# REMark®

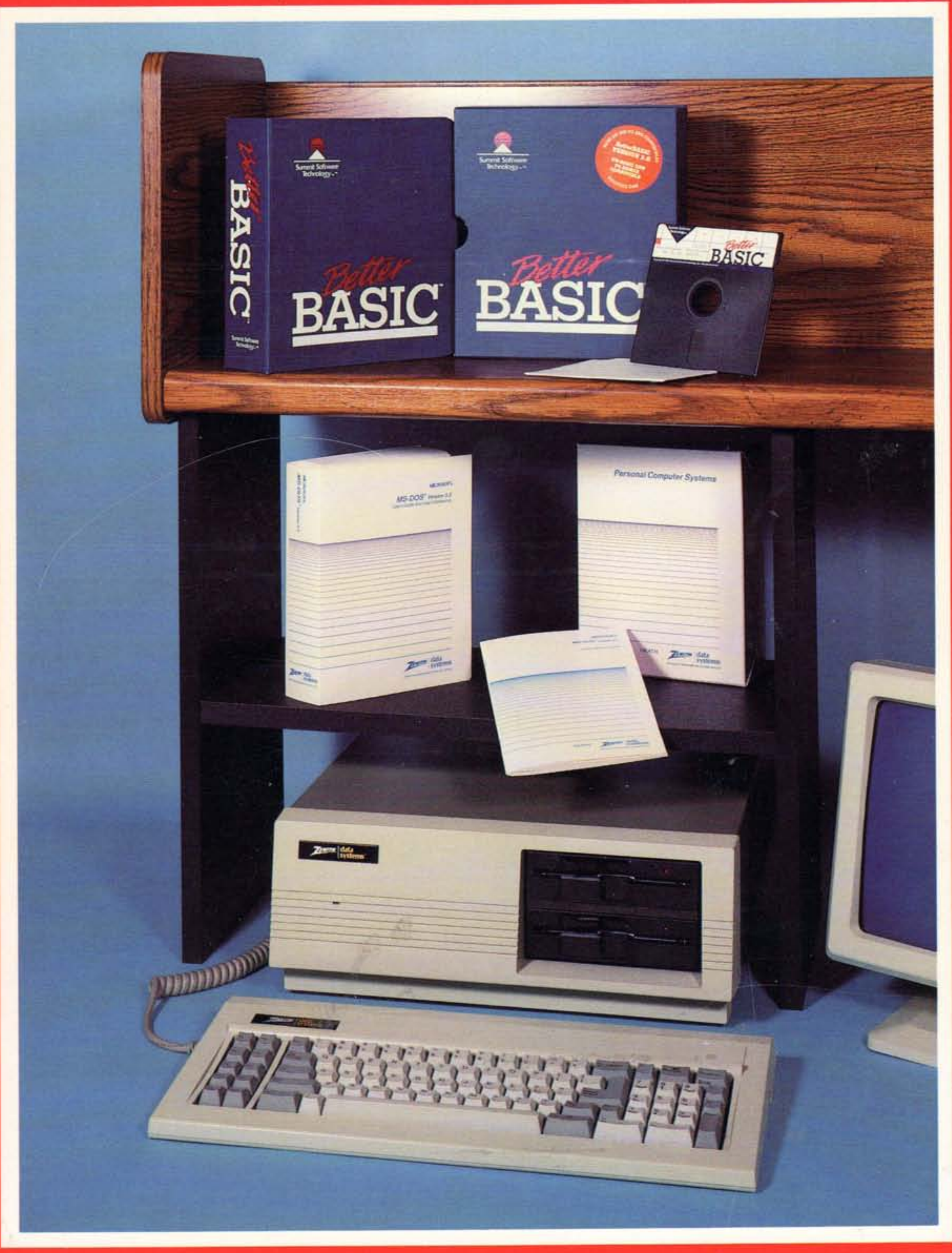
Volume 8, Issue 5 • May 1987

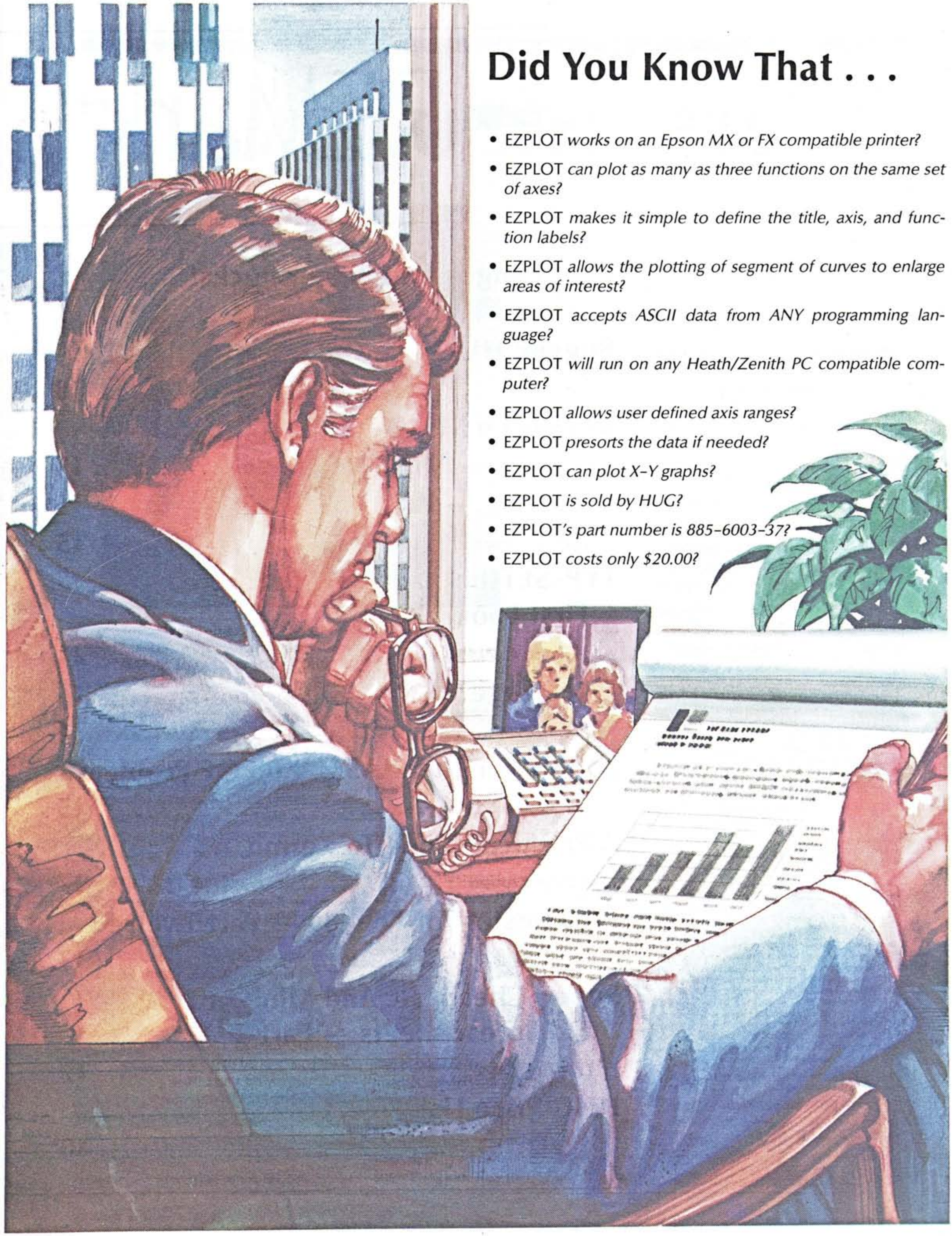
P/N 885-2088 Issue 88

Official magazine for users of HEATH ZENITH computer equipment.

$2.50

A New Look And Bigger Writer Incentives
See Jim's Editorial On Page 7





Heath/Zenith Users' Group

Managing Editor ........ Jim Buszkiewicz
(616) 982-3837

Software Engineer .......... Pat Swayne
(616) 982-3463

Software Coordinator ...... Nancy Strunk
(616) 982-3838

Production Coordinator ....... Lori Lerch
(616) 982-3794

Secretary .............. Margaret Bacon
(616) 982-3463

HUG Bulletin Board ...... (616) 982-3956

Contributing Editor ....... William Adney

Contributing Editor .......... Joseph Katz

Printer ................ Imperial Printing
St. Joseph, MI

|  | U.S. Domestic | APO/FPO & All Others |
|---|---|---|
| Initial | $22.95 | $37.95* |
| Renewal | $19.95 | $32.95* |

* U.S. Funds

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

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

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

Articles submitted by users and published in REMark, which describe hardware modifications, are not supported by Heath/Zenith Computers & Electronics Centers or Heath Technical Consultation.

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

# REMark®

Official magazine for users of HEATH ZENITH computer equipment.

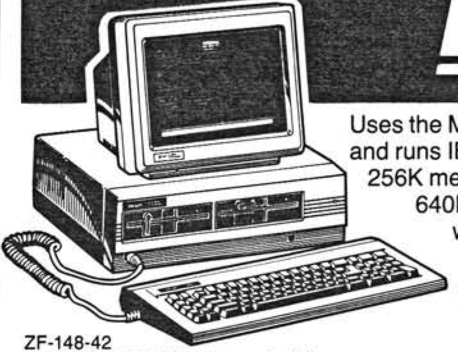
Volume 8, Issue 5 • May 1987

## 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:*** *For MS-DOS fans, a review of new commands, changed commands, hard disk support, and other noteworthy items of V3.2 can be found on Page 11. Do you already know BASIC and are reluctant to learn another language? You will want to check out the review of BetterBASIC on Page 57.*

# IBM PC/XT Compatible Personal Computer with *FREE* Monitor



Uses the MS-DOS operating system and runs IBM software. Includes 256K memory (expandable to 640K), two 5¼″ disk drives with 720K storage, and both mono and color graphics capabilities. Comes with cable.

ZF-148-42
with ZVM-1220A Monitor and cable

ONLY **$999***

## IBM PC/XT Compatible Personal Computer with C.ITOH RGB color monitor and cable

ZF-148-42
with C.ITOH Color
Monitor and cable

ONLY **$1359***

*NO ADDITIONAL DISCOUNTS

## DISKETTE SPECIALS!

Get a Keystone Pocket Everflash® Camera **FREE** when you buy 50 Nashua MD2D High Quality double-sided double-density 5¼″ diskettes at the **special price of $49.95.**

– OR –

Get a Fuji VHS T120-HS High Standard video tape **FREE** when you buy 20 Fuji MF2DD Micro Floppy high density 3½″ diskettes or 20 Fuji AT MD2HD Micro Floppy high density diskettes. **Your Choice ONLY $44.95.**

## 20 Meg Drive Kits SPECIAL – $449

**NOTE: Accessory expansion card required for Winchester Drive Installation in 148 computer.**

## Available NOW at Heath/Zenith Computers & Electronics Centers In The U.S.

**PHOENIX, AZ**
2727 W. Indian School Rd.
602-279-6247

**TUCSON, AZ**
7109 E. Broadway
602-885-6773

**ANAHEIM, CA**
330 E. Ball Rd.
714-776-9420

**CAMPBELL, CA**
2350 S. Bascom Ave.
408-377-8920

**EL CERRITO, CA**
6000 Potrero Ave.
415-236-8870

**LA MESA, CA**
8363 Center Dr.
619-461-0110

**LOS ANGELES, CA**
2309 S. Flower St.
213-749-0261

**POMONA, CA**
1555 N. Orange Grove Ave.
714-623-3543

**REDWOOD CITY, CA**
2001 Middlefield Rd.
415-365-8155

**SACRAMENTO, CA**
1860 Fulton Ave.
916-486-1575

**WOODLAND HILLS, CA**
22504 Ventura Blvd.
818-883-0531

**WESTMINSTER, CO**
8725 Sheridan Blvd.
303-429-2292

**JACKSONVILLE, FL**
9426 Arlington Expressway
904-725-4554

**HIALEAH, FL**
4705 W. 16th Ave.
305-823-2280

**PLANTATION, FL**
7173 W. Broward Blvd.
305-791-7300

**TAMPA, FL**
4019 W. Hillsborough Ave.
813-886-2541

**ATLANTA, GA**
5285 Roswell Rd.
404-252-4341

**HONOLULU, HI**
98-1254 Kaahumanu St.
808-487-0029

**CHICAGO, IL**
3466 W. Devon Ave.
312-583-3920

**DOWNERS GROVE, IL**
224 Ogden Ave.
312-852-1304

**INDIANAPOLIS, IN**
2112 E. 62nd St.
317-257-4321

**MISSION, KS**
5960 Lamar Ave.
913-362-4486

**LOUISVILLE, KY**
12401 Shelbyville Rd.
502-245-7811

**KENNER, LA**
1900 Veterans Memorial Hwy.
504-467-6321

**BALTIMORE, MD**
1713 E. Joppa Rd.
301-661-4446

**ROCKVILLE, MD**
5542 Nicholson Lane
301-881-5420

**PEABODY, MA**
242 Andover St. (Rt. 114)
617-531-9330

**WELLESLEY, MA**
165 Worcester St. (Rt. 9)
617-237-1510

**FARMINGTON HILLS, MI**
29433 Orchard Lake Road
313-553-4171

**EAST DETROIT, MI**
18149 E. Eight Mile Rd.
313-772-0416

**ST. JOSEPH, MI**
2987 Lake Shore Drive
616-982-3215

**HOPKINS, MN**
101 Shady Oak Rd.
612-938-6371

**ST. PAUL, MN**
1645 White Bear Ave.
612-778-1211

**BRIDGETON, MO**
3794 McKelvey Rd.
314-291-1850

**OMAHA, NE**
2311 N. 90th St.
402-391-2071

**OCEAN, NJ**
1013 State Hwy. 35
201-775-1231

**FAIR LAWN, NJ**
35-07 Broadway (Rt. 4)
201-791-6935

**AMHERST, NY**
3476 Sheridan Dr.
716-835-3090

**JERICHO, LI**
15 Jericho Turnpike
516-334-8181

**ROCHESTER, NY**
937 Jefferson Rd.
716-424-2560

**N. WHITE PLAINS, NY**
7 Reservoir Rd.
914-761-7690

**GREENSBORO, NC**
4620C W. Market St.
919-299-5390

**CINCINNATI, OH**
131 West Kemper Rd.
513-671-1115

**CLEVELAND, OH**
28100 Chagrin Blvd.
216-292-7553

**COLUMBUS, OH**
2500 Morse Rd.
614-475-7200

**TOLEDO, OH**
48 S. Byrne Rd.
419-537-1887

**OKLAHOMA CITY, OK**
7409 South Western
405-632-6418

**FRAZER, PA**
630 Lancaster Pike (Rt. 30)
215-647-5555

**PHILADELPHIA, PA**
6318 Roosevelt Blvd.
215-288-0180

**PITTSBURGH, PA**
3482 Wm. Penn Hwy.
412-824-3564

**WARWICK, RI**
558 Greenwich Ave.
401-738-5150

**DALLAS, TX**
12022C Garland Rd.
214-327-4835

**FORT WORTH, TX**
6825-A Green Oaks Rd.
817-737-8822

**HOUSTON, TX**
1704 W. Loop N.
713-869-5263

**SAN ANTONIO, TX**
7111 Blanco Rd.
512-341-8876

**MIDVALE, UT**
58 East 7200 South
801-566-4626

**ALEXANDRIA, VA**
6201 Richmond Hwy.
703-765-5515

**VIRGINIA BEACH, VA**
1055 Independence Blvd.
804-460-0997

**SEATTLE, WA**
505 8th Ave. N.
206-682-2172

**VANCOUVER, WA**
516 S.E., Chkalov Dr.
206-254-4441 – Washington
503-241-3776 – Oregon

**MILWAUKEE, WI**
845 N. Mayfair
414-453-1161

***Phone orders accepted***
**OFFER GOOD THROUGH JUNE 15, 1987**

**Your TOTAL SERVICE computer center • Service • Support • Software • Accessories • User Training • Competitive Prices**

**Heath® ZENITH®**

Units of Veritechnology Electronics Corporation

Computers & Electronics

Keystone and Everflash are registered trademarks of Keystone Camera Corporation.

HZC-298

# Big Changes And Bigger Bucks!

If you haven't flipped through the pages of this issue of REMark yet, you may be in for a surprise! We've decided to try going from our two column format to a more standard three column one. Although somewhat harder to do, Lori assures me there shouldn't be any problem. I think you'll find that because each column is a little shorter, it'll be easier to read or scan each line of text. Source listings (depending on the language used and line length), can be printed vertically, occupying one or two columns, or horizontally sideways occupying two columns as we've done in the past. In any case, I hope you like it. A few articles may still appear in the older two column format, but should be gone by the July issue.

Another improvement I think you'll find, is in the format of the HUG Products Price List. Even I had a difficult time understanding its organization. The list needed a 'simple-minded' approach (right up my alley), and I think we've succeeded. First, when you're trying to find a piece of software, the most important question needing to be answered is, "Will it run on my machine?" To answer that question, we divided our products into five 'machine-type' categories: 1) H8 and H/Z-89/90, 2) H8, H/Z-89/90 and H/Z-100 (not PC), 3) H/Z-100 (not PC), 4) H/Z-100 (not PC) and PC Compatibles, 5) PC Compatibles. Under each 'machine-type' category, the products will be listed in alphabetical order by their title. Following the title will be the HUG part number, the operating system under which the product runs, a short two or three word synopsis, and finally, the price. Now, at a glance, you'll be able to tell what software products are available for your particular computer system.

Ok, now for the 'biggy'! Having written a few articles myself, I'm aware of the amount of work that goes into each writing, and feel that each author should be awarded appropriately and competitively. I've taken a good long look at what we've been offering as an incentive in our major article program, and really didn't come up with very much to see! So, as of May 1st of this year, any authors submitting major articles (2000 words or more), will have their choice of one of the following:

1. Any single Heath/Zenith Software Product, excluding educational products or packages.
2. $250-$400 Cash (amount determined by article content, size, style, etc.)
3. $350-$500 Worth of HUG BUCKS (amount determined by article content, size, style, etc.)

Following the publishing of the article, the author will be notified of the amounts he/she is eligible for, and then allowed to make his/her choice. All other policies regarding the major article program still apply. Any would-be authors should refer to the January 1987 issue of REMark for information on writing and submitting articles to REMark.

The HUG Binders (P/N 885-0004) are finally coming to an end. Once the remaining stock is used up, they will no longer be available, and will not be included in the new membership package. This will also eliminate the need for the three 'unprofessional' .25 caliber holes someone's been shooting in each issue of REMark.

Time is drawing short for this year's International HUG Conference, and I'm presently in the process of lining up speakers for our discussion groups. If you think you'd like to talk about one of your favorite topics (computer related) at the conference, drop me a line, or give me a call. On a similar note, if there's some topic you'd like to hear about, or a topic you particularly found informative last year, I'd like to hear from you, too. Every year we try to make our HUGCON bigger and better than the last, and this year is no exception. As I've mentioned previously, we plan to have Dick Hardwick as our keynote speaker this year. Dick is an expert in his own field, the field of comedy. He's listed in the Guiness Book of World Records as being the star of the longest running comedy stage show at the Golden Horseshoe Review in Disneyland and has appeared in several movies. I can personally guarantee a night to remember for each and every dinner attendee.

Jim

Jim Buszkiewicz
Managing Editor

# BUGGIN' HUG

### dBASE II Version 2.43*

Dear HUG:

I am using dBASE II version 2.43* on a Z-89 with Z-37 drives. I find that the dSORT utility that was distributed with version 2.43* consistently fails at exactly the 50% point in the sort with a message:

```
d33 An undetermined error occurred^2
Sort cannot continue
```

Phone calls to Ashton-Tate netted the statement that they had never heard of this problem reported with this equipment. An early release of 2.43 did have a similar problem for Osborne and TRS-80 machines, but that was fixed before mass shipment.

