL

Disk C will be used in drive B (or drive C if you are using three (3) drives) and it does NOT have to be bootable!

Also, lets make another Disk C (Version 4.6) with these files:

LD80.COM
COBLBX.REL
RUNCOB.COM
COBLOC
CRTDRV.REL

This Disk will be used like Disk C above!

NOTE: If your drive B has 160K or more storage, Disk B and Disk C can be combined on one Disk B! I would always keep Disk A as specified above. This allows the use of a New Disk A for each COBOL program. This provides backup protection and a nice program filing method for storing your COBOL Corner programs for review from time-to-time.

Version 4.6 System Verification

To verify our Disk A, Disk B and Disk C, put Disk A in drive A and Disk B in drive B. Boot the system and type B: and RETURN! To compile we MUST be logged on the Disk that contains the compiler (Disk B), since the COBOL overlays are always read from the current disk! NOW, since Disk A has the test program--SQUARO.COB--type one of the following command lines:

B>COBOL A:SQUARO.REL, TTY:= A:SQUARO.COB and RETURN

or

B>COBOL A:SQUARO, TTY:= A:SQUARO and RETURN

or

B>COBOL ,TTY:= A:SQUARO/R and RETURN

All three (3) commands produce the same results! For (2), the compiler will furnish the default extensions (.REL and .COB). The /R compilation switch will force generation of an object file that defaults to the filename SQUARO.REL for command (3). If you would like a hard listing, substitute LST: for TTY: in the above commands. When the compilation is complete, you should find the following message on your screen:

NO ERRORS OR WARNINGS

You should find the file--SQUARO.REL--on your Disk A. This verifies the compiler Disk B. If you do not get these results, recheck your work as described above. Also, check any ERROR MESSAGES you get with your COBOL-80 Manual to help you find your errors.

Now that the Source Program has been compiled, we replace Disk B with our first Disk C in drive B. (If you have 3 drives you do not need to do this.) Also, if you have combined Disk B and Disk C on a 160K Disk you would skip this step.

The final step before execution is to load SQUARO.REL into the linking loader, L80. This step converts the relocatable object program into an absolute version and combines it with the COBOL-80

runtime system. L80 causes the absolute version to be built in-memory, where it may be saved to disk, executed from memory, or both.

The following command will load and execute SQUARO without saving the absolute version:

```
B>L80 A:SQUARO/G and RETURN
                (Use C> if 3 drives are being used)
```

The extension .REL is assumed. The /G switch directs the loader to complete the loading process and begin execution of the program. Once SQUARO has been executed, you can not execute it again without repeating the above command, since the absolute version was not saved.

If we wish to save the absolute version in a disk file without executing it directly, type:

```
B>L80 A:SQUARO/N, A:SQUARO/E and RETURN
```

Then, to execute the program, simply type:

```
B>A:SQUARO and RETURN
```

The /N switch directs the loader to save the executable program on disk when the loading process is complete. The /E switch directs the loader to complete the loading process and exit to the CP/M operating system.

We can also combine the two methods so that the absolute version is saved on disk and then executed directly by typing:

```
B>L80 A:SQUARO/N, A:SQUARO/G and RETURN
```

In both of the above examples, you will find SQUARO.COM file on Disk A. If SQUARO performs the math function correctly, the system is verified!

Now, lets verify Disk C Version 4.6. The process is very similar. To use this Disk C, substitute LD80 for L80 in the command lines given above. By using a separate Disk C, you avoid the use or not the use of the /X switch when compiling! If you want more information about the /X switch, look up Compiler Switches in your COBOL-80 Version 4.6 Manual. Did you notice any differences with this Disk C? Did you get the same results? If you had a problem, it may be that you did not change the second line of the COBLOC file to either a colon (;) or B:. This is required because we have our runtime system on Disk B in drive B! If SQUARO performs its math function correctly, the system is verified!

I am going to let you try the CRTEST.COB Demonstration Program by yourself. Just follow the above command lines replacing A:SQUARO with A:CRTEST! CRTEST.COB is a COBOL source program that tests the functions of the interactive CRT driver. If you are not using a standard Heath/Zenith system, refer to Appendix A of your COBOL-80 Version 4.6 Manual.

