REMark®

Volume 7, Issue 2 • February 1986
P/N 885-2073 Issue 73

$2.50

Official magazine for users of HEATH ZENITH computer equipment.

HS-158-41 Expandable IBM PC Compatible Computer Now in Stock Only $1599

HS-158-41
IBM-PC Compatible
Computer Kit with
HVM-122A Monitor

Expandable Computer Kit $1599

HVM-122A Monitor Kit $79.95 **Use your HUG discount**

Heathkit HS-158-41 computer kit includes MS-DOS operating system, 256K of memory expandable to 640K, six IBM-compatible accessory board slots, and 360K disk storage.

NOW AVAILABLE AND IN STOCK FOR IMMEDIATE DELIVERY

50% OFF! EXPANDABLE PC $AVERS

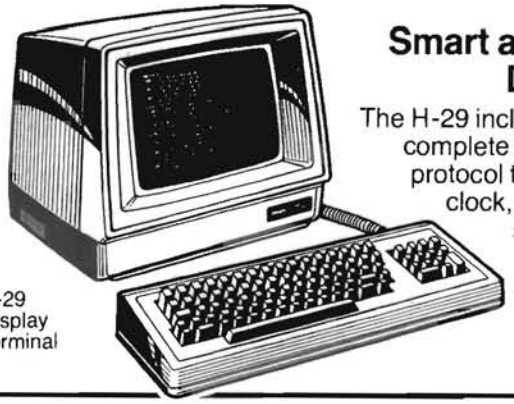
Get 50% off the price of these selected items when you purchase them within 90 days of buying an HS-158-41 computer. A copy of the computer purchase receipt is required. **No additional discounts can be applied to these items.**

Any three Heath/Zenith software packages.

Z-205-4, 256K Memory Chip Set.
Z-316-8, 8 MHz 8087 Numeric Co-Processor.
Z-207-7, Optional 5.25" floppy disk drive.

CHECK THESE SPECIAL BUYS

No additional discounts apply to products below



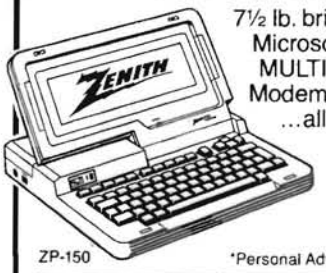
H-29
Display
Terminal

Smart and user-friendly Display Terminal

The H-29 includes built-in flexibility, complete compatibility with ANSI protocol terminals, user-settable clock, keyboard help key, and screen saver capability.

Kit only $299.00

Sensational price on this new PAL* LAPTOP COMPUTER



ZP-150

7½ lb. briefcase size unit has Microsoft WORKS, WORD, MULTIPLAN and BASIC... Modem...Calendar...Alarm ...all built in. Much more, too. Act now for this unbelievable price.

$799.00

*Personal Advanced Laptop

Smart Outlet Box

The GD-1495 has single switch control of up to seven electrical devices, surge protection, and noise filtering. Unswitched socket also included for clocks and equipment you want left on.

Now only $59.95

ZOOM/MODEM PC 1200 INTERNAL 1200/300 BPS modem for PC, XT and AT computers. The PC-132 is Hayes-compatible. Includes touch tone decoder, onboard RAM for programming, battery backup, and more.

Reg. $489 .. **Special $299.00**

SYQUEST 10 MEG Removable Winchester Cartridge Drives. Includes drive, one 10 Meg cartridge, controller board. **With free installation** **$1195.00**

NEW MASTERPIECE 4-Function Power Controller. Includes swivel base, 5 outlets, master switch, power surge protector, static ground **$99.95**

Lotus 1-2-3. Easy to use business software with spreadsheet, graphics, and database all in one program ..... Now just **$299.00**

NASHUA DSDD Diskettes ..... Per Box **$12.95**

Experts in **CAD,** Networking and Communications

Inquire about special corporate and government sales and service programs.

Phone orders accepted.
Prices good through March 1, 1986.

ZENITH data systems Authorized Dealer

Fine Electronics Since 1962

Your TOTAL SERVICE computer center

• Service • Support • Software • Accessories • User Training • Competitive Prices

HZC-173

Heath® ZENITH®

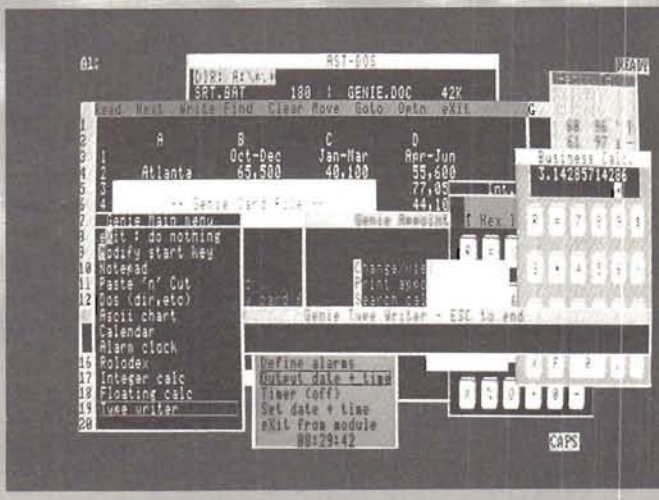
Units of Veritechnology Electronics Corporation

Computers & Electronics

NOT PROTECTED

Genie™

Software Magic from Advanced Software Technologies



Shown here is Genie "popped up" on a Z-110 running Lotus 123. From the left are: The Genie main menu, the Genie rolodex style card file, the Genie notepad containing data cut from Lotus, the Genie DOS performing a directory command, the Genie alarm clock (at the bottom,) the Genie typewriter, Genie calendar, Genie Cut and paste, Genie Calculators, and the Genie Ascii table.

Genie is a memory resident application. This means that once you load Genie it is always available for you to use. Just hit the magic keys and Genie will appear (Shift-Shift: No function keys lost.) You can have Genie perform various tasks, and when you finish Genie goes away and you are back where you started.

Here is what you get with Genie:

- **KEY MAPPER** - Redefine any key on the fly, store long commands in a single key, and save many maps on disk.
- **TrueDOS**™ - Run any MS-DOS program inside the window. FORMAT, SORT, BACKUP, compile, all without leaving your application.
- **CALCULATORS** - you get two calculators, one regular floating point calculator, and programmer's calculator for base conversions and bit operations.
- **NOTE PAD** - you'll never have to hunt for paper and pen again. Simply call up Genie's Note Pad and jot down what you need. Or expand the buffer to 56K: Instant editor.
- **CUT AND PASTE** - Cut text from any place on the screen and output it later. Cut long commands off the screen and into your KEY MAPPER, move data from your spread sheet to your word processor: Instant integration.
- **CALENDAR** - schedule appointments for any year up to 9999 keep track of expenses, search and print the calendar.
- **ROLODEX** - a name address and telephone number list that you can search any time. No limit on the number of cards. You can even output an address directly to your word processor.
- **ALARM CLOCK** - Have your Genie remind you of appointments, set alarms to ring at any time on any day. A window appears with a reminder of what each of the 8 alarms is for.
- **ASCII Table** - programmers never have to leaf through big books to find the ASCII value of a character.
- **TYPEWRITER** - Knock out quick memos any time, even send ESCAPE codes to your printer. Cut a portion of a spread sheet and paste into the typewriter for quick printouts.
- **SCREEN SAVER** - Automatic phosphor protector for your tube. Genie will even let you blank the screen manually to discourage peepers.
- **COMMAND STACK** - Lets you access up to the last 2K worth of commands you typed.

Best of all with Genie you only have to load what you use. If you don't need to use an ASCII chart don't load it. Need extra alarms? then load two alarm clocks. Your own personal Genie.

And now Genie user's have the advantage of Genie accessories, which can be installed and accessed like any Genie function - any time:

- a **Scientific Calculator** more powerful than most real calculators. Has a full range of math, science and statistical functions and very high accuracy.
- a **Spelling Helper**™ To help you find the correct spelling without leaving your word processor.
- a **Print Spuffer**™ (spooler/buffer) to speed and manage all your printing.
- a development pack to let you write your own pop up programs, including many routines for windowing, and source code for our Rolodex. Writing pop ups is a snap. (Please call for details)

Only Genie Gives you so much for so little.

ADVANCED SOFTWARE TECHNOLOGIES

452 West 47th Street
New York, NY 10036
(212) 247-0150
ORDERS ONLY CALL

1-800-437-4400 Ext 800

Yes please rush me the following software for my
_____ Z110/120 _____ PC compat.

_____	Copies of Genie	@ $54.95	_____
_____	Spelling Helper(s)	29.95	_____
_____	Print Spuffer(s)	24.95	_____
_____	Scientific calc.	19.95	_____
_____	Genie pack (I want it all) Saving $40	$89	_____
		Sub total	_____
Check encl ☐		shipping	4.00
M/C ☐		NY Tax (8.25%)	_____
VISA ☐		TOTAL	_____

Card Number _____

Expires _____ Your Tel Num _____

Name _____

Address _____

Company _____

City/ST/Zip _____

Heath/Zenith Users' Group

REMark®

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

Editorial and Advertising Assistant ..... Lori Lerch (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

Official magazine for users of HEATH ZENITH computer equipment.

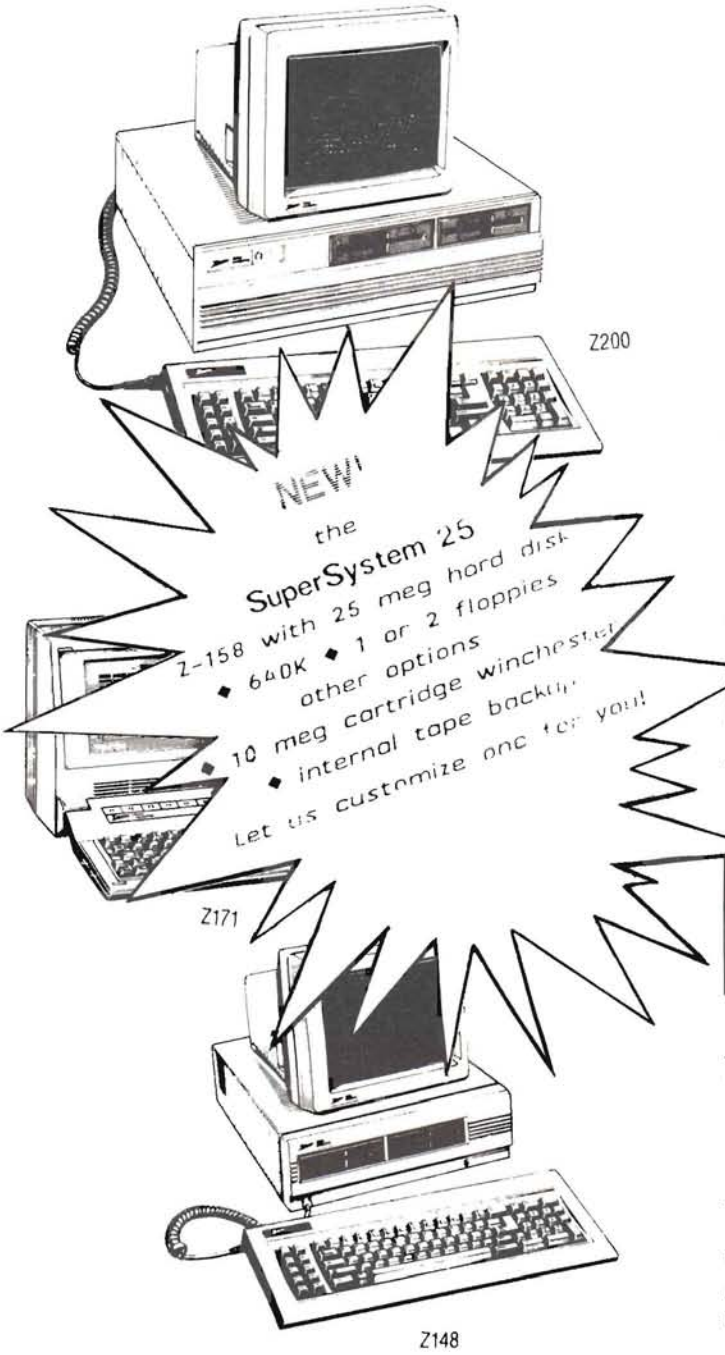
Volume 7, Issue 2 • February 1986

Index of Advertisers

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

On The Cover: *Pictured is the ZF-158, the expandable personal computer. See article by Pat Swayne on Page 65.*

First Capitol Delivers Zenith. Your Way.



Have it your way.

If you're looking for a PC, you couldn't do better than Zenith. And you couldn't buy your Zenith at a better place than First Capitol. We'll put together a system the way YOU want it. And we'll do it for a price that competes with the guy who just sells boxes that fall on his loading dock.

Great Prices. And A Great Machine.

We're committed to Zenith. And to you. the standard Zenith models that come from the factory are a good buy. And we'll be glad to sell you one at a price that will match anyone. But if you want the full potential, we'll fix it up for you. Larger disk drives. More memory. Internal tape back up. Removable winchesters. All at a budget pleasing price.

We make Zenith Scream.

When a new model comes out, we're among the first to get it. Our Zenith fanatics take it apart, prod it and poke it until it screams for mercy. Then we put it back together (usually with some new parts) so that it REALLY screams. With power.

You'll be back.

Check with us. Check elsewhere. We'll put together a custom quotation for you we GUARANTEE you'll find interesting. We think you'll come back to us. There's a reason for the growing list of companies, government agencies, and end users that know us as the source for Zenith. Your Way.

First Capitol Computer

1106 First Capitol Dr.
St. Charles, MO 63301

1-800-TO-BUY-IT (800-862-8948) Orders and quotes

1-314-946-1968 Technical support and order status

The StockSystem Challenge.

We will meet or beat any dealer's price on new stock Zenith systems. And for enhanced systems, no one can touch our combination of value, performance, and custom service.

Before you buy . . . . . .

Call Us First. And Last.

ZENITH | data systems

AUTHORIZED SALES AND SERVICE

Gearing Up For HUGCON86

Bob Ellerton
HUG Manager

It's time for the Heath/Zenith Users' Group to begin considering the trials and tribulations of the International Heath/Zenith Users' Group Conference. Each year we are looking for new and exciting products to be exhibited during a fast-paced weekend in Chicago. Also, we try to provide seminars on a variety of subjects we have been contacted about during the months leading up to the Conference. We need your HELP!

Participation by the users has been one of the major reasons that the Conference has been such a tremendous success in the past. We would hope that each of you would sit down and drop us a note on the events and topics you want to see most. We will do our utmost to bring each suggestion to life at HUGCON86. Remember, this is YOUR conference and we would like you to help us make this year a special year for all who attend.

Again, the International Heath Users' Group Conference will be hosted by the O'Hare Hyatt Regency located in Rosemont Illinois. The Hyatt is only a short shuttle ride from O'Hare Airport. Those of you who attended last year will remember the effort the Hyatt put into making our weekend extremely enjoyable. The dates for the Conference weekend have been established. Plan now to be in Chicago August 15, 16 and 17.

We have included the Official Registration Form in this issue of REMark. If you are interested in seeing the Chicago area while you are visiting the Conference, please indicate what you would like to see or do on the line provided for this purpose. This may help us in planning some special side activities. Related computer equipment manufacturers should take notice of the registration information to insure that you receive your Exhibitors Package outlining HUGCON86 and the Exhibit Area.

We will need your input as soon as possible to make things happen during HUGCON86. You might think about your favorite hardware or software projects. If you have attended the International Heath/Zenith Users' Group Conference in previous years, how about your choice for speakers you would like to hear most. Maybe a suggestion about a particular exhibit you would like to see at HUGCON86. Any ideas you can come up with will help us in selecting activities for the weekend.

Keep in mind that the total dinner registration can not exceed 1,100 people. Register NOW! Don't miss the fun! We hope to see as many of you as possible during a special weekend in Chicago. *

Heath/Zenith Users' Group

INTERNATIONAL
HEATH/ZENITH USERS' GROUP CONFERENCE
Official Conference Registration Form

O'Hare Hyatt Regency
Rosemont, Illinois
August 15, 16, 17

Name(s): _____

_____

_____

Company: _____

Address: _____

City: _____ **State:** _____ **Zip:** _____

Enclosed is $25.00 for each of the individuals listed above to attend the International HUG Conference being held the weekend of August 15, 16, and 17, 1986. Please send tickets along with information regarding hotel reservations and transportation.

Amt. Enclosed: _____ No. Attending: _____

For Our Information:

Which Heath/Zenith computer do you now operate? _____

Are you a Non-User-Attendee?	Yes	No
Are you a computer related manufacturer?	Yes	No
If yes, would you like exhibit information?	Yes	No
Are you, or anyone in your party, interested in activities in or around the Chicago area other than the Conference?	Yes	No

If yes, please indicate any suggestions you may have:

_____

Special Notice To Exhibitors:

Exhibitor Information Packages are available on request from the Heath/Zenith Users' Group. Those of you interested in exhibiting your products should contact us as early as possible to ensure a position at this year's event.

For Your Information:

The $25.00 you are paying for your reservation to the International HUG Conference entitles you to all functions of the Conference. Visitor tickets, for those of you simply attending the seminars and exhibits, are available for $10.00. Visitor tickets do not include eligibility for prizes or food while attending the Conference.

Please send your completed registration form or suitable copy to:

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

Registration(s) must be post marked no later than July 31, 1986. Cancellation will not be accepted after this date.
Sorry, We cannot accept purchase orders

BUGGIN' HUG

GRADE Correction

Dear HUG:

Thank you for your generous coverage of my teacher's electronic record book program, GRADE, in the November issue of REMark. The description contained a couple of errors which could cause difficulty for a purchaser of the program. First, it requires that the user have a printer. Second, the MS-DOS version allows a maximum of 90 students and 90 scores for each student in each file. The CP/M version allows a maximum of only 40 students and 40 scores for each student in each file. I am sorry for any problems which these errors may have caused.

Sincerely,

R. R. Ludeman, Ph.D.
Professor,
Engineering Technology
Andrews University
Berrien Springs, MI 49104

"Mighty Mini RTTY" In December

Dear HUG:

There is a discrepancy between so much of the schematic diagram (Figure 1) and the text (second paragraph page 59) as concerns driving the signal LED. The schematic is correct. The patch of the controlling signal is "*** from pin 5 of U2 to pin 9 of U3, thence from U3-8 to Q1 ***". Note should be made of the other error in that paragraph. Q1 "sources" rather than "sinks" the LED current. I apologize for failing to catch these before their publication.

Sincerely,

Ray Isenson
4168 Glenview Drive
Santa Maria, CA 93455

"Automatic Ramdrive Utilities"

Dear HUG:

REMark Volume 6, Issue 12, December 1985 had an article, "Automatic Ramdrive Utilities," by Joseph Hebda. This was a good article, but it implied it was limited to the H/Z-100. The same or a similar approach is also applicable to the H/Z-89. My other problem with it was the use of menus. Menus are fine if you are writing programs for your secretary or data entry person to use; however, a direct approach is much faster and can still be used by non data processing personnel with a short instruction sheet.

I have installed a 1 meg. C.D.R. Super Ram 89 with Expander board and Clock in my H-89 with Magnolia. It was purchased from Peter Shkabara, ANALYTICAL PRODUCTS. The following is the procedure I use to activate Super Ram 89 and load the ramdisks with the desired programs.

My primary use is with WordStar and dBase II and the speed of working with 1 meg. of RamDisk is very impressive, particularly with large files.

The CDR Super Ram 89 software generates a COM file, SRAM .COM, when the install program INSRAM.COM is run. SRAM allows you to reactivate the ramdisks without going through the menu and re-answering the parameter questions.

I used the Magnolia SETAUTO.COM program to install SRAM in the Cold Boot routine so that Ram 89 is automatically activated on Cold Boot. SRAM provides the option of erasing the Directory when first powering up or of not erasing the directory if it becomes necessary to Cold Boot during a session with the power on. With Super Ram 89 you can Cold Boot and not lose what is in the RamDisk memory, as long as the power has not been turned off.

I split the RamDisk into two disks, A: = 248k for programs, and B: = 676k for data. I use the Warm Boot from RamDisk option as it significantly speeds up switching back and forth between WordStar and dBase II.

Once the RamDisk is activated, it is necessary to transfer WordStar, dBase II, and other programs to it. This should be as automatic as possible. Because I elected to use the Warm Boot Option, I cannot use a SUBMIT file in the Cold Boot routine. When the RamDisk is activated, Ramdrive A: becomes the logged on drive, but Directory A: is blank. It does not contain SUBMIT.COM; therefore, a SUBMIT file will not run.

The following procedure will rapidly transfer desired files to the RamDisk with a minimum number of keyboard entries. It requires writing and storing SUBMIT files with the extension, ".SUB" on a disk. In my case, they are on the BOOT disk which becomes drive C: when the RamDisk is activated.

STEP	KEYBOARD ENTRY	PROGRAM
1	BOOT<ret>	Use SETAUTO.COM to install SRAM in the Cold Boot routine so the RamDisk is activated on Cold Boot.
2	A0>type C:<ret>	Log On C:, floppy w/programs.
3	C0>type SUBMIT PIPC/A<ret>	**FileName: PIPC/A.SUB** PIPV A:=C:SUBMIT.COM PIPV A:=C:PIP??????.* PIPV A:=C:D.COM

(This .SUB file transfers SUBMIT.COM, D.COM, and any filename with "PIP" as the first three letters to Drive A:. All SUBMIT files, FileName.SUB, to be used on drive A: start with the letters "PIP").

4	C0>type A:<ret>	Log On RamDrive A:
5a.	A0>type SUBMIT PIPWS<ret>	**FileName: PIPWS.SUB** PIPV A:=C:WS??????.* PIPV A:=C:KEYWS.*

(Transfers WordStar to Drive A:)

or 5b.	A0>type SUBMIT PIPDBII<ret>	**FileName: PIPDBII.SUB** PIPV A:=C:DB??????.* PIPV A:=C:SB.COM

(Transfers dBase II and Spellbinder to Drive A:)

or 5c. A0>type SUBMIT PIPWSDB<ret>

FileName: PIPWSDB.SUB

```
PIPV A:=C:KEYWS.*
PIPV A:=C:DB??????.*
PIPV A:=C:SB.COM
```

(Transfers WordStar, dBase II and Spellbinder to Drive A:)

PIPV.COM is the CP/M PIP.COM modified to automatically Verify every time a file is copied. If you are using PIP.COM, change the lines in the .SUB files to read: PIP A:=C:filename.*[V].

The number of steps could be reduced by PIPing all of the program files to A: in Step 3. This could be done by Listing them all in the PIPC/A.SUB file. However, this would require waiting for all the programs to be transferred when for some sessions only one is needed.

I did not write SUBMIT files to transfer the DATA files from floppy to RamDrive B:. Generally, I am working with only one or two data files or with a series of data files that all have the same first three or four letters in the filename. Thus, it is easy to use PIPV (or PIP) to transfer the data files to the RamDrive B:.

A caution in using RamDisks!! If you have made any changes in a program or data file, you MUST REMEMBER to transfer those files back to a floppy before you turn the power off. If this is not done, ALL THE CHANGES WILL BE LOST when the power is turned off.

Either PIP.COM or a SUBMIT file, filename.SUB, can be used to transfer changed or new files from RamDisk to floppy. In Joseph Hebda's article he used AUTOSAVE.SUB and SHUTDOWN.SUB, which is a good approach, to SAVE all files which might have had changes.

I have found the speed of a RamDisk makes it well worth the cost. Elimination of the time waiting for drive access to take place has also made the 4 mhz modification previously installed worthwhile. WordStar commands: Find and Replace, Move Block, etc. and dBase commands: Find, Index, Sort, Replace, etc. really fly.

Sincerely,

Leon C. Daugherty
28719 Blythewood Drive
Rancho Palos Verdes, CA 90274

Radiation In CRTs?

Dear HUG:

If they haven't already, your readers will probably soon receive an ad for a "radiation shield" for their computer monitor. Carefully worded so as to remain legal (maybe), the ad will imply that CRTs (cathode-ray tubes) emit ionizing (X-ray like) radiation, that this is harmful to the operator and that the way to save yourself is to purchase the advertised shield to put between you and the CRT. One ad I just received from a national distributor of computer products shows an operator dressed in a full anti-contamination suit complete with face shield and rubber gloves. It is emblazoned with the yellow and red radiation symbol and is really frightening. The CRT shield from this company costs $161+, which is even more frightening. Such devices are utterly worthless.

In my 12 years as a health physicist for the University of California at Irvine, I have yet to measure ANY ionizing radiation from ANY CRT, old or new, color or monochrome. Since we are exposed to somewhere around 100 millirems of ionizing radiation each year from the sun, the stars, naturally occurring radioactive materials in the ground, in our buildings and in ourselves (much, much more if we live in Denver, fly in airplanes or get a medical X-ray), and this normal background radiation is readily measurable with even the simplest gieger counter, then the failure to detect any radiation from a CRT using much more sensitive detectors means that CRTs do not emit hazardous radiation. Period. Any claim to the contrary is misleading to the point of fraud. You would do your readers a service if you would warn them of this fraud and reject any advertising copy of this nature submitted to you.

Thank you.

Yours truly,

William G. Nabor
27172 Huerta Street
Mission Viejo, CA 92692

ZCPR3 Semi-Automatic Installation

Dear HUG:

I am disappointed! In reading through your two (count them two!) articles on ZCPR3 installation (Nov. Issue) for the H-89, I found that neither one of them addressed the semi-automatic installation for ZCPR3 that has been available for over a year, Mine! HEATHZ3.LBR is available from the ZCPR3 bulletin board (I sent them a copy), the individual files are available on compuserve in DL4 of the Heath Users' section (Use Key: ZCPR3), and I have seen the library file on nearly every ZCPR3 bulletin board around the country.

Now that I have that off my chest, I feel obligated to point out several misconceptions prompted by the two articles. First, Rick Swenton's article says there is no "In Between" for ZCPR3 installation, either buy a $149.00 auto-install program, or have an extended knowledge of assembly language programming... not true! My installation programs are public domain! Secondly, he shows the wheel byte location at 3BH, while this is normally okay, 3BH is a reserved memory location and I repeatedly ran into problems putting additional information in page 0, best to have it in upper memory as Richard Allen's article suggests. In Mr. Allen's article, CONFIGUR can be run on ZCPR3 system disks as long as the modify cold boot command option is not exercised. There is a program in my library called PATCON.SUB that patches this option out, but as long as you remember not to use that option then it will be okay. (It will foul up the startup routine if you do.)

My installation consists of several utilities, plus a bunch of submit files that modify the BIOS and install ZCPR3. The text file Z3H89.TXT is a step-by-step installation guide and the whole process can be done in an hour or two. Additional files are also available on compuserve to install ZCPR3 on CP/M-80 vers. 2.2.04. I guess I should have written this letter a little sooner, HUH?!

Jerry Furst
17049 Vista Bluff
San Antonio, TX 78247

Interactive Graphics Controller

Dear HUG:

I have recently installed the Interactive Graphics Controller made by SigmaSoft and Systems in my H-89. The system produces

Continued on Page 82

Adding An Internal Dual Disk Drive To The H-89 Series!

R. Kenneth Strum
P.O. Box 35502
Brooks AFB, TX 78235

Recently, I added a couple of internal Mitsubishi 4851 double-sided, double-density 48 TPI drives to my Z-90 computer. They are terrific! My Z-90 now uses the same type of drive and disks that the Z-100 and Z-100 PC computers use. This gives me: (a) very reliable 640 K internal disk space, and (b) use of the most common and inexpensive blank disks around. I can buy a box of 10 soft-sectored double-sided disks at my local Heath/Zenith Computers for about $15. Compare that to the $40 list price for a box of the single-sided hard-sectored my old faithful H-89A uses. (Lindley Systems sells both hard-sectored and soft-sectored disks at $15 for ten. I've used them for several months now without any problem.) I think you'll want to consider these factors, plus a few others, if the idea of dual internal drives appeals to you.

There are several sources for internal dual disk drive setups. Prices for most (if not all) of them have dropped dramatically since the concept was first introduced. In some cases, the price reduction has approached 50 percent! I bought mine from PAYLOAD for about $275. The kit included drives, mounting screws, second cable connector, "Y" power connector, and of course, instructions. I didn't get the shielded mounting case because I didn't need it. With mounting shield the PAYLOAD package costs $312. Another vendor, QUIKDATA, sells the same drives in their setup package for about $308. No mounting shield is included at this price. QUIKDATA includes a preconfigured data cable whereas PAYLOAD gives you the components to alter your current cable. H & H COMPUTER ENTERPRISES offers their DSM series of internal dual drive setups for $340 ($355 for hard sector). According to their literature, they include everything needed to install the drives.

Which vendor to buy from is an important consideration. The increased prices usually provide you with an easier setup. My setup, using the Mitsubishi drives (a very reliable, proven design) was purchased at a rock bottom price. Installation was simple enough, but the instructions assumed a great deal of expertise. As Murphey would have it, I made almost every installation error possible, plus invented some new ones. The one error that I didn't make, was one I thought I did. I Spent considerable time trying to correct something that wasn't wrong. Luckily, Charles Porter at PAYLOAD was very helpful. He sent me, at his expense, a new part, to help find my trouble. He researched Heath/Zenith manuals on a saturday to verify the Z-89-37 jumper connections. Charles was, of course, unaware that I planned to write this article.

Table 1
My Z-89-37 Jumper Configuration

J1	170
J2	170
J3	1
J4	No Jumper
J5	No Jumper
J6	Jumper on
J7	No Jumper

Henry Fale at QUIKDATA has also helped me on other non-related problems, even though he didn't know me from Adam. For me, this kind of personal help is very important. I'll gladly pay more to get it. Another vendor consideration is: "Which brand drive do you want to install?". Both PAYLOAD and QUIKDATA sell the Mitsubishi drives. H & H doesn't list who makes their

drives. They've been shipping their drives since January 1983. According to their brochure, they've shipped well over 1,000 setups, with only one returned. Heath and Zenith seem to be using Shugart drives in the Z-100's. My local Heath/Zenith Computers and Electronics sells the Mitsubishi for add-on drives.

Now that you've decided on the vendor and brand drive you want, what else do you need to consider? Tracks Per Inch (TPI). Life is not simple. Almost every vendor offers their dual drive setups in both 48 TPI and 96 TPI versions. Both H-DOS 2.0 (with either HOS-5-UP or the HUG Soft-Sectored Support Package) and Soft-Sectored Heath/Zenith CP/M 2.2.04 will support dual-sided 48 and 96 TPI operation. If you have a Hard-Sectored controller, then you can still get double-sided use out of the new drives. HUG's Hard-Sectored Support Package for H-DOS has a double-sided SY: driver. Check your manual to see if your CP/M BIOS will support double-sided, hard-sectored drives.

Livingston Logic Labs' BIOS-80 V5 will support hard-sectored, double-sided 48 and 96 TPI drives. BIOS-80 is available from QUIKDATA for about $40. QUIKDATA also sells a similar driver for H-DOS. H & H sells their setup with hard-sectored software for an additional $15. The second operating system modification is $25. Charles Porter, proprietor of PAYLOAD COMPUTER SERVICES, told me that a special shielded case for 96 TPI drives was necessary, while most 48 TPI drives could be mounted inside the '89 using the existing case. Most people with the dual drives, and those who repair the drives, recommend 48 TPI. If you use the existing case, you'll have to drill a couple of mounting holes, if you want to securely mount both drives. On my Z-90, I attached one drive using the existing mounting holes. I just slid in the other drive. It sits there nicely. If I get around to it, I'll remove the case, drill the holes, and mount the second drive.

As I see it, the advantages of going with either 48 or 96 TPI are:

> **48 TPI:** Less expensive disks, more reliable operation (less loss of data), "compatibility" (many vendors can't supply software in 96 TPI format), less maintenance to keep the drives working, cheapest installation.
>
> **96 TPI:** Lowest cost per byte of information, most online storage, fastest drives, able to use some large programs that require more disk space than available on 48 TPI drives.

Now that we've discussed some of the benefits of dual internal drives, let me tell you how I installed (and should have) mine. My Z-90 was equipped just as it came from the factory. It had the soft-sectored controller board (Z-89-37), no hard-sectored card, and one Siemens single-sided "Barn Door" drive. The package from PAYLOAD contained instructions, two drives, a "Y" connector, eight mounting screws, and a non-conductive brace plate. One of the drive's doors would not close. A burr was holding one of the sensor bars down. I fixed it by gently pulling up on the bar and rubbing the burr off with my finger.

I started off installation by configuring the new drives as described in the PAYLOAD instructions. You must configure "Head Load", "Drive Select" and "Terminator" options. "Head Load" was preset on my drives to HM, the PAYLOAD recommended position. Secondly, "Drive Select" simply labels one drive as system unit 0 (for drive SY0: under H-DOS, or A: under CP/M) and the other as system unit 1 (SY1: or B:). Selection is via a jumper. Much easier than cutting a program block as on the old Siemens. Next, remove SY0: or A: drive's terminator Integrated Circuit. That's all to configuring the new drives.

The second "big" task is to remove the Siemens drive and the disk drive data cable per instructions. I took the data cable out and attached a second drive connector about two inches in from the factory drive connector. You need a vice to do it right. It's easy to put the connector on wrong. The connector must be on "straight". If you look at the factory connector, you'll see how "little feet" each touch a separate wire in the cable ribbon. Use your vice and put on the supplied connector in the same way. It's simple to do. Just be careful to keep the connector straight so each foot is centered over the right wire. I did it right the first time. (This was the step I thought I had messed up.) The way I kept the connector straight was to line up the pin 0 and pin 34 "feet" on the same wires as the factory connection. Then, I checked the other 32 "feet" to see if they were centered on the wire. Finally, I held the connector in place with my hand while I pressed the connector on with my vice.

Table 2
Some Vendors Who Carry Dual Drive Setups

H & H COMPUTER ENTERPRISES, INC
P.O. Drawer H
Blacksburg, VA 24060
(703) 552-0599

PAYLOAD COMPUTER SERVICES
15006 Sun Harbor
Houston, TX 77062
(713) 486-0687

QUIKDATA COMPUTER SERVICES, INC
2618 Penn Circle
Sheboygan, WI 53081
(414) 452-4172

Once this is done, you are ready to connect the data cable to the drives and the controller board. Remove the old drive by unscrewing the mounting screws and disconnecting the power supply cable. Slide in the new drives. Then, connect the "Y" power supply, and connect the drives to the controller card. On the Z-89-37 board, you must connect to P4 at the bottom, rather than the "normal" position at P3. In order to use P4, you'll have to disconnect the cable going to the external disk drive connector at the back. The Z-89-37 connector at P3 doesn't seem to support more than one drive. The PAYLOAD instructions didn't mention this, although it's important. You configure the J jumpers on the Z-89-37 board as though the internal drives were external. See Table 1 for my jumper configurations. Also, be sure your precompensation is set for none. I had a WANGO drive attached to my Z-90 at one time, so my board was set for precompensation. The Mitsubishi drive's won't work at double-density or double-sided if the precompensation isn't set for "none". Now that this is done, all you have to do is test your drives with either H-DOS or CP/M utilities to verify they're working correctly. If they are, screw down the drives and close it up.

If you have a hard-sectored disk controller, then your installation is very similar. There's only one plug on the hard-sectored board for you to connect the disk drives to. There aren't any jumpers to set.

Needless to say, I'm very pleased with my new drive setup. The Mitsubishi drives are very, very quiet. They have a 6 Millisecond seek time, which is twice as fast as my fastest Siemens single-sided drive. My "new" Z-90 really provides the computer tool I need. Since my modest computer program needs are met by my current (or available) H-DOS and CP/M programs for the H-89/Z-90 series, I can continue to use my system. This increases the return on my monetary and time investment.

If you need more disk storage, especially to keep your "all-in-one" a true "all-in-one", consider getting an internal dual drive setup from your favorite vendor.

Table 3
Other Vendors Mentioned

LINDLEY SYSTEMS
21 Handcock Street
Bedford, MA 01730
(617) 275-6821

*

FIRST ANNUAL

MAY 2nd, 3rd, and 4th

at HENRY VIII INN & LODGE

Bridgeton, St. Louis County,
Missouri 63044

OR MORE INFORMATION CONTACT

MIDWEST HUGCON
HEATH/ZENITH Computer Center
3794 McKelvey Rd.
Bridgton, MO 63044
(314) 291 1850
(VENDORS Contact Steve Hart)

Sponsored by the
St. Louis Heath
Users' Group

SLHUG BBS
(314) 291 185
(MODEM)

768k In The ET-100

WARNING! WARNING! WARNING!

INSTALLATION OF THIS MODIFICATION REQUIRES THAT YOU SOLDER JUMPER WIRES DIRECTLY TO THE FOILS ON YOUR CPU AND ACCESSORY BOARDS. THIS WORK SHOULD ONLY BE DONE BY PERSONS EXPERIENCED IN MAKING SUCH CHANGES TO PRINTED CIRCUIT BOARDS. THE POSSIBILITY EXISTS OF PERMANENTLY DAMAGING YOUR CPU AND/OR ACCESSORY BOARDS. THIS MODIFICATION IS NOT SANCTIONED IN ANY WAY BY HEATH COMPANY OR ZENITH DATA SYSTEMS. NEITHER THE HEATH/ZENITH TECHNICAL CONSULTANTS NOR THE HEATH USERS' GROUP STAFF MAY BE ABLE TO HELP YOU IF YOU ATTEMPT THE MODIFICATION AND YOUR COMPUTER DOES NOT WORK.

Pat Swayne
HUG Software Engineer

The ET-100 is a computer trainer that is based on the H/Z-100 computer. When it is expanded with the ETA-100A accessory, it almost becomes a complete H/Z-100 computer. However, the maximum amount of memory you can put "in the box" is 192k, and there are no expansion slots that would allow you to add more memory. The internal memory is arranged as three banks with 64k each. In this article, I will describe a method for modifying the expanded ET-100 to use 256k chips in each bank, giving you the ability to place up to 768k total in the unit.

Preparation

To perform this modification, you will need at least 9-256k dynamic RAM IC's (Heath part no. 443-1268), a 74AS00 IC (443-1243), a 14-pin IC socket (434-298), a 16-pin IC socket (434-299), a 20-pin IC socket (434-311), 4 feet of 30 or 28 gauge "wire wrap" wire, and some double-stick tape. You will also need the EA-100 Learning Computer Accessory Assembly Manual that came with your ETA-100A package. **Note:** In place of the 74AS00 IC, you can use a 74ALS00 or a 74LS00, but if you plan to try one of the commercially available speed modifications, such as the one described in the April 1985 REMark in your ET-100, you should use the 74AS00.

Disassembly

( ) Unplug your ET-100 and remove all accessory cables (disk and printer cables, etc.). Unplug the keyboard cable.

( ) Using the instructions starting on page 7 of your EA-100 assembly manual as a guide, open the cabinet of your ET-100 and remove the power supply heat sink and the main (CPU) board. Set the main board and cabinet top aside, and remove the accessory board from the bottom of the cabinet.

Note: You will have to unplug several connectors when you remove the cabinet top, the main board, and the accessory board. Be sure to note where these connectors go.

Accessory Board Modification

( ) Remove all IC's that may be in locations U655 through U681. Save the ICs by wrapping them in foil, or placing them in conductive foam. These IC's can be used to expand your video board to color operation if you have not already done so. Just follow the instructions on page 22 of your EA-100 assembly manual.

Make the following connections on the foil side of the accessory board:

( ) Jumper P603 pin 2 to U650 pin 2. **Note:** Pin 2 of P603 is the first pin at the end of P603 that is closest to P604. It is on the side of P603 that is closest to U635. There may be a numeral "1" printed on the component side of the board beside this pin (even though it is pin 2). There is no foil trace at this pin, just a pad.

( ) Jumper U650 pin 18 to RP606 pin 16.

( ) Jumper RP606 pin 1 to U675 pin 1.

( ) Jumper all pin 1's of U675, U674, and U673 together.