Is no one even trying to use dSORT or version 2.43* on Heath/Zenith 89s? The thing appears to halt with the error message no matter what the size or content of the dBASE file, every time. Did no one buy the upgrade to fix all the bugs in 2.41? Did no one complain when it didn't work? (Apparently not!)

Complaints to Ashton-Tate do have some effect. The "bug" in version 2.41 where "edit record 7" rather than "edit 7" would silently go and edit the last record you had accessed, not necessarily record 7, and give no syntax error was fixed in 2.43*. This fix occurred a few months after they got unquestionably the nastiest phone call they had EVER received, after I discovered (had a record clobbered by) this BUG.

If there is anyone actually using version 2.43* (8 bit only), I would like to hear of any problems you are having. Until my money or patience runs out, I will pay ten dollars a bug reported to me. It has to be dBASE II version 2.43*, CP/M-80 or the utilities supplied. The bug has to be internal to these, not your programming error. I have to be able to repeat it, you have got to explain how to do it. It has got to be new, duplicates or known published bugs don't count. Ashton-Tate claims that only 3 or 4 are known.

I still swear that there is the "old dBASE end-of-file bug" lurking in 2.43*, I have been bitten by this with a "sum to, delete for, append, edit" sequence one time, but have not been able to repeat it. Contrary to Ashton-Tate's hotline I DID NOT, 1) "shut down the machine without exiting dBASE", 2) "hit some WRONG KEYS or SOMETHING!!!", or 3) "static electricity did it", i.e. wrote a ^Z in the middle of my file!

Thanks,

Don Taylor  
12270 SW Center Street #63  
Beaverton, OR 97005

### MACAT: A Disk Cataloging System

Dear HUG:

Since you published my article, MACAT: A Disk Cataloging System (August '86), I've added an H-158 to my computing stable. I switched my catalog of files from my Z-100 to the 158 and discovered MACAT does not run faithfully. I delved into the system's works and eliminated the anomalies. Thinking others may encounter the same disappointments as I, let me share the fixes with your readers.

Only the batch files are involved; no modifications to the BASIC programs are required. In all the appropriate batch files make the following changes:

Change lines reading "BASICA WHAT" to read — "BASICA WHAT/F:5". This is necessary because GW-BASIC for the PC series drops the files opened by default from 5 to 3.

Change "HFM" to "HFM2" or rename "HFMF2.COM" to "HFM.COM".

Change lines reading "ATTRIB WHAT.BKP" to read — "ATTRIB -R WHAT.BKP" to accommodate a difference between MS-DOS 2.x and MS-DOS 3.x.

Change lines reading "ATTRIB WHAT.BKP R" to read — "ATTRIB +R WHAT.BKP" for the reason given above.

Sincerely,

Louis M. St. Martin  
860 Hillcrest Drive  
Pomona, CA 91768

### *Fastcard///*

Dear HUG:

In the December '86 REMark, Peripheral Marketing, Inc. of Scottsdale, AZ advertised *Fastcard///* on page 3. This board originated with Thesys, Inc., also in Scottsdale and later also Nogales, AZ. My experience with this device and the response of PMI who appear to be the successor to Thesys has not been good.

I purchased one from a local distributor about a year ago. It was installed in my H-151 with 2 megs of 256K DRAMS. Excellent support solved an initial problem and it worked well. After about six months, the board failed and it was returned for repair. The 800 phone was gone.

The board came back in about four weeks with a new ROM, software and manual. The board cache wouldn't serve either of my hard drives, but it worked well otherwise. (It did give a performance test of access improvement on the C drive even though it ended with "no hard drives attached".)

After about six weeks, the board failed on access by a program which recognizes expanded memory. The printer went crazy and data files on drive C were trashed.

The *Fastcard///* was pinpointed as the culprit. An hour and a half on the phone (part my money, part their's) with numerous setup changes and reboots yielded only erratic results and it was decided to return the board to them.

The *Fastcard///* came back from PMI at the Thesys, Nogales after eight weeks with a note that the BIOS ROMs in my IBM PC should be updated. Three calls with two promises of call backs have not provided any answer as to what the problem was or to what their fix refers.

Regards,

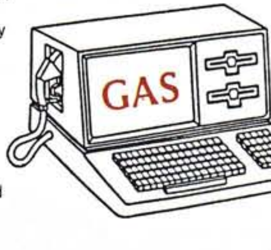
Al Cohen  
Consulting Engineer  
725 Pine Street  
Deerfield, IL 60015

### BSR X-10 Powerhouse System

Dear HUG:

I purchased a BSR X-10 POWERHOUSE system and ran into some difficulty after finding out that the software package that was included was fixed to use the COM1 port. This caused me a little heartburn in that I've dedicated COM1 for use with my modem and that there was no type of configuration utility included to change this. After spending a few long hours in DEBUG and a short time in editing, I've written a simple set of files that will allow you to configure the program X10.EXE to use any COM port from 1 through 4.

The method that I chose uses the batch and I/O redirection facilities of MS-DOS along with DEBUG. This should help other readers with the same type of problem and I hope that others will follow this method



when releasing patches to other programs to the public. In addition, maybe Pat Swayne or someone else might use this information to run the software under ZPC2 on the H/Z-100.

In closing, I would like to thank all those who have supported HUG and REMark over the years and throw in a request for anyone who has written a similar program for the X-10 on an H-8. (Yes, I am still a proud owner and it does see regular use.) As long as I'm asking, I'm also interested in any color graphics software for the HA-8-3 running CP/M.

Thank you,

Joseph B. Travis
10336 Great Rock Road
Santee, CA 92071

```
FILE: XCONFIG.BAT

echo off
if "%1" == "COM1" set install=x1Øcom1.dbg
if "%1" == "COM2" set install=x1Øcom2.dbg
if "%1" == "COM3" set install=x1Øcom3.dbg
if "%1" == "COM4" set install=x1Øcom4.dbg
if "%install%" == "" goto error
echo Configuring X1Ø.EXE for %1
ren x1Ø.exe x1Ø.bin
debug x1Ø.bin <%install%
ren x1Ø.bin x1Ø.exe
set install=
echo Configuration complete
goto done
:error
echo Usage: xconfig COMx
echo where x = 1 through 4
:done

FILE: X1ØCOM1.DBG   FILE: X1ØCOM2.DBG

e31c4 fb Ø3         e31c4 fb Ø2
e31d2 f8 Ø3         e31d2 f8 Ø2
e31df f9 Ø3         e31df f9 Ø2
e31ed fb Ø3         e31ed fb Ø2
e31fa f9 Ø3         e31fa f9 Ø2
e320e f8 Ø3         e320e f8 Ø2
e326a fd Ø3         e326a fd Ø2
e328e f8 Ø3         e328e f8 Ø2
e32ff fd Ø3         e32ff fd Ø2
e3338 f8 Ø3         e3338 f8 Ø2
w                   w
q                   q

FILE: X1ØCOM3.DBG   FILE: S1ØCOM4.DBG

e31c4 eb Ø2         e31c4 e3 Ø2
e31d2 e8 Ø2         e31d2 eØ Ø2
e31df e9 Ø2         e31df e1 Ø2
e31ed eb Ø2         e31ed e3 Ø2
e31fa e9 Ø2         e31fa e1 Ø2
e320e e8 Ø2         e320e eØ Ø2
e326a ed Ø2         e326a e5 Ø2
e328e e8 Ø2         e328e eØ Ø2
e32ff ed Ø2         e32ff e5 Ø2
e3338 e8 Ø2         e3338 eØ Ø2
w                   w
q                   q
```

# "Now I Can Handle More Of You!"

Hi, my name is HUGPBBS, I know that many of my friends have tried to reach me, only to hear that annoying buzz... buzz... buzz... busy signal. It's hard to believe that I've become so popular in so short a period of time. Well, now I'm multi-user, and can respond to more of you! My human figured out a way to clone me, and now I'm not sure which one of me is the original! We're all still available 24 hours a day, now with over 20 megabytes of free software for downloading. Most of my software is for the PC Compatible computer, however I do have a lot to offer the H-8, H/Z-89, and H/Z-100 systems. I've got games, utilities, business, scientific, home finance, real estate, point-of-sales, graphic applications, and even HDOS version 3.0 just to give you an idea. In addition to all this software, users have been using my message base to exchange ideas, get on line help, and even sell things. Remember, there's no charge for my service to HUG members, and I can be as slow (300 baud) or as fast (1200, 2400 baud) as you like. I'm a little touchy as to 'word' type, so make sure you're set to 8-data bits, NO-parity, and 1-stop bit. Call me at (616) 982-3956, type a carriage return several times to get my attention, and register your human. Just leave his first and last name, his HUG ID number, and a password (up to 16 characters). Although not as interesting to talk to, your human can call mine (Jim Buszkiewicz) at (616) 982-3837 and register in English.

# Review Of MS-DOS V3.2

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

In the latter part of July and August, I had the opportunity to Beta Test MS-DOS V3.2 (the PC version) for Zenith Data Systems. The review of the documentation was done in September. In this article, I would like to share with you the new features of MS-DOS V3.2 versus V3.1 from the general user's viewpoint rather than from the programmer's viewpoint.

The reason for not covering the inner workings of MS-DOS V3.2 is because I did not have access to a "Programmer's Utility Package" for V3.2. So I will not cover the MS-DOS interface level changes at this time. When I get that information and it appears that there have been some significant changes, I will share that with you via another article.

What I will be covering in this article are the new commands, commands that changed from V3.1, some changes in the hard disk command area, and some miscellaneous information. Before I forget, let me say that one of the major enhancements in MS-DOS V3.2 is for the support of the 3-1/2" floppy disk drives that are part of the new Zenith Z181 microcomputer. The capacity for those drives is 720KB.

**New Commands**

The first new command in alphabetic order is the DSKSETUP command. This command has three major functions. The first is to partition your hard disk into up to four partitions. You may recall that in V3.1 there was an automatic partition assignment flag for hard disks and this was part of the CONFIGUR command. This function has been removed from CONFIGUR and it is now a part of DSKSETUP. The automatic assignment flag tells MS-DOS to assign the first MS-DOS partition on your hard disk to the next available drive letter. This option only supports one partition. If however, you have multiple partitions, then you select manual assignment which means that after the first MS-DOS partition, all other partitions must be assigned a drive letter before they can be accessed. That is accomplished with the ASGNPART command. This function is the same as that in MS-DOS V3.1.

The second function of the DSKSETUP command, is to configure your floppy disk drives. Basically, you tell MS-DOS whether your floppy disk drive(s) is a 5-1/4" 360KB, 5-1/4" 1.2MB, or a 3-1/2" 720KB drive. You indicate what each of the 4 possible floppy disk drives are.

The third function of this command is to enable/disable format protection. The purpose of this feature is for protection against accidentally reformatting a partition and losing all your valuable files. You can enable or disable the protection for each of the partitions that you have. If the partition is enabled, you will not be able to reformat that partition until you run DSKSETUP again and disable the protection for this partition.

The next new command for the Zenith version of MS-DOS is the GRAPHICS command. This command was available in PC-DOS back in V3.00. This command allows the contents of a graphics display screen to be printed on a printer when using a color/graphics monitor adaptor. This is a memory resident program that is activated via the SHIFT-PrtSc key sequence. GRAPHICS allows one to print color screen images depending on the colors of the printer ribbon. For example, if the ribbon is black, up to 4 shades of gray are printed. If the ribbon is RGB (Red, Green, Blue, and black), then the image is printed in these colors. The other colored ribbon supported is the CMY ribbon (Cyan, Magenta, Yellow, and black).

REPLACE is the next new command. The REPLACE command allows one to update files on a hard disk with new versions in a very easy manner. This command has a number of functions. First, by default, this command replaces files in the destination directory with files in the source directory that have the same name. Wildcard characters can be used in source file names.

REPLACE has a number of switches which restrict what can and cannot be replaced. The /A switch adds files that exist in the source directory, but not in the destination directory, to the destination directory. In other words, the /A switch adds new files to the destination directory instead of replacing any.

The REPLACE /D switch replaces files in the destination directory only if the source files are newer than the corresponding destination files. The /R switch replaces read-only and unprotected files. If this switch is not specified, trying to replace a read-only file will result in an error which stops REPLACE. The /S switch searches all subdirectories of the destination directory while it replaces matching files. Please note that REPLACE never searches subdirectories in the source path. Also, please note that REPLACE will not replace a hidden file or system file.

The next command, the RTCLOCK command, is a bit confusing. The purpose of this command is to set and read a real-time clock device, and then set the MS-DOS system clock to that date and time. The documentation does not specify what clock devices are supported. I have a Hyperclock from Hyper-Micro Inc. I tried to read the date and time on my Z-160 using this command and it seemed to work OK except the year was off by a few years. This was reported to Zenith, and hopefully will be corrected before it is released.

XCOPY is new and it could be looked at as an extension of the COPY command. It copies files and directories, and lower level directories if they exist. XCOPY will work on both floppy disk drives and hard disks. The subdirectory structure on the source device is duplicated on the destination device. This means that XCOPY will copy a whole tree structure from one device to another if the destination device has enough capacity.

XCOPY has a number of restrictions and options. Hidden files cannot be copied by XCOPY and XCOPY cannot copy to a read-only file on the destination device. Switch /D selectively copies files based whether they were modified on or after a specific date. Switch /E copies any subdirectories even if they are empty. Archive files can be copied using the /A switch. The /S switch copies all directories and lower level subdirectories.

With all these capabilities, XCOPY could be used as an alternative to the BACKUP command, but there are some limitations. One limitation is that XCOPY cannot copy a file that is larger than the destination device. BACKUP will allow files to span disks. Also, it was mentioned before that hidden files cannot be copied, as well XCOPY cannot copy to a read-only file.

The last new command is ZCOM. This command transfers files between two computers using direct serial communications or modems. In order to transfer files using this command, both computers must be running ZCOM. One must run in the 'server' mode, while the other must run in the 'user' mode. A number of switches are available which allow the user to enter a password, specify whether a Hayes compatible modem is being used, and whether COM2 is used rather than COM1. ZCOM itself has a number of commands, such as CONNECT, DISCONNECT, QUIT, ABORT, RECEIVE files, TRANSMIT files, set BAUD rate, PASSWORD, etc.

### Changed Commands

That covers the new commands. Now let me cover the commands that changed from MS-DOS V3.1. The first command is the ATTRIB command. This command now supports another file attribute flag called the 'archive attribute' flag. This flag is used by the BACKUP, RESTORE, and XCOPY commands to control a selective 'backup', 'restore', or an 'xcopy' on files that have been modified. In other words, it lets you select which files you want to copy or restore by setting/clearing this archive attribute flag.

Another switch has been added to the COMMAND command. The /E switch lets the user specify the size of the MS-DOS environment in bytes. The range is 128 to 32768 bytes.

I mentioned before that one of the functions of CONFIGUR was removed and added to the new command DSKSETUP. That was the Automatic Partition Assignment Flag function. The other functions remain and seem to be the same as those in V3.1.

Three switches have been added to the DISKCOMP command. They are as follows. The /L switch allows DISKCOMP to compare just the first side of a floppy disk even if the drives and disks being used are double-sided. Switch /8 compares only the first 8 sectors per track, even if the disk contains 9 or 15. The third switch is the /R switch which causes the computer's speaker to sound each time DISCOMP prompts you for some input.

Just as with the DISKCOMP command, DISKCOPY too has three switches added. In previous versions of MS-DOS, DISKCOPY automatically formatted the destination disk first before doing a copy. It does not do that in MS-DOS V3.2. If you want that to happen, then you must specify the /F switch. Otherwise, DISKCOPY will copy the contents of the source disk to the destination disk with the files on the destination disk possibly being fragmented. Fragmentation occurs when disks have had many files created and deleted over time. Then this disk is used as the destination disk. The files currently residing on the destination disk are erased and replaced with those from the source disk. In a fragmented situation, the destination disk is not a track-for-track copy of the source as would be if the destination disk were reformatted before the copy operation. If you want DISKCOPY to work as in the past, specify the /F switch.

Switch /I copies only one side of a disk even if the disk is a double-sided disk. The /R switch sounds the computer's speaker when DISKCOPY requests input from the user.

A number of changes have been made to the FORMAT command. The one I like is the way that it now displays the track and head numbers as the formatting operation is in progress. Now you can see what is going on. Because information is now displayed on the screen while formatting, the CTRL-C and CTRL-BREAK can now be used to abort the operation. This was not possible before.

The /M switch in V3.1 has been changed to /L. It appears to work the same, just a name change. A new switch has been added to support the 3-1/2" disks, /D. This switch is used in conjunction with either the /L or /8 to force FORMAT to format all 80 tracks on a single-sided disk and/or 8 sectors per track. Without this switch, the /L would only format 40 tracks. Normally, a double-sided 3-1/2" disk is formatted 9 sectors per track and 80 tracks per side. The /D switch is used for single-sided disks.

The /C switch, that was used in conjunction with the /M switch in V3.1 with respect to formatting double-sided disks as single-sided disks, has been removed. Its function has been incorporated into the replacement for /M, which is /L.

A couple of switches have been added to the ZSPOOL command that was introduced in MS-DOS V3.1. The /A switch indicates that the buffer to be allocated is in expanded memory rather than in the main RAM memory. Switch /P:n has been added to allow the user to specify a printer 'priority'. The parameter 'n' specifies the number of characters that can be sent to the printer every clock tick. This value can be 1 to 100. Default is 10. Slow printers should have this value set at the low end of the range, so that performance of the system will not be degraded. Faster printers can set the value to a higher figure to increase printer output but yet not degrade the system. There are no fixed values that can be recommended for best performance. That is something the user must do by 'trial and error'.

### Hard Disk Support

In the area of hard disk support, there is a major enhancement to the PREP command. In the Z-200 ROM, there is a new command (facility) called SETUP that allows a user to set some facts about the disks that are being used, such as the size and number of floppy disks, the specific hard disk configuration used, whether the boot operation should be automatic on power-on, whether the boot operation should be from hard disk, etc. Some of these functions are performed by dip switches and/or jumpers in earlier PC type machines.

Until MS-DOS V3.2, Zenith had a few standard types of hard disks that it supported, and I believe the largest disk in the SETUP configuration, for example, was 65MB. Recently, I had the opportunity to install an 80MB hard disk in a Z-200. MS-DOS V3.1 would not allow me to install this disk or at least take advantage of the full disk capacity. This is where the enhancement to the PREP command solved the problem.

Zenith calls hard disks that are not described in ROM 'enhanced' disks. The PREP command has a switch, /Q, which allows the user to describe his/her disk via a number of parameters which are prompted by the ROM. With this switch, I was able to describe my 80MB disk and the installation went quite well. I am now up and running with 3 partitions.

PREP with the /Q parameter prompts the user for a number of facts about the enhanced hard disk that is to be installed. The information requested is the same for both the Z-200, as well as a Z-100-PC, except that for the Z-200, the user is asked for the zone used on the disk for shipping, while for the Z-100-PC the user is asked for the reduced write current cylinder. The other information requested is as follows: type of hard disk drive (cartridge or fixed), whether servo track information is needed, the number of cylinders used for the disk, the number of heads used, and the starting write precompensation cylinder.

What all this means is that Zenith computers can now support nearly any type of hard disk drive.

### Miscellaneous

The MS-DOS line editor, EDLIN, has four new commands as follows. The COPY command allows one to copy a range of lines to a specified line number. The MOVE command moves a range of lines to a specified line. The PAGE command allows one to change the number of lines displayed per page. The default is 23. The fourth new command is the TRANSFER command. This command transfers the contents of another file into the file that you are editing at the specified line number.

There is a couple of new CONFIG.SYS commands. DRIVPARM defines parameters for block devices that will override the original MS-DOS device driver settings. A number of block-type devices are supported with this command: 320/360KB disks, 1.2MB disks, 720KB disks, 8" single-density, 8" double-density, hard disks, and tape drives. The information that can be provided in addition to the device type is as follows: logical drive number, maximum head number (range is 1 to 99), number of sectors per track (range is 1 to 99), and the number of tracks per side (range is 1 to 999).