Closing

Special Note: I tried the Version 4.6 on my H/Z-100 with CP/M-85 and it seems to work fine! I will test further. Also, you COBOL-86 users will now find that this COBOL-80 Version 4.6 is very similar.

This COBOL Corner article got longer than what I would like. You will have a lot to learn and try this month. However, I would still like to give you some "homework!" Please use the Version 4.6 Set-Up Disks and try compiling, linking and executing each of the COBOL Corner programs (including #5) we have covered so far using both the Disk C and the Disk C Version 4.6. This will be an excellent review and it will help you decide if you like the new Version 4.6 COBOL the best. Also, I want you to complete Program #5, because next month I am going to have you do Program #6 nearly all by yourself!



MAP PROJECTIONS OF THE WORLD

- Produce high-res color graphics maps for the H/2-100 and H/2-150 (or IBM)
- Track up to 10 hurricanes in animation
- Two historical ones are included
- Choose any of 7 different projections- 5 Azimuthal & Polyconic & Mercator
- Pick either of 2 data files: World Coastlines with 5070 digitized points or a very-high-res file of the Caribbean from the Carolinas to Mexico to the Amazon (11940 points)
- Vary the origin, rotation, window and scale(Global or down to 20 miles/inch on most projections)
- Choose any grid spacing for the parallels and meridians
- User's manual includes screen dumps of 16 sample maps to help you learn how to reproduce these and many others
- Superfast execution- over 1000 pts/min from an optimized compiled basic program that requires only DOS and color graphics. A color monitor does help

"...fascinating look at the world from every conceivable angle...The maps are excellently drawn. It's amazing that Dr Pittman could squeeze that much detail onto a single 5 1/4-inch diskette..."-- Dave Felstul--BUSS # 91

Pick up a copy at many Heathkit Centers or order direct by sending \$49.95 post-paid (specify Z-100 or Z-150)to: Dr M E Pittman, 57 Emile Av, Kenner, LA 70065

WATCHWORD

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

FRIENDLY - FAST - FLEXIBLE

- See subscripts, superscripts, underlining and boldface directly on the screen
- Create your own fonts and special characters
- Remap any key and configure to your printer
- Written in native 8088 assembly language for fast response

Other features: centering, formatting, microjustification, arbitrary line length, automatic horizontal scrolling, split screen, macros, and color.

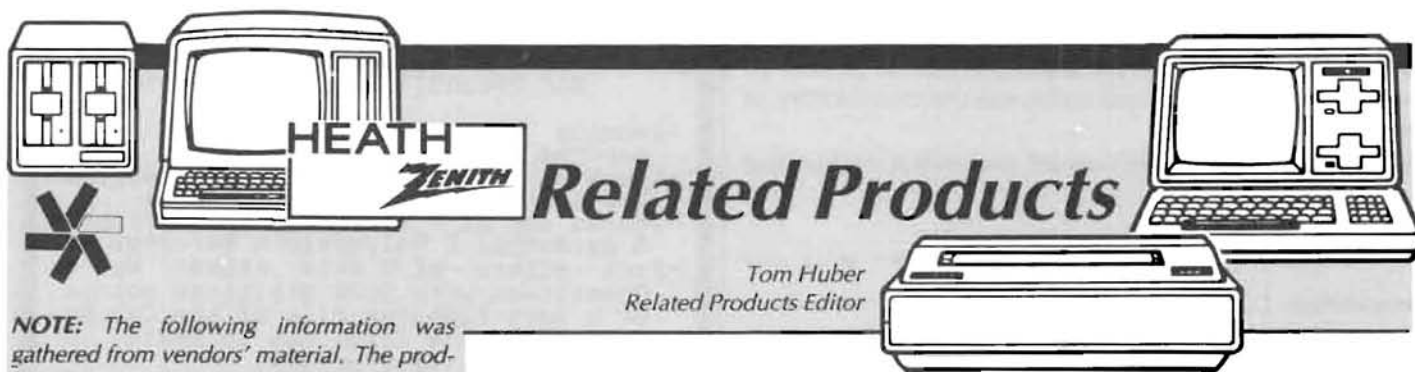
Demo available at all Heathkit Electronics Centers.