( ) Jumper all pin 1's of U657, U656, U655, U666, U665, and U664 together.

( ) Jumper all pin 1's of U676 through U681 together.

( ) Jumper all pin 1's of U667 through U672 together.

( ) Jumper all pin 1's of U658 through U663 together.

( ) Jumper all pin 1's of U675, U656, U680, U671, and U662 together. **Note:** There will be other jumpers at each of these pins.

( ) Carefully inspect all work. If you have an ohmmeter, use it to ensure that all of the jumpers are connected correctly.

( ) Insert 9–256k dynamic RAM ICs at locations U655 through U663.

( ) If you have 9 more 256k ICs, insert them at locations U664 through U672.

( ) If you have 9 more 256k ICs, insert them at locations U673 through U681.

( ) Set the accessory board aside for now.

Main Board Modification

In this section, you will make the connections that are shown in Photo 1. Refer to this photo for placement of the 74AS00 and the jumper wires. When I made the original modification that is shown in the photo, I soldered some of the connections directly to IC pins. However, I do not recommend that you use that technique, but rather that you use sockets as described below.

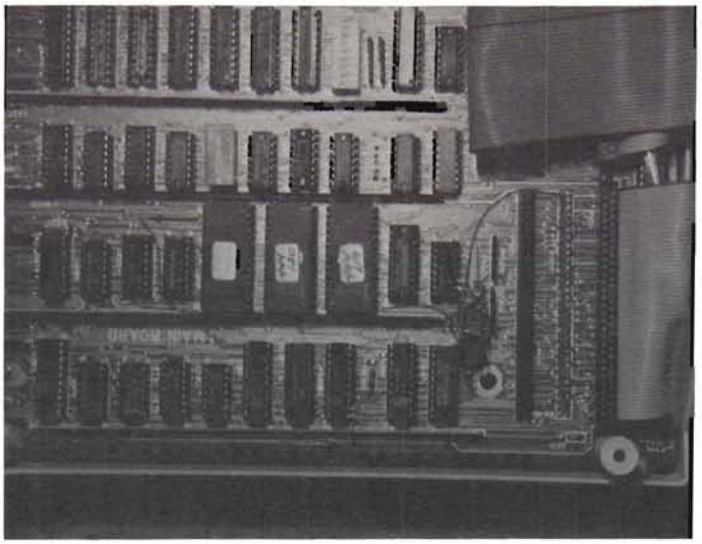


Photo 1
Main board modification.

( ) Remove the IC at U260 and temporarily store it wrapped in foil or inserted into conductive foam. Remove the IC at U256 and store it temporarily in foil or conductive foam.

( ) Bend pins 1 and 2 of the 20-pin socket outwards to 90 degrees from the other pins. Connect one end of a 1.5" (3.75 cm) length of jumper wire to these pins, so the pins are connected together. The other end of the wire will be connected later.

( ) Plug the prepared 20-pin socket into location U260. Make sure that pins 1 and 2 of the new socket do not contact pins 1 and 2 of the socket beneath them.

( ) Bend pins 2 and 3 of the 16-pin socket outwards to 90 degrees from the other pins. Connect one end of a 2.5" (6.75 cm) length of jumper wire to pin 2, and another 2.5" length to pin 3. The other ends of these wires will be connected later.

( ) Plug the prepared 16-pin socket into location U256. Make sure that pins 2 and 3 of the new socket do not contact pins 2 and 3 of the socket beneath them.

( ) Make sure that 5 of the feed-through holes in the area between U227 and U256 are clear (have no solder in them). If there are not 5 clear holes, use a suction type solder removal tool to remove the solder from 5 holes. You will be passing wires from the component side of the board to the foil side through the feed-through holes. Do not use the holes by pin 1 of RP202, because they will be covered later.

( ) Bring the end of the jumper wire connected to pins 1 and 2 of the socket in U260 around the edge of the board and connect it, on the foil side, to the pin of B205 that is closest to pin 1 of U260. This pin is grounded.

( ) Pass the wire from pin 2 of the socket in U256 through a feed-through hole and connect it to pin 2 of U260 on the foil side of the board.

( ) Pass the wire from pin 3 of the socket in U256 through a feed-through hole and connect it to pin 1 of U260 on the foil side of the board.

( ) Using double-stick tape, temporarily mount the 14-pin socket upside down (with the pins up) to the main board between the end of B204 and RP202. Pin 1 of this socket will be at the end closest to the mounting hole (where a screw holds the board down when it is installed), and on the side closest to U227. Note that pin numbering will be clockwise around the socket, instead of counter-clockwise, because it is upside down.

( ) Scrape about 1/8" (.2 cm) of the solder resist (green paint) from the wide foil between the end of B204 and pins 10 and 11 of U227 on the component side of the board. Connect a jumper from this scraped foil to pin 14 of the upside-down socket. This foil carries +5 volts when the unit is in operation.

( ) Scrape a small amount of the solder resist from the edge of the foil pad at the mounting hole by the upside-down socket. Connect a jumper from this foil to pin 7 of the upside-down socket. This foil is grounded.

( ) Connect pins 3 and 4 of the upside-down socket together with a small length of jumper wire. Connect pin 5 to pin 11 with jumper wire.

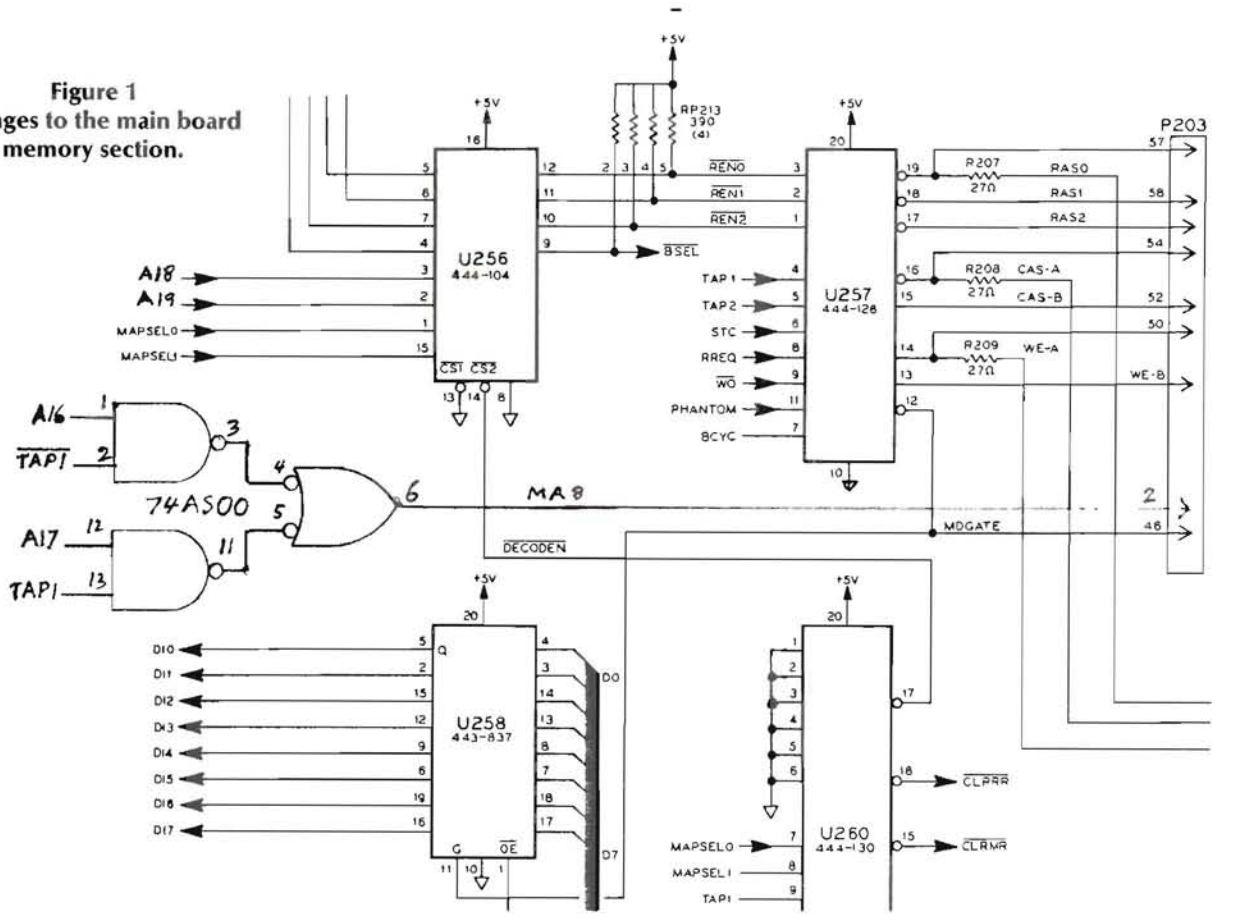
( ) Jumper pin 2 of the upside-down socket to a feed-through hole near U255. See Photo 1 for the location of this hole. If you would like to check for the correct hole with an ohmmeter, it connects to U255 pin 1. Solder the wire to the feed-through hole. You may have to scrape some solder resist from the hole first.

( ) Jumper pin 6 of the upside-down socket to pin 2 of P203, on the component side of the board. Pin 2 is at the end of P203 where a "1" is printed, on the opposite side of P203 from the "1". There is no foil trace at this pin.

( ) Connect a jumper wire from pin 1 of the upside-down socket through a feed-through hole to pin 3 of U256 on the foil side of the board.

( ) Connect a jumper wire from pin 12 of the upside-down socket through a feed-through hole to pin 2 of U256 on the foil side of the board.

Figure 1
Changes to the main board memory section.



( ) Connect a jumper wire from pin 13 of the upside-down socket through a feed-through hole to pin 9 of U260 on the foil side of the board.

( ) Straighten the pins of the 74AS00 as described on page 15 of your EA-100 assembly manual.

( ) Carefully pry the upside-down socket from the double-stick tape, leaving the tape on the board. Slide the 74AS00 under the socket with the pins up, with pin one at the end nearest the mounting hole, and press the socket onto the pins of the 74AS00. Make sure that the IC sticks to the double-stick tape.

( ) Replace the ICs that you removed from U256 and U260, inserting them into the new sockets at those locations.

( ) Carefully check all work done to the main board. If you have an ohmmeter, use it to ensure that all of the jumpers are connected correctly.

( ) Using the instructions starting on page 16 of your EA-100 assembly manual as a guide, reassemble your ET-100. You may want to perform the test on page 21 before you reconnect the video board. If you do, the screen should show 256k of RAM if you have 9-256k ICs, or 512k if you have 18 of them, or 768k if you have 27 of them installed.

( ) After you have finished assembling your ET-100, ensure that it runs disk software and graphics correctly.

Theory of Operation

Figure 1 is part of the schematic of the main board memory section, showing the changes made by the modification. Normally, the ROM at U256 uses address lines A16 and A17 to select one of the three 64k banks of RAM. After the modification, lines A18 and A19 are connected to this ROM, which causes it to select 256k banks instead of 64k banks. The PAL IC at U260 normally uses address lines A18 and A19 to ensure that U256 only selects 64k banks from the first 256k of address space. After the modification, these inputs to the PAL are grounded, so that U256 can select 256k banks out of the full 1 megabyte address space.

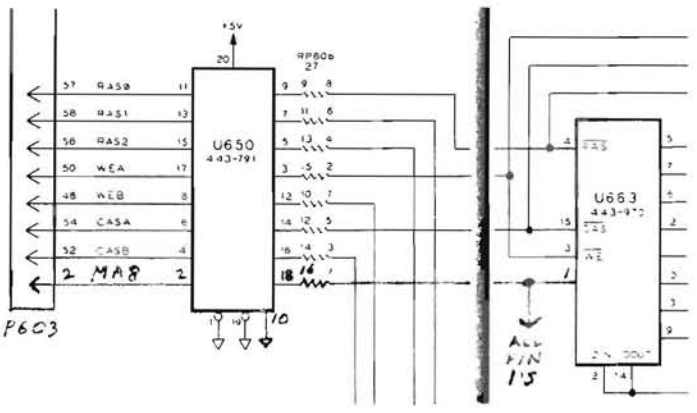


Figure 2
Changes to the accessory board memory section.

The extra 74AS00 that was added to the main board in the modification is used to decode address lines A16 and A17 along with a multiplexing signal called TAP 1 into the extra multiplexed ad-

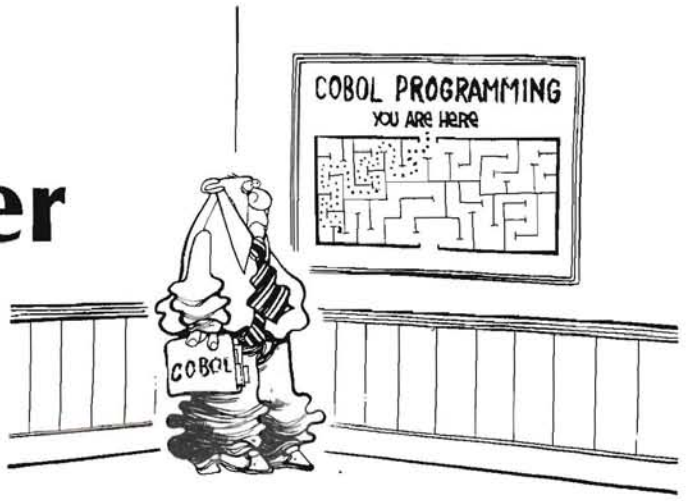
dress line (MA8) required by the 256k memory ICs. The output of this circuit goes to P203 pin 2 (normally unused), so that it can be passed to the memory ICs on the accessory board.

Figure 2 is part of the schematic of the accessory board memory section, showing the modification changes. The new multiplexed address line created on the main board arrives at P603 pin 2, passes through an unused gate in buffer U650 and an unused resistor in RP606, and then is applied to pin 1 of each memory IC.

*

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

COBOL Corner XVI



Introduction

If you were successful with Program #8, you are ready for our next program challenge! If you are having problems, write to me with complete details and examples. DO NOT forget the stamped, self-addressed business-size envelope.

Many computer programming languages use Tables (sometimes called arrays). Also, spreadsheet and data-base software use tables. In computer jargon, a table is a collection of data in which each item is uniquely identified by a label, by its position relative to the other items, or by some other means. (Nearly everyone has seen and used the income tax tables). Tables allow data to be stored compactly and the data can be referenced or retrieved efficiently by a program. Every table contains "table entries". A table entry may contain a "table argument" and one or more "table functions". A month-name table (we will use one) has 12 table entries. Each month table entry contains a table argument (the month number) and one table function (the month's name or month-name abbreviation) for the month represented by the month number. The field that would be used to locate the appropriate table entry (month-number field) is called the "search argument". We could carry this month table further with more than one table function. For example, the maximum number of days in each month.

COBOL Table Lookup Methods

The COBOL programmer provides a Table organized so that a program can efficiently access and retrieve data fields that relate to a given search argument. The process of locating the specific data field is called "table lookup". COBOL provides several general table-lookup methods:

1. Serial Search.
2. Binary Search.
3. Positional Addressing.

Serial Search

A serial search (works like it sounds) compares the search argument with the argument of the first entry. If they are the same, the correct table entry has been found. If they are not the same, the program logic causes the search argument to be compared to the following table arguments, one-by-one, until either of these following conditions are reached:

1. A table argument is found that matches the search argument.
2. The end of the table entries is reached.

If there is no matching table argument, the following exists:

1. There are erroneously coded search arguments.
2. The table contains only "exception" entries that will require special handling in the program.

A COBOL programming enhancement can be programmed into a serial search if the table is sequentially organized. This is called "serial search with early exit" (We will be using this in our programs). Sequential organization just means that the table entries are arranged in sequence (ascending, for example) according to the value of the table entry code. The early exit can reduce the number of comparisons required. Rather than just checking for an equal (same) condition between search and table arguments, the program logic tests to see if the search argument is lower than the table argument. If it is, the search continues. If the search argument is NOT lower than the table argument and not equal to it, a table entry will not be found and the lookup will be terminated. Thus, early-exit programming means fewer comparisons are needed and less time is used for lookups!

Binary Search

When the program's table contains many table entries (over 30), a serial search for arguments whose entries are located down deep in the table can be time consuming. For these "long-sequentially" organized tables, the programmer should use a

"binary search" in place of a serial search. The binary search will greatly reduce the average time required to locate a table entry. **Note:** The table MUST be sorted sequentially and have at least 30 entries.

The binary search technique starts making comparisons in the middle of the table rather than the first table entry as is done with serial search. Then, either the top half or the bottom half of the table is searched, depending upon the relationship of the search argument to that half-way table argument. If the search argument is less than the middle table argument, the upper half of the table becomes the search area. Next, the search argument is compared with the argument of the middle entry in the selected half of the table to determine the relationship. Again, depending upon the result of that comparison, the table is split in half and the middle entry of that portion of the table is tested. This halving is repeated until an equal table argument is found or there are no remaining portions of the table to divide. (We will do a program using binary search after the serial search program).

Positional Addressing

The positional addressing method does not require a SEARCH statement. It is a rapid table lookup method. The search argument specifies the relative position of the corresponding table entry. A month-name table could be an example. Remember, the positional addressing applies ONLY to positionally organized tables. Such tables require an unbroken sequence of numerical table arguments. Before trying to access a positionally addressed table, the programmer MUST validate to ensure that the search argument is within the range of the table. For example, in a month-name table it is a must that the search argument value be greater than zero and less than 13. If an out-of-range value is used, erroneous processing occurs. In a few months, we will delve deeply into data validation.

Multiple Level Tables

The tables we are going to be working with at this time will be examples of single-level, or one-dimensional tables. Some programs require tables with two or more dimensions. Such tables are called multiple-level or multi-dimensional tables. The federal withholding tax tables are an example.

Table Data Source

The table data can be coded directly into the program or read into the program from input data record files. The direct coded approach is referred to as a "hard-coded" table. The second one is called an "input-loaded" table. The hard-coded table is cumbersome if additions, deletions, or modifications to the table become necessary very often. They require a programmer to make the changes and then the programmer would have to recompile the program. This quality could be beneficial or detrimental, depending on the application. Sensitive or security dependent data-items such as salaries by job level or confidential data are best handled by hard-code tables, to name a few.

Input-loaded tables are best suited for volatile data. Changes or updates could be handled by data-processing clerks as required. It is, however, more difficult to ensure that unauthorized or erroneous changes are not made. Furthermore, the correct input-loaded table must be processed each time the program is run. This could be time consuming.

New COBOL Elements

We must learn some new COBOL reserved words and their usage:

1. The OCCURS clause.
2. Subscripts.
3. The VARYING phrase of the PERFORM statement.
4. The INDEXED BY clause.
5. The SET statement.
6. Index data-items.
7. The SEARCH statement.

I will define these elements now and we will really learn them as we use them in the next few programs. Also, this might be a good time to state — Tables are defined in the WORKING-STORAGE SECTION of the DATA DIVISION and accessed in the PROCEDURE DIVISION!

Program #9

Program #9 is going to use the same input file — Payroll Records (FILEL4.DAT) — that we used for Program #8. However, we are going to produce a fairly simple Employee-Department Report so that we can emphasize the Table Handling Concepts. The program will use two hard-coded tables in the WORKING-STORAGE SECTION:

1. Month-name Table.
2. Department Number and Name Table.

The program will use serial search with indexing and the SEARCH statement.

Month-Name Table

Let's look at the first table:

```
XXXXXX   Ø1  WS-PROGRAM-TABLES.
XXXXXX   Ø5  MONTH-CONSTANTS.
XXXXXX   1Ø  FILLER     PIC X(Ø9)  VALUE "  JANUARY"
XXXXXX   1Ø  FILLER     PIC X(Ø9)  VALUE " FEBRUARY"
XXXXXX   1Ø  FILLER     PIC X(Ø9)  VALUE "    MARCH"
XXXXXX   1Ø  FILLER     PIC X(Ø9)  VALUE "    APRIL"
XXXXXX   1Ø  FILLER     PIC X(Ø9)  VALUE "      MAY"
XXXXXX   1Ø  FILLER     PIC X(Ø9)  VALUE "     JUNE"
XXXXXX   1Ø  FILLER     PIC X(Ø9)  VALUE "     JULY"
XXXXXX   1Ø  FILLER     PIC X(Ø9)  VALUE "   AUGUST"
XXXXXX   1Ø  FILLER     PIC X(Ø9)  VALUE "SEPTEMBER".
XXXXXX   1Ø  FILLER     PIC X(Ø9)  VALUE "  OCTOBER"
XXXXXX   1Ø  FILLER     PIC X(Ø9)  VALUE " NOVEMBER"
XXXXXX   1Ø  FILLER     PIC X(Ø9)  VALUE " DECEMBER".
XXXXXX   Ø5  MONTH-TABLE   REDEFINES MONTH-CONSTANTS.
XXXXXX   1Ø  MT-MONTH-NAME   PIC X(Ø9)  OCCURS 12 TIMES
```

To establish a hard-coded storage table, the programmer must first define the month's names with VALUE clauses. Notice that each of the 12 table elements has been specified as a separate data-item. (Such separate definitions are not always used as we will find in later programs). The approach of using separate data-items for each item is recommended in this program because:

1. The table entries are easier to read.
2. The table is easier to modify if additions, changes, or deletions to this table should become necessary (say we wanted to change to abbreviated month names).

Often, data for table elements will not be of equal length, like our month names. MAY contains 3 characters and the longest name, SEPTEMBER, has 9 characters. So, to make our coding easier we like to have uniform field lengths. I have padded the shorter name months with RIGHTMOST blank spaces to make them all equal.

Notice that each of the data-items was not given a user-defined name but instead named FILLER. This is appropriate because the data is referred to only by the "redefined" data-name associated with the OCCURS clause.

OCCURS Clause

After the table data has been specified with VALUE clauses, the table must be "redefined" with an OCCURS clause. The OCCURS clause is used to indicate how many times a particular field or group of fields is repeated in the table. The FORMAT of the OCCURS clause is as follows:

```
OCCURS integer TIMES
```

This OCCURS clause can be used with any data-item description that has a level number from 02 thru 49; it cannot be used as an 01-level! Also, an OCCURS clause cannot contain a VALUE clause. This prohibition that a data-item cannot contain both an OCCURS and a VALUE clause explains why the table must be defined with the required VALUE clause data and then "redefined" within the OCCURS clause. Whenever an OCCURS clause is used with a data-item, either a "subscript" or an "index" must be used when referring to that item in the PROCEDURE DIVISION. I will discuss "subscripts" now and "indexes" later in the article.

Subscripts

Subscripts are used to reference a specific occurrence of a repeated field defined with the OCCURS clause. This subscript indicates which occurrence of the field is being referenced. For example, in our month-name table, JANUARY is occurrence 1, MAY would be occurrence 5, etc. There are two forms of subscripts: "literal" and "variable". An example of a literal subscript would be:

```
MT-MONTH-NAME (5)
```

Of course, this would refer to MAY. Literal subscripts have limited use because they must be "pre-coded" in the program. Variable subscripts are much more powerful and therefore more frequently used. The variable subscript uses a value field to indicate the occurrence number. For example:

```
MT-MONTH-NAME (WS-MONTH)
```

Here, we would have to look at what value has been assigned to WS-MONTH field in the WORKING-STORAGE SECTION. If we found the following:

```
WS-MONTH                    PIC 9(02)  VALUE  8.
```

The month-name would be AUGUST.

The variable subscripts must be an elementary numeric integer data-item defined in the DATA DIVISION. The length of the subscript field must be sufficient to contain the number of occurrences specified in the OCCURS clause. When the program is executing the statement containing the subscript, the value of the subscript must be greater than zero and NOT greater than 13 for the month-name table. If the value is not in this range, erroneous results or program termination would be the result. Thus, it is imperative that the subscript value be validated before it is used. (We will go into great detail on data validation in future articles).

Department Number And Name Table

Now, let's look at the second hard-coded table that we will be using:

```
XXXXXX 05  DEPT-NBR-NAME-DATA.
XXXXXX 10  FILLER  PIC X(20)  VALUE "1000ADMINISTRATION  "
XXXXXX 10  FILLER  PIC X(20)  VALUE "1100PURCHASING      ".
XXXXXX 10  FILLER  PIC X(20)  VALUE "1200PERSONNEL       ".
XXXXXX 10  FILLER  PIC X(20)  VALUE "1300MARKETING       "
XXXXXX 10  FILLER  PIC X(20)  VALUE "1350ADVERTISING     "
XXXXXX 10  FILLER  PIC X(20)  VALUE "1900PUBLIC RELATIONS"
XXXXXX 10  FILLER  PIC X(20)  VALUE "2000ENGINEERING     "
XXXXXX 10  FILLER  PIC X(20)  VALUE "3000ACCOUNTING      "
XXXXXX 10  FILLER  PIC X(20)  VALUE "3500DATA PROCESSING "
XXXXXX 10  FILLER  PIC X(20)  VALUE "4000MANUFACTURING   ".
XXXXXX 05  DEPT-NBR-NAME-TABLE  REDEFINES DEPT-NBR-NAME-DATA
XXXXXX 10  DEPT-TABLE-ENTRY     OCCURS 10 TIMES
XXXXXX                          INDEXED BY ST-INDEX.
XXXXXX 15  TD-DEPT-NBR    PIC 9(04).
XXXXXX 15  TD-DEPT-NAME   PIC X(16)
```

Compare this table with the previous month-name table. You will note two differences:

1. Data for the table elements are not of equal length again. To make them a uniform field length, I have padded them with LEFTMOST blank spaces. (You will see why later).
2. I have added the "INDEXED BY ST-INDEX" to the OCCURS clause.

This is a sequential organized table where I will use "indexes".

INDEXED BY

The FORMAT of the OCCURS clause has changed a little. Here is the new FORMAT:

```
OCCURS integer TIMES
INDEXED BY index-name
```

NOTE: No period is used until the end of the INDEXED BY clause! When the INDEXED BY clause is written, a user defined index-name must be specified. This index-name is used like a variable subscript for identification of a specific occurrence. There are three important differences between subscripts and indexes:

1. When the index-name is specified in the INDEXED BY clause, the index is provided for by the COBOL compiler. That is, the programmer does not establish a data-name in the DATA DIVISION as would be required for a variable subscript.
2. Subscripts are used for the storage of occurrence values where indexes contain displacement values. Displacement refers to the number of positions from the starting position of the table. (Number of name characters from the start.)
3. Since index values are different from normal data values, initialization of the index cannot be done with a MOVE statement as can be done with subscripts. Also, arithmetic statements, such as ADD or SUBTRACT, cannot be used for incrementation or decrementation of the index value. The SET statement is used for the index value initialization, incrementation, decrementation, or other modifications.

SET Statement

There are two FORMATS for the SET statement. They are (I will show only the formats we will be using at this time):

```
FORMAT-1
        SET index-name TO integer

FORMAT-2
                          UP BY
        SET index-name           integer
                        DOWN BY
```

Here are some examples:

```
1  SET ST-INDEX          TO 1
2  SET ST-INDEX          UP BY 1
3  SET ST-INDEX          DOWN BY 1.
```

As you can see, FORMAT-1 can be used to initialize the index. FORMAT-2 is used for index incrementation or decrementation. The SET statement takes occurrence numbers and converts them to index displacement values, and vice versa.

Index Data Items

Index data-items are specified in the DATA DIVISION to allow for storage of index displacement values for later use without their conversion to occurrence numbers. To specify an index data-item, the INDEX clause is used. Remember that PICTURE and VALUE clauses are not allowed in the INDEX clause; index lengths are uniform and they are always assigned by the COBOL compiler. You should see the distinction between indexes and index data-items. Indexes are specified with the INDEXED BY clause and they are used like a subscript to refer to a certain occurrence of a data-item containing an OCCURS clause. The only function of an index data-item is to store, without conversion, the contents of an index. In the PROCEDURE DIVISION, index data-items can only be used in a SET statement or in a relation condition test with an index or another index data-item.

SEARCH Statement

The SEARCH statement is used in the PROCEDURE DIVISION to lookup table entries. It has two FORMATS (I will show only variations we are currently using.):

```
FORMAT-1
        SEARCH table-name
            AT END imperative-statement
            WHEN condition

FORMAT-2
        SEARCH ALL table-name
            AT END imperative-statement
            WHEN condition
```

When FORMAT-1 is used, the table specified is stepped through, entry-by-entry, until the condition expressed in the WHEN phrase is satisfied. As soon as the condition is satisfied, the SEARCH ends. If the end-of-table is reached, the condition has not been satisfied and the AT END imperative-statement is executed. Therefore, with the SEARCH statement, index incrementation is handled "automatically", condition testing is accomplished by the WHEN phrase, and the AT END phrase specifies the end-of-table processing. But, initialization of the index must be done by coding a SET statement prior to the SEARCH statement.

Binary Search

The FORMAT-2 SEARCH statement is used to provide a binary search. A binary search requires the use of a KEY clause in the OCCURS clause of the DATA DIVISION and of the ALL phrase in the SEARCH statement in the PROCEDURE DIVISION.

When a table is to be used in a binary search, the KEY phrase must be specified to indicate whether the table arguments are arranged in ASCENDING or DESCENDING sequence. The FORMAT of the OCCURS clause with a KEY phrase follows:

```
OCCUR integer TIMES
    ASCENDING
                KEY data-name
    DESCENDING
INDEXED BY index-name
```

It is the programmer's responsibility to make certain that the table is actually arranged in accordance with the KEY phrase specification! COBOL will not arrange the data!

Specification of the reserved word ALL triggers the binary search logic. REMEMBER THIS! When the ALL is specified to obtain a binary search, the SEARCH statement handles the index initialization. Therefore, the programmer need not code a SET statement as must be done for the serial search!

PERFORM/VARYING Statement

Another way to do a serial lookup is with the use of the PERFORM statement including the VARYING phrase. The sequential lookup coding is minimized with this method. The FORMAT (Simplified to fit our current need.) follows:

```
PERFORM procedure-name
   VARYING identifier-1
      FROM identifier-2 BY identifier-3
         UNTIL condition
```

Notice that a subscript would be used that would be initialized and incremented by the FROM and BY phrases. It is important to note that, once the match between search and table arguments has been found, the subscript must be DECREMENTED by "one" to cause it to reflect the proper table occurrence number! This is because the subscript has been incremented one time after the matching condition has been found by the PERFORM/VARYING statement.

Summary

I have covered the items required for table processing in this article and I know that you will not understand it all at this time. I have stated in prior "COBOL Corner" articles that the way to learn COBOL was by doing. This is what we will do again! We will do programs using these various table processing items and by using them you should see their logic. I will not go into multiple tables or input-loaded tables at this time. We have a lot to cover without these. We will do three programs:

1. Serial search with indexing and the SEARCH statement.
2. Binary search with indexing and the SEARCH ALL statement.
3. Serial lookup with subscripts and the PERFORM/VARYING statement.

Let's review some of the material we have covered to help you understand a little better:

1. Tables are defined in the WORKING-STORAGE SECTION of the DATA DIVISION and they are accessed in the PROCEDURE DIVISION.
2. To establish hard-coded table data (the type we are going to use for the next three programs), the table is specified with VALUE clauses and then REDEFINED with an OCCURS clause.
3. The OCCURS clause is used to indicate how many times a certain field or group of fields is repeated. An integer must be specified to indicate the number of repetitions.
4. The OCCURS clause:
   A. Can be used with any data-item description that has a level-number from 02 thru 49 (It cannot be an 01-level).
   B. Cannot be used with a data-item that contains a VALUE clause.

5. Whenever an OCCURS clause is associated with a data-item, either a subscript or an index must be used when referring to that item in the PROCEDURE DIVISION.

6. Subscripts are used to reference a specific occurrence of a repeated field defined with the OCCURS clause. They are ALWAYS enclosed in parentheses and may be of either literal or variable form.

7. The field used for a variable subscript must be:

   A. An elementary numeric data-item defined in the DATA DIVISION.
   B. Of sufficient length to contain the number of occurrences specified in the OCCURS clause for the field being referenced.

8. At execution time, the subscript field must contain a value greater than zero but not larger than the number of occurrences of the field referenced.

9. Indexes are used like subscripts, but there are three important differences:

   A. When an index-name is specified in the INDEXED BY clause, the index is "automatically" provided for by the COBOL compiler. The programmer does not establish a data-item in the DATA DIVISION as would be required for a variable subscript.
   B. Indexes contain displacement values. Be sure you know what this means. Subscripts contain occurrence values.
   C. Initialization, incrementation, and decrementation of an index must be done with a SET statement.

10. SET statements can have two (2) FORMATS:

    A. FORMAT-1 is used for index initialization.
    B. FORMAT-2 is used for index incrementation.

11. Table lookups can be accomplished by use of the following:

    A. PERFORM/UNTIL statement.
    B. PERFORM/VARYING statement.
    C. SEARCH statement.

12. SEARCH statements can be used for:

    A. Serial search.
    B. Binary search.

Closing

I did not start a program this month because we had too much material to cover. Therefore, you should have some time for a "homework" assignment! I would like to have you use the COBOL-80 or the COBOL-86 Manual to lookup, read, and STUDY the following:

1. Table handling.
2. Indexes.
3. Subscripts.
4. OCCURS clause.
5. INDEXED BY clause.
6. SET statement.
7. SEARCH and SEARCH ALL statement.
8. PERFORM/VARYING statement.

We will be using tables in most of the programs that will follow in this series. This is an important COBOL programming requirement that you should know. So, we will practice with its many formats many times. Be sure to have yourself prepared! Next month I will go over most of the coding required for a program using the tables that we discussed this month. *

Controlled Data Recording Systems Inc.

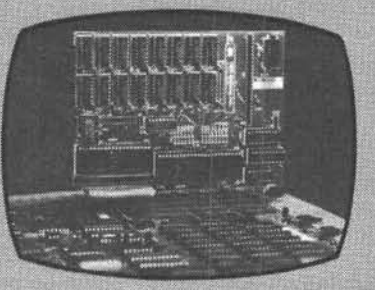
Quality Products and Support for the Heath/Zenith Community

New for the H/Z89-90 Computer Users:

The Super RAM 89 Package

Now get the speed and power of a high capacity Ram Drive System at a reasonable price.

Using the Ram 89 package, standard software shows an immense improvement in speed. Depending on the software being run, programs may execute 10 times faster than when run through standard floppies.



Ram 89 comes as a two card set that plugs into the left hand (16k expansion) side of the computer. No computer modifications required.

Board 1 has two banks of 256k chips possible for a total of 512k. Either or both banks are able to use 64k chips instead. Board 1 can be used by itself, with board 2 added at a later date.

Board 2 has an additional 512k, plus it has a real time clock capability, and a SASI interface hardware capability. Board 2 piggybacks onto board 1.

Now get RAM 89

HDOS DRIVER Software Only **$35**

RAM 89 DVD supports RAM SY0: and SY1:, added autoload features, plus more!

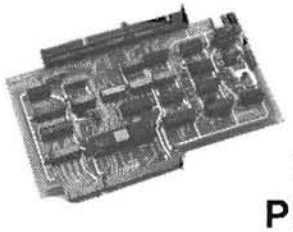
The Ram Drive Software (SRAM) allows one or two logical ram drives. The ram drive(s) can be located starting anywhere from logical A: to O: (standard drives get relocated). SRAM can be set to start at logical A: and warm boot with ram (no floppy disk accesses needed). Ram drive attatches to any of the versions of CP/M 2.2 bios used in the H/Z89-90.

GET A FULL MAGABYTE OF RAM FOR YOUR H/Z89-90

Board 1 includes hardware manual and ram drive software with no ram: $190.00. Each 256k bank, add $56.00. Board 2 (must have board 1). With no ram, no clock, no SASI $90.00. Each 256k bank, add $56.00. Ask about clock, SASI pricing.

FOR THE H/Z89-90 COMPUTERS

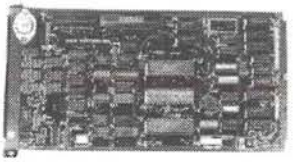
THE ONE CONTROLLER FOR 8" & 5.25" DRIVES THE FDC-880H PRICE: $395



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

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. The FDC-880H operates with or without the Heath hard sectored controller.

THE FDC-H8 DOUBLE DENSITY 8" AND 5.25" CONTROLLER FOR THE H8 COMPUTER PRICE $495



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

- 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.

MODIFY89

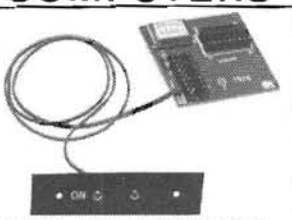
Allows owners of the FDC-880H or the FDC-H8 to read/write to the following disktypes:

Access S.S.D.D., Cromemco S.S.D.D., DEC VT180 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-8001 A S.S.D.D., Osborne S.S.D.D., Osborne S.S.D.D., Otrona D.S.D.D.*, Zorba D.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.D.D., Zerox 820-II S.S.D.D. Plus more, Standard (Tests for H/Z37 and C.D.R. Disk types)

PRICE $49.95

FOR THE Z100 COMPUTERS

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.

SUPER DRIVES 85

A powerful floppy drive package for the Z100 or any computer that can use 8" floppy drives.

Two extended technology 5.25" drives in a package with interface cables, case and power supply, that look exactly like standard 8" floppy drives to the computer. Now have **2.4 megabytes** of **floppy file storage capacity** in one 5.25" package.

ONLY **$695** LIST

Dealer inquiries invited.

For information about these products contact:
Controlled Data Recording Systems Inc.
7210 Clairemont Mesa Blvd.
San Diego, Ca. 92111 Phone (619) 560-1272 or contact a Heath/Zenith dealer that carries the C.D.R. Systems Inc. line of products.
New - C.D.R. Systems Bulletin Board (619) 560-8929

Using The H-100 As A Time Machine Part III

Arnold W. Seibel
851 Cypress Street
Monterey, CA 93940

The preceding articles in this series developed a pair of Z-DOS assembly language procedures that can be used to time the execution of any procedure that is CALLed between them by a main program. We also modified Randy Meyers' SETPOINT procedure to be used with the timing procedures, and wrote a main program to pull it all together.

The remainder of this series is devoted to a more mundane purpose than benchmarking subroutines: finding out the date and time with a minimal number of keystrokes.

Another Simple Sub-Subroutine: DROPZERO

What we're after in this set of procedures is a readable version of the time and date. One thing that will help make these numbers readable is to drop leading zeros from the month, the day, and the hour. When H2ASCII leaves me with a string like '01' for 1 o'clock or '03' for March, I don't want to see that '0' on the screen. DROPZERO gets rid of it for me. It inspects the ASCII tens' digit in the two-byte string constructed by H2ASCII, and gets rid of it if it's a '0'.

The first instruction in this procedure is CMP AL,'0'. This instruction subtracts the value of '0' (30 hex) from whatever is in AL; however, it doesn't leave the difference anywhere, it only sets some flags in the 8088's flag register. A flag is a single bit; it's set to 1 or 0, depending on the results of certain instructions. One flag is ZF, the Zero Flag; it gets set to 1 if the result of a computation is zero, and to 0 if the result is non-zero. (This is obviously confusing, but think of it as being "true" if it's set to 1: when ZF is 1, it's true that the result was zero.) If the value that H2ASCII left in AL is '0', or 30 hex, then this subtraction will result in zero, and ZF will be set to 1.

The next instruction includes JNE, which stands for Jump if Not Equal. This instruction tests ZF; if ZF is 0, then the two values just compared are not equal, and the jump is made. The instruction JNE DROP_RET tells the 8088 to jump to the instruction labeled DROP-RET if the things compared were not equal, i.e., if the byte in AL was not an ASCII '0'. At DROP-RET it finds the RET instruction, so it does nothing more in this procedure, just goes back to where it left off in the calling procedure.

If JNE finds that ZF is 1, however, then that means that the values compared were equal, and the jump is not made. The 8088 then executes the next instruction, MOV AL,0. This puts the value 00 hex, also known as ASCII NUL, in AL. A NUL in a character string causes nothing at all to be printed — not even a blank. Thus, if DROPZERO finds that H2ASCII has left it a '0' in AL, it replaces that '0' with nothing at all — the string as it appears on the screen will be one character shorter.