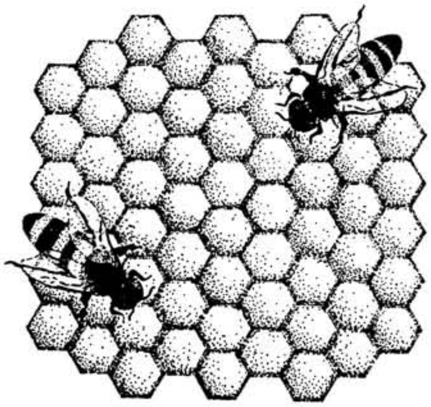
The second new command is STACKS. This command lets the user define the size of the stack used by MS-DOS, by specifying the number of stack frames (8 to 64) and the size of the stack frames (32 to 512). The defaults are 9 and 128, respectively. Each time an interrupt occurs, MS-DOS allocates one stack frame. The frames are released after the interrupt has been processed. The default allows up to 9 interrupts deep to occur before getting a stack error condition. For most situations, the default value is sufficient. If a stack error condition occurs, try setting this to a larger value to eliminate the problem. It may also be necessary to increase the size of the stack frame depending on what the interrupt routine does. This is just something to be aware of.

A new installable device driver, DRIVER .SYS, has been added to the two previously available device drivers: ANSI.SYS and VDISK.SYS. DRIVER.SYS supports external disk drives. I suspect one of the main reasons for this driver was to support external 3-1/2" disks. One loads a copy of DRIVER .SYS for each external drive. Parameters that can be specified with DRIVER.SYS are: device type (same as those I included above with the CONFIG.SYS DRIVPARM command), the maximum number of heads, the number of sectors per track, the number of tracks per side, and whether the device is nonremovable. For example, if one wanted to add an external 3-1/2" disk to a computer that contains 2 internal floppy drives, the following would be added to the CONFIG.SYS file: "DEVICE= DRIVER.SYS /D:02". The /D: is the switch that specifies the device type. From the description of the DRIVPARM command above, you can see that device 2 is a 720KB drive which is the 3-1/2" disk.

That covers the changes that I am aware of. Hopefully, this gives you some idea whether it would be worth your time and dollars to upgrade from MS-DOS V3.1 to V3.2. For those of you still at MS-DOS V2.11 (H/Z-100-PC owners) or MS-DOS V2.21 (H/Z-100 owners), I would recommend the upgrade to V3.2 for a number of reasons. Some of those reasons are described above and some have been described in a number of MS-DOS V3.1 articles over the past year.

Happy computing . . .

# Beyond IBM PC Compatibility:



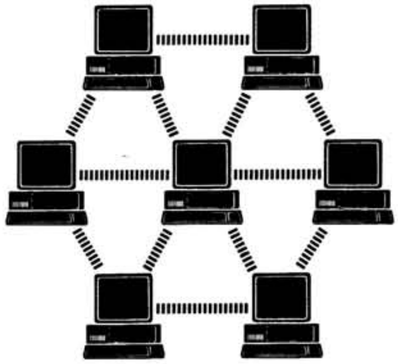
Z-100 users who are ready for PC compatibility, should keep one thing in mind. The future. New, state-of-the-art PC software is creating ever greater performance demands on computer hardware. Large scale, memory intensive programs for networking, graphics, number crunching and communications are quickly becoming the rule, not the exception. But software is only as good as the system that runs it.

## EasyPC will run it all.

Even the newest copy-protected best sellers. That's because you actually boot up PC hardware, not software translation routines.

## EasyPC runs faster, too.

Faster than an IBM PC, faster than a Z-150, and faster than any other emulation scheme you can buy.



*EasyPC will support a fully integrated Local Area Network System (LANS) of Z-100's, PC's and AT's communicating at 115Kbaud.*

## Networking is now a reality for Z-100 users.

EasyPC will fully support a network of up to *sixteen* Z-100's, PC's and AT's. EasyPC is the core of an integrated system of components that includes RAM expansion, I/O, math co-processing, hard disk subsystem and more. For serious users, it's the only choice.

## Ask to see an EasyPC in action.

Visit your Heath/Zenith Electronics Center or independent UCI dealer and see how easy it is to turn your Z-100 into a PC compatible system. For the name of your nearest dealer, just call 800-UCI-COMPUTER.

EasyPC is a trademark of UCI, Corp. Z-100 and Z-150 are trademarks of Zenith Electronics Corp.
IBM and IBM PC are trademarks of International Business Machines.

UCI CORPORATION

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

# C_POWER

# Part 3

***John P. Lewis***
*6 Sexton Cove Road*
*Key Largo, FL 33037*

This article will serve to enhance the reader's knowledge of random records by furnishing a complete code module for insertion into the program we listed in Part 2. We will then examine each line of code in an effort to provide the reader with a thorough understanding of how the computer writes random records and what must happen in order to append a record.

The included code module will, in addition to providing a means of creating a new record, present the input module in an attractive format which is easy for the lay–person to use and will enhance the value of the program. These routines will rely heavily on the "funclib.c" files which we developed in Part 1 of this series, particularly the "locate" and "gofor" routines.

I promised you additional routines for your "C" library as an added benefit of this series. What we are really going to do in this issue, is modify one of the library routines included with "C/80" to enable our program to function the way we want it to. This is one of the features of the language which make it so powerful. The modified function will be used in Part 4 of this series when we develop the option to "Search for a record(s)."

OK, let's get to work. First, make a backup copy of "CPOWER.C" (program listing from Part 2). Now make an additional copy on a separate disk which you have labeled "CPOWER development". This will be our working disk for use in writing our new version of "CPOWER". Retain the first version (as insurance) until you have the new version completed and tested. Other files needed on this disk are: "FUNCLIB.C", "PRINTF.C", "SEEK.C" and "CLIBRARY .ASM". We created "FUNCLIB.C" in Part 1 of this series and the other three files are part of the C/80 package. Now use your word processor or text editor to insert the code listed below into the appropriate places within our program skeleton. When you have finished this project, the bulk of the new code should be between "case '3':" and "case '4':". This should completely replace any previous code found in this area. You will also need to add the new functions, opnforrd() and o_pen(), that have been included. There are two new definitions, as well as a new include file at the top of the listing which will also have to be added.

Study this listing carefully, there are some interesting code fragments here for those of you who are new to the "C" language, in general, and random records, in particular.

```
/* --- "CPOWER" by John P. Lewis --- */

#include "printf.c"
#include "funclib.c" /* routines cls, locate, fetchc and gofor from "CPOWER"
                        (part 1) */
#include "seek.c"  /* one of random access library files */
#define DEL 127   /* this char used for an "end of records" delimeter */
#define SIZE 2048  /* size of future "search" area */
#define DELAY 10000 /* number of iterations in delay loop */

unsigned fd, offset; /* global variables to be used by functions and main */
opnforrd ()
   {
   fd=fopen("data","r"); /* open "data" file in read mode */
   if ( fd == 0 )
     {
     fd=fopen("data","w");seek(fd,0,0); /* if file does not exist, create */
     fprintf(fd,"%c%c",DEL,EOF);seek(fd,0,2);fclose(fd);
       /* one & insert */
     fd=fopen("data","r");  /* delimeter */
     }
   }

o_pen()
   {
   fd=fopen("data","u");  /* open file in "u" (update) mode */
   }

main ()
  {
    int i, k;
    char men[2], compny[25], name[25], street[35],
      city[33], string[SIZE];
    char phone[10];
    do
    {
    cls ();putchar(27);putchar(70);locate (4,5);
      /* enter graphics mode */
    printf("ppppppppppppppppppppppppppppppppppp");
    printf("ppppppppppppppppppppppppppppppppppp");
    putchar(27);putchar(71);  /* exit graphics mode */
    printf("\n\n\t         Please ....");
    printf("\n\n\t              1. Search for record.");
    printf("\n\t              2. Review existing record names.");
   printf("\n\t                3. Create new record(s).");
   printf("\n\t                4. Print entire mailing list (to printer).");
   printf("\n\t                5. Sort list (on specified field).");
   printf("\n\t               6. Exit to operating system.");
   printf("\n\n\n\t   Enter the number corresponding to your choice ");
   gofor(men,1);
   switch (men[Ø])
    {
    /* "case 1 & 2 from "CPOWER, PART2" should be in this position */
 case '3':
    cls();locate(6,6);printf("Please be patient, I'm looking");
    locate(8,6);printf("for the end of this file ");
    for ( i=Ø; i <= DELAY ; ++i );
    opnforrd();offset=Ø; /* open file for read */
    seek(fd,offset,Ø);   /* set pointer tc byte Ø */
    while ( string[Ø] != DEL )    /* test for delimeter */
       {
       for (i=Ø; i <=127; ++i)
          {
          string[i]=getc(fd);  /* get a record */
          }
       offset +=128;seek(fd,offset,Ø); /* bump pointer to next record */
       }
    offset -= 128;fclose(fd);string[Ø]='Ø'; /* Purge string[Ø] */
  for ( ; ; )
     {
       cls ();locate (4,10);printf("Mail List\n");o_pen
         ();locate(6,6);
       printf("Enter Company (if applicable) __________¦");
       locate(8,6);
       printf("Enter name _________________________¦");
       locate(10,6);
       printf("Enter Street ___________________________¦");
       locate(12,6);
       printf("Enter City, State, Zip _________________¦");
       locate(14,6);
       printf("Enter Phone __________¦");
       locate (6,36);gofor(compny,24);locate(8,17);
         gofor(name,24);
       locate(10,19);gofor(street,34);locate(12,29);
         gofor(city,32);
       locate(14,18);gofor(phone,10);
       seek(fd,offset,Ø);
       fprintf(fd,"%-25s%-25s%-35s%-33s%-10s",compny,name,street,city,phone);
       seek(fd,offset+128,Ø);
      fprintf(fd,"%c%c",DEL,EOF); /* insert delimeter & end of file marker */
       seek(fd,Ø,2);fclose(fd);locate (16,4);
       printf("Enter Ø to exit, any other character to continue ");
       gofor(men,1);
       offset +=128;
       if ( men[Ø] == 'Ø') /* test for more input actvty */
       break;
     }       /* end of for loop */
   break;   /* two "break" statements necessary here, one for "for" loop and
                one to delimit the "case" */
   /* "case" 4 & 5 belong in this position */
    }  /* end of switch loop */
  } /* end of menu do - while loop */
  while ( men[Ø] != '6');
}   /* end of main */
```

Now we have acquired some meat on the bones of our program skeleton. It is not terribly useful as it stands, but it writes and appends random records with a lot of class! Soon, with a little luck and a lot of persistence, we will have a program that we can use as is or modify to do many different jobs for us.

Starting at the top of our listing, you will find the additional define statements and one more include directive. Putting the define statements here enables the programmer to change the value of a variable at the top of a program, without searching through the entire listing for a constant. A useful ploy for those of us who are a bit lazy. The two define statements are used to, in the first case, create a 2K buffer which will be used for a couple of different jobs, and the second define statement is used to set the delay in a timing loop (allows reading a message). The new include statement points the compiler towards the "seek" library file. Seek is a routine which is used extensively when dealing with random records.

Next, we encounter two variables which are being declared outside of "main". Why here? These are "Global" variables "known" to the rest of the program, both functions and main. We declare them as global in order to avoid having to add to the complexity of the program by passing values back and forth between the functions they are found in and the body of the program. If a variable is declared within a function, it is "known" only to that function. Hence, "fd" and "offset" are known to the entire program.

Moving along, we find two new functions, "opnforrd" and "o_pen". The first of these is designed to open file "data" in the read mode. If success eludes us (fd is equal to zero, file does not exist), the file is opened in the "write" mode, a delimeter is inserted in byte zero of the file, EOF (CP/M end of file) in the next byte, and then the file is closed in the write mode before it is opened in the read mode. This is a way of insuring that the file is indeed opened whether one existed before or not, and if it is a new file, we have also identified the first byte of the first record.

Our next function, "o_pen", lives only to allow file update. This mode is used whenever we are writing to an existing file, and that is what we have been trying to do all along. At this point of program development, we use this bit of code only to allow us to append a record. Later, we will invoke this function to enable the editing of a record. No test is made for successful file opening here since this routine is only called when we have previously opened the file in the read mode. Thus, we are assured there is such a file.

You will note the existence of "main" in this listing even though it is redundant. I left it here only for clarity and you don't have to type this in again! Also, it is here for my convenience, I can test this program, just as you see it, for any possible errors before submitting it to REMark, thereby possibly foiling Murphy's Law.

Now we have arrived at the point of program evolution that you have been waiting for (I hope). You will discover "case '3':" hidden in all the rest of this code. We have incorporated a user message right at the top of this routine that asks the user to "Please be patient, I'm looking for the end of this file ". Below that, you will find the delay loop that we discussed earlier using 10,000 iterations to allow the user to read this message before going on to clear the screen after the routine, which may introduce the delay that we are asking the user to be patient about. If the existing file is rather short, the delay will be brief, but we can't guarantee this, hence the message.

Next we find the "opnforrd" function discussed earlier followed by code which resets "offset" to zero, and then sends the computer read/write head to the first byte of this file [seek(fd,offset,0)]. Now we encounter a "while" loop with a nested "for" loop. The while loop causes a test of "string[0]" for the "DEL" character each iteration, while the "for" loop causes character retrieval of this record, building string [i] into our record. We increment the offset of the pointer (by 128) each time we fall out of the "for" loop and test the value of string[0] at the same time. When this test is NOT successful ( string[0] != DEL ), we fall out of the "while" loop and decrement the pointer by 128 bytes. The logic of this arrangement might seem a bit obtuse, but the proof is in the pudding as the saying goes and the algorithm does work! Another cliche might be used to describe this code fragment, "don't fix it if it ain't broke".

There are other ways to do the same job and some of them are more "elegant" I'm sure, but I'm the author of this little gem and the part that I do like is that I can understand it. The idea here is that the computer finds our delimeter in byte 0 of a record and adjusts the offset accordingly. Now it is ready to append our random file. The file is open in the read mode, so first we close the file, purge string[0] (in case we need to use it again), and then open the file in the update mode by using the function, "o_pen", which we discussed earlier. We also enter an endless loop at this time for the purpose of data input. An endless loop in "C" is created with the statement: for(;;). The computer will execute this loop until it encounters a "break" statement. We will test the user input at the end of record input to escape the loop.

Next, we encounter the input module. This has been designed to facilitate user input in such a way that it makes the user aware of the maximum number of characters in the field being entered. This routine also prints a return and locates the cursor on the next field when the entry parameters are exceeded. For instance, when the user is inputting a name in the appropriate field and he or she types the twenty-fifth character, the input is terminated for that field and the cursor is placed on the next input field. The cursor activity is a reflection of what is happening internally at the same time, since the string being entered is terminated by "gofor" when the 25th character is entered. Thus, we have an input routine that is nearly foolproof.

The main reason for the creation of this function (gofor) was not merely to aid the user, but to avoid some rather erratic results which were encountered when the declared string length was exceeded during user input. If we declare name[25] and input twenty-six or more characters using another input function, the characters will overrun their assigned space in memory. This can cause some rather unpredictable results. Remember Murphy's Law? When writing random records in BASIC, much of this is academic since the housekeeping is performed for us. "C" is not so forgiving of sloppy practices, it requires that we operate within the declared parameters.

Now that we have all this data, we need to write it to the disk in the proper format. Here again, the programmer must do some of the work that would be performed by BASIC if we were using that language. However, it is not too complicated and we will know a lot more about what goes on internally when writing random records when we get through this routine.

The program must provide two very important pieces of information in order to accomplish this job. It must position the read/write head at the beginning of the appropriate record, and it must format the writing operation precisely. As an example, when writing to the third record in a file, where each record is 128 bytes in length, the write operation must start at byte number 256 within that file. Each byte of each record must be placed where we can go back and find it, and that is what we are getting ready to do as part of this routine. The statement, seek(fd,offset,0); will force the placement of the read/write head at the beginning of the record starting at byte number OFFSET. The next job that must be performed by the program is the correct placement of each field within the individual record. How can we do that without a field statement similar to that used by BASIC? I thought you would never ask!

The printf statement in "C" has some very sophisticated features that we will put to work to accomplish this job for us. We will use fprintf which is the disk write version of printf. The statement, "fprintf(fd,"%-25s %-25s",compny,name); will cause the writing of the string "compny" in the first twenty-five bytes of this record, and the string "name" in the succeeding twenty-five bytes of the same record assuming a record at least fifty bytes in length and the read/write head was positioned at the beginning of the record.

The above code will also force left justification when writing these strings to the disk. This gives us absolute control over formatting of data input, which in turn enables the finding of any field within a record or even a single byte of any given record. For example, the statement, "seek(fd,offset+25,0);" will position the read/write head to commence writing beginning at byte 0 in the name field of the above example, and the record number will be determined by offset which must be a multiple of the record length. The offset must always be incremented or decremented by the number of bytes in a complete record. The failure to do this will result in the erratic reading/writing of random records. We are soon going to deal with a routine which does just that, cause offset to be equal to a number other than a multiple of the record length, but I'm going to show you how to restore it to it's proper value before proceeding. This is part of the housekeeping that I mentioned previously.

Since we declared offset to be an unsigned integer, we can divide it by the record length and then multiply it by the record number to arrive back at a multiple of the original record length. To wit, assume that we have incremented offset by a value equal to the "compny" field (25), the record length is 128 bytes and that we are dealing with the third record. This would cause offset to be equal to 256+25 or 281. The code fragment: "num=offset/128; offset=num*128; would result in restoring the value of offset to 256. This is a very important concept to remember when dealing with random records.

I have strayed a bit from the topic we were covering, but the successful use of random records in "C" hinges on very precise operations. Hence the dissertation.

Back to our record writing. We have now arrived at the line of code which writes our record to the disk (fprintf, etc.), and we now see how the data is written in the proper format. The next operation will serve to enable us to find the first vacant record the next time this routine is used. Remember, we wrote over our record delimeter when writing the last record to disk. We'll just advance the read/write head and insert another delimeter to take care of this job.

Notice the next line of code, seek(fd,offset+128,0);. Here we have advanced the read/write head to the first byte in the next record. Next, we'll write two characters into that record with fprintf(fd,"%c%c", DEL,EOF);. Now we have put our unique marker in the first vacant record and we are ready to exit the loop or complete another iteration. First, we will seek to the end of the file and close it. We can't simply close the file as we did when reading it, since this will result in a possible overwrite of the data just entered. Murphy is just waiting in the wings to visit you! The code fragment seek (fd,0,2); fclose(fd); will accomplish this for us.

Now we are nearly finished with this input module. We next ask the user to indicate whether or not he or she wishes to continue with data input or exit (to the menu) and test the input for a 0. If men[0] is equal to 0, we "break" out of the input loop, otherwise, we start over at the beginning of the input module. This completes the coding for the "Create a new record(s)" option from the menu. With a few changes, it could become the nucleus for any number of random records programs.

I promised you a new function for your "C" library and now I'm going to deliver. Locate the "index(s,t)" function on your distribution disk within "stdlib.c". PIP or otherwise copy this file onto the disk containing "funclib.c". What this function does is give the position of a substring within another string. Now picture, if you will, all the names in our data file concatenated so they form a very long string. Picture this string in memory where we can find the beginning of it. Remember, the "string [2048]" that you were wondering about? Well, in Part 4 of this series, we are going to read the names (after concatenation) into memory (string[2048]). Then we could use index to find a substring within string [2048]. The problem with "index(s,t)" is that it always starts at the beginning of string(s).

Let's suppose that you are using your new database program to locate someone named Smith in your database. You have forgotten the first name, but you know the last name. If we use index to find the substring "Smith" within our database, it will find the first occurrence of our substring and will go back to the same place no matter how many times it is invoked. If we could make "index" start at the point where it stopped the last time it was used to search for the same string, then we could "peruse" through the string looking for the right "Smith". We are going to modify "index(s,t)" so it will do just that. Then we will have the right tool for the job at hand.