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



Tom Huber
Related Products Editor

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

ROOTS II for the H/Z-100 Computers

Comsoft has announced that ROOTS II is now available for the H/Z-100 computers, as well as the H/Z-100 PC family. It provides automatic linking for up to 4096 relatives going back 99 generations. Data on any individual or family can be found by searching name, date, place, relationship, or multiple fields. The program stores names, dates, and places of important events, in addition to unlimited biographical information for each individual. LDS ordinance dates are supported. Roots II accepts ROOTS/M files and converts them to the ROOTS II expanded format. Z-150 support includes maps and photos. Four types of genealogical forms are supported and a complete family book can be printed (up to 999 pages long). It assigns page numbers and will print an alphabetized index of all names found in the book. It comes complete with a 220 page manual. Comsoft has added a utility to ROOTS II called "Split and Merge," which is used for database servicing. The new utility will allow you to separate one branch from the tree or merge in a new branch. If other researchers are working on the same family, the results of their work can be integrated automatically. With the addition of this utility, any restrictions to the size of the database (and/or the number of people in a given file) is effectively removed.

ROOTS/M for CP/M and ROOTS89 for HDOS provide similar features, but are not as expanded in their capability. Specify operating system and disk format.

Vendor:	Comsoft, Inc. 2452 Embarcadero Way Palo Alto, CA 94303 (415) 493-2184	
Prices:	ROOTS II:	\$195.00
	ROOTS/M:	\$49.95
	ROOTS89:	\$39.95

CAD Program for H/Z-100, PC

Computer Integrated Manufacturing Systems, Inc. has announced a three-dimensional design and drafting system. CADC requires no computer knowledge and is menu driven. It may be used in a wide variety of applications, including architectural, electrical, chemical, civil, mechanical, and structural engineering. There are six modules: CADC Install, Model Mode, Detail Mode, Properties Mode, Plot Mode, and Spaceframe. A demonstration disk and more information is available from the vendor.

Vendor:	CIMS, Inc. 20859 Old Spanish Trail
---------	---------------------------------------

New Orleans, LA 70129

(504) 254-0545

Prices:	CADC Model Mode:	\$995.00
	CADC Detail Mode:	\$295.00
	CADC Properties Mode:	\$195.00
	CADC Plot Mode:	\$195.00
	CADC Spaceframe:	\$495.00
	CADC System (all programs):	\$1995.00
	CADC Demonstration Disk:	\$75.00

VENIX Runs on H/Z-100, H/Z-100 PC Computers

VenturCom has announced a new release of VENIX, derived from UNIX System V, for a number of computers, including the H/Z-100 and H/Z-100 PC Series Computers. System V VENIX is based on AT&T UNIX System V, Release 2. Application software developed for other versions of UNIX can be compiled under VENIX with little or no modification to the source code. It also provides binary compatibility for microprocessor families. Additional information about VENIX is available from the vendor. Color and graphics are supported on the H/Z-100 PC but not the H/Z-100. The 8087 Coprocessor is supported on both machines.

Vendor:	VenturCom, Inc. 215 First Street Cambridge, MA 02142 (617) 661-1230
---------	--

Price:	\$875.00
--------	----------

RamPal Supports 256K Chips in H/Z-100 PC Series

Software Wizardry has announced the RamPal memory upgrade modification for H/Z-100 PC computers. RamPal is a replacement PAL IC for the main memory card and allows the use of the 41256,256 kilobit dynamic RAM ICs. The RamPal will allow the computer to be configured to 640K (the maximum memory) on the main memory card without requiring additional memory cards in the expansion slots. RamPal allows 64 kilobit and 256 kilobit chips on the memory card. Some 64 kilobit chips must be replaced with 256 kilobit chips in systems already having 320K.

Vendor:	Software Wizardry 1106 First Capitol Drive St. Charles, MO 63301 (314) 946-1968
---------	--

Price:	\$39.95
--------	---------

New Book Is Computerist's Guide to Telecommunications

The Chilton Book Company has announced Plugging In: The Microcomputerist's Guide to Telecommunications, a reference that "helps the user get the most out of the personal computer" by providing samples of many private and public telecom-

munications services, including CompuServe, The Source, and Dialog's Knowledge Index. The realm of timesharing services, electronic data bases, consumer and specialized online services, information utilities, and the free services available on hundreds of community bulletin boards are explored. The reader is told how to find reasonably priced data bases that provide information similar to more costly specialized markets. The book covers self-help techniques for using an acquired service to best advantage and designing a good data base search strategy. Information is included on how to become a personal information broker, how to start one's own bulletin board, and how to transform the terminal into a portable encyclopedia. The book is available in bookstores nationwide or may be ordered directly from the vendor.