Note that the code in this procedure could have been set up to jump if the byte in AL is '0', rather than to jump if it is not '0'. The reason it is set up the way it is has to do with execution time. A conditional jump takes four times as long to execute when the jump is made, as when it is not made and the 8088 goes on to the next instruction. For this reason, it's always a good idea to set up code with a jump in it so that, in the most common situation, the jump will not be taken. In my program, DROPZERO is going to be used on months (1-12), days (1-31), and hours (1-12). There are more month/day combinations that are both one-digit numbers than are both two-digit numbers, and there are more one-digit hours on the 12-hour clock than two-digit hours, so this code is "optimized" — about two microseconds' worth.

A New Subroutine: CLOCK

The procedure CLOCK is one of the two (the other is CALENDAR) that do the work in my NOW program. CLOCK reads and prints the time in a manner very similar to that of TIMEROFF, the second of the two subroutines that I wrote to time the execution of SETPOINT. However, CLOCK has a few tricks of its own. Its main trick is translating the time from the 24-hour-clock format, which confuses me on a daily basis (I admit it, I have trouble add-

ing and subtracting 12 in my head), to the 12-hour format, which has been second nature since I was about four. The translating is done by the procedure AM_OR_PM, which I have written and assembled in the same module with the procedure CLOCK.

Why not assemble AM_OR_PM as a separate module? I had two reasons. First, I can't imagine wanting to call it from any other procedure than CLOCK. Second, it refers to a structure that's defined in CLOCK's module, and I didn't want to go to the trouble of declaring that structure PUBLIC in CLOCK — and I can't figure out how to declare it EXTeRNal in AM_OR_PM. (Any suggestions?) There is one drawback to assembling AM_OR_PM in the same module with CLOCK: Whenever I decide to revise CLOCK, I will have to reassemble AM_OR_PM along with the revised CLOCK. That will take a little extra assembly time, but for a procedure as short as AM_OR_PM, I don't think it matters.

CLOCK begins with two structure definitions which are very similar to those in TIMEROFF. The first, TIME_BYTES, is identical to the structure of the same name in TIMEROFF. The second, TIME_STRING, differs from its counterpart in only one part: the characters added on at the end of the string. In TIMEROFF, the string ends with ' elapsed'; the space and the word appear immediately following the time when the elapsed time is printed. In CLOCK, the string ends with a space; then the word (two-byte) string 'mp', which is 'pm' spelled backwards; and then five spaces. The fives spaces set the time off from the date, which will follow on the same line. The bytes in 'mp' will be swapped when that two-byte value is moved into memory, so it will come out 'pm' as it should. The 'mp' is segregated from the spaces before and after it, and given a label (SUFF) of its own. This is necessary because, if AM_OR_PM discovers that it's morning, it will need to find the bytes labeled SUFF and replace them with 'ma'.

In DATASEG we reserve space for two structures, just as we did in TIMEROFF. In CODESEG we declare two EXTeRNal procedures, H2ASCII and DROPZERO, each of which is assembled in a different module; AM_OR_PM is not declared EXTeRNal because it's in this module (right after the end of the procedure CLOCK).

After initializing the DS register we use INT 21h to call the Get Time function, indicated by the 2C hex in the AH register, and move the bytes into the structure labeled TIME_IN.

In the next block of five lines we get the hour byte into AL and call three successive subroutines to process it: AM_OR_PM, which converts it to the 12-hour clock; H2ASCII, which converts it to ASCII digits; and DROPZERO, which gets rid of its leading zero if it has one. That done, we move the two-byte string in AX into the appropriate word in the structure TIME_OUT, where we are building the output string.

The next three blocks of code translate the minute, the second, and the hundredths of a second into ASCII and move them into the output string. Then we use DOS function 9, Print String, to print the time on the screen. RET instructs the 8088 to RETurn to the calling procedure, and CLOCK ENDP marks the end of this procedure.

Down With The 24-Hour Clock: AM_OR_PM

For such a short procedure, AM_OR_PM has a lot going on in it. Before examining the lines of code, let's look at the structure. Note, first, that the RET is in the middle of it, not at the end. Lesson 1: RET doesn't have to be at the end, it just has to be the last instruction executed in the procedure.

Note also that there is only one RET in the procedure, even though there are actually four different ways it can end: Do nothing if the hour is twelve (the initial pm is correct and the number is also correct for the twelve-hour clock); change the hour to 12 and change pm to am if the original hour is zero (midnight); change pm to am and leave the hour alone if it's 1, 2, 3, ..., 11; or subtract 12 from the hour and leave the pm alone if the original hour is 13,14,...,23. Why not have four different RETurn instructions, one for each outcome? Lesson 2: It's good programming practice — in fact, it's a cardinal rule in structured programming — to write subroutines so that they have only one exit point. Why? Because that makes it a lot simpler for you to figure out why they don't work (on those rare occasions when they don't), and also makes it a lot easier for other people to follow your code when they read it. You'll see how easy in a minute.

Next, note that this procedure has a lot of jumps in it. Depending on the hour (given the four options described above), the procedure has to send the 8088 to different instructions; jumps are the instructions that do that. Only one of the jumps is spelled JMP, which means jump unconditionally. The other three are conditional jumps: JB (Jump if Below); JE (Jump if Equal); and JNE (Jump if Not Equal, which we also used in DROPZERO). Whether or not any of these instructions actually causes a jump to be made depends on the outcome of a preceding CMP (CoMPare) instruction.

When you have that many jumps in a procedure, there are many different ways you could sequence them. How do you pick a sequence? Lesson 3: Arrange the jumps so that, in the most common case, no jumps are taken. The reason for this is that it takes a lot longer to actually make a jump than to "fall through" to the next instruction following the jump.

This procedure is organized so that it takes the least time to execute when the hour is between 1:00 pm and midnight. (Why? Because that's when I'm most likely to be asking this machine what time it is.) Hours from noon to 1:00 pm take about half again as long; hours from midnight to 1:00 am take more than three times as long; and hours from 1:00 am to noon — when I'm least likely to be using the machine — take about four times as long. When we've gone through this code and you understand the instructions, see if you can figure out how you would write this procedure if you work mainly during the morning hours.

AM_OR_PM starts off with a CMP AL,12. This subtracts 12 from the hour byte in AL, but doesn't leave the difference; it just sets the flags. This instruction is followed by not one, but two conditional jumps. The first, JB MORNING, sends the 8088 to the label MORNING if the hour is less than (Below) 12. If that jump is not taken, the 8088 tests the flags again with JE AM_RET. This one sends the 8088 to the label AM_RET (and out of the procedure) if the hour is equal to 12. If that jump is not taken either, it means that the hour is greater than twelve (it's not less than 12, and it's not equal to 12). The 8088 executes the next instruction, SUB AL,12, which subtracts 12 from the hour byte. It then comes to the RET, which sends it back to the calling procedure. Thus, if we're in the pm hours (not counting the hour between noon and 1:00 pm), we get out of this procedure without taking a single jump.

If the JB sends us to MORNING, we first change the 'pm' in the output string to an 'am'. (Remember that the bytes we are MOV-

ing — 'ma' to begin with — will be swapped when they are placed in memory; the printed characters will come out in the right order.) Next, we test the hour to see if it's zero (midnight), with the instruction CMP AL,0. Then we Jump if Not Equal to AM_RET and out; that is, if it's morning but the hour byte is not 00 hex, we're finished. But if the hour is zero (that jump was not taken), we move the value 12 into AL, because midnight on the 12-hour clock is called 12, not zero. Having done that, we execute JMP SHORT AM_RET and return to the calling procedure. The SHORT in that instruction tells the assembler that the label it's jumping to is between -128 and +127 bytes away from the JMP instruction, and that allows the assembler to save one byte in the .OBJ file. (It doesn't save any time.)

Next we come to the end of this procedure, and immediately after that to the end of the code segment. END signifies the end of the module.

A Simpler Problem Than Time: CALENDAR

The procedure CALENDAR is designed to be called right after CLOCK. It prints the date on the screen in the mm/dd/yy format, with leading zeros dropped from the month and day. It won't tell you the day of the week; if you want to know that, you'll have to use the resident DATE function.

CALENDAR is very similar in structure and content to CLOCK. It doesn't, of course, call AM_OR_PM. It does call H2ASCII and DROPZERO. It also uses a new DOS function — number 2A hex, the Get Date function.

The two structures defined at the top of the module are closely analogous to those used in TIMEROFF and CLOCK. The first is used to address the area in DATASEG where we will store the three values returned by the Get Date function. The main difference between this structure and the one we used for the time bytes is that this one has a word-sized (two byte) label. This is used to store the year, which is returned by the Get Date function as a two-byte hexadecimal number. The second structure in CALENDAR is used for the data area where we will build the output string. In DATASEG we use those structures to set aside storage areas labeled DATE_IN and DATE_OUT, and to initialize DATE_OUT with the slash characters and the '$' marking the end of the string.

In CODESEG we declare the two EXTeRNals, identify the segment registers to be used for CODESEG and DATASEG, and then go into the procedure. There we initialize the DS register, then use INT 21h to execute the Get Date function. The three values are moved into the appropriate locations in DATE_IN. Then come two blocks of code that do exactly the same thing to two of our stored values, the month and the day. Each is first converted to ASCII characters by a CALL to H2ASCII; then each is stripped of its leading zero, if it has one, by a call to DROPZERO; then each is MOVed to its place in the output string.

The next block of code processes the year. It first subtracts 1900 from the two-byte year in the AX register. This leaves a value less than 100 in the AL register, so we can treat it just like all the other values we've been dealing with in these procedures. (Come the year 2000, I'll have to rewrite this block.) The call to H2ASCII translates the year into ASCII characters. Then we MOVe the characters into the output string.

Finally, we call the now-familiar Print String function, and RETurn.

A Really Simple Program: NOW

NOW is the main program that calls CLOCK and CALENDAR to print the time and date on the screen. And that is exactly all it does. The entire working code of the procedure consists of the following two lines:

```
CALL    CLOCK
CALL    CALENDAR
```

Everything else in the module is housekeeping required of a main procedure. For explanations, see the discussion of PIXTIMER in the second article of this series.

Assemble, Link, And Run

To use NOW, first type the code into .ASM files, one each for DROPZERO, CLOCK, CALENDAR, and NOW. Use MASM to assemble each .ASM file into an .OBJ file.

At this point, if you've set up a library file with H2ASCII in it, you can add the new subroutines (all the new .OBJ files except NOW.OBJ) to the library file by responding to LIB's prompts as follows (use the same name for the .LIB file that you used before):

```
Library File  b:timing (RETURN)
Operations: +b:dropzero+b:clock+b:calendar (RETURN)
List file: b:timing.cat (RETURN)
```

Then you can put together NOW.EXE by responding to LINK's prompts as follows:

```
Object Modules [.OBJ]  b:now (RETURN)
Run File [A:NOW.EXE]: b:now (RETURN)
List File [NUL.MAP]. b:now/m (RETURN)
Libraries [.LIB]: b:timing (RETURN)
```

Summing Up

If you've typed in all this code, assembled it, made a library file of the subroutines, and linked PIXTIMER.EXE and NOW.EXE, you have a pretty good — if basic — understanding of the power of MASM and the utilities supplied with it. You have a couple of general purpose subroutines — H2ASCII, to translate a hexadecimal byte (with a value less than 100) into a two-character ASCII string, and DROPZERO, to drop the leading zero from such a string. You have another pair of subroutines — TIMERON and TIMEROFF — that you can use to time the execution of anything that you can write as a Z-DOS assembly language procedure.

If you're curious about the execution time of SETPOINT in various languages (languages that will let you get at the hundredths-of-a-second byte), you've got a standard to measure them against. If you're a lot more ambitious than I am, you might even want to try improving on the execution time of the assembly language version. (My guess is that you won't get far.)

You also have a program that will tell you the time and date on one line without requiring you to hit (RETURN). One use for that is time and date-stamping a screenload of information that you want to dump to the printer with PSC.COM.

And finally, if you've felt just a bit intimidated by MASM and the array of powerful tools that accompany it, then I hope you've found at least a little corner of it in which you feel that you understand what's happening. I have.

References

The following books have been invaluable guides during this excursion.

Leo J. Scanlon, *8086/88 Assembly Language Programming*. Bowie, MD: Robert J. Brady Co., A Prentice-Hall Publishing and Communications Company (1984).

This is a good book for beginners in assembly language. It keeps clear the distinction between the 8086/88 instruction set and the directives used by a specific assembler to put together programs made up of those instructions. All the directives described exist in MASM; all except the macro directives are compatible, and very little is said about macros. There are sample programs; the housekeeping directives need a little work to satisfy MASM, but the information in this series of articles should enable you to fix them. Some MASM directives (including those for structures) are not covered.

The iAPX 88 Book. Santa Clara, CA: Intel Corporation (1981).

This book appears as part of Appendix D in the second volume of the Z-100 Technical Manual. It includes very detailed descriptions of all instructions in the 8088 instruction set, as well as useful information about registers, the 8088's way of addressing memory by way of a segment and an offset, and a profusion of more technical things. It is a good book to look into after you've read Scanlon.

Z-DOS. Benton Harbor, Michigan: Heath Company (ND).

The two-volume Z-DOS manual includes complete, if somewhat terse, descriptions of all the MASM directives; careful instructions on how to assemble a program; and thorough guides to the use of LIB and LINK. The tiny sample .EXE program in Appendix P, "Notes on Writing Z-DOS Programs," is the only documentation I've found for the way you set up a return address for the exit to the operating system.

Sponsored by the
St. Louis Heath
Users' Group

DROPZERO

```
page    54,132
title   DROPZERO.ASM  check for '0' in al and change it to NUL

; Uses register ax

; Code segment.

codeseg         segment para    public  'code'
                assume  cs:codeseg

; Procedure.

                public  dropzero

dropzero        proc    near

                cmp     al,'0'          ; compare the ASCII digit to '0'
                jne     drop_ret        ; jump to ret if it's not '0'
                mov     al,0            ; it's '0', so change it to NUL

drop_ret:       ret                     ; return to calling procedure

dropzero        endp                    ; end of procedure

codeseg         ends                    ; end of code segment

                end                     ; end assembly
```

CLOCK

```
page    54,132
title   CLOCK.ASM  print the time with am or pm (12-hour clock)

; This procedure reads the time, stores it in a structure, translates the hour
; to the 12-hour clock, translates the bytes into ASCII, stores the characters
; in another structure, and prints that structure
; Calls the external procedures h2ascii and dropzero
; Uses registers ax, cx, dx

; Define a structure for saving the bytes from the registers.

time_bytes      struc
bhr             db      ?               ; gets hour from ch
bmin            db      ?               ; gets minute from cl
bsec            db      ?               ; gets second from dh
bfrac           db      ?               ; gets 100ths of second from dl
time_bytes      ends

; Define a structure for building the string as the bytes are translated.

time_string     struc
hr              dw      ?               ; hour will go here
                db      ':'
min             dw      ?               ; minute will go here
                db      ':'
sec             dw      ?               ; second will go here
                db      '.'
frac            dw      ?               ; 100ths of second will go here
```

```
                db      ' '                     , space to separate suffix
suff            dw      'mp'                    , low byte is stored high
                db      ':'                     ; separate time from date
                db      '$'                     , end-of-string for printing
time_string     ends

; Data segment.

dataseg         segment para    public  'data'

time_in         time_bytes      <>              , declare byte structure
time_out        time_string     <>              , declare string structure

dataseg         ends                            , end of data segment

, Code segment

codeseg         segment para    public  'code'

                extrn   h2ascii:near            , declare external proc's
                extrn   dropzero:near

                assume  cs:codeseg,ds:dataseg

, Procedure.

                public  clock

clock           proc    near

                mov     ax,dataseg
                mov     ds,ax

                mov     ah,2ch                  ; function number for Get Time
                int     21h                     , interrupt for Function Request

                mov     time_in.bhr,ch          ; move hour from ch
                mov     time_in.bmin,cl         ; move minute from cl
                mov     time_in.bsec,dh         ; move second from dh
                mov     time_in.bfrac,dl        , move 100ths of second from dl

                mov     al,time_in.bhr          , get hour in al to work on it
                call    am_or_pm                ; translate to 12-hr clock
                call    h2ascii                 , translate to ASCII
                call    dropzero                ; drop leading zero
                mov     time_out.hr,ax          , build into output string

                mov     al,time_in.bmin         ; get minute in al
                call    h2ascii                 ; translate to ASCII
                mov     time_out.min,ax         , build into string

                mov     al,time_in.bsec         ; get second into al
                call    h2ascii                 ; translate to ASCII
                mov     time_out.sec,ax         ; build into string

                mov     al,time_in.bfrac        , get 100ths of second in al
                call    h2ascii                 , translate to ASCII
                mov     time_out.frac,ax        , build into string

                mov     dx,offset time_out      , offset of string into dx
                mov     ah,9                    , function no. for Print String
                int     21h                     ; interrupt for Function Request

                ret                             ; return to calling procedure

clock           endp                            ; end of procedure

;

, Subroutine am_or_pm

, This procedure converts the hour from the 24-hour to the 12-hour scale and
, changes "pm" in the output string to "am" if it's morning.
; The hour is passed as a hex byte in al and returned in the same register

; Procedure

am_or_pm        proc

                cmp     al,12                   ; compare hour (0-23) to 12
                jb      morning                 ; jump if hour < 12
                je      am_ret                  ; jump to ret if hour = 12
                sub     al,12                   ; it's 13-23, so adjust it

am_ret          ret                             ; return to calling procedure

morning.        mov     time_out.suff,'ma'      ; it's morning, so fix label

                cmp     al,0                    , then check for midnight
                jne     am_ret                  , jump to ret if it isn't
                mov     al,12                   , it is, so change 0 to 12
                jmp     short   am_ret          , then jump to ret

am_or_pm        endp                            , end of procedure

;

codeseg         ends                            ; end of code segment

                end                             , end assembly
```

CALENDAR

```
page    54,132
title   CALENDAR.ASM.  print the date on the screen

; This procedure reads the date, moves it into a structure, translates the bytes
; into ASCII, drops leading zeros from the month and day, moves the characters
; into another structure, and prints the string on the screen
, Calls the external procedures h2ascii and dropzero
, Uses registers ax, cx, dx

; Define a structure for saving the date from the registers.

date_bytes      struc
bmo             db      ?                       , gets month from dh
bday            db      ?                       , gets day from dl
byr             dw      ?                       ; gets year from cx
date_bytes      ends
```

```
; Define a structure for building the string as the bytes are translated

date_string     struc
mo              dw      ?                       ; month will go here
                db      '/'
day             dw      ?                       ; day will go here
                db      '/'
yr              dw      ?                       , year will go here (2 digits)
                db      '$'                     ; end-of-string for printing
date_string     ends

; Data segment

dataseg         segment para    public  'data'

date_in         date_bytes      <>              , declare input structure
date_out        date_string     <>              ; declare output structure

dataseg         ends                            , end of data segment

, Code segment

codeseg         segment para    public  'code'

                extrn   h2ascii.near            ; declare external proc's
                extrn   dropzero:near

                assume  cs:codeseg,ds:dataseg

; Procedure

                public  calendar

calendar        proc    near

                mov     ax,dataseg
                mov     ds,ax

                mov     ah,2ah                  ; function number for Get Date
                int     21h                     , interrupt for Function Request

                mov     date_in.bmo,dh          ; move month from dh
                mov     date_in.bday,dl         ; move day from dl
                mov     date_in.byr,cx          ; move year from cx

                mov     al,date_in.bmo          , get month in al to work on it
                call    h2ascii                 ; convert to ASCII
                call    dropzero                ; drop leading zero
                mov     date_out.mo,ax          ; build into output string

                mov     al,date_in.bday         ; get day in al
                call    h2ascii                 , convert to ASCII
                call    dropzero                ; drop leading zero
                mov     date_out.day,ax         , build into string

                mov     ax,date_in.byr          , get year in ax (two bytes)
                sub     ax,1900                 ; subtract 1000s and 100s
                call    h2ascii                 , convert 10s and 1s to ASCII
                mov     date_out.yr,ax          , build into string

                mov     dx,offset date_out      , offset of string into dx
                mov     ah,9                    ; function no  for Print String
                int     21h                     , interrupt for Function Request

                ret                             ; return to calling procedure

calendar        endp                            ; end of procedure

codeseg         ends                            , end of code segment

                end                             , end assembly
```

NOW

```
page    54,132
title   NOW.ASM:  print the time and date on the screen

; Calls the external procedures clock and calendar
; Uses no registers

, Stack segment

stackseg        segment para    stack   'stack'
                db      0ffh dup (?)
stackseg        ends                            ; end of stack segment

; Data segment

dataseg         segment para    public  'data'
wrtaddr         label   word
rtaddr          dd      0
dataseg         ends                            ; end of data segment

; Code segment

codeseg         segment para    public  'code'
                extrn   clock:near
                extrn   calendar:near
                assume  cs:codeseg,ds:dataseg,ss:stackseg,es:nothing

; Procedure.

now             proc    far

                mov     ax,dataseg
                mov     ds,ax
                mov     wrtaddr+2,es

                call    clock                   , print the time
                call    calendar                ; print the date

                jmp     rtaddr

now             endp                            , end of procedure
codeseg         ends                            ; end of code segment
                end     now                     ; end assembly, specify now
                                                ; as entry point
```

SPREADSHEET Corner

Part 13

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

This month, we will create a very simple worksheet that I am going to call BUDGET, to review the last article's "pointing" and "printing". When we have finished this review, we will have a good worksheet to start the subjects of GRAPHS and MACROS. I would suggest that you use the Help Screens and the manual for your software to get all the information that you have on these subjects. I will be using the LOTUS 1-2-3 software on my Z-150 PC with an EPSON F-80 printer. (The same things apply to the H/Z-100 series of computers with slight changes that you either know or can read about.)

If you are not using either of these computers, or this printer with LOTUS 1-2-3, you will not be able to do some of this project concerning Graphs and/or MACROS. I would suggest that you read your manuals to see what Graph and/or MACRO possibilities your set-up can do. I choose the LOTUS with the Z-150 and the Epson printer because most of the readers that answered my questionaire in "SPREADSHEET Corner — Part 7" were using them. Several readers indicated that they were using MultiPlan because it came packaged with their computer.

MultiPlan is a good spreadsheet software package for many uses, BUT when it comes to getting into advanced usage, it falls short of filling the need. I would suggest that these readers do as much of the project as they can.

SuperCalc3 Rev 2 has better graphics than the LOTUS 1-2-3 and it can even print the graphs right into a text presentation. I will do a review on this SuperCalc3 Rev 2, as soon as I have the space in one of the upcoming articles. You will not find the MACRO capability with it though.

If you would happen to have SYMPHONY by LOTUS, it will do a good job on this project, but you will have to work out how to convert from my 1-2-3 procedure yourself. I cannot devote the space for the very few readers that might have the software. I will do a short review on SYMPHONY by LOTUS in an upcoming article, so that you readers can know it's strong points and weak points.

I might tell you that I will be doing a short review on dBASEIII and FRAMEWORK by Ashton-Tate, again so that you will know how I feel about these important software products in case you were considering buying any of these software packages.

Before we start this article's project, I want to tell you that, if you have dBASEII V2.41, Ashton-Tate has a new version 2.43, that they are supplying without charge if you will send your system disk to the UPDATE DEPARTMENT. It does have a few new things and it will put the manual into a usable one compared to the old manual. I have just received my copy and I have not had time to use it yet. The next "SPREADSHEET Corner" that requires dBASEII will include a discussion on this upgrade.

Again, the questionaire showed that more readers have dBASEII than any other if they have such software. For those buying, I would suggest buying dBASEIII, if you are using the H/Z-150-PC type of computer. I think that it is time to get started on this article's project. Several readers suggested that I discuss the equipment needed at the start of the article so that they could read that portion to see if they want to do the project!

Do not forget the preparation work that I expect all readers to do before starting any entries on the worksheet. I am not going to spend the time or space discussing this. If you do not remember the requirements, go back a few articles and reread them! My preparation form and the data that we are going to use for this project follows below:

	A	B	C	D
1	YEAR	ORDERS	BACKLOG	SALES
2	1980	2918	1598	1320
2	1981	3580	2559	2619
3	1982	4617	3899	5277
4	1983	5458	2239	7118
5	1984	7324	1080	9483
6	1985	9000	980	10000

Before we start adding the labels and the data, let's set the numbers to dollar values with no decimal places (called currency for-

mat). We do this by typing /WGFC (/Worksheet Global Format Currency) and at the "Enter number of decimal places (0..15): 2", type 0, and press the Return key. Budgets are usually presented with currency format so this looks like a wise choice.

What does your preparation form show? Let's also center all of the labels. We will type /WGLC (/Worksheet Global Label-Prefix Center). Now enter the labels for the worksheet and then the data as shown on my preparation form. Do you find the numbers in currency format and the labels in the center of their columns? A dollar sign should be in front of each currency value. Does this look good to you? I am disappointed with it. The YEARS should not be in currency format, but we have a dollar sign if front of them. Our Global Currency Command above was not so smart. A good preparation form should tell you this before you try it! We must change this column, press the HOME key to move the cursor to the "A" column, type /RFT (/Range Format Text), and in response to "Enter range to format:", type A5.A10, and press the Return key.

Another thing, don't you think that we should have these YEARS labels in the center of their column A? GOTO Cell A2, press the F2 key (Edit key), Press the HOME key to move the cursor to the beginning of the edit line, type ^ (center prefix), and press the Return key. Repeat this process for all the cells in the column — A3 thru A10. How do you like what we have now? Save the worksheet by typing /FS and name the file BUDGET1.

Now, we will "dress-up" our worksheet. First, We will change the column widths by typing /WGC (/Worksheet Global Column-Width), in response to "Enter global column-width (1..72): 9", type 14, and press the Return key. Second, we will add two blank rows above the column labels. Press the HOME key to bring the cursor to Cell A1, type /WIR (/Worksheet Insert Row), at the "Enter row insert range: A1..A1", type A1.A2, and press the Return key.

Also, we will insert a row between the column labels and the first value line. GOTO Cell A4 (don't forget the use of the F5 key), type /WIR, and at the "Enter row insert range: A4.A4", press the Return key. Third, let's add a column between YEAR and ORDERS columns. Move the cursor to any cell in column B and type /WIC (/Worksheet Insert Column), and at the "Enter column insert range:", press the Return key. Change this new column B width to 5. Type /WCS (/Worksheet Column-Width Set), and in response to "Enter Column-Width (1..72):", type 5, and press the Return key. Save your worksheet by typing /FS and name it BUDGET2.

I think that we need to give our worksheet a Title. Press the HOME key to put the cursor at A1. Type, SPREADSHEET Corner COMPANY:—BUDGET IN THOUSANDS, and press the Return key. It looks to me that we could use another divider line. GOTO A2 and type \=, and press the Right Arrow key. With the cursor now at B2, type \=, and press the Right Arrow key. Repeat this procedure for cells C2, D2, and E2. Note: Do you remember the better method that we used in the last project? If not, reread the article.

Which method do you like best? There are nearly always more than one way to do something with 1-2-3, but you will find that some methods make it faster and reduce the chances for errors by reducing the amount of typing that must be done.

GOTO Cell B3, and type ^ | |, and press the Down Arrow key. Do you know why I used the ^ key? It is the prefix to center a label and while it was not important in this case, it is important to use it so that you will remember in other applications. This narrow column is only 5 characters wide. The cursor should now be at B4, type ^ | |, and press the Down Arrow key. Continue this procedure until you are down to Cell B10.

I think that a couple of more divider lines are needed. Do you see where? GOTO A4, type \-, and press the Return key. Copy Cell A4 to Cells B4 thru E4 by typing /C (/Copy), and in response to "Enter range to copy FROM: A4.A4", press the Return key, and at the "Enter range to copy TO:", type B4.E4, and press the Return key. GOTO Cell A10, type /C. In response to "Enter range to copy FROM:", type A4.E4, and press the Return key and at the "Enter range to copy TO: A10", press the Return (accept) key.

How do you like this method vs what we did above for a Copy procedure? This shows that there are many ways to use commands with 1-2-3 and still obtain the correct results. How do you like the worksheet now? Save it by typing /FS and name it BUDGET3. After I studied my worksheet, I decided that the column sequence should be rearranged! I think that a good BUDGET worksheet should have the BACKLOG column last. Do you agree and if you do, how would you do it? I will Move the BACKLOG column to column F, then Move the SALES and BACKLOG columns one column to the left. Here is my method:

1. Move the cursor to any cell in column D and type /M (/Move). In response to "Enter range to move FROM:", type D3.D10, press the Return key, and at the "Enter the range to move TO:", type F3, and press the Return key.

2. Move the cursor to any cell in column E and type /M. In response to "Enter range to move FROM:", type E3.F10, and press the Return key, and at the "Enter range to move TO:", type D3, and press the Return key.

Note! Did you notice what the MOVE command can do? It moved the data in column D to column F, leaving column D blank. It then moved both columns E and F into column D and E as a block move where we only had to define ONE CORNER — D3. Also, look and you will find column F is now blank. Do you see how the MOVE command differs from the COPY command? This is an important thing to remember. We will use this many times.

Save this worksheet by typing /FS and name it BUDGET4. I hope that this roundabout method that I used to create this worksheet will SHOW YOU that a poor PREPARATION FORM will add a lot of reworking to the project! Count how many times I had to rework the worksheet to get one that looked useful and this was a simple worksheet, at that. Thus, you must make out a GOOD PREPARATION FORM before you start to enter anything into the worksheet you are creating! PLEASE! DO NOT ever do your projects this way. You will have to make enough changes even if you do a good preparation! GOTO Cell C12, type FIGURE 1, and press the Return key.

Finally, for review we will print this spreadsheet. Turn your printer ON and align the paper to the top of the page. Type /PPR (/Print Printer Range). Press the HOME key to move the cursor to Cell A1, type a period to anchor the top left-hand corner of the worksheet, press the END key, and press the HOME key again. This should "paint" the entire "active area" that we want to print. Press the Return key. Type A (selects Align option which resets the line-number counter to 1). Type G (selects the GO option which will start the printer typing the "painted" area). When the printing is complete, type Q (selects QUIT option which returns

the program to the Ready Mode). Did you end up with a good printed BUDGET? Save again by typing /FS and name it FIGURE 1!

SPREADSHEET Corner COMPANY:--BUDGET IN THOUSANDS

YEAR	ORDERS	SALES	BACKLOG
1980	$2,918	$1,320	$1,598
1981	$3,580	$2,619	$2,559
1982	$4,617	$5,277	$3,899
1983	$5,458	$7,118	$2,239
1984	$7,324	$9,483	$1,080
1985	$9,000	$10,000	$980

Figure 1

This completes the Review! If you are having any problems, NOW is the time to find them. Go back over last month's article and this project until you CAN DO IT! If you find that you cannot correct your problems, send me copies of your worksheet and your questions. BE SURE to enclose a stamped, self-addresses envelope (business size) if you would like an answer from me. I will try to give you some help!

Now, for this month's project. I will start this project this month and continue it over to next month. The subjects of GRAPHS and MACROS take time to introduce them to any reader who has not worked with them and they cover a wide range of problems to learn how to handle. We will start with the worksheet we have been using for the review!

GOTO Cell E1, type @TODAY, press the Return key, and Format to DATE1: by typing /RFD1 (/Range Format Date 1), and press the Return key. If you use the Help key F1 and take the 1-2-3 manual and read about Date, you find that 1-2-3 is capable of many functions with dates. We will be using just one type now. Save the worksheet by using /FS and naming the file BUDGETGR. You will now have a date shown on the Title line of your worksheet.

First, we will create a Line Graph. Press /GT (/Graph Type), press F1 (the help key) for information on line graphs and when completed press the ESC key. Select LINE. Select X ("Enter X-axis range:"), move the cursor to Cell A5, type a period, move the cursor to Cell A10, and press the Return key. Select A ("Enter first data range:"), move the cursor to Cell C5, type a period, move the cursor to Cell C10, and press the Return key. Select B ("Enter second data range:"), move the cursor to Cell D10, and press the Return key. Select C ("Enter third data range:"), move the ("Enter third data range:"), move cursor to Cell E5, type a period, move the cursor to Cell E10, and press the Return key. Select VIEW to see your graph! Press the spacebar when you are ready to return to the Graph Menu.

Now, we will add Legends to our Line Graph (again, use F1 key to read about Legends). Select Options. Select Legend. Select A ("Enter legend for the A-range:"), type \C3 (or type ORDERS), and press the Return key. Select Legend. Select B ("Enter legend for the B-range:"), type \D3 (or type SALES), and press the Return key. Select Legend. Select C ("Enter legend for C-range:"), type \E3 (or type BACKLOG), and press the Return key. Select QUIT. Select VIEW and now how do you like the Graph? Press the spacebar to return to the Graph Menu.

Next, we will add Titles. Select Options, Select Titles. Select First ("Enter graph title, top line:"), type SPREADSHEET Corner COMPANY, and press the Return key. Select Titles, Select Second ("Enter graph title, second line:"), type BUDGET IN THOUSANDS, and press the Return key. Select Titles, Select X-axis ("Enter X-axis title:"), type \A3 (or type YEARS), and press the Return key. Select Titles, Select Y-axis ("Enter Y-axis title:"), type DOLLARS, and press the Return key. Select Quit, Select View. How does the Line Graph look to you now? Press the spacebar to return to Graph Menu.

Now, we will name this Line Graph. If you wish to use this graph later, it is necessary to give it a name! When this has been done, the next time that the worksheet is saved the graph will automatically be saved also! Select Name, Select Create ("Enter graph name:"), type LINEBUDG, and press the Return key.

If you wish to change the graph type, Select Type (allows the graph type to be changed), Select Bar. Select VIEW and look over your new graph. What did you find that the 1-2-3 program did for you? Press the spacebar to return to Graph Menu. Select Name, Select Create ("Enter graph name:"), type, BUDGBAR, and press the Return key. If you wish to print the Graph later, it must be saved on the data disk as a .PIC file with a unique filename. Select Save ("Enter graph filename:"), type, BARBUDG, and press the Return key. Select QUIT (exit the Graph Manu). This will save a Picture File with the extension .PIC automatically added.

How do you "SPREADSHEET Corner" reader's feel about trying something NEW! I know that some of the readers most likely know MACROS. We are just going to scratch the surface about MACROS in this project, but I want to stimulate your interest in MACROS so that you will use the HELP SCREENS and the 1-2-3 Manual to study them, as well as using them with this project. The 1-2-3 Manual is a documentation that has been better prepared than most manuals, but it falls far short of covering MACROS! At least that is my impression. I am going to add a few special little twists that I feel make MACROS more useful. Also, I am going to add the MACROS directly below the worksheet in this project for ease of printing the worksheets. Normally, to make sure that the MACROS do not overlap into the active worksheet, I start them way over in column AA or further. There are no firm rules on their location.

We will use the same worksheet as above. We will first create a keyboard MACRO that will SAVE the current worksheet automatically. If you look at what we have done so far on this project, you will notice that we have SAVED the worksheet many times and we will be SAVING them many more. That is one of the first reasons to use a MACRO! Use them where the same command strings are going to be used many times, so that they will save you time and reduce the chance for typing errors! I am going to use the letter S for this MACRO's name (it tells something that might make you think of SAVE). I am going to teach MACROS by creating and using them. That is the method we have been using in this series of "SPREADSHEET Corner" articles right along.

GOTO Cell A14, type '\S (Apostrophe Backslash S) and Return. This will be the name of this MACRO! Before I proceed, let me back up a little. The definition that I think of for a MACRO is TYPING ALTERNATIVE, because they are used to replace a series of 1-2-3 keystrokes or commands "tied" to a single ALT key, pressed down with the MACRO letter name (S) — for H/Z-100 users the Shifted CTRL key with the MACRO letter name.

Some people call them a keyboard MACRO. I visualize MACROS as a collection of keystrokes and/or commands. This is a little too simple description because as we use and become better

aquainted with MACROS, we will find that LOTUS 1-2-3 has a number of special commands that can only be used with MACROS! I will expand on the explanation of MACROS as we use them in future projects. MACRO functions can be complex but as we have found in "SPREADSHEET Corner" many times, if we break them down and take them step-by-step the tough problems all become easy! Do not let anyone tell you that they are too hard to learn or that you have to be a "programmer" to create and use MACROS!!! You DO NOT have to know everything about 1-2-3 before taking advantage of MACROS. They are too valuable a tool for reducing time and ERRORS to wait. Again, I want to restate that anytime you will be repeatedly using the same sets of keystrokes or commands with your worksheet, THINK about using a MACRO! Again, how many times have we saved our worksheet already for this article? Do you get tired of doing this very important, but not very interesting procedure? That is why I chose the SAVE CURRENT WORKSHEET as our first MACRO!!

First, you should choose a letter name for the MACRO that will give a little hint about what it will be doing, if possible. It MUST be a unique LETTER name. The MACRO name will consist of a \ (backward slash) followed by a letter that can be either Upper or Lower case. Thus, \A, \N, or \Y are valid names for examples. The name must be started with a label-prefix (apostrophe), if the name is going to look like a label to 1-2-3. We will find some names that will not require this apostrophe later.

Next, GOTO Cell B14, type '/FS~R~ (apostrophe, backslash [not a slash], File, Save, Tilde (Return key in MACROS), and Replace, (Tilde), Return key. Are these the keystrokes that you use to save the current worksheet? As you can see, we have supplied a series of keystrokes that would save the current worksheet. If you are not sure, try it. Does it fit our definition?

We must name the MACRO so that LOTUS 1-2-3 will recognize our work as a MACRO! GOTO Cell A14, type /RNLR (/ Range Name Labels Right), and press the Return key. This step MUST not be forgotten, otherwise our 1-2-3 software will not read these entries as a MACRO! It also tells the program that the name will be in the column to the right of the command portions. This is not a required location, but it is the way I like to have them. As we use MACROS, you may decide that you may like a different method. That is the nice thing about 1-2-3, you can do the same things more than one way, many times. We should include a short note that will explain what the MACRO will do. If we do not do this step, we could come back to this program later and not know what the MACRO will do. Also, some other person may use the project and not know the purpose of each MACRO. So, be sure to document each MACRO! This one is almost self-explaining, but not all MACROS will be. GOTO Cell D14, type SAVES CURRENT WORKSHEET, and press the Return key.

Now, execute the MACRO by holding down the ALT key and press the S key. The current worksheet (BUDGETGR) with current named Graph will be saved automatically. Compare your worksheet screen with my Figure 2. Press the F10 (Graph) key. This should now display the last graph that we created (the BAR Graph). Press the spacebar to return to the current worksheet.

With the next "SPREADSHEET Corner" article we will use this same worksheet and create a PIE and a STACKED BAR GRAPH. We will FORMAT the Graph Scales as we have done above. We will also create a MACRO that will display all four named graphs. I am going to start the MACRO for this below, so that the readers can enter and study it to see if you can determine how it will work. Of course, I will explain the MACRO in the next article.

1. GOTO Cell A16, type '\G, and press the Return key.
2. GOTO Cell B16, type '/GNULINE~Q, and press the Return key.
3. GOTO Cell B17, type '/GNUBAR~Q, and press the Return key.
4. GOTO Cell B18, type '/GNUPIE~Q, and press the Return key.
5. GOTO Cell B19, type '/GNUSTACK, and press the Return key.
6. GOTO Cell D16, type—DISPLAY LINE GRAPH—, and press Return.
7. GOTO Cell D17, type—DISPLAY BAR GRAPH—, and press Return.
8. GOTO Cell D18, type—DISPLAY PIE GRAPH—, and press Return.
9. GOTO Cell D19, type—DISPLAY STACKED GRAPH—,and press Return.

Now GOTO Cell A16, type /RNLR, and press Return key. That is the complete MACRO! Did I hear someone say, "How does LOTUS 1-2-3 know when to stop"? The answer is that a blank cell tells the program that it has reached the end of that MACRO. Therefore, NEVER HAVE a blank cell unless you want to tell the 1-2-3 program to stop the execution of the MACRO.