Look at the program listing for "jindex-(s,t,p)". This function does the same job as index, but starts it's search at position "p". Now compare it to "index(s,t)" from your C/80 stdlib.c file. You will notice a very strong resemblance. The first was derived from the latter, but it is better suited to the job at hand. Now either copy "jindex-(s,t,p)" from this listing or modify your library file of "index(s,t)" to look exactly like the listing. Either method is fine. After you have made sure it is error free, make a backup copy on your backup disk. You may be wondering what "jindex(s,t,p) does that index won't. The biggest difference is that we pass to the function the position from where it is to start it's search for a match. If we have already found a match (string t is a match with substring s), but the record is not the one we are looking for, we can restart "jindex" at the next "name string" within string[2048]. We will build our "Search for record" routine around this function in "C__POWER — Part 4".

The method you use for copying and inserting listings or library files will depend largely on your editor or word processor. I use PIP for concatenation of files, and then use the Block Move function of my word processor for inserting the text into the proper place in the listing. Some of you might not be aware of the concatenation feature of PIP. The command:

```
"PIP NEWNAME.EXT=OLDFILE1.EXT,
  OLDFILE2.EXT"
```

will copy "OLDFILE1.EXT" and "OLDFILE2 .EXT" to the disk under the name "NEWNAME.EXT" and they will be joined to-

```
cls ()
{
putchar(27);putchar(69);/* escape, "E" sequence for H89's */
}

locate (row,col)
int row, col;
{
row+=31; col+=31;
putchar(27);putchar(89);   /* escape, "Y" sequence for direct */
putchar(row);putchar(col); /* cursor control */
}

fetchc ()
{
#asm
DIR:    MVI E,0FFH     /* "direct" call to CP/M Dos */
        MVI C,006H      /* for I/O (input) */
        CALL 5
        CPI 0
        JZ  DIR
#endasm
}

gofor (c,s)
int s;
char c[];
{
int i;
for ( i=0 ; i <= s && c[i-1] != 13 ; ++i )
{
c[i]=fetchc();/* c[i]=toupper(c[i]) */ putchar(c[i]);
if ( c[i]==8 )                           /* this subroutine
                                            "calls" */
   {                                     /* fetchc () for char-
                                            acter I/O */
   i-=2;putchar(32);putchar(8);
   }
}
if ( i-1 == s )         /* a return or s+1 characters will */
c[i-1]='\0';             /* cause this routine to return */
if ( c[i-1]==13 )        /* and will replace the return with a null */
c[i-1]='\0';          /* ('\0'). This is a "C" requirement */
}

jindex (s,t,p)
char s[],t[]; /* finds position of substring t within string s  */
unsigned p;   /* starting at position p */
{
int i, j, k;
for ( i=p; i <= q ; i++) /* pass starting position to i, q=buffer size */
   {
   for ( j=i, k=0; t[k] != '\0' && s[j] == t[k] ; j++, k++);
   if (t[k] =='\0' ) /* found match, return with position */
   return (i);
   }
return (-1);      /* no match, indicate so */
}
```

gether. This is a big help in file manipulation.

Next month, we will utilize our new function within the program option "Search for record". We will add code to provide for searching for a record, record retrieval, and printing the record to the screen. You will find this method of finding a record is very fast, since it does all the searching through memory. Another factor in reduced search time is the fact that all the routines are written in "C" or assembler which enhances the speed considerably (over BASIC).

If you have any problems with the program or material we have covered, please feel free to write me with your questions. Please enclose an SASE if you wish a reply. I'm also open to suggestions, so write if you have some ideas that you would like to pass along to others.

"C" you in "C_POWER — Part 4".

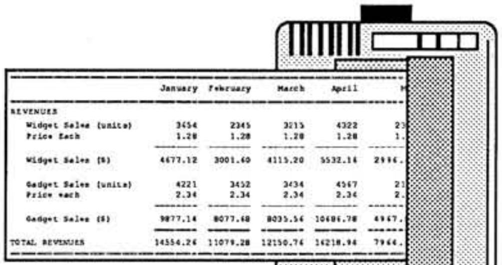
**Products Mentioned**

C/80 from The Software Toolworks
15233 Ventura Boulevard, Suite 1118
Sherman Oaks, CA 91403

CP/M from Digital Research Inc.
P.O. Box 579
Pacific Grove, CA 93950

# LONG & LOUD!

## *Sideways & Banner Printing Utility for Dot-Matrix Printers*



```
                          January  February     March     April       M
REVENUES
   Widget Sales (units)      3454      2345      3215      4322      23
   Price Each                1.28      1.28      1.28      1.28      1.
                          -------  --------  --------  --------  ------
   Widget Sales ($)       4477.12   3001.60   4115.20   5532.16   2996.

   Gadget Sales (units)      4221      3452      3434      4567      21
   Price each                2.34      2.34      2.34      2.34      2.
                          -------  --------  --------  --------  ------
   Gadget Sales ($)       9877.14   8077.68   8035.56  10686.78   4967.
                          -------  --------  --------  --------  ------
TOTAL REVENUES           14554.26  11079.28  12150.76  16218.94   7964.
```

For *any* CP/M or MS/DOS computer (IBM compatibility is not required), just...

**$34.95**

*Special Offer:* one MS/DOS and one CP/M version for only...

***$49.95***

We've improved our popular TWIST & SHOUT! package and given it a new name! **LONG & LOUD! Version 2.0** is easier to use and install, includes new typefaces in both LONG (four sizes) and LOUD (Times, Sans Serif, Olde English, Script and Symbols — in both upper and lower case). **LONG** lets you print out your spreadsheets (or any file) the long way (sideways) on your dot-matrix printer. No more cutting and pasting to put together a fragmented printout. **LOUD** prints giant banners in letters from two to eight inches high. Make banners & posters with ease!

# PRESTO!™

## *Multi-Function Software Supercharger for CP/M*

## *NEW for Heath/Zenith*

Profiles magazine wrote, ***"PRESTO still has the edge over Write Hand Man in features and general polish..."*** And now we've improved it even more! PRESTO adds features to any program you run. Just hit a special trigger key and PRESTO suspends your current program and opens a window on-screen. You can then call up a floating point calculator, a programmers calculator (hex, binary, octal, decimal), a notepad, a perpetual calendar, a Rolodex™, and perform screen dumps. Hit another key and you're right back where you left your original program.

PRESTO! (Version 3) uses almost 5K less memory than previous versions, yet includes great new features like:

***NEW CP/M Commands:*** From within any program you can now do a directory, copy and rename files, erase files, and type files to the screen.

***NEW Keyboard Macro Processor:*** Throw away SmartKey and XtraKey because PRESTO now includes its own key processor. The keys module includes powerful features like the ability to automatically load special key definitions for each program you use. One key can do the work of hundreds — a real time saver!

And best of all — the price is just **$39.95**. Available for all Heath/Zenith CP/M computers using H19/89 type terminal. Versions are also available for Morrow, Osborne, Kaypro and Otrona. Specify computer and hard or soft sector format.

# Rembrandt

*Complete Business Graphics Toolkit™*

Finally there's an easy and fun way to create graphics on your H/Z-89, H/Z-90, H/Z-100 *(CP/M only)* computer or any H/Z-19 equipped machine.

***No extra hardware required!*** It works with a standard unmodified machine yet also supports the TMSI SuperSet ROM, and the Font19 Character ROM.

***Freehand drawing:*** You can easily draw lines, boxes, circles and write on the screen in large characters. Full block operations are also supported — move, delete, fill, copy and more! Your graphic creations can be saved to disk and recalled at any time for further editing. Layout forms, design logos, draw diagrams and pictures. It's easy and fun to use.

***Business graphics:*** REMBRANDT lets you create horizontal and vertical bar charts, pie charts and xy plots (scatter graphs). Use hand-entered data or read numerical data from virtually any source including dBase II, SuperCalc, MBasic, Wordstar and ASCII files.

***Slide shows:*** Sequence your graphics on-screen using eleven cinematic special effects like wipes, fades and spirals. Produce electronic 'slide shows' without any programming.

***Print your graphics:*** Print your graphic screens on most dot-matrix and daisy wheel printers. Interface with all word processors so that your reports can include charts, graphs or any graphic creation — intermixed with your text!

***Compatible:*** It even reads, displays and prints *Ed-A-Sketch* files!

***Affordable:*** Even with all of this power, REMBRANDT is available for an amazingly low price of... ***$39.95***

REMBRANDT runs on H/Z-89's, 90's, 100's and H/Z-19 equipped machines.

***Other Stuff:*** **MILESTONE** Business Project Planner (CP/M and MS/DOS) $99.95, **MEDIA MASTER** Disk Conversion (Z-100, PC-DOS) $39.95, **MEDIA MASTER PLUS** Disk Conversion & CP/M Emulator $59.95, **ACCELERATE 8/16** including V20 chip $99.95

*TECHNOLOGIES, INC.*
22458 Ventura Blvd., Suite E
Woodland Hills, CA 91364

We accept **VISA, MASTERCARD** and **AMERICAN EXPRESS**

Order by mail or call our 24 hour toll free order line from anywhere in the US or Canada:

**800-628-2828 (Extension 918)**

Technical questions, orders: 818-716-1655 (9-5 PST)

Add $4 per order postage/handling. Overseas, add $12. US funds only. CA residents add 6% tax (LA County 6.5%)

# TYP-SET(tm) Lettering Software First Look

*William G. Nabor*
*27172 Huerta Street*
*Mission Viejo, CA 91692*

Have you ever had to make a sign for your garage sale, church bazaar or something similar, to advertise for your lost dog, or just to post the current price on the office coffee machine? Perhaps you are in charge of your organization's newsletter or want to ready a flier for printing and would like to have lettering that looked professional. Perhaps you had a report to prepare and wanted your illustrations to be identified with good looking letters that were much larger than a typewriter's and didn't look like dot-matrix jobs. Perhaps you had to give a talk and wanted some slides made. Perhaps you have had many other lettering jobs come your way. No doubt you have and many times, too. What have you done? Probably hand lettered whatever you wanted to say and accepted the fact that it looked amateurish and was hard to read.

Wouldn't it be nice to use the power of your Zenith PC to do the job for you? Of course it would, and there are any number of software programs out there that offer typesetting capabilities. The trouble is that they use a dot-matrix printer that looks tacky, or require either an expensive laser printer (which, even so, still looks like a xerox copy), or else an HP or other plotter, which is also out of your price range. Well, now there is a product from Enter Computer of San Diego, California, which allows you to produce professional looking lettering on paper or plastic transparencies using the inexpensive ($229.95 in the August 1986 Heathkit Catalog) Heathkit 5208 plotter. It should not be surprising that support for this plotter should come from Enter Computer, since the 5208 is the Heathkit version of Enter's Sweet-P plotter. What is surprising is that there are so very few commercially available programs that do support the 5208. Alas, this one is currently available only for Zenith PCs and their compatibles. No support, yet, for the Z-100.

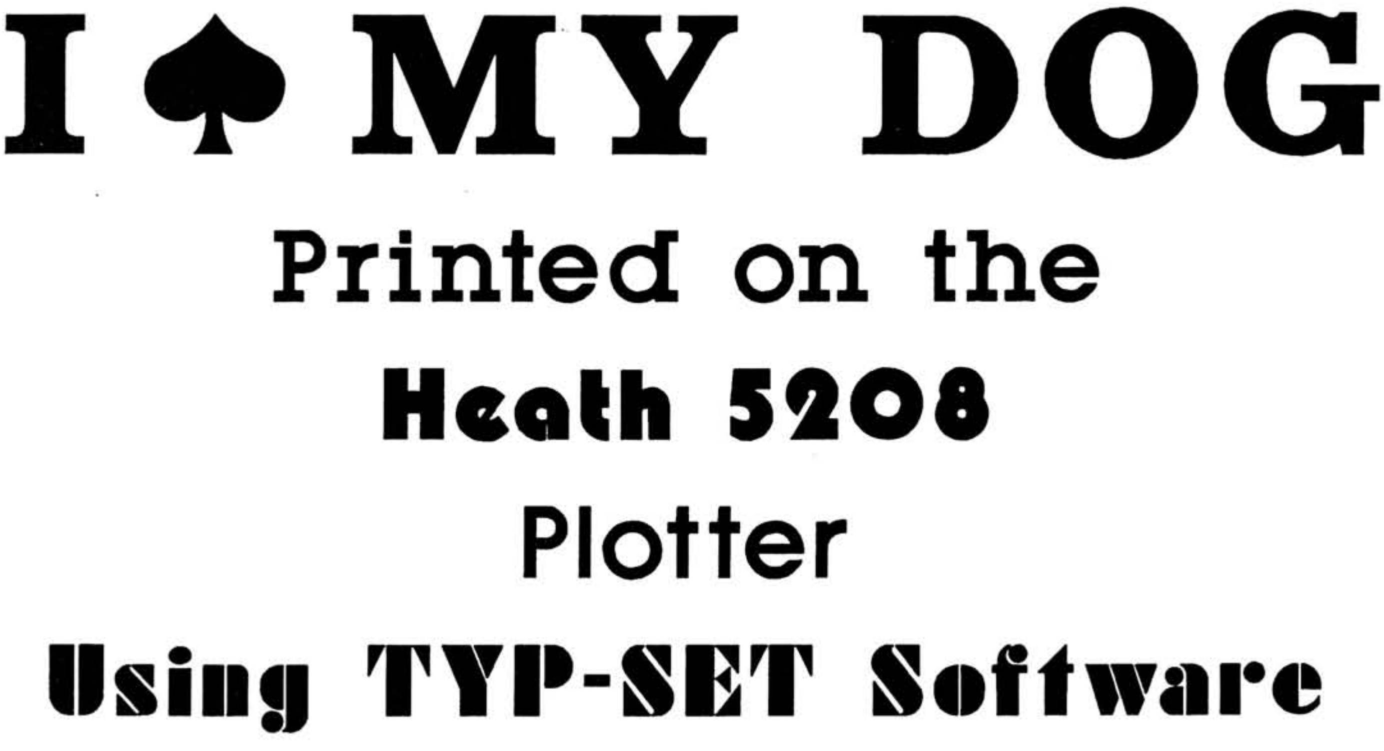
The software is called TYP-SET(tm). It costs $149.99, plus $50.00 if you want dot-matrix printer support, too. The current version (V2.0) requires a Zenith PC or compatible, 256K RAM, DOS 2.0 or higher, two floppy drives (one will do, but poorly) or one hard disk, and a plotter (plus a printer if you have purchased that option). In addition to the 5208, TYP-SET supports the following plotters: Sweet-P 100 (=5208), Sweet-P 600, Hewlett/Packard plotters, Houston Instruments plotters, Western Graphtec plotters, and Ioline plotters. The printer option supports the following dot-matrix printers: Epson printers, Gemini-10X, Gemini-15X, and IBM Proprinter.

The program is extremely versatile. Extreme versatility usually means extreme complexity and difficulty in use, but Enter has made it as easy as possible. The program is not copy protected, so you can install it anywhere by just copying the distribution disk onto your working disk. To invoke the program, simply type TYP-SET (CR) at the DOS prompt. This brings up the TYP-SET logo. This comes up so slowly, however, that you will put up with it only the first time. After that you will want to push (esc) twice and then a (CR) as soon as the logo begins. Doing this will bring up the main menu. The first time you run the program, you will have to enter the utilities section and set yourself up according to the type of plotter and printer you have. Just about every variable that is important can be set here and then saved to disk: The disk drive where the program resides, the drive where the output is to be sent, parallel or serial output, Com1 or Com2, baud rate, resolution, plotter page dimensions, number of plotter pens available (only one with the 5208), as well as the printer and plotter types you have, and metric or English notation. Once this is done and saved to disk, you needn't set anything again unless a parameter changes.

To use the program, call up the main menu. From here you can do the following:

1. List all the designs you have so far created. These are called "jobs".
2. Delete any job you no longer need.
3. Begin a new job.
4. Call up an old job.

If you begin a new job, you will be asked for a 1-8 character ID. This 8-character limit is



I ♠ MY DOG

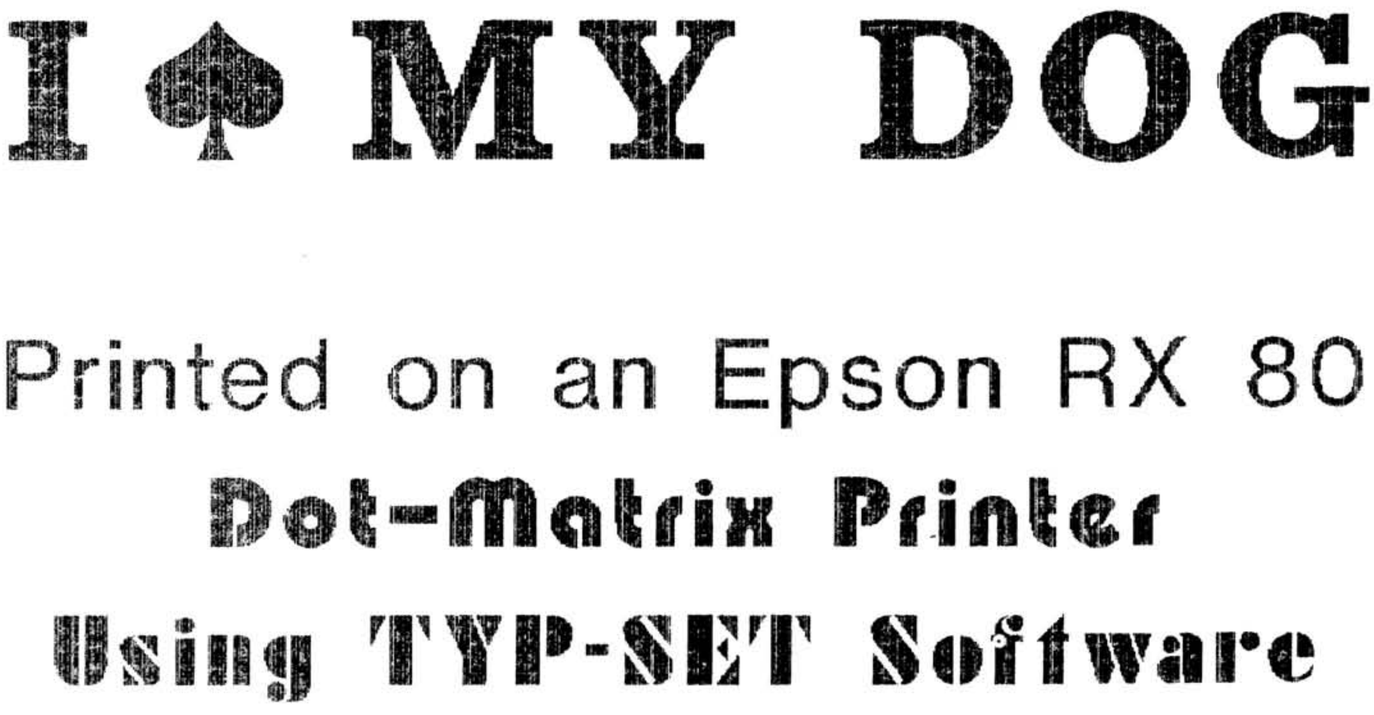
Printed on the

Heath 5208

Plotter

Using TYP-SET Software

**Illustration #1**
An example of TYP-SET's output on the Heath 5208 plotter. Each line shows a different font. There is a sixth font which is a thinner version of the one for "Plotter". In addition, the first line shows how an icon might be used.