Vendor: Chilton Book Company
Radnor, PA 19089
(800) 345-1214
Price: 11.95 + \$1.75 Shipping and
Handling

HelpDOS Ends DOS Confusion

Help Technologies has announced HelpDOS, The Help System, for PC-DOS and MS-DOS. It is a menu-driven program that provides onscreen reference information and examples for DOS commands, special keys, and batch commands. A unique "hints" feature assists the user to find the right DOS facility for the task. HelpDOS is menu driven and may have additional menus or ASCII files added to it by the user. For MS-DOS version 2.0 or later.

Vendor: Help Technologies
P.O. Box 50834
Palo Alto, CA 94303
(415) 856-3431
Price: \$49.95

Pro Driver Updated to MS-DOS Version 2

Studio Computers has announced that Pro Driver has been upgraded to MS-DOS version 2 for the Z-100 and Z-100 PC Series computers. Pro Driver is a communications package that allows transmitting and receiving of both ASCII and binary files. Menu driven, it provides online help messages, supports both acoustic and autodial modems, generates reports for tracking calls, has 32 user-definable automatic logon sequences, emulates VT-52 and VT-100 operation, and accepts up to 40-digit telephone numbers. Specify Z-100 or Z-100 PC version. Requires a minimum of 192K.

Studio Computers has also announced the availability of its 24 page, 1985 catalog of Zenith computers, peripherals, and software. Phone or write for a free copy.

Vendor: Studio Computers, Inc.
999 South Adams
Birmingham, MI 48011
(313) 645-5365
Prices: Pro Driver 2: \$49.00
Pro Driver 2 upgrade for
registered owners: \$25.00
1985 Catalog: free

Unique Chocolate Byte Has a Bite Missing

A milk-chocolate diskette (4.8 oz. of Mercken's milk chocolate) is now available as the perfect gift for computer users who love chocolate and chocolate lovers who love computers. It is called The Original Chocolate Byte and comes wrapped in a reusable software case that holds up to three non-edible 5.25-inch disks. Available through computer retail outlets and by mail.

Vendor: The Chocolate Software Co.
15233 Ventura Blvd. #1118
Sherman Oaks, CA 91403
(818) 986-4885
Price: \$9.95

dBASE II Coding Guide Aids Software Developers

A dBASE II coding guide has been announced by Associated Technology to help software departments formulate their own database standards for microcomputers and programmers to create testable and easily maintained database software. 53 pages.

Vendor: Associated Technology
Route 2 Box 448
Estill Springs, TN 37330
Price: \$22.00

Z-BASIC Graphics and Games Collection

Intuitive Logic has put together a collection of the companies best full-color graphic displays and fast-action games. Sixty-four programs are included to provide an interesting diversion and answer to the classic question, "What can a computer do?" The graphic programs range from simple animation to detailed graphic displays. The games include memory games, action games, and "Beat Down the Grubs." Documentation consists of brief program descriptions and explanations. The programs are all unprotected and may be examined and used as a tool for learning new techniques, not covered in the ZDS documentation. Requires Z-DOS, Z-BASIC, 128K and color video memory.

Vendor: Intuitive Logic
412 Taylor Street
Rochester, MI 48063-4327
(313) 651-3859
Price: \$29.95

Wildflower Identification Kit

Intuitive Logic has announced a program to identify 90% of the wildflowers that grow East of the Mississippi. The programs work by asking a series of questions. Once the questions are answered, the plant is identified by family. The book, included with the program, can then be used to look up the species, common name and other facts about the plant. The program has a built-in glossary of botanical terms that can use, as an aid, detailed graphic pictures of a flower, leaf forms, and inflorescent types. Requires Z-DOS, Z-BASIC, 192K and color video memory.