SPREADSHEET Corner COMPANY:--BUDGET IN THOUSAND 05-Sep-85

YEAR	ORDERS	SALES	BACKLOG
1980	$2,918	$1,320	$1,598
1981	$3,580	$2,619	$2,559
1982	$4,617	$5,277	$3,899
1983	$5,458	$7,118	$2,239
1984	$7,324	$9,483	$1,080
1985	$9,000	$10,000	$980

Figure 2

```
\S /FS~R~          SAVES CURRENT WORKSHEET
```

Figure 3

Hold down the ALT-key and press the S key at the same time. What did this do? It saved the current worksheet displayed on the screen! Compare your display with my Figure 3. Is it the same?

In the next article, we will Review printing the current worksheet and files with a Header line. We will then go over the procedure for the four GRAPHS. Before we do this, I would suggest that you study your LOTUS 1-2-3 Manual section about PrintGraph! You get there from the LOTUS ACCESS System Menu. Did you remember this?

For a closing review on the subject of MACROS for this article, I would like to dwell a little longer on a few important points while MACROS are fresh in your mind. I will show you a few of the more interesting items that will be coming. Remember, a MACRO is nothing more than a named Text Cell! Can you determine what this MACRO would do?

```
'/RFC0~~
```

Enter it into a blank worksheet and enter the keystrokes for help. Type an apostrophe (label-prefix), followed by the /(slash, not a

backslash), then press R (Range), press F (Format), press C (Currency), press 0 (zero), and press the Return key twice (tildes). We have used this series of keystrokes for the /Range Format Currency with zero decimal places. Why the two tildes? When you enter the keystrokes on the keyboard you would press the Return key after you have entered the zero and then use another Return key to tell 1-2-3 that this command applies to the current cell. Did you follow this? This is the tough programming that some people will tell you what is needed to do with 1-2-3 MACROS! Did you find this hard to do or follow?

For one more example, let's make a small change to demonstrate something for you to think about:

```
'/RFC0~.{END}{RIGHT}~
```

This MACRO looks a lot like the one we just analyzed, except that now the command will cause the cursor to move. Do you see this? Notice that we have only one tilde where we had two. The period would be the anchor that we have used in a few projects before. The phrase, {END}{RIGHT}, in this MACRO stands for press the END key followed by Press the Right Arrow key. The last tilde is the press the Return key. The phrase has the same effect in this MACRO as if these two keys would have if they were pressed in sequence from the keyboard. Use a blank worksheet and enter these keystrokes to help you follow them.

We use "symbols" like the above to represent FUNCTION KEYS. {EDIT} would be the F2 key and {GOTO} would be the F5 key on the Z-150-PC. Therefore, I am sure that the reader can see how powerful and useful MACROS can be! It is well worth the little extra effort to learn where and how to use them, but do not expect to learn them all at one time. Our future projects will make good use of MACROS and help you learn more about them.

The readers that are not using LOTUS 1-2-3 will not have the advantage of MACROS, but they can still do the projects using the keystrokes from the keyboard that the MACROS represent. This will, of course, accomplish the same worksheets using a little more typing. I have just touched the surface on the subject of MACROS, but I did want the readers to know what they are, what they can do, and why you would want to use them! The MACRO is leading us down the path where we will eventually have a USER MENU for our worksheets so that anyone, whether they know anything about spreadsheets or not, will be able to use our worksheets. Then, we have reached the MOST USEFUL PROJECTS! Don't you agree?

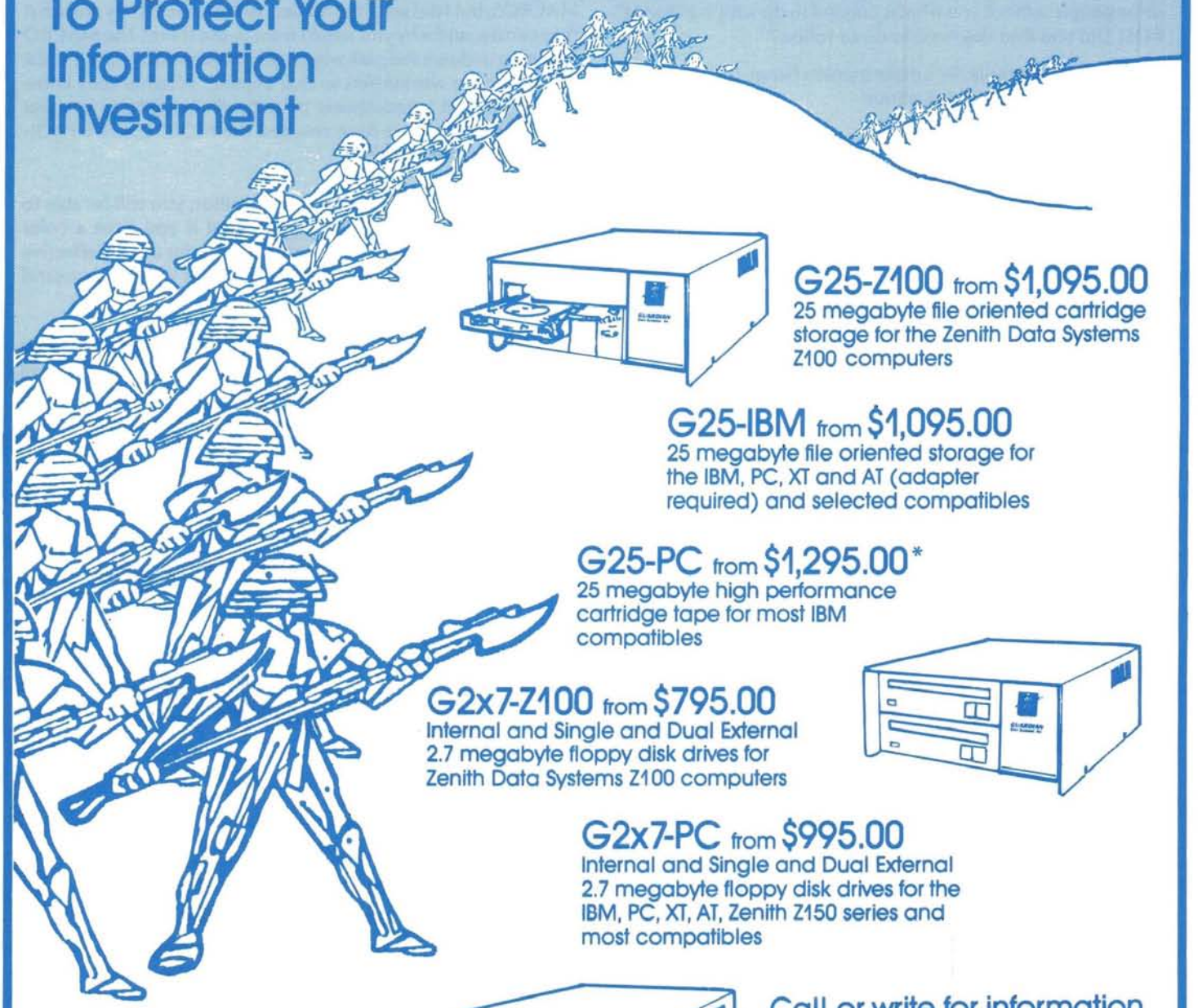
Also, in closing, if you have a color monitor, you will be able to display the graphs in "living color", and if you have a color printer, the graphs can be presented in color for a really effective presentation! What do they say? A picture is worth a thousand words, or is it 10,000 words?

Happy spreadsheeting! I hope that I have started some of the readers thinking about advanced LOTUS 1-2-3 (sometimes called the "hidden power" of LOTUS 1-2-3! *

HUG Catalog Update Coming Soon!

THE GUARDIANS ARE GATHERING

To Protect Your Information Investment



G25-Z100 from **$1,095.00**
25 megabyte file oriented cartridge storage for the Zenith Data Systems Z100 computers

G25-IBM from **$1,095.00**
25 megabyte file oriented storage for the IBM, PC, XT and AT (adapter required) and selected compatibles

G25-PC from **$1,295.00***
25 megabyte high performance cartridge tape for most IBM compatibles

G2x7-Z100 from **$795.00**
Internal and Single and Dual External 2.7 megabyte floppy disk drives for Zenith Data Systems Z100 computers

G2x7-PC from **$995.00**
Internal and Single and Dual External 2.7 megabyte floppy disk drives for the IBM, PC, XT, AT, Zenith Z150 series and most compatibles

10 to 40 Meg Hard Disk Upgrades for Zenith Data Systems Z150 and IBM Compatibles

*Available Soon

Call or write for information and the name of your nearest Guardian Data Systems Dealer.

GUARDIAN
Data Systems, Inc.

Distributed by
SYSTEMS INNOVATIONS INC.
44 Stedman Street
Lowell, Massachusetts 01851

HOURS: 9 AM-6 PM MON.-FRI.
THURS. TIL 8 PM • SAT. 10 AM-3 PM
(617) 459-4449

"Your Complete Computer Resource Center"

Professional Writing

Thoughts On Word Processing And A Professional Text Processor

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

What do you do with your computer? In the Bad Old Days, we lied to each other a lot, and talked about balancing checkbooks, keeping recipe files for our spouse, maintaining Christmas card mailing lists and telephone numbers. In seven years of working with Heath computers, you could count on the fingers of one hand the number of people I've met who really do these things. If you were to ask me, I'd say that the vast majority of personal computer users do three main things with their machines: they "tinker," they play games, and they write. I'm guilty of all three, and I've long since reached the point where I appreciate that the computer is the greatest toy in the world. However, that's another subject.

I use my Zenith 120 mainly for writing. Everything from personal letters to work-related papers gets done on my computer. I use pen and pencil only when absolutely necessary, and then only with great gnashing of teeth. I haven't touched a typewriter in years, and I hope never to have to again. I like to write, but I've become spoiled by my Z-100. I'd like to share some of the things I've learned about the programs that make your computer a high-powered typewriter, and in the process, show you some programs you might like to investigate.

If you have a computer, sooner or later you'll need a text-oriented program. The games are wonderful (ask my wife, she'll tell you about Y-Wing II and Vega-Bound!) and the spreadsheets, like SuperCalc and Lotus 1-2-3, are marvelous and essential to my business, but eventually the need will arise to write things and print them out. Your choice of software for this task will determine how pleased you are with the results. How pleased you are will largely determine how much use you make of the text capabilities of your computer, and will have a surprising effect on how well you write. Writing is hard enough without adding unnecessary obstacles, technical or otherwise.

I don't propose to evaluate all the text-oriented programs you could choose from. There must be literally dozens, if not hundreds. Some are minor improvements on older programs, often taken from other machines and altered just enough to run on your Heath. Others fall into the category of special-purpose editors, having features that tailor the program to a particular use. What I'd rather do is discuss a few programs that take advantage of our own particular computers and offer enough features to be widely useful.

Writing this article is simplified by the fact that I'm restricting my audience to users or potential users of Heath/Zenith computers. There are lots of other machines available, and some are quite good at one specific function or another. I believe that the Heath computers are the best general purpose computers you can get for your money. When I talk about Heath's computers, please keep in mind that I'm referring to the Heath/Zenith H/Z-89 and H/Z-100 series (the 110 and the 120, not the IBM look-alike.) I also consider the H-8 connected to an H/Z-19 or H/Z-29 terminal is the functional equivalent (or better) than the H-89. And when I use the term Heath (or H) I mean the assembled versions (the Z machines), as well as the kits. I just have an inbuilt bias for the kits. In fact, what I'm going to share includes a lot of biases and personal opinions. That can't be helped; software selection is a personal matter.

The H-89, Heath's follow-on design to the H-8 and the one that attracted Zenith's attention when they were looking for a company to acquire to get into the microcomputer market, was clearly aimed at both the technically proficient, as well as the applications-oriented user. I know of several professional writers who selected the H-89 and never so much as "opened the hood" to look inside. My wife's H-89 had performed flawlessly for over four years with zero maintenance when a bridge rec-

tifier failed. I think that's remarkable, given the mean time between failure of the "mainframe" computers I use at work. Also of note is the fact that I fixed it myself for less than $10.00, here at the "end of the pipeline" in Germany. Of course, the Z-100 is in a class by itself; I think it's the best general-purpose computer under $10,000, bar none. In other words, you already know that you have an excellent computer, but you may not know how good it is at handling text.

First, some terminology. I've referred to text-oriented programs to keep my initial discussion general in nature. Some folks call the programs we're going to look at text editors. Others call them word processors. There really is a difference. Let's see why.

Text editors generally refer to programs written to allow you to manipulate text at a fairly low level of complexity. Classic examples, still to be found in use, are HDOS' EDIT and CP/M's ED. These are line-oriented, meaning they "look" at one line of text at a time, and if you want to move on to another, you must tell the editor what line to go to. Not bad for programming in assembly language, they're very cumbersome when it comes to writing a letter.

An order of magnitude better than line-oriented text editors are the screen-oriented editors represented by one of the earliest available for the H-89 (and the H-19 terminals), called PIE. (PIE initially stood for "Programma's Improved Editor," written for the Apple II by Programma International; Walt Bilofsky of the Software Toolworks knew a good thing when he saw it, and recognized the potential of the basic idea for the H-19 terminal, soon to be followed by the H-89 computers.) PIE was not the first of the screen-oriented editors, but was the first for Heath computers. It allowed you to use the terminal's screen as a "window" on your text file, and let you swim about your file by using the cursor control keys to get you to the part you wanted. You then made the desired changes, and saw those changes immediately reflected on the screen. Walt followed the precept that "what you see is what you get," and what we got was wonderful. Initially provided for HDOS (beginning a tradition of providing excellent independent software support that inspired a great many others), PIE is now available for CP/M and ZDOS (and MS-DOS 2), as well.

Like all good ideas, it wasn't long until someone thought of a way to improve on PIE. One of the early improved screen-oriented editors came from Ron Rocheleau's Newline Software, and it was initially called Video Scribe. This program added a number of features that Walt left off. The differences in the programs were due more to philosophy than anything else. Walt's editor was small, and is intended to be used with a text formatting program like TEXT. Ron's Video Scribe provides some of the amenities like line centering, text fill and justification, and on-screen function key definitions that change according to their use. The drawback is that Ron's program is longer and somewhat slower to run. Like PIE, there are several versions for HDOS, CP/M, ZDOS and MS-DOS 2. Ron's Video Scribe, however, marks a transition to our next topic, word processors.

Word processors, unlike text editors, are intended to be more or less "full service" in the sense that most functions can be done from within the word processor itself, without resorting to text formatting programs. The intent is to allow the user to see on-screen the way the text will look when it's printed out, and handle things like resetting margins and formatting the text. One of the first programs of this sort available for the H-89 was AutoScribe, from Soft Words. This was one of the first programs issued under the Zenith label after they acquired Heathkit, and it was a most impressive word processor. The document could be created, edited, formatted and printed without leaving the program. It was fast, easy to teach and learn, and provided many office functions that could only be bought for $10,000 and up in 1979. Alas, it was something of a kluge, being made up of many smaller programs strung together not unlike some of today's "integrated" software. AutoScribe didn't fare too well, in part because the H-89 wasn't widely accepted in the business community at first, and in part because it wasn't well-advertized. There were problems of adapting AutoScribe from HDOS Version 1.5, under which it first appeared, to HDOS Version 2, the "finished" version that became the standard in-house disk operating system and reigned until the advent of CP/M 2.2. Despite these difficulties, and the fact that being an HDOS program made it device driver-dependent (you could only use printers supported by a piece of software called a device driver), at the time of it's appearance it was excellent.

Then WordStar came to Heath, and by now it must be available for almost every computer in the the Known Universe. While the various versions have been well-received (and had wonderful reviews!), they suffer because of the same universality that makes them popular. Written to run on almost anything with an Intel microprocessor (there's a rumor that there was once a version for the 8008), a great deal of customizing is required to obtain a comfortable "fit" to any given computer. Part of the problem lies in the generalized keystrokes required to get WordStar to do anything. It's often required to press up to four separate keys to perform an editing function, and computers equipped with special function keys can't make use of them without some kind of customizing. Sometimes the WordStar installation program allows this kind of customization, but sometimes not. Until the advent of Version 3.3, it was necessary to use programs that redefined the Z-100 special function keys in order to get the greatest benefit from WordStar. Nevertheless, WordStar has become a "de facto" standard of sorts. I've been told that it's among the most pirated programs in the history of microcomputing, suggesting that it's widely circulated. However, of all my computer-using acquaintances, I only know two people who actually use WordStar. That suggests that it may not be all that widely used, and has a higher "fascination factor" than a "use factor." Almost all my acquaintances use something faster and simpler. So on to the faster and simpler.

We've come to the word processing program I've been wanting to tell you about, one that's already fully customized for your particular computer. This type is the outgrowth of the special-purpose word processors best illustrated by the IBM Displaywriter and the Wangs, Laniers, CPTs and the like. These are, of course, nothing more than computers that have only a certain kind of software available, the word processing program. It's often built-in, so that it's there when you turn the machine on. Sometimes you have to load it from a disk, either a central hard disk associated with the CPU, or from a floppy disk at the work station or terminal. Regardless, the computer will only process words. Generally there is no programming, numerical analysis or game-playing capability available. Often more efficient at their intended function, these systems are inherently limited, tend to be quite expensive, and usually found only in large offices where concern over duplication of equipment costs may be less. Secretaries love them, but it's been my experience that secretaries (or others who do a lot of typing) love any text processing program that works well, does not require three weeks to learn, and relieves them of the tedium of countless retyping of drafts.

We Heath users are blessed with a variety of specially configured word processing programs. The best I've found so far for my H-100 is Newline Software's "Professional Text Processor," also called PTP. It's so good that I think it's worth a close look. I'll compare it to Newline's older "TxtPro" editor, available for the Z-100 under both CP/M-85 (very fast and efficient) and ZDOS or MS-DOS 2 (not so fast.)

First of all, the documentation for all three versions is clear, with enough examples to help you master the program's features. There's an on-line help function available for all versions, obtained by pressing the HELP key. It's only three screens long for PTP instead of eight for TxtPro and Video Scribe, and they cover only the basics, but I've found that for each program the amount of on-line help is generally adequate. The paging is fast enough to help rather than hinder. If what you need isn't there, especially for PTP, it usually means that the help you need is on one of the special Setup Keys or the Option Key menus.

TxtPro labels the special function keys on the 25th line, a very nice feature that makes learning easier. PTP does that differently, providing a full screen of definitions whenever you select the Options Key or one of the Setup Keys. For instance, to configure the program for direct output to the printer (meaning that you don't have to save a file to disk before you can print it), you just press the keypad 8 key. A full screen of options appears, and you select the one you want by pressing a single keyboard key corresponding to the first letter of the option. This is actually a lot simpler and faster than it sounds, and doesn't require you to memorize any escape codes at all. If you want to underline some text, or print it in boldface, or change tabs or reformat a paragraph, you press the Options Key, 0 on the keypad, and either a one-line menu or a full screen of options is presented, depending on what you need. As a result, there's rarely any need to refer to the documentation.

If you find the switching back and forth between text and help screens bothersome, you can take advantage of an "expert mode," which allows you to do the same things by just typing an escape sequence of two keys. For example, to enter the underline mode, just press CTRL/SHIFT U. All the other optional features have a corresponding sequence, usually a CTRL/SHIFT and the first letter of what you want to do, like B for bold print or W for word wrap.

The most impressive feature of PTP is the screen display. Most readers know that the Z-100 is extremely fast, and PTP takes full advantage of the speed capabilities. While the ZDOS version of TxtPro is disappointingly slow on screen scrolling, especially when in the text insert mode, PTP is almost instantaneous. Press the keypad 1 key and you scroll one page (22 lines) immediately, a page at a time, not a line at a time. Press keypad 2 and you go back a full page. The ENTER key allows you to ask for multiple repetitions of any command, such as the page. Press the HOME key to get to the top of the file, and SHIFT/HOME to get to the bottom. If your file won't fit in memory, you can work with part of it on disk, as in TxtPro.

All the standard word processing functions are available for all three programs, but they work best with PTP. Cut and paste, inserting and deleting text, and reformatting paragraphs are fast. You can work with text files larger than memory, and determine if you want to create backup files or not. You can replace disks during the editing session, which gives virtually unlimited file access. Finally, you can customize the program with respect to margins, the insert mode and so on.

If you get the impression that I recommend PTP, you're right. I started with Video Scribe, and bought Newline's combination pack of TxtPro for CP/M-85 and ZDOS, and finally moved up to PTP. The advantage of common commands among all the versions has been a tremendous time-saver, especially when I switch from the 8-bit side of the Z-120 to the 16-bit side. My wife also appreciates the commonality between the H-89 and the Z-120 programs. You can spend a great deal more, but I don't believe you can do better for your money.

Professional Text Processor
Versions for CP/M-86, Concurrent CP/M, ZDOS & MS-DOS
$100.00 ($150.00 for the Z151/161 or IBM PC)

TxtPro
Versions for HDOS, CP/M-80, CP/M-85 and ZDOS $59.95 ea.
ZDOS & CP/M-85 versions together, $99.95

Newline Software
P. O. Box 289
Tiverton, RI 02878

*

HUG Members only!

HUGE HUGGER DISCOUNTS now available on these selected products...exclusively for members only!

That's right. We're offering HUG members these close-out, last call and limited quantity specials from Heath Company.

The items listed are not necessarily representative of complete systems. Please call your nearest Heath/Zenith Computers & Electronics Center or Heath Company toll-free at 1-800-253-7057 for more information before ordering.

Watch for additional MEMBER ONLY discounts in upcoming issues of REMark®!

REMark Discount List

	ITEM	LAST ADV'D PRICE	HUG MEMBER PRICE
H-19-3	H-19 to H-88 Mod Kit	220.00	199.95
H-88-1	Floppy Disk Accessory	150.00	99.95
H-88-5	Cassette I/O Card	25.00	24.95
H-88-9	Drive Mounting & Inst.	25.00	24.95
H-125	Dot Matrix Printer	750.00	499.95
HCA-9	Print Stand For H-125	99.00	89.95
PC-140	DG Magic RAM Expn. OK	129.00	99.95
PC-300	MMS CP/M+ & 128K Mem	395.00	395.00
PC-310	DG Super '89 CPU	699.00	499.95
PCA-120-1	6-Pak Plus Game Acc'y	39.95	34.95
PP-110	Trippe Emer Power System	349.00	299.95
PP-260	DTC 380Z Daisy Printer	1295.00	899.95
PPA-260-2	Cut-Sheet Feeder /380Z	574.95	399.95
PPA-260-3	Ribbons (3PK) /380Z	9.95	5.99
PPA-260-4	Cable - RS232 /380Z	49.95	34.95
PPA-260-5	Cable - Parallel /380Z	49.95	34.95
PPA-270-1	Trac Feed - Stylewriter	149.95	99.95
PPA-270-2	Sheet Feed - Stylewriter	249.95	174.95
ZA-100-4	S-100 Extender Board	59.00	29.95
ZA-219	Amber CRT /H-121	75.00	19.95
ZG-219	Green CRT /H-121	75.00	19.95
ZW-219	White CRT /H-121	75.00	14.95

Heathkit®
Heath Company

A subsidiary of Zenith Electronics Corporation.
Some items may not be available at all store locations.
Prices, product availability and specifications subject to change without notice

CP-236

"Where's That File"

MSDOS Disk Catalog Procedure

Paul F. Fuhrmeister
271 Princess Place
Williamsburg, VA 23185

Introduction

Providing all MSDOS 5" disks have a volume number recorded at Format time, this procedure provides for the cataloging of these disks in a manner that the FIND utility can be used to find a file or group of files. Or FIND can be used to find, for example, all the files that were written on a particular date, or all files with a specified extension, etc.

Detail Procedure (192K Of Memory Is Required)

1. Prepare a bootable program disk that includes the following files:

   Standard MSDOS programs

   ZBASIC.COM
   FIND.EXE
   SORT.EXE
   MDISK.DVD

   Specific programs for this procedure: (complete listings at end of article)

   CATWORK.BAT
   CATMISC.BAT
   CAT.BAT
   SCH.BAT

   DSKCAP.BAS
   CAT.BAS

   CONFIG.SYS that will install MDISK (example follows)

   BREAK= ON
   SWITCHAR= /
   AVAILDEV= TRUE
   BUFFERS= 10
   DEVICE= MDISK.DVD

   The use of MDISK greatly speeds up the cataloging process.

2. Reboot the computer to install MDISK.

   The program files have been prepared for running the program from disk B: and inserting the disks to be cataloged into disk A:. This can be changed by proper alteration of CATWORK.BAT and CATMISC.BAT.

3. DELETE THE ASSOCIATED .RAW FILE for the disk group that is to be updated.

   *** IF THIS IS NOT DONE, THE NEW CATALOG INFORMATION WILL BE APPENDED TO THE OLD FILE ***

4. Run any or all of the CATxxxx .BAT programs.

   Use The F3 key for each additional disk on each .BAT program.

5. Run CAT.BAT

6. Search for files as desired.

   SCH <string>

   Any string may be used as <string>
   Examples:

file name	COMMAND.COM
extension	.COM
date	2-02
disk	MDOS 8010

   Note: SCH.BAT uses the MSDOS FIND utility that searches for a string. If a file such as "C.COM" is specified, the FIND utility will find all of the following:

   C.COM
   PSC.COM
   ZBASIC.COM

7. Any editor can be used to examine any file in detail.

   Note: It is possible to update the .RAW data file for only those disks that have been changed. This is done by using an editor

to CAREFULLY DELETE THE DIRECTORY INFORMATION FOR ONLY THOSE DISKS TO BE UPDATED, then running step 4 only on the update disks, and then running CAT.BAT to create the updated search files.

File And Program Description

I have set up the program to allow for grouping the disks to be cataloged into two different groups. This was done to eliminate the need to insert all the disks whenever an update is desired. By seperating those disks that are infrequently changed into one group, it then is only necessary to insert those disks whithin the group that has changed in the update process. All disks are still searched.

The procedure could be changed to keep all disks in a single group, or additional groups could be maintained. The program files would have to be modified if this is desired.

Data Files

The extensions are as follows:
.RAW for the raw data directories
.SCH for the search files

Data Groups

CATWORK.xxx: For work disks
CATMISC.xxx: For the other disks. Disk groups can be added or deleted by changing the variables in the two ZBASIC programs. A new batch file is required for creating the .RAW file for each group that is added.

Program Files

ZBASIC Files

ZBASIC.COM: The ZBASIC program
DSKCAP.BAS: Prints out the number of files and free space for all disks.
CAT.BAS: Converts the .RAW files to .SCH files.

MSDOS Files

FIND.EXE: MSDOS utility
SORT.EXE: MSDOS utility

BATCH Files

CATWORK.BAT: For writing the .RAW file of work group disks.
CATMISC.BAT: For writing the .RAW file of misc. group disks.
CAT.BAT: Converts all .RAW data files to .SCH files and sorts the .SCH files by file name. Also prints out the number of files and free space on each disk and the date and time that the .RAW files were created.
SCH.BAT: Searches for a string on all disks.

Batch File Detail

CATWORK.BAT

```
DIR A:>I:TEMP.FIL
TYPE I:TEMP.FIL>>CATWORK.RAW
DEL I:TEMP.FIL
```

CATMISC.BAT

```
DIR A:>I:TEMP.FIL
TYPE I:TEMP.FIL>>CATMISC.RAW
DEL I:TEMP.FIL
```

CAT.BAT

```
ZBASIC DSKCAP
ZBASIC CAT
SORT <CATWORK.SCH >CATWORK.SRT
COPY CATWORK.SRT CATWORK.SCH
DEL CATWORK.SRT
SORT <CATMISC.SCH >CATMISC.SRT
COPY CATMISC.SRT CATMISC.SCH
DEL CATMISC.SRT
DIR *.RAW | SORT >PRN
```

SCH.BAT

```
FIND "%1" CATWORK.SCH CATMISC.SCH
```

ZBASIC File Detail

```
10 ' DSKCAP.BAS  LIST NUMBER OF FILES AND FREE SPACE
                 ON 5" DISKS
12 '             BASED ON CURRENT CONTENTS OF  RAW FILES
14 '    2-19-85
16 '
18 CAT$(1)="CATWORK"
20 CAT$(2)="CATMISC"
22 ' Add additional variables for other groups if desired
24 ' and change line 100 to reflect the total
26 '
90 LPRINT TAB(70);DATE$
100 FOR I=1 TO 2
110 PRINT TAB(60);CAT$(I):LPRINT TAB(60);CAT$(I)
120 OPEN "I",#1,CAT$(I)+".RAW"
130 IF EOF(1) THEN 250
140 INPUT #1,A$
150 IF LEFT$(A$,6)="Volume" THEN 200
160 IF LEFT$(A$,6)="Direct" THEN 190
170 A=INSTR(A$,"File(s)")
180 IF A<>0 THEN 220
190 GOTO 130
200 DSK$=MID$(A$,22)
210 GOTO 130
220 LPRINT DSK$,A$
230 PRINT DSK$,A$
240 GOTO 130
250 CLOSE #1
260 NEXT I
270 SYSTEM
```

CAT.BAS

This ZBASIC program converts all .RAW data files to .SCH files. The blank, Volume, Directory, and Files lines are deleted and the Volume number is appended to each record. In addition, the file name format is changed from that with spaces between name and extension to name.ext with no spaces. This is done to aid the search and sorting processes.

```
10 '    CAT.BAS          WRITE SEARCH FILES FOR 5" DISKS
12 '    2-16-85
14 '
16 CAT$(1)="CATWORK"
18 CAT$(2)="CATMISC"
20 ' Add additional variables for other groups, if
22 ' desired and change line 100 to reflect the total
23 '
100 FOR I=1 TO 2
110 OPEN "I",#1,CAT$(I)+".RAW"
120 OPEN "O",#3,CAT$(I)+".SCH"
130 PRINT #3,'
140 IF EOF(1) THEN 330
150 INPUT #1,A$
160 IF A$="" THEN 140
170 IF LEFT$(A$,6)="Volume" THEN 300
180 IF LEFT$(A$,6)="Direct" THEN 140
190 A=INSTR(A$,"File(s)")
200 IF A<>0 THEN 140
210 NME$=LEFT$(A$,8)
220 EXT$=MID$(A$,10,3)
230 BYTES$=MID$(A$,14,9)
240 DAT$=MID$(A$,23,10)
250 TIM$=MID$(A$,33,8)
260 B=INSTR(NME$," ")
270 IF B=0 THEN NM$=LEFT$(NME$,8)
```

```
    ELSE NM$=LEFT$(NME$,B-1)
280 PRINT #3,NM$;".";EXT$;TAB(20);BYTES$;"  ";DAT$;
    "  ";TIM$;"  ";DSK$
290 GOTO 140
300 DSK$=MID$(A$,22)
310 PRINT DSK$
320 GOTO 140
330 CLOSE #1,#3
340 NEXT I
350 SYSTEM
```

Conclusion

This procedure provides a simple procedure for cataloging 5" disks that is easily implemented. Searching the resulting directory files is fast and quite useful.

The procedure has been described for 5" MSDOS disks, but it could be easily modified to include 8" disks and/or path names for hard disks.

In addition, the procedure demonstrates the power of MSDOS I/O redirection, pipes and filters.

```
10 '    CAT.BAS          WRITE SEARCH FILES FOR 5" DISKS
12 '    2-16-85
14 '
16 CAT$(1)="CATWORK"
18 CAT$(2)="CATMISC"
20 ' Add additional variables for other groups, if
22 ' desired and change line 100 to reflect the total
23 '
100 FOR I=1 TO 2
110 OPEN "I",#1,CAT$(I)+".RAW"
120 OPEN "O",#3,CAT$(I)+".SCH"
130 PRINT #3,' I don"t know why this is needed,
              but it is!!!
140 IF EOF(1) THEN 330
150 INPUT #1,A$
160 IF A$="" THEN 140
170 IF LEFT$(A$,6)="Volume" THEN 300
180 IF LEFT$(A$,6)="Direct" THEN 140
190 A=INSTR(A$,"File(s)")
200 IF A<>0 THEN 140
210 NME$=LEFT$(A$,8)
220 EXT$=MID$(A$,10,3)
230 BYTES$=MID$(A$,14,9)
240 DAT$=MID$(A$,23,10)
250 TIM$=MID$(A$,33,8)
260 B=INSTR(NME$," ")
270 IF B=0 THEN NM$=LEFT$(NME$,8)
    ELSE NM$=LEFT$(NME$,B-1)
280 PRINT #3,NM$;".";EXT$;TAB(20);BYTES$;"  ";DAT$;
    "  ";TIM$;"  ";DSK$
290 GOTO 140
300 DSK$=MID$(A$,22)
310 PRINT DSK$
320 GOTO 140
330 CLOSE #1,#3
340 NEXT I
350 SYSTEM
```

```
ZBASIC DSKCAP
ZBASIC CAT
SORT <CATWORK.SCH >CATWORK.SRT
COPY CATWORK.SRT CATWORK.SCH
DEL CATWORK.SRT
SORT <CATMISC.SCH >CATMISC.SRT
COPY CATMISC.SRT CATMISC.SCH
DEL CATMISC.SRT
DIR *.RAW | SORT >PRN
```

```
DIR A:>I:TEMP.FIL
TYPE I:TEMP.FIL>>CATMISC.RAW
DEL I:TEMP.FIL
DIR A:>I:TEMP.FIL
TYPE I:TEMP.FIL>>CATWORK.RAW
DEL I:TEMP.FIL


FIND "%1" CATWORK.SCH CATMISC.SCH
```

HUG NEW PRODUCTS

SPELL5	.COM	SPELL5F	.COM
DICTION	.SPL	DICTIONF	.SPL
SPELL5	.DOC	SPELL5F	.DOC
SPELL5	.128	FRENCH2	.CHR
README	.DOC		

HUG P/N 885-3035-37 MSDOS
SPELL5 & SPELL5F $20.00

Program Author: Ronald Perrella

Program Content: The SPELL5 program is a memory based spelling checker. It loads a dictionary called DICTION.SPL from the disk and checks a file for spelling errors. When an unknown word is encountered, it can either be skipped or integrated to the dictionary. The dictionary is a sorted list of words separated by a carriage return and linefeed. This dictionary can be edited by any text editor, if necessary. Multiple dictionaries are supported and so is IN CONTEXT spelling checking.

There is also a French version of this program, and in order to be used properly, the ALTCHAR.SYS file must be changed to the FRENCH2.CHR file.

The SPELL5 program was designed to be easy to use, not to be a 50,000 word speller. Its true capacity depends on the amount of memory you have. In a 192k byte system, you should be able to store about 10,000 words, and about half that many in a 128k byte system.

TABLE C Rating: (10)

Introduction: SPELL5 and SPELL5F are two spelling checkers designed to make spelling checking easier and truly useful. SPELL5 is a spelling checker for the English language, and SPELL5F is a spelling checker for French.

Requirements: Both programs require the H/Z-100 computer system (not PC) with 192k of memory. Included, are versions that will also work with 128k systems. Both programs will work with either monochrome or color CRTs. SPELL5 will work with any version of MSDOS.

The following programs or files are included on the HUG SPELL5 disk P/N 885-3035-37:

HUG P/N 885-8038-37 MSDOS
RF-CAD Ver. 3.50 ..... $30.00

Introduction: RFCAD is an integrated collection of programs written in BASIC which will aid the user in designing many types

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.

of electronic circuits and antennas. This package should be invaluable to radio amateurs, electronic engineers, or anyone who has occasion to design circuitry or antennas for RF or Microwave frequencies. It is particularly useful to persons interested in satellite earth station design and construction, and also those who wish to purchase a ready made earth station. RFCAD is also useful as an instruction aid in teaching electronic theory. The user is free to experiment with various configurations and values without having to build the circuit to determine the results.

Requirements: RFCAD requires an H/Z-100 PC/200 PC system with a minimum (128k) amount of memory. A single 5-1/4" disk drive is required, and an 80 column printer is optional. Any version of MSDOS and Microsoft's BASIC is also needed.

In addition to the printed users' manual that comes with this product, the following programs or files are included on this HUG RFCAD disk P/N 885-8038-37:

AELIPT	.BAS	AFILTER	.BAS
AFPLOT	.BAS	BFILTER	.BAS
CADMENU	.BAS	CHORN	.BAS
COIL	.BAS	CONFIG	.BAS
DBCONV	.BAS	DISH	.BAS
EME	.BAS	EQUALZ	.BAS
FED	.BAS	FREQWAV	.BAS
HELIX	.BAS	HELP	.BAS
HORN	.BAS	LHFILTER	.BAS
LPI	.BAS	MATCHER	.BAS
MFILTER	.BAS	MOD	.BAS
MSCONV	.BAS	MSTRIP	.BAS
NE555	.BAS	NFCONV	.BAS
OSC	.BAS	PATH	.BAS
PELIPT	.BAS	PFILTER	.BAS
README	.DOC	RESNET	.BAS
RESONNCE	.BAS	SATANT	.BAS
SPUR	.BAS	TLINE	.BAS
WNDLD	.BAS	YAGI	.BAS
ZCONV	.BAS		

Program Author: Gary A. Field

Program Content: Ease of use is one of RFCAD's primary features. Wherever possible, the user is presented with menus of choices and examples of input data being requested. Since full source code is provided, the user may if he wishes, modify any of the programs to suit his own needs.

The designs covered by this package have been selected based mainly on needs of the author. Whenever the need to design a circuit involving extensive calculation arose, a program was written to perform the necessary task. After a large number of functions were available, a menu driver was created to make access to all the functions simpler and more uniform. The entire set of programs was written in the simplest form of BASIC with particular attention to readability and maintainability, so that the user can modify it for his own use.

The information required to perform the calculations was obtained from a great many sources; magazine articles, textbooks, papers and verbal information from friends (Joe Reisert - W1JR and Peter Reilley - KA1LAT). The particular source is listed in the program listing. Any formulas or algorithms not credited in this manner were derived by the author or taken from his own memory.

Upon execution of the program, the following menu of design aids is available:

(1) Low Frequency circuit design
(2) High Frequency circuit design
(3) UHF and Microwave circuit design
(4) Antenna and Transmission line design
(5) Filter design
(6) Space Communications Aids
(7) Miscellaneous conversions
(8) Strays
(9) Help
(10) Return to O/S

Included with the documentation are schematics of various design aids, which are referenced by the actual program itself.

TABLE C Rating: (10)

HUG P/N 885-6008-37 MSDOS
NAVPROG $20.00

Introduction: NAVPROGseven is a database management system designed for pilots flying cross-country. The system is built around a latitude/longitude referenced navigation program designed to prepare a flight log that is ready for use in the cockpit. It is equally adaptable to the needs of VFR and IFR flight.

The system stores performance data for each aircraft the user flies, navigation data about each checkpoint, airport, or navaid the user flies over, and saves this information for easy access on subsequent flights.

Requirements: This version of NAVPROGseven is designed to run on the MSDOS operating system version 2.0 or greater on the H/Z-100 PC with at least 64k of memory. NAVPROGseven requires a dual-disk drive system and a line printer. The programs are written under and requires the Microsoft GW-BASIC version 1.0+.

The following files are included on the distribution disk 885-6008-37:

NAVPROG7	.BAS	DISKAID	.DAT
DISKBID	.DAT	MENU	.BAS
AIRINPUT	.BAS	AIRCRAFT	.BAS
AIRROUTE	.BAS	OLDROUTE	.BAS
RNAVREF	.BAS	AIRALPHA	.BAS
AUTONAV	.BAS	N73116	
N81259		AIRINDEX	.RND
AIRPORTS	.RND	ROUTINGS	.DAT
RNAVLIST	.DAT	BEH	.LUK
LUK	.BEH	README	.DOC

Program Authors: Originally written by Alan Bose for use on the H-89 under HDOS. An update for CP/M was prepared by Glen Hassebrock Jr. This version was adapted for the PC by Peter Ambrose.

Files: The main program content will be described below. The following is a brief description of the data files used by NAVPROGseven:

N73116 and N81259 — These files are sample aircraft data, which may be deleted manually or by the program after the user has experimented with NAVPROGseven.