I ♠ MY DOG

Printed on an Epson RX 80

Dot-Matrix Printer

Using TYP-SET Software

**Illustration #2**
The same job as #1, but modified and printed on a dot-matrix printer.

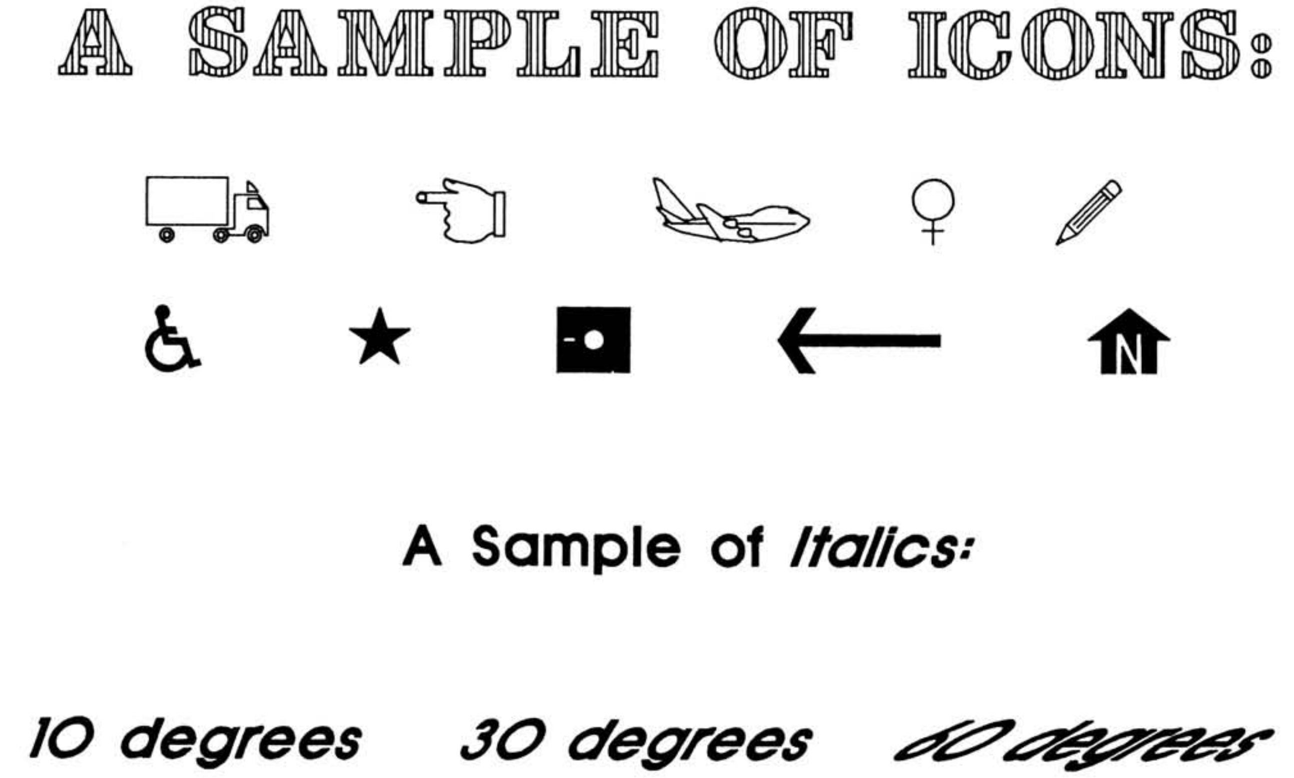


**Illustration #3**
Plotter output showing an example of icons and italics.

the DOS limit on file names. Your job will be stored on the drive you have selected by this name with a .SYM extension, and therefore, it cannot be longer than 8 characters. Since this is not enough to give you an adequate idea of the file's content (in many cases), you will next be asked to input a 1-40 character description, which subsequently can appear next to the file name when you list the files.

Having named your file and described it, the edit menu will appear. Hereby entering one-letter codes, you select all of the things you need to create your "job". There is no mandatory order to this, but I have found the following sequence to be best for me. First select the font you wish to use. There are 6 fonts available (more can be purchased at steep $50.00 each). Five of these are letters and the sixth is a series of 88 graphic icons, such as spades, clubs, diamonds, hearts, a floppy disk, etc. See the illustrations. Pushing F will call up the font menu. Select the font number you wish and the edit menu will reappear, this time listing the selected font.

Next, select the height of the letters. This is entered as the actual size to be plotted, in inches (or metric units, if you are so inclined). This takes some getting used to at first, since you usually expect to enter such things in some sort of "computer unit". Here if you want your letters to be 1.33 inches high on the finished page, you enter 1.33 as the height. The height may be anything from 0.1 inch up to the physical size of the plotter paper. Letters smaller than 0.25 inches don't show up well, however. The width of the letter can either be selected automatically according to the design of the font, or if you wish to elongate or compress the width, you can set the width at any size you wish, other than zero. A negative number will cause the letter to be printed backward.

Next, select the position on the page you wish your character string to be printed. This is done by imagining the page, held horizontally, to be marked with an imaginary cartesian grid, marked out in inches, with the 0,0 point at the lower left. The horizontal distance (from the left side of the paper) is entered as the X value and the vertical distance as the Y value. Both, again, are put in in actual inches and fractions. Thus, if you want to place your characters 5.56 inches up from the bottom and 3 inches from the left side of the paper, you enter an x value of 3 and a y value of 5.56. There are various editing aids to allow you to position the letters exactly where you want them.

A character string may be printed out proportionally, with the distances between the letters adjusted to the shape of the letters in question. VA, for example, is closer together than AA. This is called kerning. If you want, you can turn off this feature and space all letters the same distance, which you can specify. If you want the total length of the string to exactly fill a line on the paper, you can specify the length of the line and the letter sizes will be adjusted to suit. Note that this is not done like a printer, which justifies by increasing the distances between an arbitrary selection of letters or word spaces, but by actually adjusting the size of the letters to be printed.

If you want the string to start with the bottom left of the first character at the XY value you selected, you select left justification. If you want the string to start with the bottom right of the last letter to be at the XY point, select right justification. If you want the string centered about the XY point, select Center justification.

The letters are usually printed with the paper held horizontally. If you want them at any other angle, you can rotate them in any of 360 degrees, one degree at a time. Thus, if you want the characters printed out like a typewritten sheet, select a rotation angle of 270 degrees. If you want an italic effect, you can have the characters printed at a slant of 1-60 degrees.

Finally, you are ready to input your character string, which may be up to 40 characters in length. TYP-SET will then calculate the string length, page position, and plotter dimensions. If this length would result in some of the string going off the paper, a flashing display warns you of this. It doesn't tell you what is wrong, however. You might have selected an X value of 10, with left justification instead of right, the height may just be too much for the string length, or some other factor is wrong. Only a careful look, together with experience will allow you to know what to fix. This completes your first line. Pushing (esc) saves it and sets the edit menu to the next line, keeping all of the previous parameters except the text. Thus, you need to reenter only those parameters you wish to change for the new line.

When you have done all this, you are ready to output your job. This can be done to the CRT screen or to the screen and the plotter or the screen and the printer. You do this from an output menu, from which you initialize the plotter, select hollow or filled letters, and output one or a select few consecutively numbered lines. If you select filled letters, you can select (from the edit menu) the shade density, from solid black to spaced lines.

If you don't like what you got, you can reenter the edit mode and edit any line, or insert and delete lines. When you are finished, you must quit using approved procedures. Turning off the computer or the plotter may cause files to be lost.

As the following examples demonstrate, TYP-SET is a versatile lettering instrument that can turn out professional results with no lettering skill on the part of the user. There are, as you might expect, a few problems, however. Although the program sounds complicated at first, it is easily learned. I think the meager supply of fonts should be expanded for less than $50 a crack. The main problem, though, is speed. Unlike other lettering programs that use canned stick-men style letters that become distorted when you change the size, TYP-SET calculates the shape of each letter using a mathematical formula. This requires many, many, many arithmetical calculations. This takes computer time. TYP-SET supports the math co-processor and without it, plotter and screen output is slower than snail's paces. Using my Z-200 without a co-processor, it took 1 hour and 15 minutes to print Example #1 on my 5208. This is not quality time, however. You can go have lunch while the plotter is drawing your job. If you have a math co-processor and a printer buffer (not yet available as part of the TYP-SET package, although it should be), you might even be able to go on computing while the plotter plots. Without these two things, however, plan on your computer being tied up for a considerable time. It takes less time than doing it by hand, however, or using press-on letters. Comparing the final result with whatever you were using, shows that it's worth the wait.

## General - 16-Bit

# Patch Page

This article presents a correction to the CP/EMulator patch in the March 1987 REMark (page 75), and a patch to the PW .COM program on the Debug Support Utilities disk (HUG p/n 885-3038-37).

### CP/EMulator Patch Correction

When you make the CPMPCH.DAT file as directed in the March 1987 REMark Patch Page, it should contain these lines:

```
E131E
90 90  0C
W
Q
```

Notice that there are two spaces between the second 90 and the 0C. Other than this difference, the instructions in the previous Patch Page remain the same.

### PW.COM Patch

On the Debug Support Utilities disk, the program PW.COM that is used to control the PWINDOW (Processor Window) program contains a bug that does not allow you to specify a segment other than zero when you specify an address. To correct the problem, load the file PW.ASM into your editor and locate the following lines.

```
MOV     CX,14               ;14 REGISTERS
MOV     DI,OFFSET REGTBL    ;POINT TO REGISTER TABLE
REPNZ   SCASW               ;LOOK FOR USERS'S ENTRY
```

Modify the lines so that they look like this.

```
PUSH    CX
MOV     CX,14               ;14 REGISTERS
MOV     DI,OFFSET REGTBL    ;POINT TO REGISTER TABLE
REPNZ   SCASW               ;LOOK FOR USERS'S ENTRY
POP     CX
```

Now, locate these lines.

```
MOV     AX,BX
MUL     BYTE PTR SIXTEEN    ;MULTIPLY ACCUMULATOR BY 16
MOV     BX,AX
```

Modify them to look like this. Notice particularly that the word BYTE has been changed to WORD.

```
PUSH    DX                  ;AND DX
MOV     AX,BX
MUL     WORD PTR SIXTEEN    ;MULTIPLY ACCUMULATOR BY 16
MOV     BX,AX
POP     DX
```

Now, locate this line.

```
SIXTEEN DB       16                    ;MULTIPLIER
```

Change DB to DW, so that you have

```
SIXTEEN DW       16                    ;MULTIPLIER
```

After you have made these changes, reassemble PW.ASM, and use LINK and EXE-2BIN to produce a new PW.COM.

***Pat Swayne***
*HUG Software Engineer*

# A Cure For The MASM 4.0 EOF Problem

*Pat Swayne*
*HUG Software Engineer*

The latest version of Microsoft's MS-DOS assembler, version 4.0, is a considerable improvement over earlier versions when it comes to speed. This version, which is now being distributed with the MS-DOS Version 3 Programmer's Utility Pack, can assemble some programs in less than half the time required by any previous version. However, it does have one minor problem that you may run into. Some editors and word processors, especially older ones that were once CP/M programs, place end-of-file (EOF) characters at the end of any file they create or modify. The EOF character (actually, a Control-Z) was required under CP/M because file sizes are always multiples of 128 bytes in that operating system, and the EOF character was used to mark the end of a text file whose size was not a multiple of 128. Converted editors and word processors, such as WordStar, still make file sizes a multiple of 128 bytes and use the EOF character to delimit the end of the file.

MASM version 4.0 can work with main .ASM files that have EOF characters, but if an INCLUDE file has one, the assembler will generate an "extra characters at end of line" error message. It will still assemble the program correctly, but the "extra characters" messages printed on your screen may cause you to miss a more important error message. Fortunately, it is very easy to get rid of EOF characters at the end of files.

I have written a little program called FIXEND that looks for any EOF characters in the last 128 bytes of a file, and if it finds one, it shortens the file so that the last non-EOF character in the file is at the end. If you type in and run the following BASIC program, it will create FIXEND.COM

To use FIXEND, copy FIXEND.COM to your MASM disk or directory, and enter

```
FIXEND pathname
```

where pathname is any valid MS-DOS path description, including the specific name of the file you wish to fix. If FIXEND is unable to find any EOF characters, or if any thing goes wrong, it will print the message "ERROR - file not changed." If it does find an EOF character, it will print "Control-Z's removed." Run FIXEND on all of your INCLUDE files, and you will not encounter the EOF problem.

Below is the source code for FIXEND, which you can use to generate FIXEND .COM instead of the BASIC program if you prefer.

```
10 REM THIS PROGRAM CREATES FIXEND.COM
20 DEFINT A-I:OPEN "O",1,"FIXEND.COM
30 S=0:S1 = 18202 :FOR I=1 TO 160
40 READ B:S=S+B:PRINT #1,CHR$(B);
50 NEXT I:IF S<>S1 THEN PRINT "TYPING ERROR!":END
60 CLOSE #1:LOCATE 23,1:PRINT "DONE!":SYSTEM
70 DATA 191,128,0,138,29,50,255,71,198,1
80 DATA 0,139,203,176,32,252,243,174,79,139
90 DATA 215,180,61,176,2,205,33,115,9,186
100 DATA 109,1,180,9,205,33,205,32,139,216
110 DATA 185,255,255,186,128,255,184,2,66,205
120 DATA 33,114,232,180,63,185,128,0,186,128
130 DATA 0,205,33,114,220,191,128,0,185,128
140 DATA 0,176,26,242,174,117,208,139,209,247
150 DATA 210,185,255,255,184,2,66,205,33,114
160 DATA 194,180,64,185,0,0,205,33,114,185
170 DATA 186,137,1,180,9,205,33,205,32,13
180 DATA 10,69,82,82,79,82,32,45,32,70
190 DATA 105,108,101,32,110,111,116,32,99,104
200 DATA 97,110,103,101,100,46,36,13,10,67
210 DATA 111,110,116,114,111,108,45,90,39,115
220 DATA 32,114,101,109,111,118,101,100,46,36
```

```
        PAGE    ,132
;       FIXEND - FIX THE END OF A TEXT FILE.
;       THIS PROGRAM REMOVES CONTROL-Z CHARACTERS AT
;       THE END OF A TEXT FILE, AND TRUNCATES THE
;       LENGTH OF THE FILE TO THE LAST NON-CONTROL-Z
;       CHARACTER.
;
;       BY PATRICK SWAYNE, HUG SOFTWARE ENGINEER

CODE    SEGMENT
        ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
        ORG     1ØØH

START:  MOV     DI,8ØH                  ;POINT TO ARGUMENT COUNT
        MOV     BL,[DI]                 ;GET IT
        XOR     BH,BH                   ;MAKE IT A WORD
        INC     DI                      ;POINT TO
        MOV     BYTE PTR [BX+DI],Ø ;TERMINATE ENTRY
        MOV     CX,BX                   ;COUNT TO CX
        MOV     AL,' '
        CLD
        REPZ    SCASB                   ;LOOK FOR NON-SPACE
        DEC     DI                      ;BACK UP TO IT
        MOV     DX,DI                   ;MOVE NAME LOCATION TO DX
        MOV     AH,3DH                  ;OPEN FUNCTION
        MOV     AL,2                    ;OPEN FOR READ/WRITE
        INT     21H                     ;TRY TO OPEN FILE
        JNC     OPENOK
EXIT:   MOV     DX,OFFSET ERRMSG
        MOV     AH,9
        INT     21H
        INT     2ØH
OPENOK: MOV     BX,AX                   ;PUT HANDLE IN BX
        MOV     CX,-1
        MOV     DX,-128                 ;CX:DX = LSEEK POINTER
        MOV     AX,42Ø2H                ;LSEEK FROM END FUNCTION
        INT     21H
        JC      EXIT                    ;PROBLEM
        MOV     AH,3FH                  ;READ FUNCTION
        MOV     CX,128                  ;READ 128 BYTES
        MOV     DX,8ØH                  ;PUT THEM HERE
        INT     21H                     ;DO THE READ
        JC      EXIT
        MOV     DI,8ØH                  ;POINT TO DATA READ
        MOV     CX,128                  ;SET COUNTER
        MOV     AL,'Z'-'@'              ;LOOK FOR CONTROL-Z
        REPNZ   SCASB                   ;LOOK FOR IT
        JNZ     EXIT                    ;NONE FOUND
        MOV     DX,CX                   ;COUNT TO DX
        NOT     DX                      ;NEGATE (COUNT-1)
        MOV     CX,-1                   ;CX:DX = -COUNT
        MOV     AX,42Ø2H
        INT     21H                     ;PUT LSEEK POINTER THERE
        JC      EXIT
        MOV     AH,4ØH                  ;WRITE FUNCTION
        MOV     CX,Ø                    ;TRUNCATE FILE
        INT     21H
        JC      EXIT
        MOV     DX,OFFSET FIXMSG
        MOV     AH,9
        INT     21H                     ;SAY "FIXED"
        INT     2ØH                     ;AND EXIT

ERRMSG  DB      13,1Ø,"ERROR - File not changed.$"
FIXMSG  DB      13,1Ø,"Control-Z's removed.$"

CODE    ENDS
        END     START
```

*

# A Practical Guide To Kermit

## About The Author

*Vincent Alfieri, Ph.D., is a graduate of Columbia University and is a Microcomputer Software Specialist. He has worked on micro-to-mainframe communications at the University of Southern California and is the author of "Mastering WordStar" (Hayden) and a forthcoming book on WordPerfect.*

*Vincent Alfieri*
*2738 Medlow Avenue*
*Los Angeles, CA 90065*

This article is about Kermit, the excellent file transfer program developed at Columbia University. Kermit is one of the best software bargains around and rates as my personal choice for the software find of this or any year.

What makes Kermit so likeable? Three things: it does its job extremely well, is easy to use, and is absolutely free! How many programs do you know that can fill that bill? (Although Kermit is free, it is not in the public domain. Kermit is copyright and owned by Columbia University, and those who attempt to sell Kermit for a profit are breaking the law. By the way, the name "Kermit" is used with the permission of Henson Associates, Inc., the "Muppets" people.)

In this article I'll take you through a brief tour of what Kermit is and how it works. I'll then step you through a typical sample session using Kermit to transfer files, first by uploading a file from one computer to another, and then by downloading a file in the opposite direction. I'll mention a few tips for working with Kermit, and some of its many practical uses. Finally, I'll tell you how to get Kermit and how to learn more about it.

### A Short Primer On File Transfers

Most communications programs, the software that run modems so that computers can communicate over the telephone lines with other computers, usually have some kind of built-in file transfer capability. Many include the public-domain XMODEM program, which was developed by Ward Christensen. At the very least, there is usually a way to "capture" information that is scrolling on the terminal to a disk file. Both methods, although workable in many cases, do not have the power of Kermit.

With the capture method, the information from the computer at other end (such as a mainframe) is captured in the memory area of the calling computer (the micro) as this information scrolls on the screen. At the user's request, the information is then "dumped" to a disk file. The biggest problems with this method are (1) the limitation to the amount of memory available for capturing data at any one time, (2) the unreliability of the transfer, and (3) the fact that this method only works for printable characters that can appear on the screen.

With a file-transfer program, such as Kermit or XMODEM, the downloaded file is not retained in a buffer area, which may fill up quickly and cause the transferring to be suspended until the buffer empties, and the uploaded file doesn't have to be scrolling on the screen. Both Kermit and XMODEM do direct file-to-file transfers between the two computers.

The severest restrictions to using XMODEM, however, relate to the type of files it can transfer and its availability. On some communications links (notably connections to mainframes), XMODEM might not be able to deal with the 8th bit of every byte. And although XMODEM has great popularity in the micro world and can transfer all eight bits between micros, there are few versions of XMODEM on mainframes or minicomputers.

Kermit answers all these problems. It can transfer both 8-bit and 7-bit files with ease. Indeed, it makes no difference to Kermit what operating system is in use. And because Kermit has been developed in academe (not only at Columbia, but at many colleges and universities), there are versions of Kermit for virtually all popular mainframe, minicomputer and microcomputer systems. Up until recently, Kermit users considered themselves part of an "underground," but now that Kermit has "surfaced," its popularity will increase, due in no small measure to its widespread availability and ease-of-use.

The myriad versions of Kermit are the labors of love of many selfless people who spent their own time to develop a Kermit version for a specific machine. None of them were paid to do it. They did it because they know that Kermit is a great program, and they want others who have the same machine to be able to use Kermit.

### How Kermit Works

For any file-transfer program to work properly, the same program must be running at both ends of the link. The two programs "talk" to each other; that is, they understand the codes that are sent as part of the transfer. This type of communication between programs is known as a protocol. Both ends must understand and use the same protocol.