Vendor: Intuitive Logic
412 Taylor Street
Rochester, MI 48063-4327
(313) 651-3859
Price: \$59.95

and a Radio Shack model 4, and it worked fine when the communication parameters matched.

Bob Hanan
P.O. Box 5026
Oxnard, CA 93030

GC1000 Mod

Dear HUG:

In January 1984, Jim Schuster published a timely program in REMark that interfaces the Z-100 to the GC1000 Most Accurate Clock, using the modem port (J2) for the link. In my case, however, I would rather use the serial printer port A (J1), because it would obviate having to switch cables between the clock and a modem and because I do not use that port for anything else. My printers use only the parallel centronics interface.

The changes one must make in Jim's program to use the serial printer port A are minor. At the beginning of the code, make the following changes:

```
epcib_dr      equ    0E8h    ;instead of 0ECh
epcib_sr      equ    0E9h    ;instead of 0EDh
epcib_mr      equ    0EAh    ;instead of 0EEh
epcib_cr      equ    0EBh    ;instead of 0EFh

OR            a1,00010000b    ;disable PORT A interrupt
                ;instead of 00100000b
```

This is not all that is necessary to make the conversion, however, since the two ports do not have the same pinout. One must also require the cable so that:

GC1000 (RS232 pin)	goes to	SERIAL PORT A (pin)
2		2 (instead of 3)
5		6 (instead of 20)
7		7

Whichever way you choose to use it, this is a very useful program and I would like to thank Jim for publishing it; I have not had to manually enter the time and date since.

Calvin Blakley
2007 Wilderness Point Drive
Kingwood, TX 77339

Imaginator . . . Still Learning

Dear HUG:

Several months ago, I purchased the Imaginator (Retrofit Graphics Display Board). I'm extremely happy with it, but I still haven't learned to use the "Bring in program" and the "jump to program". If there are any Huggies out there who have an Imaginator, it would be greatly appreciated if you could let me know how it's done.

I have an H89-A with 64K, and one H-17-1 disk drive. I have CP/M, CP/M-80, HDOS Ver. 2.0, and MBASIC.

Sincerely,

Walter W. Herman
Rt #4 Box 214
Abingdon, VA 24210

GUARDIAN 25
25 MEGABYTE
Z100 BACKUP



INTERFACES LIKE A FLOPPY

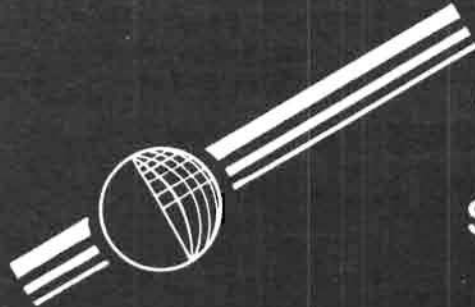
Can be used alone or as a Winchester backup

G25-Z1 Complete for Zenith Z100	\$1295.00
DC600 Certified Cartridge Tape	32.50
DC600-5 Cartridge Tape 5-pack	155.00

GUARDIAN DATA SYSTEMS
"Protecting your information investment"

44 STEDMAN ST., LOWELL, MA 01851

617-459-4449



Graphic Design Systems Inc.

Z-100
MEMORY BOARD

STARTING AT 379.00 (64K INSTALLED)

- SPECIALLY DESIGNED FOR Z-100
- UP TO 1MB CAPACITY
- USES 64K OR 256K CHIPS
- PARITY GENERATION / DETECTION

CONTACT: Graphic Design Systems Inc.
1020 ORIOLE LANE
MARIETTA GA. 30067
(404) 973-8471

WordStar And The Okidata

Dear HUG:

I am configuring WordStar for use with a Z-100 and an Okidata 92 printer. Maybe you can help me with a few problems I've encountered.

The default WordStar super- and sub-script commands produce a full-line shift, rather than a half-line shift as they should. Through the WordStar INSTALL utility, I have attempted to use the Okidata's carriage roll control codes in the carriage-roll-up and carriage-roll-down command sequences of WordStar. I find that these changes produce a completely garbled output. Again, from INSTALL, I notice that the LSB's of the WordStar carriage roll command sequences frequently change after printing a file. Setting up user-defined functions with the Okidata control codes works well, but leaves convenient default WordStar sub- and super-scripting commands useless.