AIRINDEX.RND and AIRPORTS.RND — These data files contain over 100 checkpoints already on file. These files can be revised

Continued on Page 63

HUG Price List

The following HUG Price List contains a list of all products 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	Issue
HDOS HARDCOPY SOFTWARE			
885-1008	Volume I Documentation	9.00	
885-1013	Volume II Documentation	12.00	
885-1015	Volume III Documentation	9.00	
885-1037	Volume IV Documentation	12.00	8
885-1058	Volume V Documentation	12.00	
MISCELLANEOUS HDOS COLLECTIONS			
885-1032	Disk V H8/H9	18.00	8
885-1044-[37]	Disk VI H8/H89	18.00	
885-1064-[37]	Disk IX H8/H89 Disk	18.00	
885-1066-[37]	Disk X H8/H89	18.00	10
885-1069	Disk XIII Misc H8/H89	18.00	
GAMES			
HDOS			
885-1010	Adventure Disk H8/H89	10.00	4
885-1029-[37]	Disk II Games 1 H8/H89	18.00	8
885-1030-[37]	Disk III Games 2 H8/H89	18.00	8
885-1031	Disk IV MUSIC H8 Only	20.00	25
885-1067-[37]	Disk XI H8/H19/H89 Games	18.00	12
885-1068	Disk XII MBASIC Graphic Games	18.00	10
885-1088-[37]	Disk XVII MBASIC Graph. Games	20.00	14
885-1093-[37]	D&D H8/H89 Disk	20.00	16
885-1096-[37]	MBASIC Action Games H8/H89	20.00	18
885-1103	Sea Battle HDOS H19/H8/H89	20.00	20
885-1111-[37]	HDOS MBASIC Games H8/H89	20.00	23
885-1112-[37]	HDOS Graphic Games H8/H89	20.00	23
885-1113-[37]	HDOS Action Games H8/H89	20.00	23
885-1114	H8 Color Raiders & Goop	20.00	23
885-1124	HUGMAN & Movie Animation Pkg	20.00	41
885-1125	MAZEMADNESS	20.00	41
885-1130	Star Battle	20.00	47
885-1133-[37]	HDOS Games Collection I	20.00	59
885-8009-[37]	HDOS & CP/M Galactic Warrior	20.00	32
885-8022	HDOS SHAPES	16.00	45
885-8026	HDOS Space Drop	16.00	49
885-8032-[37]	HDOS Castle	20.00	59
CP/M			
885-1206-[37]	CP/M Games Disk	20.00	11
885-1209-[37]	CP/M MBASIC D&D	20.00	19
885-1211-[37]	CP/M Sea Battle	20.00	20
885-1220-[37]	CP/M Action Games	20.00	32
885-1222-[37]	CP/M Adventure	10.00	35
885-1227-[37]	CP/M Casino Games	20.00	38
885-1228-[37]	CP/M Fast Action Games	20.00	39
885-1236-[37]	CP/M Fun Disk I	20.00	55
885-1248-[37]	CP/M Fun Disk II	35.00	69
ZDOS			
885-3004-37	ZDOS ZBASIC Graphic Games	20.00	37
885-3009-37	ZDOS ZBASIC D&D	20.00	50
885-3011-37	ZDOS ZBASIC Games Disk	20.00	52
885-3017-37	ZDOS Contest Games Disk	25.00	58
UTILITIES			
HDOS			
885-1022-[37]	HUG Editor (ED) Disk H8/H89	20.00	20
885-1025	Runoff Disk H8/H89	35.00	
885-1060-[37]	Disk VII H8/H89	18.00	
885-1061	TMI Load H8 ONLY Disk	18.00	
885-1062-[37]	Disk VIII H8/H89 (2 Disks)	25.00	
885-1063	Floating Point Disk H8/H89	18.00	
885-1065	Fix Point Package H8/H89 Disk	18.00	10
885-1075	HDOS Support Package H8/H89	60.00	
885-1077	TXTCON/BASCON H8/H89	18.00	
885-1079-[37]	HDOS Page Editor	25.00	15
885-1080	EDITX H8/H19/H89 Disk	20.00	
885-1082	Programs for Printers H8/H89	20.00	
885-1083-[37]	Disk XVI Misc H8/H89	20.00	11
885-1089-[37]	Disk XVIII Misc H8/H89	20.00	20
885-1090-[37]	Disk XIX Utilities H8/H89	20.00	22
885-1092-[37]	Relocating Debug Tool H8/H89	30.00	14
885-1098	H8 Color Graphics ASM	20.00	19
885-1099	H8 Color Graphics Tiny PASCAL	20.00	19
885-1105	HDOS Device Drivers H8/H89	20.00	24
885-1116	HDOS Z80 Debugging Tool	20.00	27
885-1119-[37]	BHBASIC Support	20.00	29
885-1120-[37]	HDOS 'WHEW' Utilities	20.00	33
885-1121	HDOS Hard Sec Sup Pkg 2 Disks	30.00	37
885-1123	XMET Robot Cross Assembler	20.00	40
885-1126	HDOS Utilities by PS:	20.00	42
885-1127-[37]	HDOS Soft Sector Support Pkg	30.00	45
885-1128-[37]	HDOS DISKVIEW	16.00	46
885-1129-[37]	HDOS CVT Color Video Terminal	20.00	46
885-8001	SE (Screen Editor)	25.00	28
885-8003	BHTOMB	25.00	28
885-8004	UDUMP	35.00	28
885-8006	HDOS SUBMIT	20.00	31
885-8007	EZITRANS	30.00	30
885-8015	HDOS TEXTSET Formatter	30.00	42
885-8017	HDOS Programmers Helper	16.00	42
885-8024	HDOS BHBASIC Utilities Disk	16.00	46
CP/M			
885-1210-[37]	CP/M ED (same as 885-1022)	20.00	20
885-1212-[37]	CP/M Utilities H8/H89	20.00	21
885-1213-[37]	CP/M Disk Utilities H8/H89	20.00	22
885-1217-[37]	HUG Disk Duplication Utilities	20.00	26
885-1223-[37]	HRUN HDOS Emulator 3 Disks	40.00	37
885-1225-[37]	CP/M Disk Dump & Edit Utility	30.00	40
885-1226-[37]	CP/M Utilities by PS:	20.00	40
885-1229-[37]	XMET Robot Cross Assembler	20.00	40
885-1230-[37]	CP/M Function Key Mapper	20.00	42
885-1231-[37]	Cross Ref Utilities for MBASIC	20.00	43
885-1232-[37]	CP/M Color Video Terminal	20.00	46
885-1235-37	CP/M COPYDOS	20.00	54
885-1237-[37]	CP/M Utilities	20.00	55
885-1245-37	CP/M-85 KEYMAP	20.00	63
885-1246[-37]	CP/M HUGFile Manager & Utilities	20.00	64
885-1247-37	CP/M-85 HUG Bkgrd Print Spooler	20.00	67
885-5001-37	CP/M-86 KEYMAP	20.00	51
885-5002-37	CP/M-86 HUG Editor	20.00	52
885-5003-37	CP/M-86 Utilities by PS:	20.00	54
885-5008-37	CP/M 8080 To 8088 Trans. & HFM	20.00	64
885-5009-37	CP/M-86 HUG Bkgrd Print Spool	20.00	66
885-8018-[37]	CP/M Fast Eddy & Big Eddy	20.00	43
885-8019-[37]	DOCUMAT and DOCULIST	20.00	43
885-8025-37	CP/M-85/86 Fast Eddy	20.00	49
ZDOS			
885-3005-37	ZDOS Etchdump	20.00	39
885-3007-37	ZDOS CP/EMulator	20.00	47
885-3008-37	ZDOS Utilities	20.00	47
885-3010-37	ZDOS Keymap	20.00	51
885-3022-37	ZDOS/MSDOS Useful Programs I	30.00	63
885-3023-37	ZDOS/MSDOS EZPLOT	20.00	63
885-3026-37	MSDOS SMALL C Compiler	25.00	65
885-3030-37	ZDOS/MSDOS Z-100 PC Emulator	40.00	68
885-3031-37	ZDOS/MSDOS Graphics	20.00	69
885-3034-37	ZDOS/MSDOS ZPC Support Pkg	10.00	72
885-8029-37	ZDOS Fast Eddy	20.00	53
885-8035-37	MSDOS DOCUMAT and DOCULIST	20.00	70
H/Z100 ZDOS/MSDOS - H/Z150 PC MSDOS			
885-3012-37§§	ZDOS HUG Editor	20.00	52
885-3014-37§§	ZDOS/MSDOS Utilities II	20.00	54
885-3016-37§	ZDOS/MSDOS Adventure	10.00	57
885-3020-37§	MSDOS HUG Menu System	20.00	62
885-3021-37§§	ZDOS/MSDOS Cardcat	20.00	63
885-3024-37§	ZDOS/MSDOS 8080 To 8088 Trans	20.00	64
885-3025-37§§	ZDOS/MSDOS Misc Utilities	20.00	64
885-3029-37§§	ZDOS/MSDOS HUG Bg. Print Spool	20.00	66
885-3032-37§	MSDOS Halley's Comet Locater	20.00	70

§ All program files run on both

§§ Program files run partially on both

Part Number	Description of Product	Selling Price	Issue
PC/IBM COMPATIBLE			
885-6001-37	MSDOS Keymapper	20.00	59
885-6002-37	CP/EMulator II & ZEMulator	20.00	59
885-6003-37	MSDOS EZPLOT	20.00	65
885-6004-37	MSDOS CheapCalc	20.00	67
885-6005-37	MSDOS Skyviews	20.00	67
885-6006-37	MSDOS Cardcat	20.00	69
885-6007-37	MSDOS DND (Dung. & Dragons)	20.00	70
885-8033-37	MSDOS Fast Edit	20.00	62
885-8037-37	MSDOS Grade	20.00	70
PROGRAMMING LANGUAGES			
HDOS			
885-1038-[37]	Wise on Disk H8/H89	18.00	
885-1042-[37]	PILOT on Disk H8/H89	19.00	
885-1059	FOCAL-8 H8/H89 Disk	25.00	13
885-1078-[37]	HDOS Z80 Assembler	25.00	21
885-1085	PILOT Documentation	9.00	
885-1086-[37]	Tiny HDOS PASCAL H8/H89	20.00	13
885-1094	HDOS Fig-Forth H8/H89	40.00	18
885-1132-[37]	HDOS Tiny BASIC Compiler	25.00	59
885-1134	HDOS SMALL-C Compiler	30.00	63
CP/M			
885-1208-[37]	CP/M Fig-Forth H8/H89 2 Disks	40.00	18
885-1215-[37]	CP/M BASIC-E	20.00	26
BUSINESS, FINANCE AND EDUCATION			
HDOS			
885-1047	Stocks H8/H89 Disk	18.00	
885-1048	Personal Account H8/H89 Disk	18.00	
885-1049	Income Tax Records H8/H89 Disk	18.00	
885-1055-[37]	MBASIC Inventory Disk H8/H89	30.00	
885-1056	MBASIC Mail List	30.00	
885-1070	Disk XIV Home Fin H8/H89	18.00	
885-1071-[37]	MBASIC SmBusPk H8/H19/H89	75.00	17
885-1091-[37]	Grade/Score Keeping H8/H89	30.00	14
885-1097-[37]	MBASIC Quiz Disk H8/H89	20.00	18
885-1118-[37]	MBASIC Payroll	60.00	30
885-1131-[37]	HDOS CheapCalc	20.00	47
885-8010	HDOS Checkoff	25.00	32
885-8021	HDOS Student's Statistics Pkg	20.00	44
885-8027	HDOS SciCalc	20.00	50
CP/M			
885-1218-[37]	CP/M MBASIC Payroll	60.00	31
885-1233-[37]	CP/M CheapCalc	20.00	47
885-1239-[37]	Spread Sht. Contest Disk I	20.00	
885-1240-[37]	Spread Sht. Contest Disk II	20.00	
885-1241-[37]	Spread Sht. Contest Disk III	20.00	
885-1242-[37]	Spread Sht. Contest Disk IV	20.00	
885-1243-[37]	Spread Sht. Contest Disk V	20.00	
885-1244-[37]	Spread Sht. Contest Disk VI	20.00	
885-8011-[37]	CP/M Checkoff	25.00	32
885-8036-[37]	CP/M Grade	20.00	70
ZDOS			
885-3006-37	ZDOS CheapCalc	20.00	47
885-3013-37	ZDOS Checkbook Manager	20.00	54
885-3018-37	ZDOS Contest Spreadsheet Disk	25.00	58
885-8028-37	ZDOS SciCalc	20.00	50
885-8030-37	ZDOS Mathflash	20.00	55
DATA BASE MANAGEMENT SYSTEMS			
HDOS			
885-1107-[37]	HDOS Data Base System H8/H89	30.00	23
885-1108-[37]	HDOS MBASIC Data Base Sys.	30.00	23

Continued on Page 83

Another Way To Upgrade The H-89:

Dual Processing — Sort Of

Phillip L. Emerson
3707 Blanche
Cleveland Heights, OH 44118

My H-89 is rather old, but very reliable. It has three disk drives, but just the hard-sector disk controller. One of my main dissatisfactions, of late, has been the paucity of on-line disk space for running word processors and certain compilers that occupy quite a lot of space together with the various associated library files. Also, though, I have become somewhat impatient with the time required for executing some of my number-crunching programs.

Disk Controller Boards

A couple of months ago, it seemed to me that the most direct solution to the disk-space problem would be the Heath H-89 soft-sector disk controller, which handles higher density formats, with double-sided disk drives. I called the local Heathkit store, and found that these controller boards were not available. After several weeks of hoping and calling, it seemed clear that these boards will no longer be available — ever. I ran across a third-party vendor who sells a similar board with some added features for about $354 (Payload Computer Services). I ordered one, but still felt a nagging about the speed of numerical computations.

Single-Board Computers

A few days later, another solution occurred to me, so I called and cancelled my order for the controller board. I then placed an order ($700) for a Slicer 80188 CPU board. The Slicer 80188 is an 8 Mhz 16-bit system with on-board floppy disk controller for two soft-sector floppy drives, handling 40 or 80-track, single- or double-sided formats. The board comes with 256 Kbytes of RAM, half of which is used for a RAM disk. A bit over 100 Kbytes is available for transient programs. This 80188 CPU board is the same size as a 5-inch floppy drive, so it can be mounted rather compactly, with due precautions for heat dissipation. My idea was to place it in the slot vacated by the internal disk drive of the H-89.

This Slicer 80188 solution seemed to offer more value per dollar than the disk-controller board, and to address the problem of speed, as well as disk space. The 80188 board has two serial interfaces, one of which must be used for terminal input and output. For an additional $85, it comes with CP/M-86 installed. An installed version of MS-DOS also is available, according to the advertising, but I have not been able to find out the price. I am using the CP/M-86 operating system for the time being. This 80188 board has a nice ROM debug monitor, and includes software and host controller for a Winchester hard disk. However, the use of a hard disk requires an interface controller board available for an additional $200.

For the present, I am using two Teac FD55F drives, for 1.6 Mbytes of whirling memory. That seems like a lot, after struggling along for years with only about 300 Kbytes. These Teac drives, and equivalent ones from other manufacturers, have been available recently for about $100 from mail-order suppliers (see the ads in Computer Shopper or Byte). They are half-height, so two can be mounted in place of a single full-height. Also, their power consumption is modest, so two of them are powered adequately by the generous power allotted by the H-89 system for a single full-height drive. Two of my old H-77-1 drives remain connected to the H-89. I am now in the process of archiving the contents of several dozen fully-loaded hard-sector diskettes on the new higher density ones. The capacity ratio is about 8 to 1. The only disadvantage of the 80-track drives is that their recording tracks are too thin to be read reliably by the older (IBM-compatible)

40-track drives. There is no problem in reading 40-track diskettes with the 80-track drives.

Hardware Connections

To get the Slicer up and running, I mounted it temporarily in a separate aluminum box, with its own power supply (Power One BAA 40W) and one of the Teacs. The other Teac is in one of the slots in the H-89 H-77 box, from which one H77-1 drive has been removed. The power requirements for the Slicer and the Teacs are quite modest. A total of +5 at 3.5 amps, +12 at 2 amps, and -12 at .05 amps is sufficient. The Slicer board itself takes +5 at 1.5 amps, +12 at .06 amps, and -12 at .05 amps. At first, I hooked the Slicer terminal port directly to the terminal board of the H-89, and of course, that makes the H-89 inoperable. Now, however, I am using the H-89 with its modem port connected into the terminal port of the Slicer. This way, I can access both kinds of disks and transfer files between them, as well as the Slicer RAM disk which is quite a pleasure.

Programming

The Slicer seems to require a 9600 baud terminal, though the documentation suggests that other rates will work. It also requires hand shaking using the RTS and CTS RS232 signals. Therefore, not just any modem-type program for the H-89 will work for this purpose. I have begun to write my own. Its present state is as shown in the program listing. The source language is C/80, the version of C from Software Toolworks. The C language seems ideal for low-level nitty-gritty programming such as this. I am very happy with some aspects of this program which I call "TRMNL". When TRMNL is loaded and running, the fundamental feature is that one is dealing with CP/M-86 on the host, almost entirely as if using a simple terminal.

A minor inconvenience is that control characters meant for the host operating system must be entered indirectly, so that they will not be intercepted and acted upon by the H-89 and its operating system. However, the advantage is that you may at any time type a control-C directly to return to the H-89 command processor, to do any H-89 chores. You can examine an H-89 disk directory, swap diskettes, edit H-89 files, and run other H-89 programs. Then, when you reload TRMNL, you are back with the host where you left off — without having to exit and re-enter whatever program was running. It is very much like switching an ordinary terminal off-line temporarily, and then back on.

Transmitting ASCII Files

The transfer of H-89 ASCII disk files to CP/M-86 disks on the host is a very slick operation, in two command steps: (1) get CP/M-86 pip and give the command FILE1.XXX=CON:, and then (2) press the ESC key (getting a & prompt from TRMNL) and type T FILE2.YYY. The contents of H-89 file FILE2.YYY will then be transferred to CP/M-86 file FILE1.XXX, and the procedure ends with the automatic sending of a Control-Z to signal end-of-file to CP/M-86. Exact file images are transferred in this way, and no post-transfer editing is needed.

Receiving ASCII Files

The transfer of disk files in the opposite direction is not quite as elegant. For one thing, about three lines of the command protocol get added into the destination H-89 file, and must be edited out later if they are objectionable. Secondly, one must oversee the operation because the closing of the destination file is not automatic. The command procedure is as follows. With CP/M-86 at the top command level (prompt A>): (1) press ESC (getting the & prompt from TRMNL) and type R FILE1.XXX; then (2) type TYPE FILE2.YYY to CP/M-86. The disk transfer from source file FILE2.YYY to H-89 file FILE1.XXX then proceeds until FILE2.YYY is exhausted. Then one must press ESC to get TRMNL to close the destination file, FILE1.XXX. This procedure works adequately, but the command string, TYPE FILE2.YYY gets added in at the top of the destination file, and an extra blank line followed by the CP/M-86 prompt, A>, gets added in at the bottom. It would be fairly easy to provide for the automatic elimination of these fixed patterns, but my main interest at present is in transferring H-89 files to the host.

Transmitting Binary Files

This program (TRMNL) is still under development, but the fundamental skeleton is quite workable. Another procedure that is built in at present is for transferring binary files from the H-89 to the host computer. The procedure is the same as that described above for transmitting ASCII files, except that the T FILE2.YYY command is replaced by U FILE2.YYY. This is a form of "uploading." Under this command, TRMNL bisects each binary byte, and transmits it as two hex bytes, inserting a carriage return and line feed after every 64th hex byte. The destination file ends up in hex form, of course, rather than absolute binary. The idea of the simple procedures of this TRMNL program is to interact only with the standard command processor of the host operating system. Obviously, many other things could be done by calling up special companion programs written for the host system.

However, the ultimate aim of this upgrade project is to gradually transfer operations to the new higher-capacity system. The interactive operations will probably become less frequent as the archiving of data progresses, and as software for the new system is acquired. Thus, there may not be much point in developing the present TRMNL program any further. On the other hand, one might consider running both processors in tandem on a permanent basis. One of them could be devoted to input-output tasks, such as printer spooling. The orchestration of such a system would go somewhat beyond the capabilities of the present TRMNL program.

Variations On The General Idea

This general approach to upgrading the H-89 certainly is one to be considered. If $700 for an 80188 board is too much, small single-board 4 Mhz Z-80 computers with built-in soft-sector disk controllers are available. There is one advertised for as low as $300 or $230 in kit form, including CP/M 2.2 installed (Ampro Little Board, from Digital Research Computers). The disk controller handles all popular formats, 40 or 80 track, single- or double-sided. This upgrade would be relatively inexpensive, though not providing as much speed enhancement. One might replace the H-89 CPU board entirely, or provide for a smoother transition by using a terminal program like TRMNL. The use of another Z-80 computer board together with the H-89 CPU should be quite straightforward, especially if both operating systems are CP/M. Most old programs acquired for the H-89 would then run directly on the new board. One convenience of the Z-80 Ampro Little Board, is that it generates its own -12 volt supply on board. Thus, it could use the +5 and +12 sources designed to supply the internal disk drive on the H-89, if that drive were to be removed.

I noticed recently that Ampro also produces an 8 Mhz 16-bit 80186 Little Board computer. The announcement quoted $499 for a 128 Kbyte version and $649 for a 512 Kbyte version, so this may be a better buy than the Slicer 80188, in the 16-bit class of disk drive sized single-board computers.

Addresses Of Some Of The Suppliers Mentioned

Ampro Computers, Incorporated
67 East Evelyn Avenue
Post Office Box 390427
Mountain View, CA 94039

Digital Research Computers (of Texas)
P. O. Box 461565
Garland, TX 75046

Payload Computer Services
15006 Sun Harbor
Houston, TX 77062

Slicer Computers, Inc.
2543 Marshall Street, N.E.
Minneapolis, MN 55418

Software Toolworks
15233 Ventura Boulevard, Suite 1118
Sherman Oaks, CA 91403

```
/* file SY1:TRMNL.C */

/* TRMNL  -- to use the H89 as a terminal to computers
     wanting a serial terminal    P  Emerson 6-19-85 */

/* Some special kludges to be aware of
 (1)  KB typed ctrl chrs don't all get through H-89 OS
      to the host when typed directly.  To get one
      through for sure, use the ` character   For example,
      type `C for control-C, and `Z for control-Z.  Also,
      `[ will pass as ESC, and `M as RETURN or ENTER.
      This special character can be changed to some other
      by redefining CTLKEY, below.
 (2)  This is an H-89 HDOS program, -- but only the two
      small routines console and getkbas are peculiar
      to HDOS, plus the substitution of CR LF for LF
      after the "else" in the main program
 (3)  HDOS never really passes a RETURN chr from kb --
      instead it converts it to LINEFEED which is ok for
      most purposes. However, the Slicer debug monitor
      and some of the SETUP routines look specifically
      for RETURN.  In those cases it is necessary to
      type `M.
 (4)  The KBEOF (ctrl-Z) file-end signal is peculiar to
      CP/M transfers from CON:
 (5)  A directly typed ESC initiates a TRMNL disk-op
      command. Therefore, an ESC meant for the host must
      be typed as `[.
 (6)  The portin and portout routines are not peculiar to
      HDOS, but they are for 8080 compatibles such as the
      Z-80, and the C/80 method of passing arguments.   */

/* misc. common constants */
#define DONOTHING   /* a filler to clarify logic of
                          source code */
#define NOMAIL -1 /* no input byte available */
#define OUTASCI 1 /* 3 file ops coded 1,2,3 */
#define INASCI 2
#define OUTHEX 3
#define EOF -1      /* eof on disk read */
#define ESC 27
#define BAKSPC 8
#define DELETE 127
#define LF 10
#define CR 13
#define KBEOF 26   /* ctrl-Z for eof on xfr from CP/M-86 CON: */
#define BELL 7     /* H89 beep */
#define CONSOL 6   /* HDOS SCALL codes */
#define SCIN 1

/* common constants for INS8250 serial interface registers beginning
   at 216 (330Q)  */
#define MODEM 216
#define BUFREG 216
#define INTREG 217
#define INTRID 218
#define LINCTRL 219
#define MODCTRL 220
#define LINSTAT 221
#define MODSTAT 222
#define LODIV 12     /* 9600 baud */
#define HIDIV 0
#define MODE 7       /* 8 data bits,no parity,2 stop bits */

/* common characters and strings for kb and screen control */
#define CTLKEY '`'
char savcurs[] = "\033j" ;
char rescurs[] = "\033k" ;
char enab25[] = "\033x1" ;
char addr25[] = "\033Y\070\040" ;
char disab25[] = "\033y1" ;
char eraslin[] = "\0331" ,
char rubout[] = "\033D\040\033D" ;

/* disk-ops common variables */
int fileop,chan,odd,count ;
char filecom[31] ;

/* setcom initializes INS8250 for baud rate, etc
   this procedure from appendix of H89 Operations manual */
setcom() { int i,
  portout(INTREG,0) ;
  portout(MODCTRL,16) ,
  portout(LINCTRL,128) ;
  portout(BUFREG, LODIV) ;
  portout(INTREG, HIDIV) ;
  portout(LINCTRL, MODE) ;
  for (i=0; i<5000; i++) ,
  portin(BUFREG) ;
  portout(MODCTRL,0);
}

/* console sets char mode, no echo, for console input */
/* kb input is via special HDOS routine getkbas rather than
   getchar so H89 can act as full-duplex terminal */
console() {
#asm
        XRA A
        MVI B,201Q
        MVI C,201Q
        SCALL CONSOL
#endasm
   return; }

/* puts concatenation of strings a and b at line 25 on screen */
screen25(a,b) char *a,*b;
{ putstr(savcurs); putstr(enab25), putstr(addr25);putstr(a);
  putstr(b); putstr(rescurs) ; }
```

```
/* erases line 25 */
erase25()
{ putstr(savcurs), putstr(enab25); putstr(addr25),
  putstr(eraslin); putstr(disab25), putstr(rescurs) ; }

/*getkbas to get a char from KB, if available, return -1 if not */
getkbas()  {
#asm
        LXI H,NOMAIL
        SCALL SCIN
        JC GETOUT
        MVI H,Ø
        MOV L,A
GETOUT  NOP
#endasm
   return, }

/* get byte from kb, and normally return it   However, (1) special
   ctlkey char is used to convert to ctrl chrs, and (2) directly
   typed ESC causes entry to disk-op command procedure */
getkb()
{ static int flag; int c; char mapup() ,
  if( (c=getkbas()) == NOMAIL) return(NOMAIL) ;
  if( ( (char) c) == CTLKEY ) {flag = 1; return(NOMAIL) ; }
  if( (char) c == ESC )
    {diskop(); return(NOMAIL), }
  if( !flag ) return(c) ;
  flag = Ø,
  return(mapup( (char) c) & 63) ; }


/* to turn on RTS line, do a portout(MODCTRL,2)
   to turn it off, do a portout(MODCTRL,Ø).
   to test if CTS, do a portin(MODSTAT) & 16
   to test if xmit reg free, do a portin(LINSTAT) & 32
   to test if new byte in rcvr reg, do a portin(LINSTAT) & 1. */

/* send a byte to host */
sendone(a) char a,
{ portout(MODCTRL,2) ;
  while( !(portin(MODSTAT) & 16)) ,
  while( !(portin(LINSTAT) & 32)) ;
  portout(BUFREG,a) ,
  portout(MODCTRL,Ø) ;
}

/* receive a byte, or NOMAIL, if none available */
recvone()
{ int c;
  portout(MODCTRL,2);
  if( (portin(LINSTAT) & 1) && (portin(MODSTAT) & 16) )
    c = portin(BUFREG) ;
  else c = NOMAIL ;
  portout(MODCTRL,Ø);
  return(c) ; }

/* send a byte from H89 file -- requires waiting for echos and
   an error check if in binary-hex mode */
filsend(a) char a;
{char b;
 sendone(a) ,
 while( (b=recvone()) == NOMAIL) DONOTHING ;
 if( (a != b) && (fileop == OUTHEX) ) errept(a,b) ,
 putchar(b) ;
 while( (b=recvone() ) != NOMAIL) putchar(b) ;
}

/* file handling according to commands initiated with typed ESC */
diskop()
{ char mapup() ;
 if (fileop)
       {fclose(chan), erase25(); chan = -1;
         if( (fileop == OUTASCI) || (fileop == OUTHEX) ) sendone(KBEOF) ;
         putchar('\n'); fileop = Ø; return;
       }
  putstr("\n&"); getstrl(filecom) ;
  screen25("Disk op in progress: &",filecom) ;
  if( mapup(*filecom) == 'T')
       {chan=fopen(filecom+2,"r") ; fileop = OUTASCI ; }
  else if ( mapup(*filecom) == 'R' )
       {chan=fopen(filecom+2,"w") ; fileop = INASCI ; }
  else if ( mapup(*filecom) == 'U' )
       {chan=fopen(filecom+2,"rb") , fileop = OUTHEX ; odd = Ø; count = Ø;}
  if( (chan == Ø) || (chan == -1) )
       {putstr("\nerror -- disk op cancelled on");
        putstr(filecom); putchar('\n') ,
        fileop = Ø; erase25() , }
}

/* map lower to upper case */
char mapup(a) char a ,
{if ( (a >= 'a') && (a <= 'z' ) )
  a += ('A' - 'a'); return(a) ; }

/* get a disk-op command string -- either DELETE or BACKSPACE
   can be used to back up and erase */
getstrl(a) char *a ;
{ int c,n; char *b, b=a; n=Ø;
  while( ( ( (char)(c=getkbas() ) ) != '\n' ) && (n < 3Ø) )
    { if (c != NOMAIL)
       {if ( (c == BAKSPC) || (c == DELETE) )
           {if (a <= b) putchar(BELL) ,
            else {a-- ; n-- , putstr(rubout) ; }
           }
        else
           { *a = (char) c ;
             putchar(*a++) ; n++ ;
           }
       }
    }
 *a = Ø; putchar('\n') ;
}

/*Get a char from H89 disk chan  If active=OUTASCI, get it directly. If
  active=OUTHEX, do split and return hex halves, with \n after each
  64 halves */
getcl(chan,active) int chan,active ;
```

```
{ static char bitehi, bitelo ; int n,
  if (active == OUTASCI) return( getc(chan) ) ;
  if (active != OUTHEX) return( EOF ) ;
  if (count == 64) {count=0; odd=0; return('\n'), }
  if (odd) {odd=0; count++; return(bitelo) , }
  if( (n=getc(chan)) == EOF ) return(EOF) ;
  bitehi = bitelo = (char) n ;
  bitelo = (bitelo & 15) + '0' ;
  bitehi = ( (bitehi/16) & 15 ) + '0' ;
  if (bitelo > '9') bitelo += ( 'A' - '9' ) ;
  if (bitehi > '9') bitehi += ( 'A' - '9' ) ;
  odd = 1; count++ ,
  return(bitehi) ;
}

/* error report for mismatch between a sent char and its full-
   duplex echo from host -- used only when transmitting
   binary files in hex form */
errept(a,b) char a,b,
{ putstr("\necho mismatch "), putchar(a), putchar(b); putchar('\n'),}


/* MAIN PROGRAM */
main()
{ int c,i; char a;
  console(),
  setcom();
  while(1)
   {if(  (c=recvone()) != NOMAIL)
       { putchar( (char) c) ,
         if( fileop == INASCI ) putc( (char) c,chan) ;
       }
    if ( ( fileop == OUTASCI) || (fileop == OUTHEX) )
       { a = getcl(chan,fileop) ;
         if ( a == EOF )
            {fileop = 0 ;
             fclose(chan) ;
             erase25() ;
             chan = -1 ;
             sendone(KBEOF) ;
            }
         else
            {if( a == LF )  filsend(CR) ;
             filsend(a) ;
            }
       }
    if ( (c=getkb() ) != NOMAIL ) sendone( (char) c ) ,
   }
}


/* routines from library file, SY1:USRLIB.B*/
/*****************************************/
#asm
EXECUTI DW 0
        RET
#endasm
portin(port)
 int port ; {
#asm
        POP D           RETURN ADDRESS.
        POP H           ARGUMENT
        MOV H,L         PORT NUMBER
        MVI L,333Q      "IN" OPCODE.
        SHLD EXECUTI    STORE INSTR.
        CALL EXECUTI    EXECUTE IT.
        MVI H,0         MOVE RESULT TO HL.
        MOV L,A
        PUSH H          RESTORE STK. PTR.
        PUSH D          RESTORE RETURN ADDRESS.
#endasm
return ; }
/**/
#asm
EXECUTO DW 0
        RET
#endasm
portout(port,byte) int port,byte ; {
#asm
        POP D           RETURN ADDRESS
        POP B           2ND ARG.
        POP H           1ST ARG.
        MOV H,L         PORT NR
        MVI L,323Q      "OUT" OPCODE
        SHLD EXECUTO    STORE INSTR
        MOV A,C         DATA BYTE
        CALL EXECUTO    EXECUTE INSTR.
        PUSH H          RESTORE STK. PTR.
        PUSH B          RESTORE STK  PTR.
        PUSH D          RESTORE RETURN ADDRESS
#endasm
return ; }
/**/
putstr(s) char *s; {while(*s) putchar(*s++) ; }
/**/
```

*

Performance.

Introducing EasyPC™ from UCI. The most advanced IBM PC emulator you can buy for your Z-100. First to provide the compatibility, speed and audio support to make your Z-100 run like a PC. Since the EasyPC provides complete PC circuitry, you actually run PC programs on PC hardware. Quite simply, it's like installing a PC right inside your Z-100. In the PC mode, your Z-100 can run virtually the entire library of PC programs, including LOTUS 1-2-3, dBase III, FLIGHT SIMULATOR, and most other copy-protected software. In the Zenith mode, your Z-100 is a Z-100, with no change in performance.

EasyPC is fast. In fact, it is noticeably faster than both the Z-150 and the IBM PC. And EasyPC can run at either 5 or 8 MHz. It supports both floppy and Winchester drives, provides crisp color video graphics, supports PC-compatible sound generation, and even comes with its own self mounting speaker.

Quality.

You already know and appreciate the kind of quality and reliability that is built into your Z-100. UCI matches that commitment to quality from first step to last. Sophisticated engineering and quality components combined with precision automatic assembly, component burn-in, and both in-circuit and full functional testing, mean superior performance and long term reliability.

EasyPC's advanced circuitry utilizes two, high quality, S-100 multilayer boards and a piggyback board. Yet its ingenious design requires only *one extra slot*. So you get all the advantages of PC compatibility, with room to spare.

Availability.

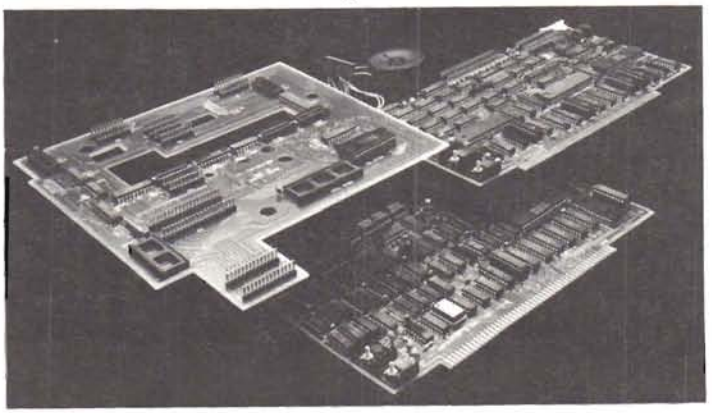
EasyPC is ready when you are. If you've been waiting too long for PC compatibility, your wait is over. EasyPC is here now. Complete documentation is included for quick and easy installation, so you'll be up and running fast. Your only problem will be which PC programs to get first.

EasyPC is available through your independent Zenith dealer. Ask him for a demonstration.

Price.

At $699, you'll pay a little more. But you'll get the best. Total compatibility. High speed performance. Unsurpassed quality and serviceability. In a word, value. You also get UCI's one year Solid Service Guarantee and full customer support. If you have any questions or problems with any UCI product, or need the name of your nearest dealer, just call 800-UCI-COMPUTER.

EasyPC™



Full IBM PC emulation for your Z-100.

UCI CORPORATION
948 Cherry Street, Kent, Ohio 44240
(216) 673-5155 / 800-UCI-COMPUTER

Benchmarking The 8087 Co-Processor In A Z-110 With Autocad

John A. G. Roach and Nancy M. Roach
JANR Technical Services
P. O. Box 10515
Alexandria, VA 22310

The 8087 numerical co-processor executes numerical instructions faster than the 8088 processor that is used in the Z-110. If your application involves a significant number of calculations and your software supports an 8087, the numerical co-processor can reduce your processing time. Benchmark articles that try to estimate the amount of time that you will save usually use code specifically written for the 8087. For example, Dougherty (The 8087 Chip, MONITOR, June 1984, 35) used FORTRAN to test the 8087 with a stream of simple mathematical functions and Boyd (Z100 Benchmark With A 8-MHz 8087 Co-Processor, BUSS, #100, 6) used "C" for a similar test. AUTOCAD is written in "C" and does not require modification of code to use an 8087 co-processor.

AUTOCAD, which was just chosen as "Technical/Scientific Software of the Year," (BUSS, #100, 8) is a computer-aided design package. It is primarily used by architects and engineers. We use it to prepare technical illustrations for geological, chemical and educational manuscripts. Computer-aided design offers several advantages over hand drawn illustrations. Chemical structures are stored in a library of shapes and may be combined or modified to form other structures in subsequent drawings. An original drawing only has to be drawn once with a computer. The drawing may then be scaled to any size and placed in a draft of a manuscript. Revision of a drawing via computer requires only a fraction of the time required to revise a drawing by hand.

The complete design system consists of the software, a microcomputer and the peripheral devices required to prepare a drawing. The recommended minimum memory for AUTOCAD 2.0 is 512 kilobytes. The user addressable memory of the Z-110 was increased to the maximum 704 kilobytes providing more than enough room for AUTOCAD. A high resolution ZVM-134 color monitor displays the drawing in memory. Data are stored on a 27 megabyte Winchester disk drive. Drawings are input to the Z-110 via a Houston Instruments DT114 digitizing tablet and the Z-110 keyboard. Output is plotted via a Houston Instruments DMP29 digital plotter.

Drawing preparation was obviously faster after installing an 8087 co-processor in the Z-110, but the time saved seemed to vary with different types of illustrations, so we selected a set of drawings to study the effect of an 8087 co-processor and other variables such as clock speed and memory size.

Two 8087 cards were tested: a Z SUPPORT BOARD with 5MHz 8087 co-processor and 256K RAM (HUDSON a.n.d. associates, Box 2957 Santa Clara, CA 95955-2957) and a 2+2 equipped with a 5 MHz 8087 co-processor (D.E.L. Professional Systems Inc., 12-11151 Horseshoe Way, Richmond, B. C. V7A 4S5). The Z SUPPORT BOARD mounts in an S-100 slot of the Z-110 and provides RAM, as well as 8087 support. The 2+2 card mounts in the 8088 socket of the Z-110 main board. Both cards will operate at 8 MHz when equipped with an 8 MHz 8087.