The basis of Kermit's protocol is that it sends or receives a file in packets. To understand what a packet is, recall the famous painting of "Washington Crossing the Delaware." Our soon-to-be first President was also using the packet technique. Because he couldn't possibly get all his troops over the river at once, he shipped them across, a few at a time, in packet boats. Aside from the logistical necessity of such a move, it also made sense: he could ensure that one group was safely transported before he sent another.

This is exactly what Kermit does with files. It breaks up each file into packets of a determined length, sends each packet individually, and makes absolutely sure that packet gets to its destination safely and correctly before it sends another.

Kermit does thorough error checking to check that each packet has been sent exactly, bit-by-bit. To do this, Kermit employs a checksum procedure. If the receiving Kermit detects that the packet does not contain the correct amount of characters — the "check sum" agreed upon by both Kermits before the transfer —then the two Kermits try again, and again and again, until they get it right.

When Kermit transfers a packet, it first sends a code that tells the Kermit at the other end something like this: "I'm sending a packet now, so everything you get until I give you the signal for the end of the packet will be data. Are you ready?" The receiving Kermit sends a code that says, in effect, "I'm ready now." Then, after the packet's data is sent, Kermit sends a closing code that tells the other side, "That's the end of one packet; are you ready for another?" Provided the checksum is correct, the receiving Kermit sends back a code to say "yes" or "no, wait a moment." The same sequence is repeated for each packet until the entire file has been successfully transferred.

If there is interference ("noise") on the line and part of the packet is not correctly transferred, the checksums won't match. So the Kermits will try again until they succeed. After a pre-established number of retries (usually 16), both Kermits give up. When this happens, disconnect and try your call again later.

As you can imagine, error checking is an extremely important procedure in file transfers because it ensures that the data is not corrupted or changed in any way. For instance, if you send programs (.COM or .EXE files), think of the hassle if the program wouldn't run after you transferred it.

Kermit is a bit-by-bit transfer program. It makes no difference to most versions of Kermit what the file contains — it could be text, text plus control codes, or binary program information. What does concern Kermit is that every bit be sent correctly. By the way, if one computer uses the ASCII coding scheme and the other computer the IBM mainframe EBCDIC codes, the Kermits will change ASCII characters to EBCDIC characters during the transfer.

### Different Versions Of Kermit

Because Kermit must be running at both ends of the communications link, there are many different versions of Kermit currently available, and more are being developed all the time. Indeed, Kermit has become so popular that there is a Kermit for virtually all microcomputers and all important mainframes, including IBM and Digital Equipment mainframes running under any number of different operating systems. At the micro end, there are Kermits for most of the Apple and Heath/Zenith computers, the Macintosh, the IBM PC and its "clones," the Kaypro and the Osborne, just to name a few. There are Kermits for many minicomputers, too.

Be advised, though, that the various versions of Kermit do not all have the same advanced features (most of which are not detailed in this article). The DEC Kermit, for instance, has a few extra commands not available on some microcomputer Kermits. And the newest version of Kermit for MS-DOS has many more features than, say, the CP/M Kermit. But the essential file-transfer protocol is exactly the same in all versions of Kermit, so that the different Kermits can communicate and work with each other for basic file transfers. And the "core" command structure is also virtually the same for all versions, so that once you know one Kermit, you can quickly learn another.

### Loading And Running Kermit

Kermit by itself cannot make the correct communications link between the two computer systems. You must still make the link yourself with whatever communications software that you use, or, as outlined in a later section of this article, you could issue the correct commands manually within Kermit if you know the command set of your modem. Most users, however, run a software package such as SmartCom or PC-Talk to make the initial connection. This means not only dialing the other computer, but also logging on to the other computer, if necessary, entering a password, and following any other necessary preliminary steps.

When you are completely logged on at the other end, then you would begin using Kermit. But how do you do this if you're currently in the communications program? You must exit the program without hanging up the line, if your communications program lets you do this. This is crucial. Some programs (such as SmartCom) automatically hang up when you exit the program. So if you can't stay on the line and exit from the communications program, you have to reboot your computer) and then load Kermit. The connection will still be there.

Many people don't realize that unless you specifically hang up, or tell the software to do it, the other computer is still on the line, patiently waiting for you. Although most mainframes will disconnect your account after five minutes or so, they generally won't hang up immediately. So you have a certain amount of time to get Kermit loaded and running (it only takes a few seconds anyway). I suggest that you have Kermit on the same bootable diskette as the communications program.

CrossTalk XVI, by the way, has the XDOS command to exit to MS-DOS without hanging up. The latest version of CrossTalk XVI is advertised to have a direct "hook" to Kermit built into the program, as do several other commercial packages. This would allow you to load and use Kermit from within the program without having to exit first. However, you may still have to procure Kermit yourself if it is not "bundled" with the program.

### Connecting Back To The Remote Computer

After you've made the connection, exited the communications program (or rebooted the computer) and then loaded Kermit at your end, you would type the command "c" for "connect." You could also type the whole word, or merely a few letters, followed by the <Escape> key — this is called escape completion. Escape completion works as long as you provide enough letters in the command so that it is not ambiguous to Kermit. There is, for example, no other command that starts with "c." All Kermit commands are sent with the <RETURN> key. You should then see the other Kermit prompt back on the screen.

Heath afficionados will like the fact that most microcomputer Kermits (including the MS-DOS version) emulate (that is, pretend to be) a Heath-19 terminal, so make sure that the other computer "knows" that that is the terminal type you're using. You can disable the Heath-19 terminal emulation if you use another terminal type.

When you are connected to the other computer with your Kermit, you must load Kermit there, too. Each version of Kermit has a different screen prompt so that you always know which end you're currently connected to. For example, Figure 1 shows the prompt for the newest version of MS-DOS Kermit.

**Figure 1**

```
IBM-PC Kermit-MS V2.28
Type ? for help
```

(The "V2.28" is the current version of Kermit as of this writing.) But the TOPS-20 version of Kermit for the DEC mainframes will show a prompt as listed in Figure 2.

**Figure 2**

```
Kermit-20>
```

Because it's important to know at which end of the link you are at all times, the different prompts guide you. When you wish to connect back to MS-DOS Kermit, you would press <Ctrl> and type c. You'll see the microcomputer Kermit's prompt. (This escape character, as it's called, may be different for other computers.)

When you are finished using Kermit, make sure that you exit from the other end (to log off and stop the meter!), which is done with the exit (or just e) command. Now let's see how to transfer files with Kermit.

### Uploading A File

As a general rule, always start any transfer — either downloading from the other computer to yours, or uploading from your computer to the other computer — from the other end. Let's say you want to upload a file to a mainframe. After loading both Kermits and with the mainframe Kermit prompt on the screen, you would type receive (you can just type re). Then you hit the <RETURN> key, and the mainframe Kermit will wait until your Kermit starts sending the file. You don't have to supply the file name unless you want its name to be different from the original name. You would then connect back to your machine (with <Ctrl>]c) and send the file from that end. The other Kermit is still waiting for your Kermit to begin the transfer. If the file name is TEXT, for example, the command is send text. You will then see the microcomputer Kermit's sending menu, which shows the name of the file that you are sending, the number of bytes and packets that are sent as they go, any retries, and any warning messages. On the current MS-DOS version of Kermit, the screen display for sending a file is shown in Figure 3.

**Figure 3**

```
          File name : TEXT
  KBytes transferred : 4
 Percent transferred : 1
            Sending : In progress
   Number of packets : 78
   Number of retries : 0
          Last error : None
        Last warning : None
```

At the bottom of the screen, Kermit lists several options for discontinuing the transfer or retrying. When the file has been completely sent, Kermit will tell you: "Completed." If it for some reason can't send the file after a certain number of retries, it will tell you "Failed." (Sometimes a mainframe is overloaded, and Kermit "times out." At other times, as I mentioned, there is noise on the line.) You should either try again, or call at another, less busy time.

One extremely useful Kermit feature is the capability to send several files at once by means of a wildcard. For example, to send all files named TEXT, no matter what their extension, the command would be send: text.*. Few other file-transfer programs have this convenience. (Note: Because Kermit normally uses the ? for its help system, use the = for the single-character wildcard symbol. Some mainframe Kermits, such as the DECs, use & for the single-character wildcard symbol.)

Kermit also renames a file to accommodate the file-naming conventions of each computer. What's more, if a Kermit receives a file with the same name as an existing file on the disk, Kermit will rename the newly-transferred file by changing one letter in its name so that the original file is not overwritten. If you wish, you can specify a new file name when you send the file. For example, the command sequence send test newtext sends the file called TEXT and renames the sent copy to NEWTEXT.

### Downloading A File

To download a file from another computer to yours, you issue the commands in reverse, but you should still start the ball rolling at the other end. Thus, to send a file called MAIN.TXT from a mainframe to a micro, you would type at the mainframe Kermit prompt: send main.txt. Then you would connect back to the micro (<Ctrl>]c) and tell that Kermit to receive.

If you want to receive the file under a different name, use the command receive <filename>. On your computer screen you'll see Kermit's receiving menu, which shows essentially the same information as the sending menu. On the latest MS-DOS Kermit, for instance, this menu will be as shown in Figure 4.

**Figure 4**

```
          File name : MAIN.TXT
  KBytes transferred : 12
          Receiving : In progress
   Number of packets : 135
   Number of retries : 0
          Last error : None
        Last warning : None
```

And that's all there is to it! Just remember to "receive" at the other computer and then "send" from your computer, or to "send" from the other computer and "receive" at your end.

### Help At All Times

As the initial Kermit prompt indicates, you can get help by typing ? on most microcomputer Kermits. Kermit will list all the commands available on that particular version. If you wish help with a specific command, type the command and, instead of hitting the <RETURN> key, type ?.

### Kermit, The Server

If you wish, you can set up MS-DOS Kermit to be in server mode when you're linked to certain computers, notably mainframes. In that way, you can do all your file transfers from the other end without switching back-and-forth between the two Kermits. So, after typing server at the MS-DOS end, you would connect to the other Kermit and then send or receive files, as you wish.

### Changing The Default Settings

Most of the problems you will encounter with Kermit file transfers have nothing to do with Kermit at all. They are generally the result of incorrect communications settings on the link or settings that don't match at both ends. As a general rule, always make sure that the two Kermits have the same settings. To check this, issue the status (or just st) command, and Kermit will show you all its current settings.

To change settings, use the set command. For instance, the baud rates for the two Kermits must be the same. If the other Kermit is set for 1200 baud, but your micro Kermit is set for 300 baud and you wish to communicate at 1200 baud, then at the micro end you would type, set baud 12.

Be especially careful of the parity setting. Often, you don't have to have any parity at all (the default for most Kermits is no parity), because that is handled by the communications software when you make the initial connection. I have found, and this is another general rule for working with any micro-to-mainframe link, that trial-and-error is usually the best rule to follow. If the default doesn't work, try one setting and, if it doesn't work, try another until you get one that does. Then remember to change the settings whenever you start working with Kermit.

Alas, even if you know that your parity setting is correct, sometimes something else will "get in the way." Recently, I helped members of a medical project at the University of Southern California transfer files cross-country with Kermit via the Telenet system so that they could save money on long-distance calls. Kermit worked fine when the connection was direct, but wouldn't cooperate with Telenet. We finally discovered that even though the normal setting for the mainframe was no parity, Telenet uses even parity, so we had to set Kermit accordingly. It then worked like a charm!

If you don't like the escape sequence Ctrl-] for switching back to MS-DOS Kermit, you can change it with the set escape command. For instance, you might like Ctrl-\, which is used in several microcomputer Kermits.

MS-DOS puts a Ctrl-Z character in a file to designate the end of the file, although you won't see this character when you look at a file. Most mainframes don't automatically add the end-of-file marker when a file is created, but you can instruct a mainframe Kermit to supply a Ctrl-Z during the transfer if necessary. Use set eof ctrl-z to activate this feature, or set eof noctrl-z to turn it off.

### Tips And Applications

Even though text is text, that doesn't mean that a mainframe can immediately handle all the characters in files that were created on your computer. There is always a certain amount of massaging of the file that must be done when you work with two different systems or even two different programs on the same system.

If you attempt to send files that contain control characters, Kermit will tell you that the file contains a "Non-ASCII character." Most microcomputer programs, such as Lotus 1-2-3 or WordStar, insert control characters to govern formatting and other operations. With word processing files, you may still be able to send the text correctly. But, as a general rule, it's a good idea to strip out these control characters before you ship the file, if possible.

For example, in some programs you could "print" the file to a disk file first. Lotus 1-2-3 operates in this fashion. Make sure that you also use the /Options /Other /Unformatted commands from the /Print menu to strip the control codes and create a "straight ASCII" file.

When writing with WordStar, use the "non-document" mode if possible. But if you do wish to send a normal WordStar "document" with the notorious WordStar eighth-bit, Kermit will do it, but make sure that you first delete all the dot commands and embedded printing effects (such as boldface or underline). In addition, don't use "soft" hyphens. Kermit will give you an error message about "non-ASCII" text, but you should still be able to use the file with another text editor at the mainframe.

By the way, you can run Kermit from within WordStar with WordStar's R ("run a program") command. You can even run your communications software and then Kermit from within WordStar, provided that your computer has enough memory. So after creating a file with WordStar, you can ship it with Kermit and then go back to WordStar in one fell swoop.

The same caveat about formatting codes holds true for other word processing packages, such as The FinalWord, but it's best to make your files as "vanilla" as possible before sending them. And since The FinalWord uses formatting commands that are similar to certain mainframe text formatters (notably, SCRIBE), you can edit your documents with The FinalWord on your computer and then ship them up for printing on a mainframe. If you merely wish to store the file on the mainframe, or use the mainframe as a "gateway" to another system, then see below.

### Kermit For Program Development

Kermit is great if you want to write computer programs with a text processor at your micro and then compile the programs at a mainframe, and vice versa. Recall that program listings don't contain control characters, since compilers can't read them. I have one friend who loves working with EMACS on his mainframe account to write and store Pascal programs, but then he uses Kermit to download his files to the micro so that he can compile his programs with the superfast Turbo Pascal.

### Transferring Binary (Program) Files

As mentioned, some versions of Kermit, both mainframe and micro, cannot send or receive command files, which as you know are in binary form. They can only deal with text files. So if you need to transfer these kinds of files, make sure that the respective Kermits can do the job. MS-DOS Kermit can transfer all types of files, provided that the Kermit at the other end can handle them.

On most mainframes, however, you would have to instruct a mainframe Kermit to receive the file in its full, 8-bit form (many mainframe Kermits disregard the 8th bit by default unless told otherwise). For instance, on DEC equipment you issue the following command before you begin a file transfer: set file bytesize to 8-bit. On UNIX (trademark of Bell Laboratories) machines, there is the -I switch added to the send <filename> command. Once you've set the eighth bit on, the mainframe will then accept your binary files exactly as they come, with each bit sent correctly.

### Archiving Files With Kermit

If the mainframe Kermit can do binary file transfers, one practical application of sending all eight bits of a file is the ability to archive your files on a mainframe. Provided you can afford it, you can save a great deal of disk space by shipping inactive files with Kermit to mainframe accounts and archiving them there. You can also use the mainframe as a gateway between two micros. For instance, if you wish to transfer a Lotus 1-2-3 file to someone else who has a mainframe account, you can use the mainframe as a "middle-man." Remember to use the 8th-bit option to ensure that the file is sent correctly.

When you wish to get your archived files back, the mainframe Kermit will send them back exactly as they were originally sent. (It keeps track of this.) You can't view the files at the mainframe, though, just as you normally can't "type" a binary file.

### Changing Kermit Defaults The Easy Way

MS-DOS Kermit allows you to supply any needed changes to the default settings in an initialization file that the program automatically looks for when you load it. This special file must be called MSKERMIT.INI and can be set up with a text editor, WordStar's "non-document" mode, or the MS-DOS copy con: command. For example, Figure 5 shows an initialization file that sets space parity, the baud rate to 1200, Ctrl-Z as the end-of-file marker, and instructs Kermit to use the B drive as the default.

**Figure 5**

```
set parity space
set baud 12
set eof ctrl-z
set default-disk b:
```

With the MSKERMIT.INI file in place, you never have to type in these commands each time you begin working with Kermit. You can also use the define command to set up frequently-employed macros. For example, you can define a macro called "u" to set up all the parameters for a UNIX computer (UNIX is a trademark of Bell Laboratories) and then the command "go u" in Kermit to load the macro.

### For The Adventurous

Although Kermit can't by itself make the correct communications link between two computers, you can make the connection yourself from within Kermit, if you know the command set of your modem. The trick is to load Kermit, type connect, even though you haven't connected yet, and then issue the commands to establish communications manually. For example, with the Hayes command set, the command ATDT followed by a number tone, dials the number.

### If The Other Computer Doesn't Have Kermit

MS-DOS Kermit's log command allows you to "capture" data from the remote computer as it appears on the screen. It works like a similar feature in most communications packages, and is most useful when there is no Kermit running on the other computer. Of course, this is not the same as a true file transfer.

### The Hard-Wired Route

If your computer is hard-wired to another computer directly, you don't have to bother with making the connection — it's already there! Just load Kermit, connect to the other end, and load the mainframe Kermit. You're then ready to go. A hard-wired Kermit can support transfers at rates of up to 9600 baud.

### Kermit And MS-DOS

The latest version of MS-DOS Kermit allows you to go directly to the disk operating system of your computer while you're working within Kermit, as well as to access certain file-maintenance functions at the remote end. This is great when you want to check the file directory (the dir command) for the exact file name or to delete an unwanted file. You can also change the default drive from within Kermit.

Kermit also has a space command which works with MS-DOS' CHKDSK.COM so that you can ascertain before you begin whether the file to be transferred will fit on this disk. You can also use dir to get a listing of the remaining bytes available on the disk.

Kermit's local and remote features allow you to do directory listings at either end of the link. There is also a remote type command to list ("type") the contents of a file so that you know it's the one you want, and a type command at the micro end, too.

You can run an external program, provided that you have enough memory for both Kermit and the other program. Or you can use the push command to "exit" to the MS-DOS prompt and run programs such as batch files. To return to Kermit, type exit. This feature is a life-saver if you're connected to another computer and want to save a file, only to realize that you don't have a formatted disk ready.

### Using Kermit For Direct Micro-to-Micro Transfers

You can have Kermit transfer files between two microcomputers via a null cable. A null cable is a serial cable with pins 2 and 3 crossed so that one micro's pin #2 connects to the other micro's pin #3, and vice versa. Check with your printer or hardware guru for help in making a null cable. They're very cheap if you do it yourself, provided you can solder.

Micro-to-micro transfers can take place at the highest speed allowed by both computers. When I wanted to transfer all my WordStar and SuperCalc files from my trusty, but antiquated, Osborne 1 to my nifty Zenith PC-160, Kermit did the trick in a snap. I could then use these files with WordStar or SuperCalc immediately on my Zenith.

The tip to remember when you transfer files between two micros is to make sure that you issue the receive command first at the receiving end before you send the file from the other machine.

### How To Get Kermit

Many of the public-domain bulletin boards have Kermit for any number of microcomputers. You could download the program with a special file written in BASIC. But the easiest way to get Kermit is to copy it from a friend (this is entirely legitimate), and the easiest way to "find a friend" is to go to a user's group meeting in your area.

It used to be Columbia University would only provide institutions with mainframe versions of Kermit, including tapes and documentation, whereas micro owners had to rely on users groups to get Kermit. Now, however, you can purchase micro Kermits and documentation for nominal fees directly from Columbia. Even though the program is free, the University can't afford to provide Kermit willy-nilly. So there is a charge for handling and other unavoidable expenses. The address to write to is:

Kermit Distribution  
Columbia University Center for  
Computing Activities  
612 West 115th Street  
New York, NY 10025

### For More Information

Every version of Kermit has its own documentation, which you should download (using Kermit, of course!) from the bulletin board where you got Kermit in the first place, or purchase from Columbia. You can also get more specific information directly from the folks at Columbia by writing, calling, or connecting with them through BIT .NET (or, as some would still have it, ARPA .NET). There is also a Kermit bulletin board where you can leave comments or questions.