Also, I find that the WordStar underlining command does not make use of the continuous underlining capability of the Okidata printer. WordStar leaves gaps between each character's underlining. Once again, the user-defined functions come to the rescue and successfully underline continuously. This has the curious side effect of preventing the last carriage return in a combined boldface/underlining operation when the printer is in the correspondence quality mode. WordStar's boldfacing is a triple-printing operation, while Okidata's correspondence quality mode is a double print operation. Thus, the print head travels over the same boldfaced correspondence text six times, doing the appropriate things until the last traverse. Changing the WordStar boldface to a double print operation solves this prob-

lem, but there goes another valuable WordStar command, inviting inadvertent error by invoking a screwed up WordStar command.

I do not know if any of these problems have come up before, but I hope you can help me. Thank you.

Very truly yours,

Craig Tucker
622 Stoney Creek Avenue
Baton Rouge, LA 70808

Plan your summer vacation **NOW!** Attend the 4th Annual **HUG International Conference** Chicago, O'Hare Hyatt Regency August 9, 10, 11



↳ Vectored from Page 33

885-8020-[37] CP/M RF Comp. Aided Design 30.00 44
885-8031-[37] CP/M Morse Code Transceiver 20.00 57

COMMUNICATION

HDOS

885-1122-[37] HDOS MicroNET Connection 16.00 37

CP/M

885-1207-[37] CP/M TERM & HTOC 20.00 26
885-1224-[37] CP/M MicroNET Connection 16.00 37
885-3003-[37] CP/M ZTERM (Z100 Modem Pkg) 20.00 34
885-5004-37 CP/M 86 TERM86 and DSKED 20.00 56
885-5005-37 CP/M 86 16 Bit MicroNET Conn. 16.00 61
885-8005 MAPLE (Modem Appl. Effector) 35.00 29

885-5006-37 CP/M 86 HUGPBBS 40.00 62
885-5007-37 CP/M 86 HUGPBBS Source List. 60.00 62
885-8005 MAPLE (Modem Appl. Effector) ... 35.00 29
885-8012-[37] CP/M MAPLE (Modem Program) 35.00 34
885-8023-37 CP/M 85 MAPLE 35.00 45

ZDOS

885-3019-37 ZDOS 16 Bit MicroNET Connect. 16.00 61

MISCELLANEOUS

885-0004 HUG Binder 5.75
885-1221-[37] Watzman ROM Source Code/Doc 30.00 33
885-4001 REMark Vol. I Issues 1-13 20.00
885-4002 REMark vol. II Issues 14-23 20.00

885-4003 REMark Vol. III Issues 24-35 20.00
885-4004 REMark Vol. IV Issues 36-47 20.00
885-4005 REMark Vol. V Issues 48-59 25.00
885-4500 HUG Software Catalog 9.75
885-4600 Watzman/HUG ROM 45.00 41
885-4700 HUG Bulletin Board Handbook 5.00 50
885-3015-37 ZDOS SKYVIEWS 20.00 55

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

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



CUT ALONG THIS LINE

HUG MEMBERSHIP RENEWAL FORM

HUG ID Number: _____

Check your ID card for your expiration date.

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

Name _____

Address _____

City-State _____

Zip _____

REMEMBER - ENCLOSE CHECK OR MONEY ORDER

CHECK THE APPROPRIATE BOX AND RETURN TO HUG

	NEW MEMBERSHIP RATES	RENEWAL RATES
U.S. DOMESTIC	\$20 <input type="checkbox"/>	\$17 <input type="checkbox"/>
FPO/APO & ALL OTHERS*	\$35 <input type="checkbox"/>	\$30 <input type="checkbox"/>
		U.S. FUNDS

* Membership in France and Belgium is acquired through the local distributor at the prevailing rate.

MEDIA MASTER™

“Foreign” File Compatibility Through Disk-To-Disk Transfers On Your Z-100™ or Z-150™

Tired of hunting for software in your computer's disk format? Do you need to exchange your Lotus 1-2-3, dBase II, Multiplan, and Wordstar data files among different CP/M and MS-DOS computers? Then MEDIA MASTER is for you!