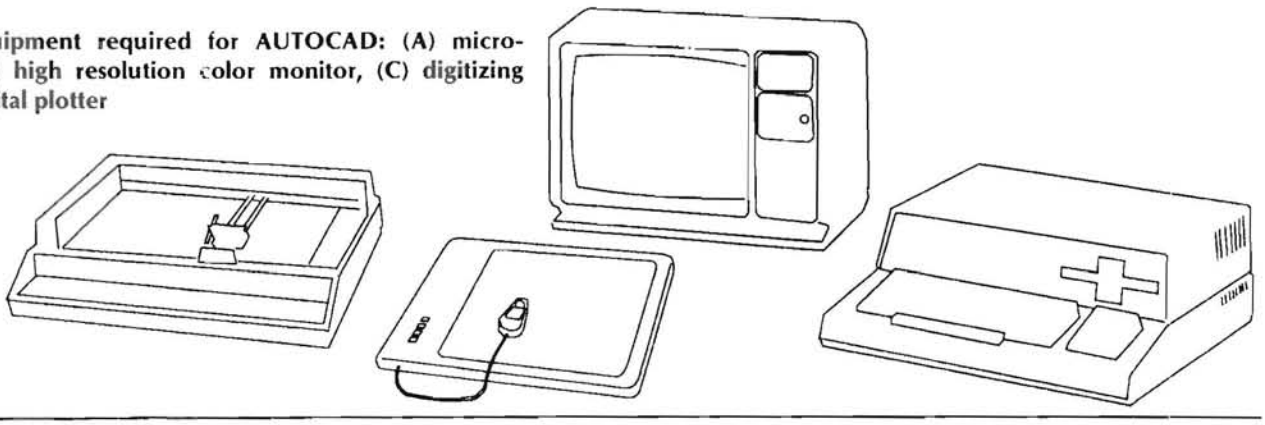


Figure 1. Equipment required for AUTOCAD: (A) microcomputer, (B) high resolution color monitor, (C) digitizing tablet, (D) digital plotter

Installation of either board is fairly straight forward, but you must take precautions not to damage any CMOS components with static electricity. Connect a ground strap from a metal watchband on your wrist to a water pipe or other earth ground when you are ready to handle these components. Special extraction and insertion tools (40 pin Grounded DIP IC extractor #41-612 and insertor #41-608, GC Electronics, Rockford, IL 61101) can be used to safely handle these components when the tools are also connected to the ground strap, but a small screwdriver and a little patience work almost as well. The directions provided with either board are an adequate installation guide.

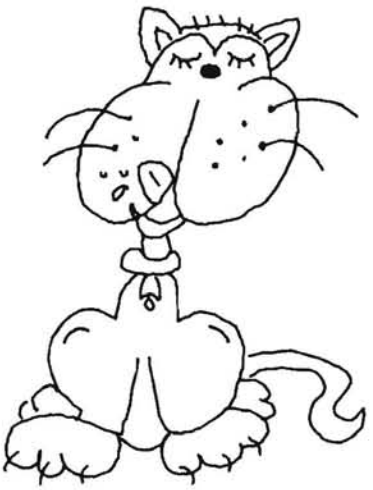
In considering an 8087, we had heard that it was a hot chip requiring careful cooling to operate. Both boards placed the 8087 in the cooling air flow of the Z-110. In order to determine if the 8087 required a heat sink for additional cooling, we monitored its temperature during operation. After 4 hours of operation in a 77 degree Fahrenheit room, the 8087 was 95°F and the adjacent 8088 was 93°F. At this point, we used an extender board to move the 8087 out of the cooling air stream. After one-half hour of operation, the 8087 was 110°F and the 8088 was 106°F. We concluded that additional cooling was not necessary.

Fellow Capital Heath Users' Group member, George Page, had a Z-100 set up for 8 MHz operation and a keen interest in 8087 co-processors, so we obtained baseline 8088 performance data at 8 MHz and 5 MHz by comparing our two machines. We verified that there was no other difference in the speed of the two machines by temporarily replacing his 8 MHz crystal with a 5 MHz crystal for some of the tests.

AUTOCAD will use available RAM to improve its performance. We kept the amount of RAM constant at 448K for most of the tests. Operations on a large drawing will be slowed down disproportionately compared to a small drawing if there is an insufficient amount of RAM. The drawings for these tests were selected by running them in 704k RAM and 448k RAM with AUTOCAD 1.4 to insure that 448k was large enough not to effect the execution speed of any of the drawings.

Each of the drawings contained specific drawing elements so that by comparing the speed of execution, we could determine which elements were using the 8087 co-processor. For example, CAT was generated in the sketch mode. It occupies 31K and consists of hundreds of small straight line segments. In contrast, TURTLE consists of text and a drawing of a turtle that is made up of arcs, lines and dots. TURTLE occupies 25K.

Figure 2. CAT



SECTION is a 59K drawing containing sketch, pattern and text elements. MAP, is a 49K drawing contains only sketch and text elements. The other test drawings contain only text and occupy very little storage. QUIET takes 2K. REMARK, an ad on the back of REMARK 5, 11 (1984), takes 2K. POSTER, the basis for the ad on page 35 of REMARK 5, 9 (1984), takes 3K.

Turtle

Lives in a shell
Swimming, sleeping, eating
A turtle will always have fun
Slow fun

by

Becca Roach

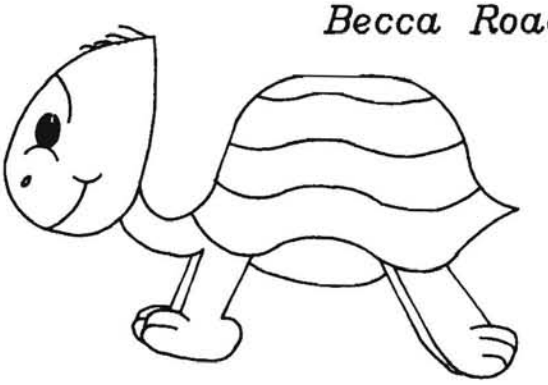


Figure 3. TURTLE

When AUTOCAD 1.4 generates a drawing from a drawing file, it calculates the location of all of the elements in the drawing and stores this in a reference file along with any subsequent changes to the drawing made by the user. The reference file can be used to quickly regenerate the drawing from the drawing file. If the reference file is erased, AUTOCAD 1.4 must recalculate all of the drawing element positions from the drawing file to generate the drawing. Thus, it is a simple matter to reproducibly time AUTOCAD 1.4 performance by erasing a drawing reference file and then noting how long it takes to regenerate the drawing.

Part of the time required to generate a drawing is spent in disk access time. In order to minimize the influence of disk access on the time recorded for each drawing, text fonts were loaded before the tests and the time trial was not started until the drawing editor appeared on the monitor. The editor appears during drawing generation after nearly all of the disk access is completed. Its appearance on the monitor was a convenient time to start the stopwatch.

Comparison A, Drawing Time vs (1) 8088 at 5 MHz, (2) 8088 at 8 MHz and (3) 8088/8087 at 5 MHz, supports two conclusions. First, an 8087/8088 operating at 5 MHz, is faster than an 8088 at 5 or 8 MHz. If the same amount of math was involved in each type of drawing tested, the 8087/8088 results would have shown the type of improvement that was observed by changing from the 5 MHz to the 8 MHz clock. But TURTLE took less time to draw compared to SECTION with an 8087. This led to the conclusion that the various elements in a drawing require various amounts of math.

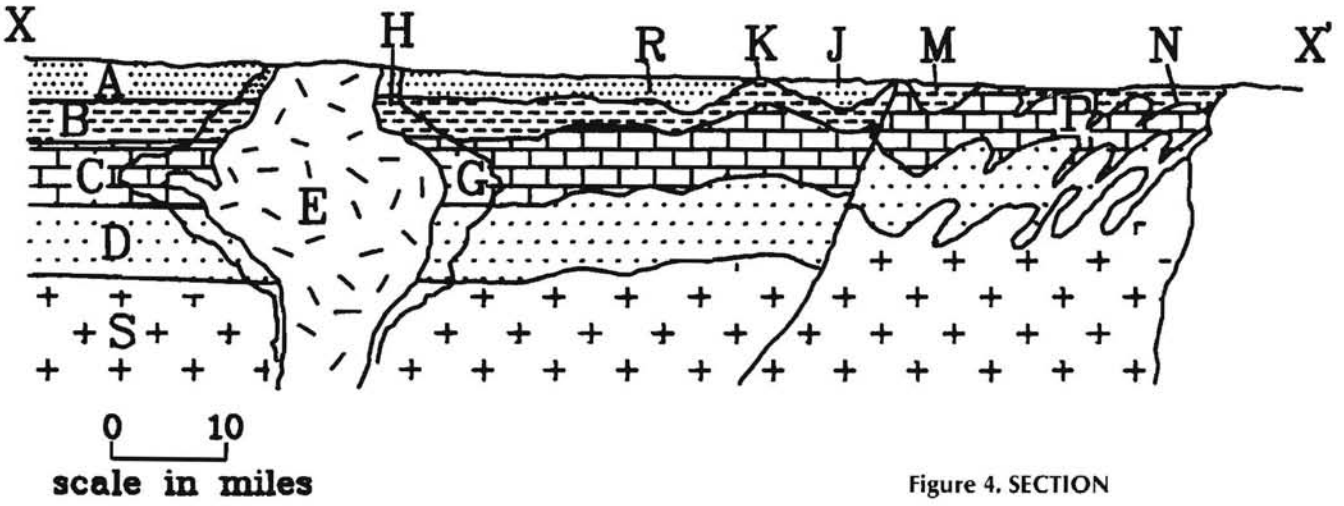


Figure 4. SECTION

TURTLE contains more text than the other drawings in Comparison A.

Comparison B, Drawing Time vs (1) 8088 at 5 MHz and (2) 8087/8088 at 5 MHz, was compiled from drawings that contain various amounts of text. An italics text font was used to generate the drawings because it is the slowest font for text with an unassisted 8088. This would produce the greatest difference in drawing speed between 8088 and 8087/8088 operation if generation of text relied heavily on the 8087. Even though POSTER was much smaller (3K) than CAT (31K), it took 18 minutes and 49 seconds to generate with an unassisted 8088. The addition of an 8087 reduced this time to 5 minutes and 20 seconds. In retrospect, this is logical. A text font is a library of shapes. Each shape consists of straight line segments that form a single letter of known size and orientation. To place text into a drawing, AUTOCAD must do the calculations to scale and reorient each letter.

This article was based on AUTOCAD version 1.4 and was originally presented at the October 1984 meeting of the Capital Heath Users' Group. During the 8087 tests with version 1.4, we learned that drawing text required a lot of calculations. This made it very beneficial to have an 8087 co-processor, but it also made text one of the slowest AUTOCAD operations. Our AUTOCAD 2.0 arrived in time for Christmas. We have found that version 2.0 has significantly altered the file structure so that we cannot conveniently erase a reference file and then clock the speed at which a drawing is generated. It is possible to use other drawing operations to determine the usefulness of an 8087 co-processor with version 2.0, but it is probably not necessary because AUTOCAD has found a nifty way to save time on manipulating text in a drawing. In version 2.0, it is possible to toggle on/off the use of two parallel lines to depict the location and size of a line of text in a drawing. This saves the CPU time required to calculate and draw text during drawing regeneration.

In conclusion, if your application is word processing, spend your money on a faster clock. If your application involves numerous calculations and you know that your software supports an 8087, spend your money on an 8087 co-processor.

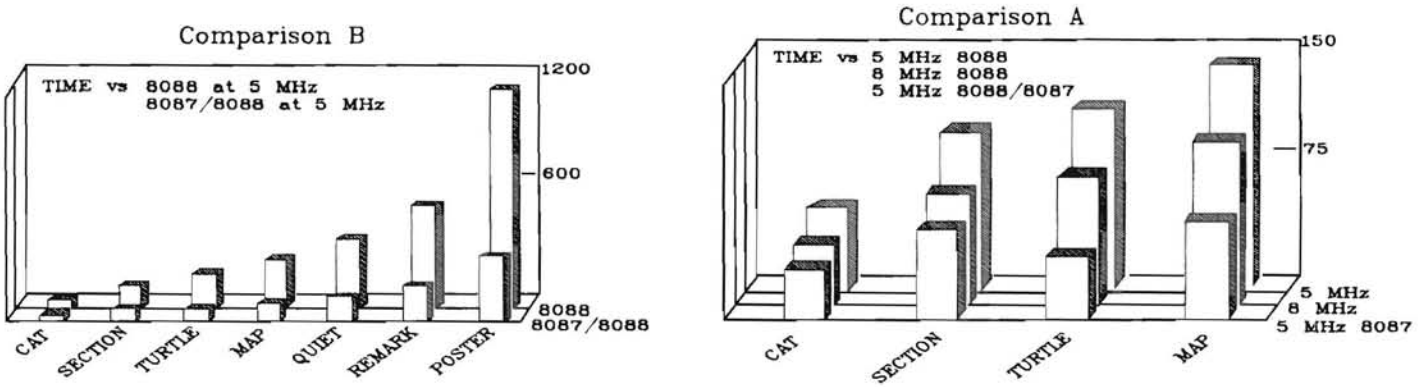
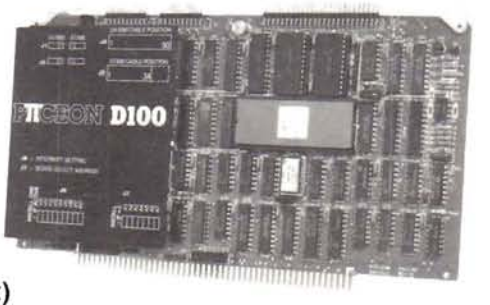


Figure 5. Comparisons A and B

Upgrade Your H/Z-100 Computer With A Piiceon Hard Disk Subsystem



Now Full Compatibiity

With Z-DOS (ver 2.1x) and PC-DOS (ver 2.xx)

Each Piiceon Hard Disk Subsystem comes complete with its own chassis, power supply, controller and installation software. Subsystems available in 10, 27, 44, 70 and 85 Megabytes at new low prices.

When the H/Z-100 Microcomputer is enhanced with the Gemini Emulator Board*, these subsystems are now compatible with both the Z-DOS and PC-DOS operating systems. Now access your files from either operating system.

Contact your Heath or Zenith Dealer for full details on these and other Piiceon **SUPERAM**™ Memory products — call today.

All you need to remember is... **Piiceon.**

PIICEON

2114 Ringwood Avenue
San Jose, California
408/946-8030

*GEMINI technology inc. **SUPERAM** is a trademark of Piiceon, Inc.

ZCPR3

We are licensed by ECHELON, Inc. to distribute the Z-System for the Heath market. This includes the ZCPR3 system with supporting software utilities, the ZRDOS-plus BDOS replacement and applicable documentation. Comes already installed and includes a bootable disk ready to run.

ZCPR Gives you many features found in UNIX and MSDOS. The ZRDOS is not in the public domain and gives you automatic logon of disks. No more "BDOS ERROR ON B:" messages just because you forgot to do a ^ C after changing disks.

Available for Heath BIOS 2.2.03 and .04 and for CDR BIOS. Check for availability of other BIOS configurations. Call or write for a brochure on this excellent product.

ZCPRS package - specify format $98

4MHz mod $45

An easy to install plug-in module. No trace cutting or soldering. Speed may be toggled with software. Includes a replacement Z80A (4MHz). Software supports Heath CP/M 2.2.03 and 04 and CDR Systems BIOS. Call or write for support of other BIOS or HDOS. Specify disk format.

EMULATE

Allows the H89 to read/write to the following disk formats.

Actrix	Eagle II	Morrow MD	Superbrain Jr.
AMPRO	Epson QX-10	NCR DecMate 5	Televideo
Beehive Tpr	Fujitsu CP/M86	NEC PC-8001A	TRS80-1 CP/M
CDR	IBM CP/M86	Osborne 1	TRS80-3 CP/M
Cromemco	IMS 5000	Otrona	TRS80-4 CP/M
DEC VT180	Kaypro II	PMC MicroMate	Xerox 820
DEC Rainbow	Magnolia	Sanyo 1100	Zorba

Now includes 42 formats! Uses a modified version of CP/M 2.203 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.

For H37 with Heath CP/M $59
Check for C.D.R. and Magnolia versions.

REP3 - Automatic Repeat

Stop wasting time - make full use of the repeat function just like the Z100. Simple plug-in installation on your keyboard assembly. Hold any key down for half a second and the key begins repeating. Combine this with our 4MHz mod and make WordStar fly! Provision for a defeat switch.

Kit $35
Assembled $45

ANALYTICAL PRODUCTS 209/564-3687

20663 Ave. 352 Woodlake, CA 93286

Specify Disk Format on Software

CALL OR WRITE FOR CATALOG PRICES SUBJECT TO CHANGE

Terms: Check or Money Order - VISA/MC - C.O.D.
Add $3 per order for shipping and handling
California residents add 6% tax

ZPC Update #3

Pat Swayne
HUG Software Engineer

This is the third in a series of support articles for ZPC, a program that allows H/Z-100 (dual processor) computers to run IBM PC and/or Z-100 PC (Z-150, etc.) software. As many of you know, ZPC has stirred up a lot of excitement in the H/Z-100 community, and with the confusion surrounding all of this, I have had a hard time keeping up. I have made little changes to ZPC as I discovered problems, and updated the masters that the disks we ship out are produced from each time, so that at least some of you would have immediate benefit of the changes. In the last ZPC Update article, I listed patches that were supposed to update the earlier releases, but I discovered that the patches do not work with all of them. So I will present new patches in this update. I discovered one more "bug", which will be corrected in these patches, so you may want to perform them even if you did the patches from the last update.

This article also contains patches for more PC software, to allow it to run under ZPC. The patches are for Enable (integrated software) version 1.1, Multimate Professional Word Processor version 3.3, Volkswriter Deluxe version 2.0, Edix version 2.05, and PC Palette version 1.0. New patches for the Heath/Zenith release of Microsoft Word are also included.

In the last ZPC Update article, it was reported that Print Shop would run without patching. The article should have said Print Master. Print Shop is copy protected, and it is not known if it will run. Programs that will run without patching include GLU-Draw, PC-File, and BSE150.

All of the patches for ZPC that are presented in this article, and all of the PC program patches from this and the last update are available on the new ZPC Support Disk (HUG P/N 885-3034-37), which sells for $10.00 plus shipping/handling. The disk also includes a program that emulates the ANSI.SYS driver provided with Z-100 PC MS-DOS. This emulator will allow you to run programs that require ANSI.SYS. See the New Products page of any REMark for ordering information.

Later in this article, I will discuss the memory requirements of ZPC, especially the 768k requirement for maximum compatibility.

The ZPC Patches

The patches presented here are for ZPC3.COM (level 3 of ZPC). Many of the changes only affect level 3, which is required to run most PC software, including all of the programs for which patches are provided in this article. It is assumed that you have prepared a ZPC system disk as instructed in the ZPC documentation, and that you have copied ZPC3.COM to it and renamed it to ZPC.COM.

Using an editor or word processor, create a file called FIXZPC.BAT on your ZPC system disk that contains this line:

```
DEBUG ZPC.COM <FIXZPC.DAT
```

Now, create a file called FIXZPC.DAT. If your original ZPC3.COM file is dated 9-5-85, FIXZPC.DAT should contain the lines in the first column. If it is dated 9-9-85 or 9-11-85, use the lines from the second column. If it has a later date, go to the next paragraph.

```
F1848,184B,90
E118E
E8 30
E18DD
80
E1A04
D0
E1BC5
27 10
E1C08
27 10
A1C45
JZ 1C74

E1C4D
27 10
E1EE4
1D
A1A31
CALL 30E0
NOP

A19EB
CALL 30F0
```

```
F1848,184B,90
E118E
E8 30
E18DD
80
E1A04
D0
E1EE7
20
A1A31
CALL 30E0
NOP

A19EB
CALL 30F0

A23A1
JMP 30A0

A30A0
CMP AH,3F
JZ 30B2
CMP AH,40
JZ 30BA
CMP AH,48
```

```
A239E
JMP 30A0

A30A0
CMP AH,3F
JZ 30B2
CMP AH,40
JZ 30BA
CMP AH,48
JZ 30C1
JMP 23DA
CMP BX,0
JNZ 30AF
JMP 23A8
CMP BX,1
JNZ 30AF
JMP 30B7
CMP BX,FFFF
JNZ 30AF
PUSHF
PUSH CS
MOV AX,30D0
PUSH AX
MOV AH,48
JMP 30AF
CMP BX,1000
JC 30DB
SUB BX,1000
STC
STI
RETF 2

A30E0
XOR DH,DH
MOV SI,DX
MOV AX,CX
RET

A30E8
MOV BYTE PTR [12E],0
JMP 11AE

A30F0
MOV ES,[10E]
JMP 1A31

W
Q
```

If your original ZPC3.COM is dated 9-18-85, use the lines from the first column below. If it is dated 9-19-85, use the lines in the second column. If it has a later date, go to the next paragraph.

```
F1849,184C,90
E118E
E8 30
E18E3
80
E1F04
3D
A1A4E
CALL 30E0
NOP

A19FF
CALL 30F0

A23BE
JMP 30A0

A30A0
CMP AH,3F
JZ 30B2
CMP AH,40
JZ 30BA
CMP AH,48
JZ 30C1
JMP 23FA
CMP BX,0
JZ 30C1
JMP 23DD
CMP BX,0
JNZ 30AF
JMP 23AB
CMP BX,1
JNZ 30AF
JMP 30B7
CMP BX,FFFF
JNZ 30AF
PUSHF
PUSH CS
MOV AX,30D0
PUSH AX
MOV AH,48
JMP 30AF
CMP BX,1000
JC 30DB
SUB BX,1000
STC
STI
RETF 2

A30E0
XOR DH,DH
MOV SI,DX
MOV AX,CX
RET

A30E8
MOV BYTE PTR [12E],0
JMP 11AE

A30F0
MOV ES,[10E]
JMP 1A31

W
Q
```

```
F1849,184C,90
E118E
E8 30
E18E3
80
E1F06
3F
A1A50
CALL 30E0
NOP

A23C0
JMP 30A0

A30A0
CMP AH,3F
JZ 30B2
CMP AH,40
JZ 30BA
CMP AH,48
JZ 30C1
JMP 23FC
CMP BX,0
JNZ 30AF
JMP 23CA
CMP BX,1
JNZ 30AF
JMP 23C8
CMP BX,1
JNZ 30AF
JMP 30B7
CMP BX,FFFF
JNZ 30AF
PUSHF
PUSH CS
MOV AX,30D0
PUSH AX
MOV AH,48
JMP 30AF
CMP BX,1000
JC 30DB
SUB BX,1000
STC
STI
RETF 2

A30E0
XOR DH,DH
MOV SI,DX
MOV AX,CX
RET

A30E8
MOV BYTE PTR [12E],0
JMP 11AE

A30F0
MOV ES,[10E]
JMP 1A4E

W
Q
```

If your ZPC3.COM file is dated 10-4-85, use the lines from the first column below to make your FIXZPC.DAT file. If you use MS-DOS version 3, and your ZPC disk contains a file called ZPC3A.COM that you are using for your ZPC.COM, use the lines from the second column below.

```
F184E.1851,90
W
Q
```

```
F1851,1854,90
W
Q
```

After you have placed FIXZPC.BAT and FIXZPC.DAT on your ZPC system disk or partition, copy DEBUG.COM (from your MS-DOS distribution disk) to it, log on to it, and enter

```
FIXZPC
```

at the system prompt, and hit RETURN. (**Note:** When you run a self installing patch like this, ZPC, if installed, must be in the Z-100 mode, and if you have the HUG FASTIO program, it must be disabled.) Your ZPC will be automatically patched, and brought up-to-date.

ZPC And MS-DOS Version 3

The ZPC3.COM file supplied on the ZPC distribution disk will not run properly under MS-DOS version 3. Late releases of ZPC contain a file called ZPC3A.COM, which is for use with MS-DOS version 3. If your disk does not have a ZPC3A.COM, you can patch ZPC3.COM so that it will work with version 3 of MS-DOS. First, create a file called DOS3.BAT, which contains this line:

```
DEBUG ZPC.COM <DOS3.DAT
```

Now create a file called DOS3.DAT. It should contain the lines in the first column below if your original ZPC3.COM file is dated 9-5-85. If the file is dated 9-9-85 or 9-11-85, use the lines from the second column. If the date is 9-18-85, use the lines from the third column. If the file has a later date, go to the next paragraph.

```
A1327
CALL 3090

E1363
A3
E1366
02 00
A1369
CALL 3098

A28FE
CMP AX,[10A]
JBE 2908
CS:
MOV AX,[10A]
ES:
MOV [2],AX
SUB AX,DX

A3090
MOV AX,CS
STOSW
MOV AL,90
STOSB
RET
NOP
MOV AX,C22B
STOSW
RET

E33B8
A3
E33C0
02 00
E33C8
2B
E33DA
02 0
E33E2
2B
W
Q
```

```
A1327
CALL 3090

E1363
A3
E1366
02 00
A1369
CALL 3098

A2901
CMP AX,[10A]
JBE 290B
CS
MOV AX,[10A]
ES:
MOV [2],AX
SUB AX,DX

A3090
MOV AX,CS
STOSW
MOV AL,90
STOSB
RET
NOP
MOV AX,C22B
STOSW
RET

E33B8
A3
E33C0
02 00
E33C8
2B
E33DA
02 0
E33E2
2B
W
Q
```

```
A1327
CALL 3098

E1363
A3
E1366
02 00
A1369
CALL 30F8

A291E
CMP AX,[10A]
JBE 2928
CS:
MOV AX,[10A]
ES:
MOV [2],AX
SUB AX,DX

A3098
MOV AX,CS
STOSW
MOV AL,90
STOSB
RET

A30F8
MOV AX,C22B
STOSW
RET

E33B8
A3
E33C0
02 00
E33C8
2B
E33DA
02 0
E33E2
2B
W
Q
```

If your ZPC3.COM file is dated 9-19-85, use the lines from the first column below. If it is dated 10-4-85, use the lines from the second column.

```
A1327
CALL 30F0

E1363
A3
E1366
02 00
A1369
CALL 30F8

A2920
CMP AX,[10A]
JBE 292A
CS:
MOV AX,[10A]
ES:
MOV [2],AX
SUB AX,DX

A30F0
MOV AX,CS
STOSW
MOV AL,90
STOSB
RET
NOP
MOV AX,C22B
STOSW
RET

E33B8
A3
E33C0
02 00
E33C8
2B
E33DA
02 0
E33E2
2B
W
Q
```

```
A132C
CALL 30F0

E1368
A3
E136B
02 00
A136E
CALL 30F8

A28FF
CMP AX,[10A]
JBE 2909
CS:
MOV AX,[10A]
ES:
MOV [2],AX
SUB AX,DX

A30F0
MOV AX,CS
STOSW
MOV AL,90
STOSB
RET
NOP
MOV AX,C22B
STOSW
RET

E33B8
A3
E33C0
02 00
E33C8
2B
E33DA
02 0
E33E2
2B
W
Q
```

After you have placed DOS3.BAT and DOS3.DAT on your ZPC system disk or partition, copy DEBUG.COM to it, log on to it, and enter

```
DOS3
```

at the system prompt, and hit RETURN. Your ZPC will now function properly under MS-DOS version 3.

Assembly Source Code Corrections

If you would like to update your ZPC source code to contain the corrections that the above patches perform, you will find them in the last ZPC update article (the December 1985 REMark). You will also need to make the following additional change.

Load the file ZPC.ASM into your editor, and locate the label WROLP:. A few lines down, locate the following code:

```
        IF      LEVEL1
        MOV     CURPOS,DX               ;UPDATE CURSOR
        ENDIF
        IF      LEVEL2 OR LEVEL3
        MOV     DS,BIOSRAM
        MOV     CURPOS,DX
        PUSH    CS
        POP     DS
        ENDIF
```

Remove these lines at this one location only.

PC Program Patches

The following patches will allow the programs listed at the beginning of this article to run under ZPC. You must have also performed the patches to ZPC itself, especially if your ZPC3.COM is dated before 10-4-85. Be sure to duplicate all of the disks provided with each software product, and use only a duplicate for the patches. Never patch an original distribution disk.

Enable Integrated Software

To patch Enable for use with ZPC, create a file called ENABPCH .BAT on a copy of the Enable Utility disk, which contains these lines:

```
REN ENABLE.EXE ENABLE.BIN
DEBUG ENABLE.BIN <ENABPCH.DAT
REN ENABLE.BIN ENABLE.EXE
```

Now create a file on the Utility disk called ENABPCH.DAT that contains these lines:

```
F4045,404E,90
F41A6,41B0,90
F41CF,41D8,90
F424E,4258,90
E485F
90
E4874
90
E487F
90
E4882
90
E48A4
```

```
90
E5041
0 0
E5044
B0
AF7A
MOV BX,FFFF
NOP

W
Q
```

Copy DEBUG.COM to the Enable Utility disk, log on to it, and enter

```
ENABPCH
```

at the system prompt, and hit RETURN. When the operation finishes, Enable will be ready to run under ZPC. When the program requires you to press a control-arrow key, press F0 followed by the 2, 4, 6, or 8 key. The separate arrow keys will not work with F0 to emulate control-arrow keys in the current version of ZPC. See your ZPC documentation for more information on the use of the 2, 4, 6, and 8 keys as arrow keys.

Multimate

To patch Multimate for use with ZPC, create a file called MMPCH .BAT on a copy of the Multimate System disk, which contains these lines:

```
REN WP.EXE WP.BIN
DEBUG WP.BIN <MMPCH.DAT
REN WP.BIN WP.EXE
```

Now create a file on the Multimate System disk called MMPCH .DAT that contains these lines:

```
FFB4C,FB50,90
FFB95,FB9D,90
FFC82,FC86,90
FFC8D,FC95,90
FFCB5,FCB9,90
FFCBF,FCC8,90
FFD29,FD32,90
EFD9C
90
EFDAF
90
W
Q
```

Copy DEBUG.COM to the Multimate System disk, log on to it, and enter

```
MMPCH
```

at the system prompt, and hit RETURN. When the operation finishes, Multimate will be ready to run under ZPC. See the instructions for Enable (above) on control-arrow keys.

Volkswriter Deluxe

To patch Volkswriter for use with ZPC, create a file called VXPCH .BAT on a copy of the Volkswriter disk, which contains these lines:

```
REN VX.EXE VX.BIN
DEBUG VX.BIN <VXPCH.DAT
REN VX.BIN VX.EXE
```

Now create a file on the Volkswriter disk called VXPCH.DAT that contains these lines:

```
F3DE0,3DEA,90
A3DF1
INT 10
MOV AL,20
INT 51
RET

A4330
JMP 3DF1

A43F1
POP BP
JMP 3DF1

W
Q
```

Copy DEBUG.COM to the Volkswriter disk, log on to it, and enter

```
VXPCH
```

at the system prompt, and hit RETURN. When the operation finishes, Volkswriter will be ready to run under ZPC. See the instructions for Enable (above) on control-arrow keys.

Edix

To patch Edix for use with ZPC, create a file called EDIXPCH.BAT on a copy of the Edix disk, which contains these lines:

```
REN EDIX.EXE EDIX.BIN
DEBUG EDIX.BIN <EDIXPCH.DAT
REN EDIX.BIN EDIX.EXE
```

Now create a file on the Edix disk called EDIXPCH.DAT that contains these lines:

```
FA5ED,A5F6,90
FA63D,A646,90
FB81C,B825,90
FB831,B83A,90
FB842,B84B,90
FB86D,B876,90
FB882,B88B,90
FB893,B89C,90
FB918,B921,90
FB926,B92F,90
FB981,B98A,90
FB992,B99B,90
EB9B9
90
W
Q
```

Copy DEBUG.COM to the Edix disk, log on to it, and enter

```
EDIXPCH
```

at the system prompt, and hit RETURN. When the operation finishes, Edix will be ready to run under ZPC. See the instructions for Enable (above) on control-arrow keys.

PC Palette

To patch PC Palette for use with ZPC, create a file called PALPCH .BAT on a copy of PC Palette disk no. 2, which contains these lines:

```
REN DRIVER.EXE DRIVER.BIN
REN EZDRAW.EXE EZDRAW.BIN
DEBUG <PALPCH.DAT
REN EZDRAW.BIN EZDRAW.EXE
REN DRIVER.BIN DRIVER.EXE
```

Now create a file on PC Palette disk no. 2 called PALPCH.DAT that contains these lines:

```
NLOGO.COM
L
E5192
90
E567B
90
E57BE
90
E57ED
90
```

```
E584B
90
E585E
90
W
NDRIVER.BIN
L
F6B9,6C8,90
E776
3
W
NEZDRAW.BIN
L
E201A
3
E3013
90
W
Q
```

Copy DEBUG.COM to PC Palette disk no 2, log on to it, and enter

```
PALPCH
```

at the system prompt, and hit RETURN. When the operation finishes, create a file called PALDPCH.BAT on a copy of PC Palette disk no. 3, which contains these lines:

```
REN EZSHOW.EXE EZSHOW.BIN
DEBUG <PALDPCH.DAT
REN EZSHOW.BIN EZSHOW.EXE
```

Now create a file on PC Palette disk no. 3 called PALDPCH.DAT that contains these lines:

```
NSHOWLOGO.COM
L
E3B78
90
E3E34
90
E3E63
90
E3EC1
90
E3ED4
90
W
NEZSHOW.BIN
L
E1CE5
3
E2CA3
90
E43F1
90
E44E1
90
E4456
90
E4467
90
W
Q
```

Copy DEBUG.COM to PC Palette disk no 3, log on to it, and enter

```
PALDPCH
```

at the system prompt, and hit RETURN. PC Palette will then be ready to run under ZPC. **Note:** The documentation on PC Palette disk no. 1 cannot be read using the G command, because the program that reads the documentation file will not run under ZPC. You can, however, copy the file PCPAL.DCA from that disk to a printer to produce printed documentation. You can also use the SEE program from HUG disk 885-3014-37 to view the file.

When you run PC Palette under ZPC, you can turn NUM LOCK on, and the keypad number keys can be used for fine cursor movement, while the separate arrow keys can be used for course cursor movement. To print pictures, load a Z-100 screen printing utility (patched as described in the ZPC documentation) before you run PC Palette. To start printing, move the arrow cursor to the PRT box in the command area and press ENTER or RETURN. After the picture has been printed, PC Palette will display a "Printer not ready" message. Ignore the message, and press ESC to get rid of it.

PC Palette is a software product in the IBM Personally Developed Software series. For a catalog, phone (800) IBM-PCSW.

Alternate Microsoft Word Patch

When Microsoft Word is patched with the patch provided with ZPC, it runs in the high resolution monochrome graphics mode. This allows it to display such attributes as bold print and italics on the screen, but it runs slow because ZPC is slow in the graphics modes. If you would like to patch Word to run in the text mode, create a file on your Word disk called MWMPCH.BAT file that contains this line:

```
DEBUG WORD.COM <MWMPCH.DAT
```

Now create a file on your Word disk called MWMPCH.DAT using the lines below. If you have a color capable system and a color monitor or a monochrome monitor that can clearly show blue letters on a black background, use the lines in the first column. If you have a monochrome system, use the lines in the second column.

```
E212A
90 90
E219E
03
E21A4
B8
W
Q
```

```
E212A
90 90
E219E
02
E21A4
B8
W
Q
```

Copy DEBUG.COM your Word disk, log on to the disk, and enter

```
MWMPCH
```

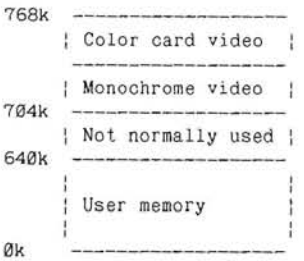
at the system prompt, and hit RETURN. If you used MW1PCH, underlining will show up as blue text, and other special attributes will show up as a faint underline under characters. Underlining and bold print together will show up as light blue (if intense color emulation is enabled in ZPC), which may be difficult to read on a monochrome monitor. The patch causes Word to think that it is using an IBM monochrome card, and the bit pattern for enabling underlining with that card is the same as for blue on the color/graphics card. Note: If you used MW2PCH, all characters will be white, with no indication of underlining. If you upgrade to color later, you can run MW1PCH even though you have already used MW2PCH.

ZPC Memory Usage

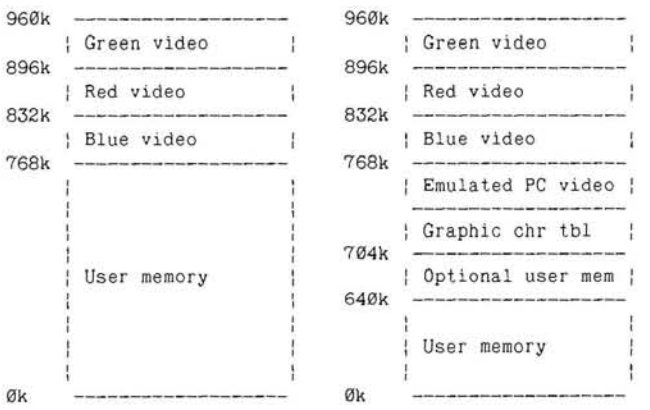
Many people have asked me to explain why ZPC requires 768k of memory in order to achieve maximum IBM PC compatibility. To help you understand, here is a chart of how memory is used in an IBM PC or compatible computer.

Notice that the first 640k of memory space in the PC is reserved for user memory. This is the memory used by your programs and the Disk Operating System (DOS). The actual user memory may not occupy all of this space, but if it does, the machine is said to be "fully populated". The segment of memory stretching from 640k to 704k is not normally used in a PC, although it is possible, by

replacing some chips, to use it in a Z-150. The segment from 704k to 768k is used for video cards. Each card has its own memory that can be written to or read from the same as user memory, but it is not used as space for programs. The PC hardware is set up so that, while the video card is in a text mode, you can write the ASCII code for a character into a location in the video card's memory and the character will appear on the screen. If the card is in a graphics mode, writing to the video card memory will produce pixel dots on the screen. There are other ways of getting characters or pixels on the screen, but the direct memory method is the fastest, so many PC programs use this method. If you want these programs to run under emulation on a Z-100, you must emulate a PC video card's memory.



The following charts describe the memory in a Z-100 computer. The chart on the left represents normal memory usage, and the one on the right represents how memory is used when ZPC (level 3) is in the PC emulation mode.



When ZPC is in the PC emulation mode, it emulates the color/graphics video card of a real PC. Level 3 of ZPC emulates the PC video card's memory, and as you can see on the chart above, the emulated memory is at the same location as the memory on a real video card. It must be there, because that is where a PC program expects it to be. That is why you must have 768k of memory in your Z-100 to run level 3 of ZPC. Notice that part of the memory reserved for a monochrome video card on a real PC is used for a graphic character table, which resides in ROM (Read Only Memory) on a real PC.

Some users have reported that they have 192k of memory on the motherboard in their Z-100s, and an additional 512k on memory cards. That leaves them 64k short of 768k, and they would like to know how to run ZPC3 under those conditions. Some have tried modifying ZPC to lower the position of the emulated video memory, but that is not the answer. Every PC program run under such a modified ZPC would also have to be modified so that it would write video to the new place. Another "fix" that users have tried, if their memory cards are from UCI, is to modify the cards so that the system contains 960K of user memory. This also does not work, because ZPC writes directly to the Z-100's video memory, and you cannot do that when the video memory is overlapped with user memory.

There are only two valid solutions to the 704k memory dilemma. One is to modify your motherboard to hold 256k chips, in which case you can populate it to 768k. The other is to purchase a 64k S-100 memory card and install it addressed to fill the missing 64k segment. A company called Digital Research Computers of Texas (not the same as the Digital Research that makes CP/M) is said to have suitable 64k memory cards at a reasonable price. You can usually find an ad for Digital Research Computers in BYTE and other computer magazines. *

Continued from Page 45

from within NAVPROGseven or the user can delete them and start over with his own data.

RNAVLIST.DAT — This file saves the fixes or checkpoints that have RNAV cross-bearings.

ROUTINGS.DAT — This file contains a sample index of routes on file. This file will be updated through the program.

BEH.LUK and LUK.BEH — These data files are sample routes on file.

Program Content: The title of NAVPROGseven comes from the features and functions designed into the system, many of which are not found in similar programs.

1. Easy input & revision of the airport/navaid database.
2. Two RNAV (area navigation) functions that return the latitude and longitude of a location based on cross-bearings from known points.
3. Aircraft performance data stored for each airplane you fly.
4. Easy access and display of airport and checkpoint information using standard ICAO identifiers as you plan your route of flight. Automatic flight planning selects navaids closest to your great circle route and prepares several alternate routings. Often flown routes can be saved for later use.
5. Great circle navigation between checkpoints using aircraft performance data, and printout of ready-to-use flight log.
6. Climb/descent profiles calculated based on aircraft performance data.
7. Multiple sort criteria to organize airport/navaid data into easy-to-read printouts.