If you want more technical information about how Kermit works, check out the two-part series written by a couple of its founding fathers: Frank da Cruz and Bill Catchings, "Kermit: A file-Transfer Protocol For Universities," BYTE, Vol. 9, No. 6 (June 1984) and Vol. 9, No. 7 (July 1984). These two articles are on the Columbia Kermit net.

This was by no means a complete tour of the power of Kermit; it was only meant to illustrate how easy it is to get your files where you want them to go with this fine program. I have described the basic procedures for working with Kermit and a few tips for specific applications, but I have only scratched the surface of Kermit's capabilities. For more detailed information, look at the excellent Kermit documentation. Join the surfacing underground: get acquainted with Kermit!

## *** UPGRADE ACCESSORIES FOR Z-100 *** SERIES COMPUTERS

**HIGH DENSITY 1.2 MEG DRIVES.** External floppy drive set-up complete with drive, power supply, case and cable. Ready to connect to your 8" floppy controller. $277.00. Dual Drive Unit ........$424.00

**ZMF100A by FBE Research.** A modification package which allows 256K chips to be used on the old-style motherboard to reach 768K. Simple assembly with no soldering or trace cutting. Compatible with Easy PC and Gemini Emulator. $60.00 alone or $148.50 with 27 256K RAM chips included.

**SmartWatch by FBE Research.** If you don't have a clock for your Z-100, get this one. More details under Z-150 upgrade listings .....$44.00

**GEMINI EMULATOR BOARD.** Makes the Z-100 compatible with the IBM PC library of programs ...............................$432.00

**UCI EASY PC.** IBM PC Emulator. Makes your Z-100 IBM Software Compatible. Full 8 MEG operation, color graphics and audio compatible. Retail $699.000, Payload ............................$477.00

**UCI EASY 87.** Add an 8087 Numeric Coprocessor. $69.00 for the board without an 8087 Chip. With 5 MEG 8087 $197.00 or with 8 MEG 8087 installed ...........................................$234.00

**UCI MEMORY UPGRADE CARD.** We recommend this one highly. The board has sockets for up to 2 MEG of RAM. With no RAM installed $328.00. With RAM installed and fully tested, 512K $387.00, One MEG $446.00, Two MEG $564.00 Add $35.00 for EasyDrive RAM Drive Software if desired.

**UCI RAMSAVER.** Maintains power on UCI MEMORY CARD RAM when computer is off. Save your programs in RAM with your computer off. Payload .............................................$177.00

**UCI EASY-I/O.** S-100 board that provides IBM PC communications port compatibility with your EasyPC. Easy I/O-1, One Serial Port $91.00. Easy I/O-2, Two Serial Ports, One Game Port, Clock-Calendar .......................................................$127.00

**UCI EasyWin.** Winchester Drive Systems at reasonable prices. Complete Hard Disk Systems for mounting inside your Z-100. Systems complete with Seagate Drives, 21 MEG $632.00, 31 MEG $727.00 System without Drive and Controller ......................$277.00

**CDR Z-100 SPEED MODULE.** Run your Z-100 Computer at 7.5 MHz. Installs easily with no soldering. Externally switchable between Speed and Normal mode. Payload...............................$48.00

## *** ZENITH SOFTWARE FOR THE *** Z-100 SERIES COMPUTERS

Zenith packages with software, manuals and registration cards for the original Z-100 computer series (not for the IBM compatibles).

| PART NUMBER | DESCRIPTION | LIST PRICE | SALE PRICE |
|---|---|---|---|
| MS-463-1 | Z-Basic (16 bit) interpreter | $175.00 | $24.00 |
| MS-463-7 | Multiplan | $195.00 | $24.00 |
| CB-463-11 | Z-Chart | $150.00 | $15.00 |
| CD-463-2 | Condor File Manager | $299.00 | $24.00 |
| PK-100-4 | All 4 listed above | $819.00 | $62.00 |
| MS-253-1 | Microsoft BASIC-80 (8-bit) | $175.00 | $24.00 |
| OS-53-2 | CP/M-85 (8 bit) | $150.00 | $24.00 |
| OS-63-4 | Z-DOS | $150.00 | $35.00 |
| CB-463-9 | PECON Peachtree to Condor | $99.00 | $15.00 |
| RS-463-5 | Peachtree Inventory Management | $499.00 | $38.00 |
| RS-463-75 | PeachText 5000 | $395.00 | $77.00 |
| WI-463-1 | Remote Batch Terminal Emulator | $899.00 | $28.00 |

## *** CHIP SPECIALS ***

The finest RAM available and at PAYLOAD prices. Order one to one thousand chips and add only $2.00 for shipping.

64K Dynamic RAM, 150 ns .............................$1.35 each
256K Dynamic RAM, 150 ns ............................$2.94 each
256K Dynamic RAM, 120 ns ............................$3.40 each

**V-20 CHIPS.** High Speed NEC V-20-8 8088 replacement. These run at up to 8 MEG and are said to increase CPU speed 10-30%. Payload ...................................................$14.75

**8087 MATH COPROCESSOR CHIPS.** Speeds and improves numeric processing. 5 MEG 8087-3.... $129.00, 8 MEG 8087-2 .... $165.00

## *** UPGRADE ACCESSORIES FOR Z-150/160 *** SERIES COMPUTERS

**SmartWatch from FBE Research.** Installs in ROM Socket on CPU Board in Zenith computer series Z-100/150/158/160. This tiny jewel of a product contains a ten year battery and keeps your computer informed of both time and date at each boot-up. Complete instructions and software included ...............................$44.00

**MEMORY KIT #150-256-18.** Includes a ZPAL chip which allows use of 256K RAM chips included (18 pieces 256K 150 ns RAM chips). Kit increases 128k memory to 640K or 256K memory to 704K. All chips plug into your existing Zenith Memory Board. Unbelievable but true .....................................................$87.00

**Winchester Hard Disk Drive Internal Set-up.** Includes Winchester drive, controller/interface card, cables and all hardware. With 20 MEG (formated) drive $448.00. May be installed in Z-148 using an Expansion Card sold below.

**PTZ-148 Expansion Card** for Z-148. Includes 2 expansion slots plus a clock/calendar. $129.00

**EVERCOM INTERNAL MODEM.** Fully Hayes compatible 1200/300 baud with powerful BitCom software included .............$128.00

## *** ZENITH SOFTWARE FOR THE *** Z-150/160 SERIES COMPUTERS

| PART NUMBER | DESCRIPTION | LIST PRICE | SALE PRICE |
|---|---|---|---|
| RS-463-75 | PeachText 5000 | $395.00 | $77.00 |
| BP-5063-71 | BPI Series Self-Training | $69.00 | $25.00 |
| BP-5063-8 | BPI Personal Accounting | $195.00 | $55.00 |

## *** UPGRADE ACCESSORIES FOR H/Z-89 *** COMPUTERS

**Magnolia Microsystems Double Density Controller.** Soft-sectored disk controller. Supports up to four each 5.25" and 8" disk drives. Complete with cables, installation instructions and CP/M ...$294.00

**INTERNAL DUAL DRIVE SETUPS.** Includes two half height double sided disk drives and all hardware and connectors required to mount inside your H-89. Steel mounting shield/case included.
MITSUBISHI MF501 Setup, 48 TPI, 6 MS seek, 320K .......$279.00
MITSUBISHI M4853 Setup, 96 TPI, 3 MS seek, 640K disks ..$284.00

## *** HALF HEIGHT FLOPPY *** DISK DRIVES

| | | | | |
|---|---|---|---|---|
| MITSUBISHI M501 | 5.25" | 48 TPI DS/DD | 320K/360K .... | $105.00 |
| MITSUBISHI M504 | 5.25" | 96 TPI DS/DD | 360K/1.2 MEG . | $177.00 |
| MITSUBISHI M4853 | 5.25" | 96 TPI DS/DD | 640K .......... | $109.00 |

## *** LAPINE HARD DISK DRIVES *** 2 YEAR WARRANTY CALL FOR PRICES

## *** SEAGATE HARD DISK DRIVES ***

ST-225 20 MEG Winchester Hard Disk .....................$355.00
With Western Digital Controller & Cables ................$448.00
ST-238 30 MEG, Requires RLL type controller .............$405.00
With Adaptec RLL Controller & Cables ..................$499.00
ST-4026 20 MEG High Speed for Z-200 ....................$508.00
ST-4038 30 MEG High Speed for Z-200 ....................$619.00
ST-4051 40 MEG High Speed for Z-200 ....................$739.00
ST-4096 80 MEG High Speed with Software.............. $1425.00

# PAYLOAD
## COMPUTER SERVICES

15718 SYLVAN LAKE, HOUSTON, TEXAS 77062
PHONE (713) 486-0687

Please **MAIL** or **PHONE** your order today and expect prompt service. **MASTERCARD** and **VISA** gladly accepted with no additional charge. All hardware carries a 90 more day warranty. Add $5.00 to all prepaid orders for handling and shipping, we pay the balance. Texas Residents please add 6.125% sales tax. We accept purchase orders from schools, government and approved accounts.

# Winchester For The '89

## Part Twelve

***Peter Ruber***
*P.O. Box 502*
*Oakdale, NY 11769*

### IBM Screen Emulation On The '89

In Part 8 of this series (September 1986), I discussed how you could turn your H/Z-89 computer, or H/Z-19/29 terminals, into a powerful 16-bit system by interfacing them to an Ampro LITTLE BOARD/186 PC-DOS single board computer. This article generated a surprising response in terms of calls and letters. Since I only gave a brief mention to the Video RAM Emulator in the article, the principal question presented to me related to software compatibility. The VRE was finally released in September 1986, so I have finally had a chance to play with it.

However, before I tell you about the VRE, I should backtrack a bit. The Ampro 186 is a complex and versatile product, and it has grown into a powerful system that includes not only the VRE, but the EXPANSION/186 board, the SCSI/IOP and the PROJECT BOARD/186. A year has now passed since I started working with the 186, so my perspective has altered and I can more fully appreciate the considerations of its original design concept.

The 186 was designed as a PC-DOS "engine" for industrial automation and as a means to upgrade terminal work stations to full computer status. It has a highly defined extension of the SCSI interface called SCSI/PLUS (tm), which provides users with a multi-master system bus for interfacing to other computers, I/O and mass storage devices through the NCR 5380 SCSI Protocol Controller. While the standard SCSI definition only allows for 8 devices on the bus, Ampro's enhancement will allow up to 64, each with 4 logical units.

PC-DOS (MS-DOS) and UNIX are not capable of handling a real time control environment. PC compatible systems can be set up to control industrial automation devices, assembly station robots, analog/digital converters, display processors, etc., on a one-to-one basis, but at the expense of the computer's native functions. Why tie up a $3,000 PC computer when a $500 board, that was specifically designed to handle these chores, can do it cheaper and more efficiently? Ampro's 186 "SCSI engine", their SCSI/IOP (input/output processor) and other new products offer the PC computing environment some extraordinary opportunities.

I spent several days this past August visiting Ampro's offices in California, and I was able to learn how their products can be used in office and industrial situations. As a result, I am currently developing a limited series for REMark that will demonstrate how the Ampro products can be used with PC computers for remote control applications, inexpensive networks, and mass storage. This series will also discuss the technical aspects of Ampro's SCSI/PLUS multi-master system bus. Although the slant of these articles will be directed to the larger user — corporate offices, universities, OEM's and the military — these concepts can be scaled down to provide users of Heath/Zenith PC/AT compatible systems with additional power in interesting ways.

### The Video RAM Emulator

The Video RAM Emulator is a PC compatible monochrome display adapter. When programs attempt to access video display hardware, a software driver converts the data written to the Video RAM into properly formatted serial data for display on the RS232C ASCII terminal connected to Serial Port A of the Little Board/186.

The Video RAM Emulator will run a greater selection of MS-DOS (PC-DOS) software beyond the "generic" category. "Generic" MS-DOS software usually means programs that write to the ROM BIOS instead of specific hardware within the PC. Software that requires extensive hardware compatibility with a PC — such as those which use bit-mapped graphics, or require a color display adapter, or make direct access to the PC's floppy disk controller, serial ports and Interrupts, generally will not load or run.

I suppose this sounds a bit confusing, so let's see if we can't narrow down the types of programs that we can use on the VRE with the '89 or '19/'29, by isolating those categories of programs we know will definitely not work unless they are extensively modified — a task that is beyond most of us, and certainly not worth the effort.

Software programs that require a color display board — the standard Color Graphics Adapter (CGA) or the Enhanced Graphics Adapter (EGA) — are easy to isolate. The packaging usually indicates which board is needed to run the program.

Software that accesses the PC's serial ports are communications programs (CROSSTALK XVI, PROCOM II, MIRROR, and Hayes' SMARTCOM II). They look for the 8250 communications controller and an attached modem.

Software that makes access to the PC's NEC765 floppy controller are those which are copy protected by PROLOK, SUPERLOCK, HDCOPY, PADLOCK II, SAFEGUARD, COUPON, PC-PADLOCK. Most of these annoying schemes use a "fingerprint" burned into a disk to foil duplication, and COPY2PC and COPYWRITE won't always provide successful backups with some of these methods.

There recently emerged an interesting program from Transec Systems, Inc., called UNLOCK, that will remove the copy protection schemes from many major protected programs that employ the PROLOK and SUPERLOCK protection schemes. When UNLOCK has removed the protection, clean duplicates can be made through the DOS DISKCOPY command, and whole programs can be placed on a hard disk without the need to keep the main program disk in the Drive A: floppy. Some of the programs that UNLOCK is guaranteed to work on, are: Lotus, Symphony, Harvard Total Project Manager, Framework, Signmaster, Chartmaster, Thinktank, DoubleDOS, Realia Cobol, Fastback, Clipper, The PFS series, dBASE III, dBASE III Plus, IBM Writing Assistant, IBM Filing Assistant, Microsoft Word, among others.

With respect to the Video RAM Emulator, I was advised by Ampro's Technical Support section that unprotected versions of Lotus and Symphony will now run on the 186. I could not confirm this because my copies are protected. However, I did have an unprotected version of dBASE III PLUS that ran like a charm, except that the on-screen clock in the upper right-hand corner was not transferred to the H/Z-89/19/29 screens. Conversely, some of the PFS series (WRITE, FILE and REPORT) worked on the 186 with the VRE even with the copy protection intact. This may be due to the fact that the degree of protection was not as violent as some of the programs I tried. I could not, however, place PFS WRITE on my hard disk. It had to remain in my Drive A: floppy while I created and edited files on my hard disk. This was a small price to pay for a program that is simple to use and adequate for letters and short documents.

Just what programs write to the PC's (8088 CPU) Interrupt structure, I have no idea of knowing. The only obvious one is clock hardware and software. When writing or reading to clocks, the device must interrupt the CPU in order to update the screen. The Heath/Zenith or FBE Research Company "SmartWatch" packages (the clock chip in a CMOS socket) will not install. The 80186 CPU in the Ampro board has a different Interrupt structure than the 8088.

However, Ampro has it's own "SmartWatch" package which plugs in under EPROM U9, when the 186 is used in conjunction with the VRE. The EXPANSION/186, which adds many additional functions (see REMark, September 1986), has its own clock.

The following is a list of programs by category that have been tested and known to work. This is not, by any means, a definitive list, because there are many programs available that are suitable for the IBM Monochrome Display Adapter. This would include programs that have an installation program to set the type of display within a PC compatible.

### Database

Dataease
dBase III
dBASE III Plus
PFS: File

### Spreadsheets

Supercalc 3
Lotus 1-2-3
Symphony

### Word Processors

Multimate 3.20
IBM Easywriter
Wordstar
Wordstar 2000 v1.01
Right Writer
PC-Write
PFS: Write
Spellbinder

### Integrated Software

Peachtree 5000
T/Maker

### Communications

MEX
Link II

### Languages

Borland's Turbo Pascal
Borland's Turbo Prolog
Microsoft "C"
MASM
GW Basic
IBM Macro Assembler
All Intel 8088, 8086, 80186 development tools and other utilities mention in the 9/86 REMark

You are not likely to need any additional software to handle the bulk of your computing requirements.

### IBM Function Key Emulation

Since many of the programs listed above make use of the IBM function keys, the H/Z driver for the VRE has incorporated key mapping that will allow about 95% emulation of the IBM's function keys. There are a couple of restrictions. The PC's keyboard system reset of CTRL-ALT-DEL can not be emulated. It has to do with the way IBM keyboards are encoded. Doing a manual RESET accomplishes the same thing. And the CTRL-ALT sequence used by memory resident programs such as SIDEKICK, has not been effected properly as of this writing. Fred Willink, head of Ampro's Technical Support, is seeking a solution for the H/Z driver that he has written for the VRE. Fred is also an old-time user of the '89, a member of the San Jose HUG, and the man to hound with questions, so he has a personal stake in this.

| PC Key | H/Z-89/19 Keys |
|---|---|
| <F1> | <ESC> <1> |
| <F2> | <ESC> <2> |
| <F3> | <ESC> <3> |
| <F4> | <ESC> <4> |
| <F5> | <ESC> <5> |
| <F6> | <ESC> <6> |
| <F7> | <ESC> <7> |
| <F8> | <ESC> <8> |
| <F9> | <ESC> <9> |
| <F10> | <ESC> <0> |
| <HOME> | HOME |
| <Pg Up> | Keypad <9> |
| <Pg Dn> | Keypad <3> |
| <Ins> | Keypad <0> |
| <End> | Keypad <.> or <Del> |
| <+> | <+> |
| <Num Lock> | <F4> |
| <Scroll Lock> | <F3> |
| <Break> | <White> <F3> |
| | **Modifier Keys** |
| <Ctrl> | <Blue> |
| <Alt> | <Red> |
| <Shift> | <White> |

**Arrow Keys**

| | |
|---|---|
| <Up> | Keypad <8> Up |
| <Down> | Keypad <2> Down |
| <Left> | Keypad <4> Left |
| <Right> | Keypad <6> Right |

<F2> — is the toggle key to move screen up or down one line. The PC has a full 25-line screen display. The H/Z has a 24-line screen with the 25th implemented as a status line under software control.

In addition to the H/Z driver, the VRE has drivers for the DEC VT100, Televideo 925, and Wyse 50 terminals. One of my two Ampro systems has a Qume VT108, which emulates the Televideo terminal. The H/Z driver seems to work best and is not as flaky in some clear screen and screen rewrite operations when using page up or down commands. This was particularly evident with my H/Z package of Microsoft Word version 1.0. I did not include this in my list because there is a slight problem when the screen clears and shifts up to display the next available workspace. The previously written text seems to linger in the screen buffer, and the only way to empty it is to use the up arrow and scroll the cursor up the screen past line one. Then scroll the cursor down. I understand this will be eliminated in a future revision. My PC version 3.0 of WORD is copy protected and dies.

### New Support Software Enhancements

Many changes have taken place with Ampro's Support Software programs. Being involved with other projects, I missed a number of interim revisions, but caught up with them during my visit to Ampro. As of September 1986, the ROM BIOS version is .005, and the Support Software version "H".

Most of the changes were a result of ROM BIOS revisions that had been optimized for PC-DOS 3.0, then for 3.1, and finally for 3.2. PC-DOS 3.2 finally included a utility to format 720K floppy drives, so Ampro dropped theirs. Ampro also revised their hard disk format routines to include the new 20-MB Seagate 225n hard disk drive with the imbedded SCSI controller chips. This eliminated the need to purchase a separate hard disk controller card.

The terminal driver source code, which had only been available previously at additional cost, is now included, so that code-lovers can play with it. An interesting user-supplied communications program, called LBCOMM, is now included with full documentation. It features auto-dialing, a library of frequently called numbers, uploading and down-loading, and is more than adequate in communicating with the several Ampro BBS systems around the country.

Utilities that will allow the LITTLE BOARD/186 to access the SCSI/IOP have been included, as well as programs for advanced users that will perform block data transfers from one hard disk drive to another. My favorite utility is the one that allows me to designate a CP/M drive on my system.