MEDIA MASTER breaks down the 5¼" disk format barrier by allowing your computer to READ, WRITE, and FORMAT over 70 different "foreign" disks.

MEDIA MASTER is easy to use. You just LOG IN your "foreign" disk prior to copying files to or from another floppy or hard disk. Through simple menu steps you can also display directories and type, print, or erase files.

To order, send check or money order for \$39.95 + \$2.50 S/H (Cal. residents please add 6%). To order COD (we pay all COD fees!), call our 24-HOUR TOLL FREE NUMBER: 800-824-7888, and ask for Operator 251. (Z-150 owners - please specify the IBM PC version). **Dealer Inquiries Invited**



4573 Heatherglen Ct.
Moorpark, CA 93021
Technical Questions? 805-529-5073

MEDIA MASTER supports the following disk formats:

Actrix (SSDD)	LNW-80 (SSDD)
Actrix (DSDD)	Lobo Max-80 (SSDD)
Avatar TC10 (DSDD)	Lobo Max-80 512 (SSDD)
Casio FP 1000 (DSDD)	Mical 9050 CP/M-80 (DSDD)
Chameleon CP/M-80	Morrow MD 11 (DSDD)
Columbia MPC CP/M-80	Morrow MD 2 (SSDD)
Cromemco CDOS (SSDD)	Morrow MD 3 (DSDD)
Cromemco w/Int'l Term (DSDD)	NCR Decision Mate 5 (DSDD)
Cromemco w/Int'l Term (SSDD)	NEC PC-8001A (SSDD)
DEC VT180 (SSDD)	NEC PC-8001A (DSDD)
Davidge (DSDD)	Olympia ETX II (SSDD)
Digilog (DSDD)	Olympia EX100 (DSDD)
Epson Multifont (DSDD)	Osborne (SSDD)
Epson QX-10 (DSDD)	Osborne 4 (DSDD)
Fujitsu Micro 16s (DSDD)	Osborne Osmosis (SSDD)
Groupil III CP/M (DSDD)	Otrona (DSDD)
H/Z Z-100 CP/M (DSDD) - later	PMC Micromate (DSDD)
H/Z Z-100 CP/M (SSDD)	Reynolds & Reynolds (SSDD)
H/Z Z-100Z-DOS 1.xx (SSDD)	Sanyo (DSDD)
H/Z Z-100 Z-DOS 1.xx (DSDD)	Superbrain (DSDD)
H/Z Z-100 Z-DOS 2.xx (SSDD)	Superbrain Jr. (SSDD)
H/Z Z-100 Z-DOS 2.xx (DSDD)	Systel II (SSDD)
H/Z Z-90 40 trk. 1 k blk (SSDD)	Systel III (DSDD)
H/Z Z-90 40 trk. 2 k blk (SSDD)	T1 Professional CP/M-86 (DSDD)
Heath w/Magnolia CP/M (SSDD)	TRS-80 III FEC CP/M (SSDD)
IBM PC CP/M-86 (DSDD)	TRS-80 III FEC T805 (SSDD)
IBM PC CP/M-86 (SSDD)	TRS-80 III Hurr. Labs (SSDD)
IBM PC-DOS 1.xx (DSDD)	TRS-80 III Mem. Merch. (SSDD)
IBM PC-DOS 1.xx (SSDD)	TRS-80 IV CP/M + (SSDD)
IBM PC-DOS 2.xx (DSDD)	TRS-80 IV Mont. Micro (SSDD)
IBM PC-DOS 2.xx (SSDD)	Teletek 40 trk (SSDD)
IDEA Britelex (SSDD)	Toshiba T100 (DSDD)
ISM CP/M (DSDD)	TurboDos (DSDD)
Insight Dev. IQ-120 (SSDD)	Wang Maws CP/M (DSDD)
Kaypro II/2 (SSDD)	Xerox 820 II (SSDD)
Kaypro 4, 10 (DSDD)	Zorba 40 trk (DSDD)

Also available for the DEC Rainbow, Osborne, and Kaypro computers. Call or write for information.



Hilltop Road
Saint Joseph, Michigan 49085

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

POSTMASTER: If undeliverable,
please do not return.

P/N 885-2063