The comprehensive flight log tells the field elevation of the departure and destination airports, navaid frequencies enroute, distances for each leg and total remaining, true and magnetic course, magnetic heading corrected for wind and magnetic compass variation, groundspeed corrected for wind and climb and/or cruise leg segments, ETE and ETA for each leg, fuel usage based on climb and/or cruise with startup/taxi/takeoff fuel accounted for, fuel remaining, and a warning if reserves will be less than IFR or VFR minimums.

In addition, a synopsis of the flight tells the fuel used, reserves in gallons and time, fuel/time/distance required to climb, how far out from the destination the user should begin his descent to maintain a 2 degree descent profile.

Master Menu: There are eight programs called by six user selectable items on the master menu. The master menu includes the following options:

* Input/Revise Airport and Navaid Data
* Input Aircraft Performance Data
* Automatic Route Preparation
* Air Navigation and Flight Planning
* Navigate Pre-Planned Route
* Sort & List Data on File

These options provide a complete capability for using the stored information to plan flights, prepare comprehensive flight logs for in-flight use, add new data to the database and to maintain the accuracy of the database.

Comments: NAVPROGseven was written for aircraft navigation but is not limited to pilots alone. The great-circle navigation and radio-beacon cross-bearings can be helpful to sailors who also have a need to navigate efficiently, though at a slower pace. The system is menu-driven and includes all necessary documentation for use. The programs are self-prompting with one-key responses and many safety checks that allow the user to go back to the menu and start over.

As mentioned above, the programs come with sample data that allows the pilot to 'test-fly' the system before creating his own database, and there are over 100 airports and navaids already on file that will get the pilot off to a running start.

TABLE C Rating: (10)

*

MEDIA MASTER

FORMATS SUPPORTED	VERSIONS	IBM PC	ZENITH Z-100	OSBORNE 1	OSBORNE 1 EXEC	KAYPRO 2/4	KAYPRO 2X/4/10	DEC RAINBOW	SANYO 550
Actrix (DSDD)		•	•				•		•
Actrix (SSDD)		•	•	•	•	•	•	•	•
Avatar TC10 (DSDD)		•	•				•		•
Casio FP 1000 (DSDD)		•	•				•		•
Chameleon CP/M-80		•	•				•		•
Columbia MPC CP/M-80		•	•				•		•
Cromemco CDOS (SSDD)		•	•	•	•	•	•	•	•
Cromemco CDOS (SSSD)				•	•	•	•		
Cromemco w/IntCl Term (DSDD)		•	•				•		•
Cromemco w/IntCl Term (SSDD)		•	•	•	•	•	•	•	•
DEC Rainbow CP/M-86/80								•	
DEC Rainbow MS-DOS								•	
DEC Mate II CP/M (SSDD)								•	
DEC VT180 (SSDD)		•	•	•	•	•	•	•	•
Davidge (DSDD)		•	•				•		•
Digilog (DSDD)		•	•				•		•
Epson Multifont (DSDD)		•	•				•		•
Epson QX-10 (DSDD)		•	•				•		•
Facit DTC (SSDD)								•	
Fujitsu Micro 16S (DSDD)		•	•				•		•
Groupil III CP/M (DSDD)		•	•				•		•
Heath w/Magnolia CP/M		•	•	•	•	•	•	•	•
Heath/Zenith Z-DOS (1.xx DSDD)		•	•		•		•		•
Heath/Zenith Z-DOS (1.xx SSDD)		•	•	•	•	•	•	•	•
Heath/Zenith Z-DOS (2.xx DSDD)		•	•		•		•		•
Heath/Zenith Z-DOS (2.xx SSDD)		•	•	•	•	•	•	•	•
Heath/Zenith Z-100 CP/M (DSDD)		•	•						•
Heath/Zenith Z-100 CP/M (SSDD)		•	•	•	•	•	•	•	•
Heath/Zenith Z90 40 track (1k block)		•	•	•	•	•	•	•	•
Heath/Zenith Z90 40 track (2k block)		•	•	•		•	•	•	•
IBM PC CP/M-86 (DSDD)		•	•				•		•
IBM PC CP/M-86 (SSDD)		•	•	•	•	•	•	•	•
IBM PC DOS (1.xx DSDD)		•	•		•		•		•
IBM PC DOS (1.xx SSDD)		•	•	•	•	•	•	•	•
IBM PC-DOS (2.xx DSDD)		•	•		•		•		•
IBM PC DOS (2.xx SSDD)		•	•	•	•	•	•	•	•
IDEA Bitflex (SSDD)		•	•	•		•	•	•	•
ISM CP/M (DSDD)		•	•				•		•
Ithaca (SSDD)								•	
Insight Dev IQ-120 (SSDD)		•	•	•		•	•	•	•
Kaypro 10 (DSDD)		•	•				•		•
Kaypro 2 (SSDD)		•	•	•	•	•	•	•	•
Kaypro 4 (DSDD) •		•	•				•		•
LNW-80 (SSDD)		•	•	•	•	•	•	•	•
Lobo MAX-80 (SSDD)		•	•	•	•	•	•	•	•
Lobo MAX-80 512 (SSDD)		•	•	•		•	•	•	•
Marral 9050 CP/M-80 (DSDD)		•	•				•		•
Morrow MD11 (DSDD)		•	•						•
Morrow MD2 (SSDD)		•	•	•	•	•	•	•	•
Morrow MD1 (DSDD)		•	•						•
MacSym 450 (SSDD)								•	
Monroe OC6820 (SSDD)								•	
NCR Decision Mate 5 (DSDD)		•	•				•		•
NEC PC-8001A (DSDD)		•	•				•		•
NEC PC-8001A (SSDD)		•	•	•	•	•	•	•	•
Olympia ETX II (SSDD)		•	•	•		•	•	•	•
Olympia EX100 (DSDD)		•	•				•		•
Osborne 1 (SSDD)		•	•	•	•	•	•	•	•
Osborne 1 (SSSD)				•	•	•	•		
Osborne 4 Vixen (DSDD)		•	•		•		•		•
Osborne Executive (SSDD)		•	•	•	•	•	•	•	•
Osborne Osmosis (SSDD)		•	•	•		•	•	•	•
Otrona (DSDD)		•	•				•		•
Philips 3000 (SSDD)								•	
PMC Micromate (DSDD)		•	•				•		•
Reynolds & Reynolds (SSDD)		•	•	•		•	•	•	•
Sanyo CP/M (DSDD)		•	•				•		•
Superbrain (DSDD)		•	•				•		•
Superbrain Jr. (SSDD)		•	•	•		•	•	•	•
Sweet II (SSDD)		•	•	•	•	•	•	•	•
Sweet III (DSDD)		•	•				•		•
S-100 40 TRK (SSDD)				•		•	•	•	•
TI Professional CP/M-86 (DSDD)		•	•				•		•
TI Professional CP/M-86 (SSDD)		•	•	•	•	•	•	•	•
TRS-80 Model III FEC CP/M (SSDD)		•	•	•		•	•	•	•
TRS-80 Model III FEC T805 (SSDD)		•	•	•		•	•	•	•
TRS-80 Model III Burr Labs (SSDD)		•	•	•		•	•	•	•
TRS-80 Model III Mem. Merch. (SSDD)		•	•	•	•	•	•	•	•
TRS-80 Model I Omikron CP/M (SSSD)				•	•	•	•		
TRS-80 Model IV CP/M Plus (SSDD)		•	•	•	•	•	•	•	•
TRS-80 Model IV Montezuma Micro (SSDD)		•	•	•		•	•	•	•
Teletek 40 track (SSDD)		•	•	•		•	•	•	•
Teletek 80 track (SSDD)								•	
Toshiba T100 (DSDD)		•	•				•		•
Televideo TurboDOS (DSDD)		•	•				•		•
Televideo 802 CP/M							•		•
Wang Mawx CP/M (DSDD)		•	•				•		•
Xerox 820 II (SSDD)		•	•	•	•	•	•	•	•
Xerox 820 II (SSSD)				•	•	•	•		
Zorba 10 track (DSDD) •		•	•				•		•

• *Requires access to "foreign" computer*

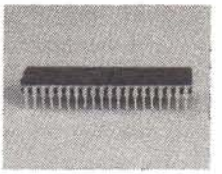
"MAKE YOUR Z-150 CP/M COMPATIBLE"

Intersecting Concepts Announces 3 Solutions To Solve Your Computer Incompatibility

But will it work on my computer? Yes! Finally, there are three easy ways to exchange information, transfer files, and run CP/M software on MS-DOS machines.

1. MEDIA MASTER is our direct disk-to-disk format conversion program. Already an accepted industry standard, this $39.95 program uses simple screen prompts that lets you read, write and format up to 70 different 5¼" diskettes from CP/M, MS-DOS and PC-DOS operating systems. So if you work on a IBM PCompatible at the office, but use a CP/M computer at home, now you can easily transfer files that would otherwise be "foreign" to your computer's operating system.

2. MEDIA MASTER PLUS goes one step further by converting 8-bit CP/M software to run on 16-bit MS-DOS and PC-DOS machines. This newly released $59.95 product combines our IBM-PC version of Media Master with ZP/EM, a powerful new emulation program. The results are amazing: CP/M programs using 8080 instructions *and* data can be transfered from popular computers like Osborne, Kaypro and Zenith to run on MS-DOS and PC-DOS machines!



3. ACCELER-8/16 is also new and dramatically improves the performance of Media Master Plus by tailoring the CP/M emulation around a NEC V20 microchip. This chip simply replaces the 8088 processor in your MS-DOS computer. Once installed, it'll run your CP/M and MS-DOS software much faster. (Speed improvements are roughly 15% faster in MS-DOS and 350% faster in CP/M!) With ACCELER-8/16 your MS-DOS computer is now CP/M compatible for only $99.95!

The Choice Is Yours

So if you just need to exchange files between different computers using different operating systems, there's Media Master at $39.95.

If you're upgrading to an MS-DOS or PC-DOS machine or already have one but don't want to spend money buying "new" software programs you already own, there's Media Master Plus for $59.95.

And if the execution speed of your CP/M programs is important, there's ACCELER-8/16 for $99.95, which provides a simple hardware and software solution to making your MS-DOS computer CP/M compatible. All three solutions save you money by eliminating expensive modems and communications software.

To Order

To order Media Master, Media Master Plus, or ACCELER-8/16, call 800-824-7888, and ask for operator 251. For additional product and upgrade information contact:

Intersecting Concepts, Inc.,
4573 Heatherglen Ct., Moorpark, CA 93021
or call 805-529-5073.

INTERSECTING CONCEPTS

Squeezing The Works

A Review Of The Heath HS–158 Expandable Personal Computer

Pat Swayne
HUG Software Engineer

The HS–158 Expandable PC is a successor to Heath's first IBM PC compatible computer, the H–150. Externally, the HS–158 looks just like a '150, but inside it has been completely redone. In both machines, the "works" are on removable printed circuit cards that plug into a passive backplane. The circuitry of a minimally configured '150 occupied 4 of these cards, but in the HS–158, everything has been squeezed onto only two cards. One of the cards contains the CPU (Central Processing Unit) and its supporting circuitry, all of the computer's memory, and a parallel port. The other card contains the video circuitry, the disk controller circuitry, and a serial port.

The "magic" that makes it possible to put everything on two cards is the use of gate array logic. This is a new technology that makes it possible to concentrate many different circuit functions on a single integrated circuit. Each of the two cards in the HS–158 contains a proprietary gate array IC that performs the functions of several IC's on the old '150 cards. These new IC's are easily recognizable by their square (rather than rectangular) shape, and the fact that they have many more connections than the other IC's. If you would like to study the functions of these IC's in detail, service modules are provided with the HS–158 kit that explain the functions and provide complete pin-outs, with a description of the signal on each pin.

Another difference between the '158 and the '150 is the addition of a switch that allows you to select the operating speed of the central processor. You can select 4.77 MHz, which is the speed at which an IBM PC runs, or 8 MHz. Most programs you run will work faster at the higher speed, but you may want to use the slower speed with certain action games. The game will run faster (and your score will probably be lower) at 8 MHz.

Building The HS–158

Those of you who built H–150 computers know that there was quite a bit of work to it. You had to build the memory board, the disk controller board, and the backplane. In the new HS–158, the only board you have to build is the backplane. There is quite a bit of soldering to do on that board, but it is simple electronically, so the chances of making a mistake are small.

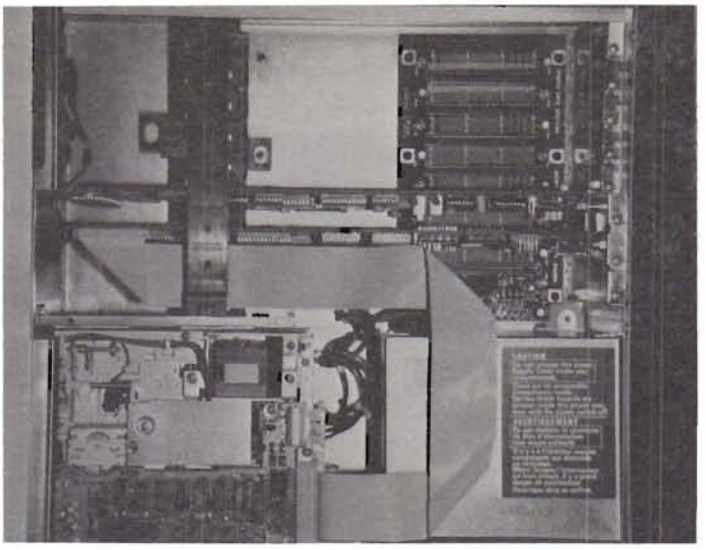


Photo 1
The interior of the HS–158. Notice the bars used to hold the cards in place, and the disk drive cable placement.

In addition to the electronic differences between the '150 and the '158, there are a few mechanical differences also. The front panel bezel is attached differently, and you no longer have to melt brass inserts into the plastic with a soldering iron (a risky procedure). The boards are also held in place differently, with plastic bars crossing the boards at the top and bottom (see Photo 1) instead of slotted brackets at the end. The two system boards are too long to allow for the end brackets anyway, and if you

thought it was difficult to insert or remove the longer system boards from the '150 without loosening some hardware, you'll find it impossible on the '158. It seems to me that some appropriately placed cuts in the sheet metal at the chassis back would make system board replacement much easier.

Construction of the HS-158 went smoothly without any major problems, and it worked the first time I turned it on. However, I did have to refold the disk drive cable to make it take a neat path from the drives to the controller. In Photo 1, you can see how I folded the cable.

Using The HS-158

Both the H-150 and the HS-158 have DIP switches on their CPU boards that are used to configure certain operating parameters. The '150 had two 8-position switches, but the '158 has only a single 4-position switch. Some of the parameters previously determined by switch settings are now determined automatically. For example, there is no longer a switch to indicate the presence or absence of an 8087 numeric processor, nor are there switches indicating the number of disk drives. The computer tests for the presence of an 8087 and the number of drives when you turn it on. That means that if you have any external drives, the power to them must be turned on before you turn on the computer's power.

To cut down on the number of DIP switches (I suppose), some of the options available on the '150 were simply eliminated on the '158. For example, you cannot defeat auto-boot, and you cannot configure the computer to start up in the 40 column text mode. You can still boot manually, though, if you just wait until the drive light on drive A comes on for the second time after power up, and then press the Escape key or Ctrl-Alt-Ins.

The CPU board does not have switches for indicating the amount of memory in the machine (that is done automatically), but it does have two plug-type jumpers for selecting whether your chips are the 64k type or the 256k type. By correctly setting the jumpers, you can mix the two types on the board. The computer comes with one bank of 256k chips (for a total of 256k of user memory). There are two empty banks on the board that can be filled with either chip type, but the computer will only "see" a maximum of 640k of user memory, even if you put in 3 banks of 256k chips. This is because the IBM PC standard dictates that 640k is the maximum amount of user memory you can have (exclusive of video memory or banked memory).

Because the HS-158 concentrates all of its functions on two cards, six of the eight slots on the backplane are available for expansion. One of the unused slots, however, was originally designed to hold an oversized Heath/Zenith card (part of the H-150 card set), and you might have difficulty putting a standard IBM-sized card into that slot without modifying the mounting bracket.

The monitor ROM (the built-in programming that controls the computer when you first turn it on) in the HS-158 is a new version of MFM-150. The version in my '158 (version 2.3B) appears to have a "bug" or two in it. For example, if you type an "x" followed by Return immediately after entering MFM-150 (Ctrl-Alt-Ins), the machine will "lock up". You can only restore operation by turning it off and back on. If you allow the the auto-boot procedure to fail by turning the computer on but not placing a disk in the drive, an error message will appear on the screen. At this point, you should be able to press the Escape key or Ctrl-Alt-Ins and then boot a disk by pressing B and Return. If you do, however, you will get the "drive not ready" error message again, and you must press Escape or Ctrl-Alt-Ins and B a second time.

In spite of the apparent problems in the monitor ROM in the '158, it seems to perform flawlessly once you boot up and run applications. The 8 MHz operating speed makes everything faster, especially screen updating, so you'll find screen intensive programs such as word processors a joy to use.

Photo 2
After I completed the HS-158, my work was inspected by an expert.

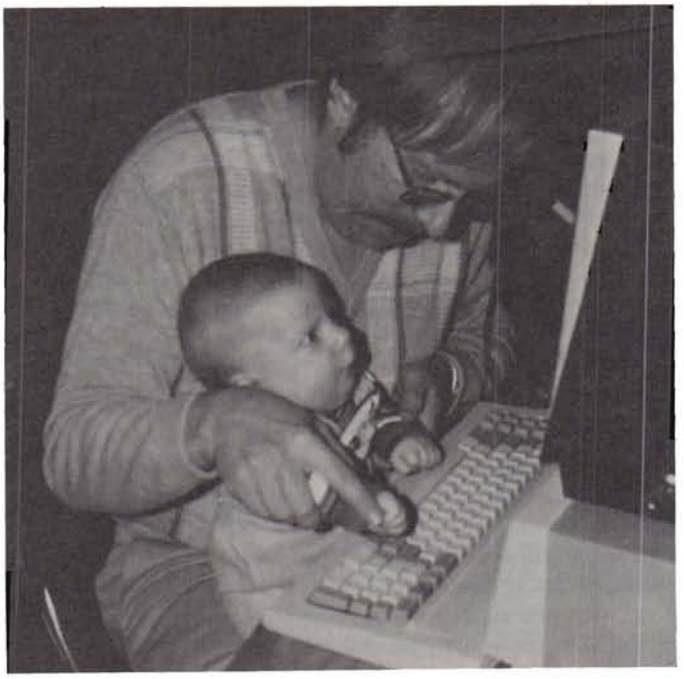


Photo 3
An expert tests software on the completed HS-158.

*

ZCLK
An Update To
The February Article

Richard L. Mueller, Ph.D.
11890-65th Avenue N.
Maple Grove, MN 55369

Just finished reading the article in the February Issue of REMark by Larry Wier on the ZCLK Calendar/Clock Module for the H/Z-100. The ZCLK Module was developed by FBE Research Company, Inc. of Seattle, Washington. The article, which is excellent on what I feel is a super product, reviews and describes the hardware in detail and touches somewhat on the supporting software.

Although considerable information was presented in Larry's article, there is, however, much more to be added in the area of supporting software and documentation. In this article, I would like to cover these two areas, mainly the supporting software and discuss my own experience with the ZCLK Calendar/Clock Module. I will not discuss the hardware itself; Larry did a fine job in covering that in his article.

First, let me discuss my experiences with the ZCLK Calendar/Clock Module. I originally purchased a ZCLK Module back in mid December and tried to install it into my H-100 shortly thereafter. However, I ran into a big problem with the Module itself. My H-100 (low profile model with 2 disk drives) is one of the first H-100's that were sold here in the Minneapolis area over two years ago. The disk drives seem to be much larger than in later models of the H/Z-100.

After inserting the ZCLK Module into the main board of my H-100 as per the installation instructions, I tried to reinstall the disk drives. Unfortunately, the height of the ZCLK Module was much too high. The lower back end of the left disk drive was resting on the top of the ZCLK Module Board components. The frame that houses the two disk drives would not seat properly on top of the main computer frame and, therefore, I could not fasten the frame. Somewhere between an eighth of an inch and a quarter of an inch is what I needed to properly seat the disk drive frame; the ZCLK Module was "too high" or simply just "in the way". This prevented me from completing the installation.

Even though the disk drive frame would not seat properly, I was anxious to try the ZCLK Calendar/Clock Module. So I left the top cover of the computer off, raised the disk drive frame a little to allow the left disk drive to clear the ZCLK Module, and finally I installed the ZDOS/MS-DOS version of the ZCLK software. I tried the three function requests (more on this shortly) and everything worked just as advertised. Just a super job.

Although the ZCLK Module worked, I could not leave my H-100 in this situation: top cover of the computer off and the disk drive frame not fastened down. I called Dave Brockman of FBE Research and discussed my situation with him. Dave came to my rescue. He had me send the ZCLK Module back so he could look at making some modifications. After about a week or so, Dave sent me a modified ZCLK Module which was lowered at least an eighth of an inch. A couple of weeks later, I was able to try again with the installation of the ZCLK Module. This time the disk drive frame rested on the main computer frame just fine. However, the lower back end of the left drive still was touching the ZCLK Module, though just barely.

After discussing this situation with Dave Brockman, he saw no problems with this. I have not experienced any problems running this way for the past several weeks. The ZCLK Calendar/Clock Module is just "humming" away. So far, I am extremely pleased with this product and would highly recommend it to all H/Z-100 users in need of a calendar/clock capability.

Larry's article talks about ZDOS/MS-DOS, so let me say a few words on this. First of all, the ZCLK software, included as part of the overall product, has three function requests: set the date and time in the ZCLK Module, read and display the date, day-of-week, and time, and process a Standard Time/Daylight Savings Time adjustment.

First let me talk about setting the date and time. Once you have the ZCLK Module installed, as well as the ZCLK supporting software, you can proceed to set the date, day-of-week, and time. This function can be done at any time and it is done via the H/Z-100 console. The user types in "ZCLK" followed by 10 digits (spaces can be inserted for readability purposes and leading zeros are required): ZCLK MM DD YY HH MM (RETURN). The current version of the ZCLK software now requests the user to enter a value of 1-7 followed by a RETURN for Sunday through Saturday. Once this has been done, then the ZCLK software waits until the user types another RETURN to enter the date, day-of-week, and time into the ZCLK Module. This delay is done so the user can synchronize the time entered with his/her watch.

Next, the ZCLK software reads back the date, day-of-week, and time and performs two tasks. The first task is to store the date and time into the ZDOS/MS-DOS system. This is the reason why I

purchased the ZCLK Module and why so many other users purchase calendar/clock products for their computers: to eliminate the task of entering the date and time every time a coldstart boot is performed. The storing of the date and time into the operating system is performed via the ZDOS/MS-DOS functions DOSF_SDATE and DOSF_STIME. Following this, the ZCLK software performs the second task and that is displaying the date, day-of-week, and time on the console CRT, then exits.

While using the system, the user can request the date and time by simply using the ZDOS/MS-DOS commands DATE and TIME, or via the ZCLK software by just typing in ZCLK followed by a RETURN. In the latter case, the user only has to type in one command to get both the date and time plus the day-of-week. This is the second function request of the ZCLK software: reading the ZCLK Module, setting the date and time into the operating system, and finally displaying this information on the console CRT.

Using the AUTOEXEC.BAT feature of ZDOS/MS-DOS, the user can have the system perform the second function above automatically each time a coldstart boot is performed. All that is necessary is to enter "ZCLK" into a file with the name of AUTOEXEC.BAT and put this file, along with the ZCLK software file, onto each bootable disk. Now, every time you power up your system and boot a disk, the ZCLK Module will set the correct date and time into the operating system and display this information to the user. This was mentioned briefly in Larry's article.

The third ZCLK software function is called the "Daylight Savings Time" adjustment function. What this basically does is either increments the current "hours" by one or decrements the current hours by one depending on the time of year. Here is how this function works. First, the user enters "ZCLK" followed by a "D" (upper or lower case) and RETURN. Upon recognizing this, the ZCLK software reads the Calendar/Clock Module, gets the current month and hours. If the current month is before July, the ZCLK software assumes that this is a request to set the time for Daylight Savings Time (i.e., the "clock" is set ahead one hour — the current hour is incremented by one). Please use this with caution since each time you request this function the hours will either be incremented or decremented by one.

If the current month is July or later, then the ZCLK software assumes this to be a request to set the time back to Standard Time (i.e., the "clock" is set back one hour — the current hour is decremented by one). This particular request provides a quick way of setting the time to either Daylight Savings Time or Standard Time. The time can also be changed by the first ZCLK function request which is to set date, day-of-week, and time.

Larry's article, as I stated before, just mentions ZDOS/MS-DOS. The distribution disk also contains the ZCLK software for the CP/M-86 Operating System. The instructions for installing this ZCLK software on CP/M-86 are included in file READ.ME which is on the same disk.

The CP/M-86 ZCLK software works very similar to the ZDOS/MS-DOS version. From the external user's point of view, the displays and the three function requests that I discussed earlier are identical. The difference is in the internals of how the date/time get set in the CP/M-86 Operating System. Remember I stated that the ZDOS/MS-DOS version sets the date and time via operating system functions. There are no equivalent system functions within CP/M-86. However, there is a System Data Area (SDA) in the CP/M-86 system, that contains the date and time among other important information. CP/M-86 maintains (updates) the date and time in this area.

The initial release of CP/M-86 and the related manuals did not discuss this System Data Area or if there was such an area, there was no way of getting to it. However, when I received my copy of CP/M-86 from Zenith about a year and a half ago, there were several "enhancement sheets" included. These sheets described the System Data Area and where within this area the date and time were stored (date is stored beginning in base address of the System Data Area plus 32, and the time is stored beginning in base address plus 41. The format of both the date and time were also described.

To get access to this area, a new CP/M-86 function was introduced (49) which returns to the caller the address of the System Data Area (ES = segment, BX = offset). The enhancement sheets also describe this new function. The CP/M-86 ZCLK software sets the date and time in the CP/M-86 system by simply writing the information directly into the System Data Area in the proper format.

Once the date and time have been set, using the same method as was described for the ZDOS/MS-DOS version, the user can verify that the Date and Time have been set correctly by using the CP/M-86 command TOD. TOD by itself will display the current contents of the date and time bytes in the System Data Area.

CP/M-86 does not have an AUTOEXEC type capability, however, there is a way that a single command can be executed each time a coldstart boot is performed. This single command can be either a standard CP/M-86 command, an application like Wordstar, or a user executable file such as ZCLK. See the CONFIGUR command in the CP/M-86 Reference Manual for details on how to set the single command to be executed each time a coldstart boot is performed.

By entering ZCLK as this single command, CP/M-86 will execute the ZCLK software each time a coldstart boot is done. This will cause the ZCLK Calendar/Clock Module to be read, the date and time will be written into the System Data Area, and the current date, day-of-week, and time will be displayed for the user. Again, the user can verify that the Date and Time were set correctly by the CP/M-86 command TOD. CP/M-86 programs/applications can now have access to the current date and time via system function 49.

I installed the CP/M-86 ZCLK software on my H-100 as per the instructions and everything seems to be just fine. In addition, I did set the single command on coldstart to ZCLK. In this way, I do not have to be bothered with entering the date and time via the TOD command each time I boot up the system.

If the ZDOS/MS-DOS and CP/M-86 versions of the ZCLK software are not enough, Dave Brockman has a version that runs on the CP/M-85 Operating System. Although CP/M-85 does not support system functions to set date and time as does ZDOS/MS-DOS or does not have a System Data Area as does CP/M-86, the ZCLK software is, nevertheless, still useful. Just having the capability of requesting the date and time while running CP/M-85 is handy.

As with CP/M-86, CP/M-85 also has the single command execution capability for coldstart boots. Refer to the CP/M-85 Reference Manual for more detail on the CONFIGUR command. The command should be just the name of the file containing the ZCLK software without the file extension (e.g., file name ZCLK).

The CP/M-85 disk is available from FBE Research for $5 as stated in the documentation that comes with the ZCLK Calendar/Clock Module. Since I occasionally use CP/M-85 on my H-100, I decided to install this ZCLK software on my CP/M-85 bootable disks. I also set the single command execution capability for coldstart boots to ZCLK. In this way, it lets me see the date and time automatically each time I perform a coldstart boot. In addition to seeing the date and time at boot time, I can request the date and time at any time just by calling the CP/M-85 ZCLK file name.

The last thing that I would like to cover is documentation. Just as with the ZCLK Calendar/Clock Module itself and with the three versions of the ZCLK software, the documentation is well done. The first few pages describe how one disassembles the H/Z-100 (low profile and all-in-one models), installs the ZCLK Module Board, and reassembles the H/Z-100. Next, the documentation covers the installation of the ZDOS/MS-DOS version of the ZCLK software. Remember, the installation instructions for the CP/M-86 version are on the READ.ME file.

The next big section (which runs for a few pages) is on programming the calendar/clock chip. When I received my first ZCLK module, the documentation did not include the programming section. The current documentation does, however, have the programming section which covers the calendar/clock chip (MSM5832 IC) and the 68A21 Peripheral Interface Adaptor (PIA). The PIA is used for the parallel printer interface and also used by ZCLK. The information is there for those who want to write and read the ZCLK module themselves.

For ZDOS/MS-DOS and CP/M-86 users, I can't see why they would want to write/read the ZCLK Module directly. The ZCLK software takes care of setting the time in areas maintained by these two operating systems, which in turn provide the users with means to access the current date and time very easily. However, that's a different story with CP/M-85. Since CP/M-85 does not provide a date and time capability, it would be nice to be able to read the current date and time by user programs. Dave provides the programming information that the user needs to accomplish this.

In conclusion, I want to say that "I'm sold on ZCLK" and definitely would recommend this calendar/clock product to any H/Z-100 user. Although there is some inconvenience when one has to install ZCLK, adjust it if necessary, and change batteries; the benefits definitely outweigh these infrequent inconveniences. *

If you have moved or are moving, please let us know as soon as possible. We need your new address if you are to receive your REMark magazine. Bulk mail is not forwarded!

Integer Arithmetic In MBASIC

Kenneth Mortimer PE
352 Green Acres Drive
Valparaiso, IN 46383

WARNING

BASIC has sometimes been referred to as the "Tower of Babel" of the computer languages. Originated in 1963 by Kemeny and Kurtz at Dartmouth as "Beginners All-purpose Symbolic Instruction Code" as a simple introductory programming language to introduce students to a new GE time sharing system, the language has grown as "bells and whistles" have been added by successive implementers until it is now a system of noncompatible dialects. This tutorial was originally developed on a H-89 using the HDOS version of MBASIC (ver 4.82 09-Oct-80). When it was tested on the CP/M version (ver 5.21 25-Nov-80), the author found that several portions of the tutorial had to be completely rewritten because of differences in the two languages both called BASIC-80. Additional investigation has led the author to assume that the two 16 bit versions of Microsoft BASIC (ZBASIC and GWBASIC) seem to conform with the CP/M version. One should, therefore, test each command explained in the tutorial to be certain how it is implemented on the particular BASIC that is loaded in his or her machine. The variations are mostly in the use of the replacement operation and the use of the integer mathematical operations when real numbers are used. This means that programs that use these operations may not be portable.

What is an integer number and what is integer arithmetic? These questions are often raised when the term integer is first noticed in computer literature. Remember way back in grade school when you were first introduced to arithmetic? You did not know anything about decimal points. You were introduced to the numbers which were whole numbers, such as 1, 2, 199, etc. You learned to add, subtract and multiply them with no trouble. When you learned to divide, you often came up with a remainder which you probably expressed as a fraction. Later, you were introduced to the decimal point and to "real numbers". Real numbers were the set of numbers that contained the decimal point and the fractional portion. When you were introduced to the computer, you learned that real numbers were broken down into two classes. The first class was called a "decimal" number and was the common form of the real number with which you were familiar. The second class was the "floating point" number which you soon recognized as "scientific notation" written on a single line.

In strongly typed computer languages, a great deal of care is taken in both defining a variable or constant as real or integer, and in the use of real or integer numbers as operands. ASCII BASIC (the standard minimum BASIC) makes no differentiation between reals and integers with the exception of introducing the INT function. BHBASIC follows this practice. Many of the more sophisticated dialects of BASIC, including MBASIC, introduce the integer constant and variable, but they are much looser in their restrictions in their use than are the more strongly typed languages, such as FORTRAN or Pascal.

Why Use Integers

There are three important reasons for using integers in the more strongly typed languages. Only one of these reasons holds in MBASIC, however, since the forgiving nature of BASIC takes care of other problems that may exist. They are:

1. An integer number requires less memory for storage than a real number. In MBASIC an integer is stored as two bytes. Since one must account for the sign of the number, 15 bits are left to store the value of the number. This makes the maximum positive value of 2^15-1 or 32767. Single precision real numbers are stored in floating form and require four bytes for storage.
2. Integer arithmetic is much faster than floating point arithmetic. However, most BASICs do all of their arithmetic, except integer division, in floating point form. This also includes many of the larger 32 and 64 bit machines.
3. Integers are required as the indices to subscripted variables and as pointers. BASIC, written to be user friendly, converts

the real number into an integer when an attempt is made to use a real number as an index or pointer. This can sometimes cause a problem since, as we will see later, there is no consistent method of converting real numbers to integers. This is one of the reasons that the proponents of the more strongly typed languages do not like BASIC.

There are, however, many programming situations where it is convenient to use integers, therefore, we should become familiar with their use.

Typing A Variable As Integer

There are two methods that may be used to type a variable as an integral. You are all familiar with the use of the dollar sign ($) to indentify a variable as a string type. Similarly, you can use the percent sign (%) to indentify a variable as an integer type. Thus "I%" is an integer variable while "I" is a real variable. This is the most common method of typing a variable as integral in BASIC. A second method of typing that is used in MBASIC is the DEFINT function. By writing the statement

```
DEFINT I,J,K,L,M,N
```

the variables I, J, K, L, M and N are typed as integers.

Integer Functions And Operators

There are three integer functions and two integer operators in MBASIC. We will test each later in this tutorial. They are:

1. The FIX Function: The function "FIX(real)" truncates or cuts off all of the variable to the right of the decimal point and returns only the integer of the real variable.

2. The INT Function: The function "INT(real)" returns the next integer that is equal or less than the value of the real variable. It is, therefore, obvious that for a positive number the function INT will merely truncate the decimal portion of the number and INT(3.14159) is 3.

   However, the mathematical definition of "less than" means that for a negative number the following happens:

   INT(-3.14159) is -4

   INT is a part of the ASCII or core BASIC, however, not all BASICs follow the official rule. BHBASIC treats the INT subroutine as the FIX subroutine described above.

3. The CINT Or Round Function: The function "CINT(real)" rounds the returns the integer that is nearest to the value of the real variable. Therefore,

   CINT(3.14159) is 3,
   CINT(3.88888) is 4
   CINT(-3.14) is -3. and by definition
   CINT(3.5) is 4

4. Integer Division: The backslash (\) [ASCII 92] is the symbol used for integer division. When performed with two integers as operands, the integer division operator acts as one might expect and returns only the integer portion or the result of the division. Since BASIC is not a strongly typed language, we must make a provision in the definition of the operator to take care of the case when the integer division operator is used with one or both of the operands being real. The formal definition of the operator in the HDOS version is:

   A\B = FIX(INT(A)/INT(B))

   This can result is some peculiar results when negative numbers are used, as we will see later.

   The formal division of integer division for the CPM and the 16 bit versions is:

   A\B = FIX(CINT(A)/CINT(B))

   Note that this may result in entirely different results from that obtained with the HDOS version for both positive and negative real arguments. However, if the arguments are integers, then the results will be the same, whichever version of the language is used. Integer division is placed after the multiplication and division operators in the hierarchy of operations.

5. The Modulus Operator: The modulus operator MOD returns the "remainder" of a division operation. Since integer division is defined differently in the two versions one would expect to get different results when using this operation on real arguments, depending upon which version is used. Its place in the hierarchy of operations is after integer division and before addition and subtraction.

Testing The Integer Functions And Operators

This tutorial is based on four versions of Microsoft BASIC. Since BASIC has been referred to as the "Tower of Babel" of the computer languages and there are no two BASIC interpreters that are exactly alike. You should now "fire up" your own computer, call your BASIC interpreter and either learn more about what I have been writing, if you are using MBASIC, or else how your particular dialect of BASIC handles these procedures and what modifications you will have to make to use the concepts. We will write some simple BASIC programs that will test each of the principles listed above.

The output routines in BASIC do not differentiate between real and integer numbers. One cannot tell from the printed number 4 if BASIC means that the output is the integer 4 or the real number 4.000. Even when one goes to "PRINT USING" one gets no additional help.

The Fix Function

We said that the FIX(arg) function truncated the argument by cutting off all of the number to the right of the decimal point. Enter the following program into your computer:

```
100 INPUT "A";A
110 IF A=0 THEN END
200 D=FIX(A)
210 PRINT "A = ",A," D = ",D
220 GOTO 100
```

Now RUN the program with the entries for A as shown in Table One. Do the results agree with mine?

Table One
D=FIX(A)

A	D
2.23	2
-2.23	-2
5	5
-5	-5

That was an easy one. The results are the same for all versions. Note that the entry of 0.0 causes an escape from the program and a return to BASIC.

The INT Function

Mathematicians define the integer of a number as the next whole number that is equal to or less than the number. This

means that INT(2.23) is obviously 2. It, however, also means that INT(-2.23) is -3. To test this function replace line 200 in your program with

```
200 D=INT(A)
```

Now RUN with the entries for A as show in Table Two.

```
Table Two
 D=INT(A)
  A       D
 2.23     2
-2.23    -3
 7.71     7
-7.71    -8
-1       -1
```

Note the difference between FIX and INT when a negative number is used. Note again that this function is the same for either version.

The Round [CINT] Function

The third method of converting reals to integers is the CINT function. To test its operation replace line 200 in your program with

```
200 D=CINT(A)
```

Now RUN the program with the entries from Table Three.

```
Table Three
  D=CINT(A)
  A       D
 2.23     2
-2.23    -2
 7 71     8
-7 71    -8
 2.50     3
```

Note the difference between CINT and the previous two conversion methods. Again, this function should return the same values regardless of which version is used.

The Replacement Or Assignment Operator

The next question that one wants to ask is what happens when one uses the replacement operator (=) to store a real number as an integer. Replace lines 200 and 210 with

```
200 I%=A
210 PRINT "A = ",A,"I% = ",I%
```

Now RUN the program using the values shown in Table One, Two and Three. If you are using the HDOS version of MBASIC, you will find that using I% = A is the same as using I% = INT(A). If you are using the CP/M or one of the 16 bit versions of BASIC, you will find that using I% = A is the same as using I% = CINT(A). **Note:** it is here that we find a major variation between the two versions.

Input Operations

In order to see what happens when when one tries to enter a real number when the input variable is an integer, store our little program and enter the following

```
400 INPUT "I%";I%
410 IF I%=0 THEN END
420 PRINT I%
```

RUN this program using the values of A from Tables One, Two and Three. When you examine the results of these operations, you should discover that the interpreter will perform a similar operation on the contents of the input buffer as it did when performing the replacement operation. Again, there are two different ways that the operation is performed.

Integer Division

In order to test the integer division operator, let us call back the program that we stored and add the line:

```
120 INPUT "B";B
```

and replace lines 200 and 210 as follows:

```
200 C=A\B
210 PRINT "C",C
```

When we run the program with integer values of the input variables A and B, the results are what one would logically expect. The results are as shown in the first two lines in Table Four. But what happens when we use the integer division operator on real numbers? The third line in Table Four shows that nothing very startling happens. The INT(8.1)=8;INT(2.1)=2; 8/2=4 and the fix of 4 is still 4, if we are using the HDOS version. The use of the CINT function in the CP/M or 16 bit versions gives the same results. Now let us go to the fourth line of Table Four. If we are using the HDOS version, INT(8.1) is still 8, but INT(-2.1) is equal to -3. Then 8/(-3) is equal to -2.66 and FIX(-2.66) is equal to -2. We have shown that 8.1\2.1 is not equal to the negative of 8.1\(-2.1) as all of our experience in arithmetic would have led us to believe. If we are using one of the other versions of MBASIC CINT(8.1) is still 8, but CINT(-2.1) is equal to -2, then 8/(-2) is equal to -4. The two versions have given us different results. Now try the other lines in Table Four and see if you can predict what will happen when you use this operator.

```
          Table Four
            C=A\B
 A          B   C(HDOS)  C(CP/M)
 9          7       1        1
 9         -7      -1       -1
 8.1        2.1     4        4
 8.1       -2.1    -2       -4
 8.1        2.1     4        4
-8.1        2.1    -4       -4
-8.1       -2.1     3        4
 8.1        2.9     4        2
 8.1       -2.9    -2       -2
```

As you can see, the results of using the integer division operator on real variables, particularly negative real variables, can result in some rather unpredictable results. For this reason, some of the more strongly typed languages prohibit the use of the integer division operator on reals. Therefore, if you intend to use the integer division operator on reals, be sure that you understand what is going to happen. Otherwise, convert the real numbers to integers using the appropriate conversion factor in order that the result will fit the mathematical model that you are using.

The Modulus Operator

The MOD operator gives us the "remainder" of the integer division operation. The results are pretty straightforward with integers. Nine divided by seven is equal to one with two left over. So, 9 MOD 7 = 2. However, when we use real numbers as arguments, we find that the results will vary because of the varying definitions of integer division. To test the MOD operator replace the \ in line 200 with "MOD". Again, run the program using the values of A and B from Table Four and compare your results with those in the Table Five.

Check the last four lines of longhand to see why one gets the results shown. Note that when the MOD operation is performed on real numbers, the effect of the algorithm is to first convert the two arguments to integers, then perform integer division on them and record the remainder. Since this conversion is dif-

ferent in the several versions considered, the results may or may not be different.

Table Five
D=A(MOD)B

A	B	C(HDOS)	C(CP/M)
9	7	2	2
9	-7	2	2
8.1	2.1	0	0
8.1	-2.1	2	0
8.1	2.1	0	0
-8.1	2.1	-1	0
-8.1	-2.1	0	0
8.1	2.9	0	2
8.1	-2.9	2	2

Real Division With Integers

As was stated earlier, MBASIC performs the four elementary arithmetic operations using real arithmetic. The use of the real division operator (/) on two integer variables is the same as performing them on real variables. If I% = 3 and J% = 4, then the result of I%/J% will be 0.75. How the results of this operation will be stored will depend on the variable to the left of the assignment operator (=). If the result is stored as a real variable, its value will be 0.75. If the result is stored as an integer variable, its value will be either 0 or 1 depending on how the assignment operator is handled in the particular version. Write a simple program yourself to test this principle.

Subscripts

It was previously mentioned that one of the uses of the integer was as the index or subscript for an array. What happens when a real number is used as the subscript of an array element? The interpreter converts the real number to its integer value and uses it as the subscript. Here is a simple program to test this proposition and see what happens when a real value is assigned as a subscript.

```
100 REM SUBSCRIPT TESTING PROGRAM
110 DIM A$(10)
120 A$(1)="ONE"
130 A$(2)="TWO"
140 A$(3)="THREE"
150 A$(4)="FOUR"
160 A$(5)="FIVE"
170 FOR I=1 TO 3 STEP 0.1
180 PRINT I,A$(I)
190 END
```

The results confirm the statement. Since the four versions of MBASIC treat this conversion in two different ways, the change from an output of ONE to TWO will occur at either I = 1.5 or I = 2.0, depending upon the way that the real number is converted to an integer. This is usually the same method used in the replacement operation. Another strange phenomenon may occur, depending upon the internal arithmetic operations of your computer. The last entry may show I = 3 but A$(I) = TWO or the last value of I may be 2.99999. This should be a warning to only use integer limits and steps in a FOR/NEXT loop, where the index of the loop is to be used as the index or subscript of an array element.

Two Simple Programs Using Integer Functions

There is no sense in having these fancy integer functions unless we can find a use for them. Below are two examples. The first was written to convert decimal hours (or degrees) to hours, minutes and seconds (or degrees, minutes and seconds). The second is to convert decimal seconds to hours, minutes and seconds (or degrees, minutes and seconds).

```
100 REM PROGRAM TO CONVERT DECIMAL HOURS TO
      HOURS/MINUTES/SECONDS
110 PRINT "ENTER DECIMAL HOURS - ENTER 0 TO ESCAPE"
120 INPUT TIME
130 IF TIME = 0 THEN END
    REM ESCAPE TEST
140 GOSUB 9000
150 PRINT HOURS;"   ";MINUTES;" : ";SEC
160 GOTO 110
9000 SUBROUTINE TO CONVERT HOURS TO H/M/S
9010 HOURS = FIX(TIME)
9020 TIME = (TIME - HOURS)*60.0
9030 MINUTES = FIX(TIME)
9040 SEC = (TIME - MINUTES)*60.0
9050 RETURN
```

```
100 REM PROGRAM TO CONVERT DECIMAL SECONDS TO
      HOURS/MINUTES/SECONDS
110 PRINT "ENTER DECIMAL SECONDS - ENTER 0 TO ESCAPE"
120 INPUT TIME
130 IF TIME = 0 THEN END:
    REM ESCAPE TEST
140 GOSUB 9500
150 PRINT HOURS;" : ";MINUTES;" : ";SEC
160 GOTO 110
9500 REM SUBROUTINE TO CONVERT SECONDS TO H/M/S
9510 TEMP = TIME - FIX(TIME).
     REM SAVE DECIMAL PORTION
9520 TIME = FIX(TIME).
     REM TO AVOID CONFUSION IN THE WAY \ IS HANDLED
9530 HOURS = TIME \ 3600
9540 TIME = TIME MOD 3600
9550 MINUTES = TIME \ 60
9560 SEC = (TIME MOD 60) + TEMP
9570 RETURN
```

You now have six new tools for your programming tool box. They are:

1. The Integer Variable (%)
2. The FIX(argument) Function
3. The INT(argument) Function
4. The CINT(argument) Function
5. The Integer Division (\) Operator
6. The Modulus (MOD) Operator

Use them well and with confidence. Check them when a BASIC program from any source does not give results that correspond to the sample output. *

PeachText And WordPerfect Compared

Alison Phillips
4012 Thoroughgood Drive
Virginia Beach, VA 23455

Comparison Shopping

Wouldn't it be nice if you could read a software advertisement and garner a true feeling for the worth of the program? When I read the advertisements, specifications, and even the reviews for a word processor, I am still left wondering just how usable and how much fun the product is to use. I have bought three word processors and used them extensively, and I can tell you now that you have to live with one a while, and perhaps accommodate to its quirks before a reasonable judgement can be made about its worth.

WordPerfect And PeachText Compared

Do you agree with the old cliche that comparisons are odious? Odious perhaps, yet helpful. Comparisons are certainly of value when it comes to decision making. After all, when we shop for a new car, do we not intuitively compare one model with another? Likewise, we need a comparative basis for an informed judgement when we spend big money for a word processing package. Many Z-100 and Z-100 PC users either own, or have used PeachText for word processing. So with this in mind, I will use PeachText as a yardstick to take the measure of WordPerfect.

Background

PeachText. PeachText 5000 (simply PeachText) has been with us a long time and was aggressively promoted when the Z-100 was in its infancy. Because it is moderately priced and has been heavily discounted, it became quite popular early in the Z-100's life. Unfortunately, PeachText was old at its birth. The word processing portion of PeachText is the old Magic Wand which was copyrighted in 1980.

Magic Wand, itself, was a good, substantial word processor in its time, and it still is for that matter, if one's needs are modest and if one can do without the extra power of the modern word processors, like WordPerfect.

Magic Wand served me well in the early years before I purchased PeachText. In fact, one of the reasons I purchased PeachText, when I moved up to the Z-100 from the 8-bit computer I owned, was that it incorporated the substantial Magic Wand. And I wanted to avoid learning a new word processor at the same time that I was learning the MS-DOS operating system and the new computer.

WordPerfect. Eventually, the time arrived to break ties with the past, and to treat myself to a new word processor. I chose WordPerfect for a number of reasons, some of which will become evident as this review unfolds.

One of the reasons was that WordPerfect supports a large number of computers including the Z-100, the Z-150, and the IBM PC along with a host of the compatibles.

Why did I spend $289 for WordPerfect when Microsoft Word was bundled for free with the new computer? The decision to go with WordPerfect was partly an act of faith (aren't all purchases that way?) and partly an informed judgment. InfoWorld, in its February 25, 1985 issue, gave WordPerfect a rare 4.0 rating and said: "We believe WordPerfect 4.0 represents a new standard of excellence for microcomputer word processors."

After studying InfoWorld's review, I made several calls to Satellite Software International, the creators of WordPerfect, and questioned them in some detail. I was impressed with their diligence and thoroughness in answering my questions. They sent me a packet of literature that was so convincing that I said, "I've got to have WordPerfect."

WordPerfect, like PeachText, is a product with roots. The original WordPerfect was developed in 1978 and 1979 by Bruce Bastian and Alan Ashton. They founded Satellite Software International and through the years have overseen the continual improvement of WordPerfect.

Bruce Bastian, who was heavily involved in the design of WordPerfect, and who did much of the machine coding that makes WordPerfect so fast, asserts that WordPerfect is an ongoing product and that he doesn't think that it will ever really be finished.

WordPerfect, like PeachText, is a complete package of word processing tools. If we were to take a satellite view of these two packages, we might be inclined to favor PeachText. Both packages have a speller, and both are capable of merging files. Here the similarity ends.

PeachText additionally includes the Random House Thesaurus, a list manager, and a spreadsheet named PeachCalc. At face value, this is very impressive. Unfortunately, only the Thesaurus is an integral part of PeachText, all of the other programs stand alone, and can only be invoked after exiting from the word processor.

There are no standalone programs with WordPerfect. Each feature of WordPerfect is truly integrated and is available from the edit screen while the text is created; this is an important distinction between WordPerfect and PeachText. The complete integration of all functions is what makes WordPerfect so convenient to use.

Distribution Disks

Distribution disks, as the name implies, are the actual disks received from the distributor when a program is purchased. You can tell a great deal about how a program is intended to be used by examining the makeup and arrangement of programs on the distribution disks. For example, if you find duplicate files on several of the distribution disks, you may infer that provisions were made for disk swapping during the operation of the program. In this respect, the file makeup of distribution disks for multi-drive systems, or hard drive systems, is quite different from that for single-drive systems.

Distribution disks may or may not be supplied with write-protect tabs. Play it safe, cover the write-protect notches on the 5-1/4" distribution disks with the pressure adhesive write-protect tabs prior to inserting the disks in the drive the very first time.

PeachText. PeachText 5000 is distributed on six, 5-1/4" single-sided disks. Of these six disks, only three contain files essential to word processing. Namely, the Word Processor disk, the Thesaurus disk, and the Speller disk.

The Word Processor disk contains learner files which need not be copied to the final working disk. The Speller disk contains 4 files which are duplicates of those on the Word Processor disk. If you have room for both the speller and the word processor on the same drive, the duplicate files are not required.

Best operation of PeachText is obtained with a two drive system. All essential files should be copied from the three distribution disks to a single disk of sufficient capacity. When this is done, disk swapping can be eliminated during word processing.

WordPerfect. WordPerfect is distributed on four double-sided 5-1/4" disks. Of these four disks, two are similar. One is the word processor configured for the IBM PC, and the other for word processor IBM PC look-alikes. Disk Three is the Speller. Disk Four is a Supplementary disk, used for learning purposes. It need not be copied to the working disk.

Because of frequent disk accesses and the large program size, best operation is obtained with a hard drive or an 8" drive. The hard drive is so fast, that I no longer bother with using a ramdrive. I have copied the Word Processor disk and the Speller disk to the same partition on the drive.

A comparison of the disk space used by PeachText and WordPerfect files essential to word processing is shown below. More on the implications of this table later in the article.

PeachText		WordPerfect	
Word Processor	93K	Word Processor	240K
Speller	101K	Speller	277K
Thesaurus	118K	Totals	517K
Totals	312K		

Technical Documentation

PeachText. PeachText's documentation consists of two spiral-bound books, each about an inch thick. Revision of spiral-bound books is not convenient. Evidently, book revision and follow-on support were not considered as important as cost containment when the books were published. One of the books is a Reference Guide, the other a Lesson Plan. Each of the books is used to document the list manager and spreadsheet, as well as the word processor. The inclusion of these standalone programs in the same books with the word processor saves money, but it also compromises the convenience of use.

The Reference Guide includes 141 pages which explain the use of the word processor, thesaurus, and speller. The Lesson Plan book includes 121 pages covering the same subjects.

The index for each manual is a composite index covering all the programs which constitute the PeachText package. The entries for the word processor are sandwiched among the entries for the other programs and this reduces the ease of using the index. But, worse than that, many useful entries for the word processor have been omitted entirely.

I find the PeachText manuals to be technically adequate, but certainly, not user friendly.

WordPerfect. The WordPerfect documentation is good, and shows the results of continual refinement. The Reference Manual is provided in a single, ring-back binder, of the type popularized by IBM, and which fits into a slide-in case. It is similar in physical structure to the MS-DOS Technical Manual, and is easy to update, should the need arise.

Also, the documentation includes 16-page Installation booklet and a 14-page Quick Reference Booklet of convenient size. The Reference Manual consists of eight sections, and each section is separated with a sturdy plasticized tab. An ample number of screen illustrations are coordinated with the tutorials and with the reference section. The manual fulfills its requirements so well that I doubt that you will ever see a book explaining how to use WordPerfect.

Ease Of Learning

Who wants to spend weeks learning to use a word processor? Wouldn't it be nice if you could just load the program and start typing? There are a host of tradeoffs between ease of learning, program power, and the speed with which the program executes. It often happens that a program that is easy to learn becomes a drag to the experienced user.

Creators of word processors have tried various methods to reduce learning time. One method is to have an on-line Help

screen. Another is to make ample use of menu options, be they in the form of graphic icons, or numbered lists. Each method is a compromise between ease of learning and ease of use. Some word processors display instructions on the screen along with the text.

I dislike having a portion of the screen tied up with distracting instructional material, borders, and the like. I prefer the screen to be like a clean sheet of paper. To the developers' credit, both PeachText and WordPerfect provide clean, uncluttered screens. But, back to the point of ease of learning.

PeachText. PeachText, for a number of reasons, is not easy to learn. The 98 pages of lessons provided in the Lesson Plan book are good, but it is a lengthy and tiring process to study them well. Even worse, it is the user's duty to memorize some 100 commands and variations thereof. He must memorize the syntax, as well. Only then, is he prepared to type the necessary embedded commands into the text being edited.

PeachText's primary Help screens are so hard to get to, that they contribute little to the ease of learning. Help screens will be considered later as a separate topic.

WordPerfect. WordPerfect is easy to learn. I am learning it as I prepare this article using WordPerfect. The judicious use of the special function keys leads to the ease of working with WordPerfect. The reference manual is clear, crisp, and well organized. There is little to require learning by rote. The way WordPerfect exploits the full potential of the special keys, when compared with PeachText is so revealing, that I will treat it as the next subject.

Use Of Special Function Keys

PeachText. When I first started using PeachText, I was pleased with the way it used the special function keys. That was good. I made a template of stiff paper to place around the keys, to reduce dependence upon memory. It seems that no two programs use the special functions keys for the same purpose, so I have made function-key templates for other programs, as well.

WordPerfect. WordPerfect also uses the ten special function keys, but Oh! how differently. A color coded white plasticized template is furnished with WordPerfect. This template fits nicely around the special function keypad. I placed dots of color on the CTRL, SHIFT, and ALT keys. By pressing either the RED, GREEN, or BLUE key and striking the selected special function key, a total of 39 functions are instantly available. For example, one key saves the text, another centers a line, while yet another will cause the text to be placed in columns. The special function keys are supported by a real-time legend which appears on part of the 25th line of the display, and then disappears the instant it is not needed. How easy it is!

Configuring For The Printer

All word processors must have a means of "talking" to the printer. But first the operating system, itself, must be interfaced with the printer. If you can print the disk directory by simultaneously pressing the CTRL and P keys, and then executing the DIR command, the operating system is interfaced with the printer.

Printers must receive control signals from the word processor to do such things as change pitch, start new pages, and establish margins. The nature of these control signals differs from printer to printer. It is the word processor's job to send the proper control signals to the printer in use. This talking to the printer is accomplished by means of a driver which is installed in the word processor when it is configured. A driver is a small program used to interface the computer with a peripheral, in this case, a printer.

PeachText. PeachText supplies a "Configurator" program on a separate disk. This disk contains the drivers for selected monitors and printers. Version 1.03 of the Configurator programs supports 17 families of printers. Configuring PeachText for a particular printer is a chore that must be anticipated before printing a document. Otherwise, you will have to terminate the working session and execute the Configurator program. If you have only one printer, this is no big problem. However, if you use more than one printer, frequent reconfiguring turns out to be a nuisance.

When the Configurator program is executed, it leads you through a menu, and ultimately the selected printer driver is copied from the Configurator disk to the word processor disk. You can only configure the word processor for one particular type of printer at a time. The whole process results in a number of disk swaps, and is tedious to perform.

WordPerfect. Configuring WordPerfect for a particular printer is a breeze. It can be accomplished at any time without exiting the edit session and without changing disks. Just press the "Print" special function key, and select the Printer Control option from the menu that pops up on the 25th line. Then follow the prompts. It is all over in a few seconds and you may proceed with editing or printing without losing your place in the text.

WordPerfect provides an on-line menu of 30 different printers or printer family options. From this menu, as many as five printers may be preselected, any one of which may be used at print time.

What You See Is What You Get?

Both PeachText and WordPerfect make an effort to display text in the manner it will appear when it is printed. Both fail. The failure results from the constraints imposed by the monitor and the way pixels are used to form a character.

A pixel is the smallest dot on the screen that can be individually lit. The typical display is 640 pixels wide. A width of eight pixels is required to adequately display a character. Hence, 640 pixels divided by eight pixels yields just 80 characters per line. If we wish to print 96 characters per line, an obvious incongruity exists between the display and the printer.

Then there is the matter of proportional spacing, where yet another incongruity exists between the display and printer. The characters displayed on the screen are of constant width, while those printed proportionally vary according to the actual width of the character. Hence, a line with a preponderance of narrow characters should contain more characters than a line of equal length in which wide characters dominate.

The dynamic display of page breaks is very important. With dynamic page breaks, the page break demarcation lines continually shift as text is inserted and deleted. It is of considerable convenience for the user to know at all times what page in the text he is working on and the line at which page breaks occur.

Based on these and other considerations, the creators of word processors are forced into compromises.

Peachtext. Word processing with PeachText is accomplished in two sessions; one with the EDIT program and one with the PRINT program. The document is created during the edit session and is formatted during the print session.

PeachText displays lines of any length up to 80 characters during the editing session. Word wrap at the end of the line is automatic. Lines longer that 80 characters can be printed — they just are not displayed while the text is being created. Most of the print formatting commands are embedded as the text is edited. These typed commands take space, remain visible, and in themselves, give a false impression of how the printed text will appear.

To get a better idea of what the hard copy will look like, it is necessary to terminate the edit session, execute the PRINT program, and issue the SCREEN ON command. The resulting display will show centering and page breaks. Each line is numbered on the display, and the space occupied by the line numbers causes the text to be "pushed" to the right on the display. A long line appears on the screen as a full line followed by a partial line. The net result is that the user must try to visualize how the printed copy will appear from what he sees on the screen.

If page breaks happen to occur at the wrong places, the user must terminate the print session, reload the edit program, reload the text file, and then insert a "new page" command at the desired point in the text. Adjustment of page breaks causes a ripple effect throughout a long text. This means that the whole process must be repeated until the end of the file is reached.

WordPerfect. With WordPerfect, the display is formatted as it is created, or edited. Centering and underlining are automatically shown on the display. Page breaks are displayed just as they will appear on the hard copy. The points at which the page breaks occur dynamically shift as text is added or deleted.

WordPerfect displays line lengths up to 250 characters. It does so by scrolling the entire screen to the left when the physical length of the line exceeds the screen's width. When the right margin is set to a value greater than 80, the scrolling commences when column 81 is reached. This scrolling continues until the last character in the line is typed. Then as word wrap takes place, the entire display "jumps" right to restore the left margin. When line lengths exceed 80 characters, the right-most characters in the line may not be displayed. This is only a minor inconvenience with line a length of 92 characters since the right edge of the screen can be read at any time by pressing the END key.

Cursor Movement

One of the essential requirements of a good word processor is to be able to move the cursor from any place in the text to any other place in the text. And to do it rapidly! Good cursor control within the line is important, and surely, we don't want to see lines rippling down the screen one at a time. These user requirements are a real challenge to programmers who must figure a way to rewrite a screen so rapidly, after an insertion has been made, that it is not distracting to the user. Let's see how PeachText and WordPerfect perform in these areas.

PeachText. PeachText does not format the text on the screen as it is created, and this tends to make repositioning of the cursor within the text somewhat faster. All-in-all, PeachText is fast enough, but there are minor exceptions. One exception has to do with moving the cursor, a line at a time, toward the front of the text from the top of the screen. Each time you back up a line from the top of the screen, the entire screen is slowly rewritten one line at a time producing a noticeable ripple effect as each line is rewritten. This is a flaw that is not apparent in WordPerfect.

WordPerfect. WordPerfect formats the text on the screen as it is created, and this tends to slow down repositioning within the text. It becomes more apparent in long files, almost to the point of exasperation. It takes about ten seconds to move the cursor from the beginning of a 12-page text to the end. PeachText performs this action much faster.

Fortunately, most of the cursor movements made while a text is being created are nearby. WordPerfect performs nearby cursor movements nicely. With a stroke of a key, you may move forward or backward a line, or screen at a time. The time required for such movements is barely noticeable, but it takes about a second to move the cursor a whole page (54 lines), and this is noticeable.

Help Screens

If a help screen is to be really useful, it must be brief, and context-sensitive, so that pertinent help is immediately available as text is being entered and formatted. Exit to the help screen and subsequent return to the cursor position within the text should each be accomplished with a single keystroke.

PeachText. PeachText's help screens satisfy none of the above requirements. The process of getting from the cursor position within the text to the help screen and back to the cursor position within the text is so tortuous that it all but nullifies the advantage of having a help screen.

WordPerfect. WordPerfect's on-line help screens are a delight. They are not needed often, but they are just a keystroke or two away. And return to the cursor position is automatic when exiting from the help screen. The help screen always relates specifically to the special function you select.

Editing Multiple Files

A good word processor, as a minimum, should allow the user to alternately edit two files during the same session. Also, it should be easy to move any portion of either file to the other file at any time.

Actually, it would be better to have three files in memory. Take the example of an author writing an article. He could use one file for text preparation, the second file for an outline, and the third for a notepad. Let's see how PeachText and WordPerfect match up to these requirements.

PeachText. Only one file can be edited at a time with PeachText. Portions of a second file can be "included" as needed into the file being edited. It is not possible to include a particular phrase, sentence, or paragraph from the second file into the first, unless advance preparation is made in the second file. The inclusion of material from the second file into the text under preparation is a one-way operation. It is usually done a screenful at a time, and at best, it is clumsy.

WordPerfect. Two files can be independently edited during a session with WordPerfect. A SWITCH key transfers the display from one file to the other. Discrete portions of text are easily transferred from either file to the other. Either file can be saved independently, and then replaced with a new file. The 25th line always shows which file is on the screen. Working with two files is smooth and natural.

Embedded Format Commands

Word processors use formatting commands embedded within the text for the purpose of controlling the printer. Such commands may be displayed along with the text, or they may be invisible, nevertheless they are there.

These embedded commands are control characters, or symbols which are to be interpreted as control characters. Often, they are customized to a given printer. Embedded commands are used for such things as centering, line feeds, page feeds, underscoring, and the like. It is best that the word processor conceal these embedded commands so that they do not clutter the display. But there are instances when the user needs to know what commands are concealed, and he must have the means to remove them when necessary.

Learning the control characters, mnemonics, and syntax of embedded commands can be burdensome with some word processors. This is particularly so if the user must type them verbatim from recall. Ideally, the user should just have to strike a labeled key for each embedded command.

PeachText. Formatting commands must be recalled and typed into the text as it is created. (A few global commands may be entered on the PRINT screen.) If the user cannot recall the command, or its syntax, then he must go to the manuals and play guessing games with the index. The embedded commands remain in full view to clutter the screen and alter its appearance from that of the printed document.

WordPerfect. WordPerfect conceals the embedded commands, and you see only the text that you type on the screen. If you wish to view the commands which are embedded in the text, all you have to do is press the REVEAL key. A reveal screen will appear immediately, and any undesired control codes can be deleted. Press the space bar and you are returned to the cursor position with the text.

It is not necessary to remember control codes, mnemonics, or syntax when embedding a printer control code with WordPerfect. For example, to center a title, simply press the CENTER key and type the title. Or if you wish to underline a word, press the underline key, type the word, and press the underline key again. The underlined word will appear underlined on the screen and will be printed that way.

Saving Files

A word processor should be able to save all or any part of the text in memory to a file on any available drive. In addition, it should support the MS-DOS hierarchical directory structure. The format in which the file is saved should be user selectable and include: (1) A format in which all control characters are embedded. (2) A format with all control characters stripped, except the carriage returns (leaving a pure ASCII file). This file must be viewable with the Z-DOS/MS-DOS TYPE command. (3) An encrypted (locked) format which cannot be read without the user's password.

PeachText. All or any portion of text in memory can be saved to disk using Format (1) above. PeachText does not support Format (2) because it removes the carriage returns which terminate physical lines before it saves the file to disk. If an attempt is made to display a PeachText file using the MS-DOS TYPE command, only the first physical line in each paragraph will be displayed. PeachText does not support the hierarchical directory structure and does not have encryption capability.

WordPerfect. WordPerfect supports all three formats listed above. Its ability to save a pure ASCII file that can be read using the operating system's TYPE command is an very desirable feature. Likewise, some users will appreciate its ability to save encrypted files.

Macro Capability

Macros are gifts from heaven. A word processor macro is actually a small file consisting of a series of keystrokes which may be made up of word processor commands and/or text to be printed. Macros can be created to do a host of chores, like printing a complete letterhead, flipping two adjacent sentences within the text, or changing all of the default functions of the word processor. Word processor macros are usually stored on disk and they work exceptionally well with a hard drive.

PeachText. PeachText does not have macro capability — and what a loss it is.

WordPerfect. WordPerfect has powerful macro capability. To create a macro, you press the MACRO DEF key and assign the macro a short name. Then go ahead and do the task at hand — say, create and center a letterhead. After the task is finished, press the MACRO DEF key again, and the macro will be filed on the default disk. In the future, when you need the letterhead, execute the macro and the letterhead will almost immediately appear on the display.

Proportional Spacing

Proportional spacing is a word processing attribute found only in the more expensive "full function" word processors. It is not to be confused with justification, or even microjustification. Some vendors are inclined to be very liberal in their interpretation of just what proportional spacing entails.

For the purpose of this review, and I believe it to be generally true, proportional spacing occurs when each character is spaced according to its actual width rather than a fixed number of characters per inch; while concurrently, line lengths are based on the accumulated widths of the characters which make up the line, and not an arbitrary number of characters per line.

A line dominated with narrow characters like in the word "illicit" would contain more characters than a line dominated with wide characters like in the word "mommy", yet both lines would be the same physical length when printed.

PeachText. PeachText supports proportional spacing as defined above. And it supports it well. This is one of the better features of PeachText.

WordPerfect. WordPerfect does not support proportional spacing as defined above. WordPerfect's promotional material and specifications claim "true" proportional spacing. My tests indicate otherwise and I wrote Satellite Software International (the developers) seeking an explanation. Their reply was not satisfying — they cite conflicting requirements and say "We are doing more research in this area and are looking for the best way to improve our printing so that future releases of WordPerfect [will] handle proportional spacing in the best way possible."

Spelling Programs

PeachText. The spelling program is severely limited by its small (20,000) word dictionary. It does not detect double words (two like words in succession). The speller is not integrated with the editor and must be used as a standalone program after the edit

session has been terminated. This precludes checking the spelling of a single word during the Edit session.

The PeachText speller marks words it cannot find in its small dictionary, and because the dictionary is so limited, many of these words that are marked may be spelled correctly. Finally, the speller cannot correct the text. The user must re-edit the file to make spelling corrections. The PeachText speller is poor on nearly every count.

WordPerfect. The spelling program has a large (100,000 word) dictionary. It detects double words that are apt to creep in during word processing. The speller is integrated with the word processor and is immediately available to check the spelling of a word as it is entered. Just press the SPELL key and you may test the word, page, or document for misspelled words. More than that, if a word is misspelled, most of the time you may correct it on screen by striking a single key. If you don't know how to spell the word, you can spell it phonetically and have a good chance that the speller will find the word you are looking for. The speller is excellent on all counts.

Thesaurus

PeachText. Peachtext has integrated a version of the Random House Thesaurus with the word processor. This thesaurus contains about 4,400 indexed words and 26,000 synonyms that occupy 118K of disk space. To use the thesaurus, the user positions the cursor on the selected word and strikes a special function key. If the word exists in the thesaurus, the synonyms will be listed across the bottom portion of the screen.

The thesaurus is the only program in the PeachText package that is integrated with the EDIT program. This makes it handy when you need an alternate word for the one that comes to mind. It would be more useful if its vocabulary were larger, and with existing 320K and higher capacity drives, enlarging the thesaurus should not be a real problem. However, if a developer should include the thesaurus and a 100,000 word spell program on the same disk, it would require about 400K. This limits the market to users with high capacity drives.

WordPerfect. The WordPerfect word processor does not include a thesaurus.

Math Functions

PeachText. It would be stretching the truth to say that PeachText supports math of any kind. It does allow embedding variables that can be incremented (added to), and this is about the extent of math that can be performed. Even this limited math is not available during the Edit session, but only becomes active when the text is printed.

WordPerfect. WordPerfect supports limited calculations while the document is being created. Both numbers and text can be typed in columns that have been defined by the tab stops. The four functions: add, subtract, multiply, and divide are supported. Calculations can be performed in rows and columns.

Other Niceties

WordPerfect has a host of additional features that PeachText does not possess, and WordPerfect does many things in a nicer way. Some of the nice features of WordPerfect, not previously discussed, are briefly listed below:

1. Case conversion of any part of existing text.
2. Automatic margin set for newspaper-type columns.
3. Insertion of date any place within text.
4. Strong merging facilities.
5. Powerful directory listing facility shows file sizes. File maintenance is accomplished by cursor control.
6. Automatic timed backup — time set by user.
7. Table of Contents and Index preparation support.
8. Powerful footnote and endnote facilities.
9. Forward and backward search not case sensitive when lower case string is entered.

Manufacturer's Support

When we buy a piece of software, it would be nice to think that there was someone, somewhere, to turn to in time of need. Don't count on it too heavily.

PeachText. I have made several calls to the Peachtree Software. On one occasion, I received an immediate answer while still on the phone. On two other occasions, they could not give immediate answers, but did reply by letter in a reasonable time.

WordPerfect. I have telephoned WordPerfect twice. The first time I asked if WordPerfect would support PeachText files. I held the long distance line while my respondent consulted with fellow workers and was finally advised that I would have to convert my PeachText files to WordStar first, and then WordPerfect would read the WordStar files. Later, tests of my own proved that these instructions were in error. WordPerfect does support PeachText files.

The second time I telephoned WordPerfect, I asked if WordPerfect would print proportionally spaced text with the number of characters within the lines based upon the combined widths of the characters which made up the line. Again, after considerable interoffice consultation, they were unable to answer the question and promised to call me back. I never received a return call. This telephone experience leads me to wonder if the people who answer the customer support lines are sufficiently knowledgeable to field questions which are not covered in their user's manual.

Then, some two months ago, I wrote a letter expressing my dissatisfaction with the way WordPerfect supported proportional spacing. I did receive a prompt reply and an offer to furnish a new proportional printwheel table to support the Cubic PS-96 printwheel. I accepted their offer, but as yet I have not received the needed table. I think WordPerfect tries, but don't count on quick response.

Summary

Many things have been left unsaid about PeachText, and even more has been left unsaid about WordPerfect. A number of features were not covered because of the need for brevity. PeachText and WordPerfect are good word processors, but WordPerfect is better because of its full integration, extended power, and ease of learning.

The principal shortcomings of PeachText is its miserable speller, and the fact that the Print and Edit programs do not perform in concert as a single program. Also PeachText does not support hierarchical files. It is a shame that PeachText, which is fundamentally good, has not been continually improved to take greater advantage of the special function keys and the increased computer memory and disk storage that has become the norm.

The shortcomings of WordPerfect that annoy me most are the "half-job" it does with proportional spacing, and repositioning the cursor several pages at a time is unnaturally slow. But, all-in-all, WordPerfect is a good product — it's modern, easy to use, and powerful. If $250 or so, is not too rich for your blood, and if you want ease of learning and the other niceties that I have previously explained, then by all means go for WordPerfect. Otherwise, PeachText can be bought for about a third of the WordPerfect's price, and you get a good sound word processor for your money.

Products Mentioned

WordPerfect
SSI Software
288 West Center
Orem, Utah 84057

PeachText
PeachTree Software Inc.
3445 PeachTree Road, N.E.
Atlanta, Georgia 30326

*

Continued from Page 10

pixel graphics on the screen, but does not execute the Print command to send pictures to a dot matrix printer. I normally run under CP/M with a serial port to an MPI-99G printer. I have also tried an Epson printer on the serial port and have used the SigmaSoft parallel interface with the parallel option of the MPI-99G. Each of these transfers text but not pixels. Extensive correspondence and conversations with the vendor, who claims to support these configurations, have not resolved the problem.

I would appreciate hearing from anyone who has successfully printed pictures with the SigmaSoft graphics package running under CP/M.

Steven M. Sussman
26 Locksley Road
Newton Centre, MA 02159

A Little Added Information For H-29 Users

Dear HUG:

We have been using two H-29 terminals at work for some time. They have been connected to a Digital Equipment PDP-11/34 in the VT-100 mode. They always powered up in Caps Lock which was fine for the RT-11 operating system. About two months ago, I started using one to communicate with Unix mail at the company computer center, and with Unix case is important so I had to unlock the upper case every time I powered up. I looked through the manual, asked the technician who assembled it, and called Heath to see if there was a way to change the default power up condition to be lower case, but no luck. Even started to look at the schematics for a hardware fix.

However, fate saved me. One day I changed the cable from the DTE port to the DCE port, and went to the SETUP mode to change ports and saved it with the SHIFT-SETUP on exit. Some time later I realized that when I powered up, the H-29 was in lower case, how come? Simple, whatever case the terminal was in at save time is apparently stored in the electronically programmable ROM. How come no one knew this?

Don't know how many H-29 users are out there, but here is a little extra information about a very versatile terminal.

Sincerely,

Adolph Fejfar
3017 Oriole Drive
Sierra Vista, AZ 85635

Halley's Comet On The H-89

Dear HUG:

I enjoyed the article on Halley's Comet by Jim Tursa in the September 1985 issue of REMark. To make use of the program, I modified it to run on my H-89A with the NOGDS color graphics board and Graph-Pac II software. While the 640 X 224 resolution of the Z-100 is nice, the program is still quite enjoyable with the 256 X 192 resolution of the TMS 9918A color graphics chip used on the NOGDS board.

The Graph-Pac II software offers many commands similar to the Z-100 graphics commands. I used a modular approach so that other Z-100 programs might be easier to translate.

Some compromises had to be made due to the resolution difference. The smallest circle that can be drawn with the Graph-Pac II software has a radius of 2 pixels. To simulate an object with a smaller diameter, but greater than 1 pixel, the circle-drawing routine plots 4 points instead of an actual circle. The number of stars printed on the screen was reduced because they made the display too confusing, probably because each star takes up more screen area.

The smallest color font available in Graph-Pac II is more than twice as large as the standard text screen font. To compensate, all text information was printed on the text screen, while only essential or interesting information was printed on the graphics screen.

The program displayed the "OUT OF MEMORY" error early during the conversion process. To produce a running program, the constellation data [lines 2520-2910] were put in a disk file (CONST.DAT), the data lines were deleted and the following lines were added or changed:

```
2900 OPEN "I",1,"CONST.DAT"
2930 INPUT#1,RA,DEC    RA=RA*PI/12    DEC=DEC*DTOR
```

Many other minor changes were required; most of which were a matter of taste.

The program runs slower than on the Z-100, but does fine at 4 MHz. The options available in the program can be used to speed up the movement of the comet and planets at the expense of some accuracy.

I will provide the H-89 version to those who send a blank disk and $2.00 (soft-sector format).

I thank Jim Tursa for submitting this program.

Sincerely,

Pete Roberson
2714 W. John Day
Kennewick, WA 99336

"MUVIT — A BASIC Boost For Spreadsheets"

Dear HUG:

I'm deeply gratified you accepted and published my article, "MUVIT — A BASIC Boost For Spreadsheets". Now I have to apologize and ask you to publish a correction in one of the BASIC modules. In Listing 4 — FOUR.BAS, there is a minor error in line 4700. The existence of 'CLS +' causes the error message in line 4640 to vanish so abruptly as to make it useless. Strike out the offensive terms and my embarrassment will be diminished. This is the price paid for submitting so carefully my unsullied original listings while I used a copy, corrected some time in the past, for family chores.

Sincerely,

Louis M. St.Martin
860 Hillcrest Drive
Pomona, CA 91768

*

Heath/Zenith Users' Group

Part Number	Description of Product	Selling Price	Issue
Continued from Page 46			
885-1109-[37]	HDOS Retriever ASM (3 Disks)	40.00	23
885-1110	HDOS Autofile (2 Disks)	30.00	23
885-1115-[37]	HDOS Navigational Program	20.00	25
885-8008	Farm Accounting System	45.00	30
CP/M			
885-1219-[37]	CP/M Navigational Program	20.00	31
MSDOS			
885-8034-37	DBZ-A Database For The Z100	25.00	69
AMATEUR RADIO			
HDOS			
885-8016	Morse Code Transceiver Ver 2.0	20.00	42
CP/M			
885-1214-[37]	CP/M MBASIC Log Book (64k)	30.00	23
885-1234-[37]	CP/M Ham Help	20.00	49
885-1238-[37]	CP/M Ascirity	20.00	57
885-8020-[37]	CP/M RF Comp. Aided Design	30.00	44
885-8031-[37]	CP/M Morse Code Transceiver	20.00	57

Part Number	Description of Product	Selling Price	Issue
COMMUNICATION			
HDOS			
885-1122-[37]	HDOS MicroNET Connection	16.00	37
885-8005	MAPLE (Modem Appl. Effector)	35.00	29
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
885-5006-37	CP/M-86 HUGPBBS	40.00	62
885-5007-37	CP/M-86 HUGPBBS Source List.	60.00	62
885-8012-[37]	CP/M MAPLE (Modem Program)	35.00	34
885-8023-37	CP/M-85 MAPLE	35.00	45
MSDOS H/Z100 - H/Z150 PC			
885-3019-37	ZDOS 16 Bit MicroNET Connect.	16.00	61
885-3027-37	MSDOS HUG PBBS	40.00	66
885-3028-37	MSDOS HUG PBBS Source Listing	60.00	66
885-3033-37	MSDOS HUG MCP	40.00	71

Part Number	Description of Product	Selling Price	Issue
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 sectored 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 ☐	$17 ☐
FPO/APO & ALL OTHERS*	$35 ☐	$30 ☐ U.S. FUNDS

* Membership in France and Belgium is acquired through the local distributor at the prevailing rate.

Heath/Zenith Users' Group

Hilltop Road

Saint Joseph, Michigan 49085

P/N 885-2073