### CP/M On The 186

Die-hard '89 users have mentioned to me that one of the basic reasons they've been reluctant to venture into the world of MS-DOS is because they have compiled large database and word processing files over the years and are afraid to lose them.

There are several noteworthy programs on the market that will allow you to run CP/M programs on a PC compatible system once the NEC V-20 CPU has replaced the 8088 CPU. A few limitations exist, particularly with programs that write to the Z-80's registers, since the V-20 only emulates the Intel 8080 instruction set.

Ampro has had a program for some time, called PLUS80 CP/M and 8080 EMULATOR. PLUS80 attaches an 9090 microprocessor instruction set and CP/M operating system software emulator to the desired CP/M program. Thereafter, the program appears to be a normal PC-DOS program, and it works with most commercial CP/M application programs.

As an alternative, if you have dBASE or WordStar in the MS-DOS versions (or any similar programs that have been transferred from 8-bit to 16-bit formats, or which will accept ASCII file transfers), you can very easily incorporate your CP/M files into your MS-DOS program with Ampro's new CPMDRVR.SYS and SETCPM.COM utilities.

They allow you to define any of the 4 logical floppy drives as a CP/M drive. SETCPM selects the desired CP/M disk format. Most H/Z-89 disk formats and the Z-100's CP/M-85 disk format can be selected. By placing CPMDRVR.SYS in your CONFIG.SYS file, this driver will be loaded automatically into the PC-DOS root directory.

The CP/M drive format can also be established during BOOT by creating a short text file called SETCPM.TXT, and containing a single letter referencing the disk format. Then you add the following line to your AUTOEXEC.BAT file:

```
SETCPM <SETCPM.TXT
```

When you boot up the system you will always have access to a CP/M drive, and data transfers between CP/M and PC-DOS disk formats will be a breeze.

### SUPERDUO: IBM Terminal Program

If you also own any of the Heath/Zenith PC compatible computers, you can do some interesting things between the '89 and the PC. SUPERDUO is a program that will turn the PC into a terminal for the LITTLE BOARD/186. Once the corresponding programs have been installed on the PC and the 186, the PC's keyboard can access either system by a simple ALT-P or ALT-A command.

This will allow you to run separate programs in each system. Each computer can be reset without affecting the other. If you have a Z-148 or Z-138 without a hard disk drive, the 186 system can act as the hard disk host for these computers. If you're working with a large database file, you can let the 186 do the onerous chore of sorting and printing, while you continue work on your PC. If you have two projects going at once, you simply toggle back and forth between the two systems and update each file as needed.

### Versatile And Fun

Since the advent of the PC, a lot of the fun many of us once had in tinkering with the '89 is gone. The PC has become such a commodity item that everything is "plug in and run". A large part of the computing world probably prefers this kind of convenience, because they don't want to be bothered learning about their computers. But true-blue hackers will be happy to know that they can still have their '89 and tinker with it as a powerful 16-bit computer.

For more information, please contact:

AMPRO COMPUTER, INC.
67 East Evelyn Avenue
Mountain View, CA 94041
(415) 962-0230

# MAKE LEARNING AN ADVENTURE

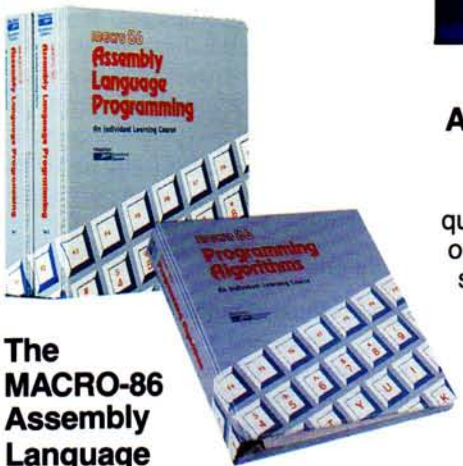
## Heathkit/Zenith Educational Systems

Feature success-oriented courses with a unique blend of technical theory and real-world applications. With our advanced course trainers and the hands-on experience they give you, you'll quickly gain valuable electronics skills. And see exciting electronics concepts come to life before your eyes. Plus our courses are approved by nationally recognized organizations and can earn you valuable Continuing Education Units for non-credit adult education. So why not embark on an exciting learning adventure today?

## HERO® 2000 Teaching Robot

Is the perfect educational trainer for learning about robot automation programming, electronics for automation, intelligent machines and robotics. HERO 2000 is equipped with an electronically synthesized voice that delivers unlimited vocabulary, music and sound effects. It also has a multi-jointed arm and gripper with sense of touch.

This computerized teacher features a 16-bit 8088 master microprocessor, 11 8-bit microprocessors and BASIC in ROM for easy programming. Plus 20 robot commands, 24K RAM expandable up to 576K, sonar, and built-in sensors for measuring light, sound and temperature levels.


MACRO 86
Assembly Language Programming
An Individual Learning Course
MACRO 86
Programming Algorithms
An Individual Learning Course

## The MACRO-86 Assembly Language Programming Course

Teaches you to program almost any computer using the popular Intel 88/86 series of microprocessors and Microsoft DOS.

## The MACRO-86 Algorithms Course

Introduces you to algorithms, an important phase in computer programming. And provides valuable programming experience.

## AutoCAD computer-aided design software

Lets you easily produce high-quality drawings and schematics on your personal computer. This superb software from Autodesk Inc. is available with several advanced drafting extensions, for both the Heath/Zenith 100 Computer and Heath/Zenith PCs. A special 3D drafting extension is also available.

## The Computer Servicing Series

Consists of three parts, from fundamentals through peripherals and on to maintenance. Uses the 16-bit Trainer which is ideal for breadboarding computer circuits that interface to the 8088 microprocessor.

**To order: Call TOLL-FREE**
**1-800-253-0570**
**Alaska and Michigan residents call: 616-982-3411.**

**Or visit your nearest Heath/Zenith Computers & Electronics Center.**

MasterCard VISA

**For more information on these as well as our entire line of educational products, see our latest Heathkit Catalog.**

ED-226

**Heathkit®**
Heathkit/Zenith
Educational Systems

Heathkit/Zenith Educational Systems is a division of Heath Company, Benton Harbor, Michigan 49022

# Super–Cheapcalc And Mini–Tutorial On Spreadsheets

***Luis E. Suarez***
*P.O. Box 66994*
*Caracus 1061–A*
*VENEZUELA*

# Part 3

## Erase Cell, Math Functions, Copy And Replicate Formula, IC Peek And DC

Once again, please remember that Math–Functions nesting is not supported. The syntax used is:

```
.C@function([number/cell][operator][number/cell]...)
```

Valid operators are + – * / ^ and [number/cell] is any number or cell designation. Valid inputs are:

```
@SIN(9)
@ATN(2-2)
@LON(A1)
@INV(B2+C5)
@COS(C3/1Ø)
@SQR(2*D3+H6)
```

See the available functions in Table 1, and refer to your BASIC manual to know limitations of the math functions. Remember that any number divided by zero will cause an Error. Similarly, a logarithm of zero yields an error and the square root of a negative number will, too. It is out of the scope of this article to review math and Trig fundamentals. However, several articles have been published in computer magazines about this subject and they may help you, if necessary. By the way, while modifying the program, I borrowed a couple of ideas from the article and program CALC .BAS by William Reese published in REMark, and also got some good routines from contributions to "My Favorite Subroutines".

I will not review all @ functions available. However, I must mention that in order to use the @RND function, you must understand the RANDOM function of your BASIC. If the program is run under MBASIC 5.25 Interpreter, @RND(0) will return the last random number, @RND(-1) restarts the same sequence and @RND(1) generates the next random number. Enter the following:

```
A1     @RND(Ø)
A2       @RND(-1)
A3       @RND(1)
```

Now, move the cursor up and down and see the result, while watching also line 2 to see the formula at the current cell. Now let's erase the cells content using the command ^^ #. Place the cursor at A1 and type #. Immediately ^^, the formula and cell content are erased. Proceed with cells A2 and A3.

This time, let's make some engineering work on electronics. The following sheet is a template to calculate circuit resonance. Yes, spreadsheets are good for engineering, despite the fact that most of the published templates are devoted to finances. Please enter the next data into a cleared sheet:

```
A1   SAMPLE4
                       B2   RESONANCE     C2   AND REACTAN
                       B3   \+            C3   \+
A5   f(MHz)=           B5   :F5Ø.3292     C5   F(Hertz)=
A6   C(pf)=            B6   5Ø            C6   C(Farads)=
A7   L(uH)=            B7   .2            C7   L(Henrys)=
A9   \=                B9   \=            C9   \=
                       B1Ø  <XC FORMULA   C1Ø  1/(2*PI*fC)
                       B12  <XL FORMULA   C12  2*PI*fL
                       B14  < RESONANCE   C14  1/(2*PI*SQR
A15  \=                B15  \=            C15  \=
A17  2*PI=             B17  :F2*3.14159

D2   CE IN RF CIR      E2   <CUITS
D3   \+                E3   <+++++++++
D5   B5*1000000        E5   < Hz
D6   B6*.000000000001  E6   < F
D7   :FB7*.000001      E7   < H
D9   \=                E9   \=            F9   \=
D1Ø  XC(Ohms) >>       E1Ø  @INV(B17*D5*D6) F1Ø  ^Ohms
D12  XL(Ohms) >>       E12  B17*D5*D7     F12  ^Ohms
D14  <(LC)        >>   E14  @INV(B17*E17)   F14  ^Hertz
D15  \=                E15  \=            F15  \=
D17  SQR(LC)=          E17  @SQR(D6*D7)
```

Erase the sheet and type ! to perform the calculation. The display shows an error at cell E14. What's wrong? Yes, you are right! The square root of the product Inductance (L) by Capacitance (C) have been calculated after cell E14 was calculated by the program. Hence, the formula at the cell was 1/6.2831*0. Any number divided by zero produces an error. Solution: Move the content of cell D17 to cell C17. To do so, place the cursor at cell D17 and use the command IL COPY, pressing the shift key simultaneously with IL and then press the shift key and move the cursor with the arrow keys until it is placed at cell C17. Then press again IL while the shift key is pressed, too. The same procedure could be done with cell E17 to be moved to cell D17. Then move the cursor back to cell E17 and erase the content with the # command ^^. Now, the square root of LC will be calculated ahead of cell E14. Finally, place your cursor at cell E14 and retype the formula to reflect the modification, as follows: @INV(B17*D17). Later we'll learn how to correct formulas or text in an faster way. But now, press ERASE and recalculate with ! RECALC command. The correct sheet will show that a 50 pf capacitor and a 0.2 uH inductance have each a reactance of 63.2455 Ohms at 50.3292 MHz. Please save the sheet under the name: SAMPLE4.DOC, and delete it with the command ESC-DELETE.

The DL REPLICATE command, is used to copy the formula to another cell modifying the formula, so that the new rows and columns match the new cell address. Type the following:

```
A1      100     B1      A1*10
A2      200     B2      A2*10
A3      300     B3      A2*10
```

Then use the REPLICATE command as follows:

**Note:** ("---> C1" means move the cursor to C1)

```
B1      SHIFT/DL ---> C1          SHIFT/DL
B2      SHIFT/DL ---> C2          SHIFT/DL
B3      SHIFT/DL ---> C3          SHIFT/DL
```

The resulted formulas at column C are displayed as follows:

```
C1      B1*10
C2      B2*10
C3      B3*10
```

If you like to move text, it is much better to use PEEK IC and POKE DC or SHIFT IL only. Should you use SHIFT DL for text moving, a strange string of characters will appear in the new cell.

If you need to repeat several times the content of a cell, use IC PEEK and DC POKE functions. Place the cursor at the cell whose contents you wish to store in the special buffer. Then DC POKE the buffer contents at whatever cell you like. Neither the peeked cell nor its cell contents are disturbed. Suppose you must repeat the string TOTAL= several times. Just type the text at one cell and IC PEEK the contents. Then DC POKE the string just peeked as many times as necessary.

### Angle, Ctrl-U, Backspace, Delete, Ctrl-D

It doesn't require one to be a scientist to use the Angle function. That's why I have included the Angle conversion function in this program. At the second line and to the right side of the screen, the word DEGREES appears to show the default value used with the trig functions. The function is toggled with the special function key f5. To change the ANGLE to RADIANS just hit the f5 key. Another hit will replace DEGREES by RADIANS and a further hit will change it into GRADS. If you hit f5 once again, the cycle starts again. I thought at the time of modification, that this was an important feature to consider, since MBASIC does all trig functions in Radians only. I borrowed the idea from CALC.BAS.

As with everything new in this life, you should practice the use of the new functions several times, so that you become familiar with the proper use of the program. However, sooner or later you need to make corrections and retyping a long formula would be time consuming. That's why the original program has the EDIT function. To access the function, just type the first letter of the string you wish to correct and then hit CTRL-U as many times as characters you need to skip until the cursor reaches the desired position. Then type the character you desire and hit again CTRL-U until the end of the string is reached. If the character to correct in the first one, type in the new character and then skip all the remaining characters with CTRL-U. If you move the cursor with CTRL-U and you hit RETURN before reaching the end of the string, the string will be truncated.

For editing purposes, while entering a new data or formula in a cell, a Backspace will move the cursor to the left without deleting the current character. The key labeled DELETE will rubout the current character.

To quit and exit to the CP/M system, type CTRL-D. But, don't do that yet! To finish this tutorial, please enter the following data:

```
A1    <SAMPLE5.DOC
A2    D = 60 ACS[sin L1 sin L2 + cos B2    L1 cos L2 co
A3    Negative for South and East    B3    coordinates
A4    \=                             B4    \=
A5    YOUR LATITUDE IN DEGREES       B5    :F10
A6    MINUTES                        B6    30
A7    SECONDS                        B7    0
A8    YOUR LONGITUDE IN DEGREES      B8    65
A9    MINUTES                        B9    30
A10   SECONDS                        B10   0
A11   OTHER PLACE LATITUDE           B11   ?ENTER LATITUDE IN
                                           DEGREES
A12   MINUTES                        B12   ?ENTER MINUTES
A13   SECONDS                        B13   ?ENTER SECONDS
A14   OTHER PLACE LONGITUDE          B14   ?ENTER LONGITUDE IN
                                           DEGREES
A15   MINUTES                        B15   ?MINUTES
A16   SECONDS                        B16   ?SECONDS
                                     B17   B6/60
                                     B18   B7/3600
A19   YOUR LAT DECIMAL               B19   B5+B17+B18
                                     B20   B9/60
                                     B21   B10/3600
A22   YOUR LON DECIMAL               B22   B8+B20+B21
                                     B23   B12/60
                                     B24   B13/3600
A25   OTHER LAT DECIMAL              B25   B11+B23+B24
                                    B26   B15/60
                                     B27   B16/3600
A28   OTHER LON DECIMAL              B28   B14+B26+B27
A29   SIN(YOUR LAT)                  B29   @SIN(B19)
A30   SIN(OTHER LAT)                 B30   @SIN(B25)
A31   COS(YOUR LAT)                  B31   @COS(B19)
A32   COS(OTHER LAT)                 B32   @COS(B25)
A33   COS(OL-YL)                     B33   @COS(B28-B22)
A34   SIN(YL)*SIN(OL)                B34   B29*B30
A35   COS(YL)*COS(OL)*COS(OLON-YLON) B35   B31*B32*B33
                                     B36   @ASC(B34+B35)

C1    ANGLE =
C2    <s (Lo2-Lo1)]
C4    \=
C5    Dist. Miles=
C6    Distance Km=

D1    < DEGREES
D4    \=
D5    B36*60
D6    D5*1.8
```

Be sure the ANGLE is set to DEGREES for this template. Maybe you would like to see what the results will be, should the ANGLE be set incorrectly. Perhaps you would like to have some practice with the EDIT functions, too. Please save this template as SAMPLE5.DOC. Now hit ERASE to erase the calculations.

The sheet is intended to calculate the distance between two locations in the earth. The formula used is shown at the second row of your sheet. The values at cells B5 to B10 are now for 10{O}30'0" North and 65{O}30'0" West. Later you may replace these ^^ values with your own coordinates. The resultant distance to a site located at 10{O}20'30" North and 10{O}20'30" West is 3315 miles and 5967 km.

Please use the command ! to recalculate the sheet and verify the answer. Should you like to calculate the distance between two different locations, maybe you would like to use the ?INPUT command ^^ instead of manually entering the data at cells B5 to B10. In doing so, don't forget the :F at the first cell to correctly display the sheet.

Well, we have reached the end of the road. We have reviewed all commands and used the spreadsheet for different tasks. Before entering the last command, maybe you would like to play around with the sheet while recalling what we did. I'm sure that you will do better and will develop by far more clever and useful templates. Now, it's time to try the last command. Just type Ctrl-D. After all, it's 3 O'clock in the morning . . .

*The following HUG Price List contains a list of all products in the HUG Software Catalog and Software Catalog Update #1. For a detailed abstract of these products, refer to the HUG Software Catalog, Software Catalog Update #1, or previous issues of REMark.*

Make the no-hassle connection with your modem today! ***HUGMCP*** doesn't give you long menus to sift through like some modem packages do. With *HUGMCP, YOU'RE* always in control, not the software. Order ***HUG P/N 885-3033-37*** today, and see if it isn't the easiest-to-use modem software available. Joe Katz says it was so easy to use, he didn't even need to look at the manual. "It's the only modem software that I use, and I'm in charge of both HUG bulletin boards!" says Jim Buszkiewicz. *HUGMCP* runs on ANY Heath/Zenith computer that's capable of running MS-DOS!

So, you got your new Heath/Zenith PC compatible. You probably sold your H/Z-89 or H/Z-100 to do it. Now you're stuck with dozens of CP/M programs just gathering dust. Well, plug ***CP/EMulator II*** and ***ZEMulator*** into your PC and blow that dust away. With HUG's *CP/EMulator II*, you can run just about all your standard 8-bit CP/M software under the MS-DOS operating system, and with *ZEMulator*, your PC will be able to reproduce the H-19 graphics character set, so you'll even be able to play all those games that use those graphics. *ZEMulator* comes with *CP/EMulator II*, and is ***HUG P/N 885-6002-37***.

Can't remember how to use the MS-DOS 'COPY' command? Forget the exact command line format for 'ASGNPART'. Too far to go for the MS-DOS manuals on the shelf on the other side of the room? Why not just type 'HELP' on the keyboard? You say it comes back with "Bad command or file name"? It wouldn't if you had HUG's ***HELP*** program. With *HELP* installed on your hard disk, all you need to do is type 'HELP' for a complete list of MS-DOS commands and transients along with a brief explanation of how each command works, as well as the format for its use. *HELP*, ***HUG P/N 885-8040-37***, works on *ALL* Heath/Zenith computers that run MS-DOS!

# HUG Price List

| PRODUCT NAME | PART NUMBER | OPERATING SYSTEM | DESCRIPTION | PRICE |
|---|---|---|---|---|
| **H8 — H/Z-89/90** | | | | |
| ACCOUNTING SYSTEM | 885-8047-37 | CPM | BUSINESS | 20.00 |
| ACTION GAMES | 885-1220-[37] | CPM | GAME | 20.00 |
| ADVENTURE | 885-1010 | HDOS | GAME | 10.00 |
| ASCRITY | 885-1238-[37] | CPM | AMATEUR RADIO | 20.00 |
| AUTOFILE (Z80 ONLY) | 885-1110 | HDOS | DBMS | 30.00 |
| BHBASIC SUPPORT PACKAGE | 885-1119-[37] | HDOS | UTILITY | 20.00 |
| CASTLE | 885-8032-[37] | HDOS | ENTERTAINMENT | 20.00 |
| CHEAPCALC | 885-1131-[37] | HDOS | SPREADSHEET | 20.00 |
| CHECKOFF | 885-8010 | HDOS | CHECKBOOK SOFTWARE | 25.00 |
| DEVICE DRIVERS | 885-1105 | HDOS | UTILITY | 20.00 |
| DISK UTILITIES | 885-1213-[37] | CPM | UTILITY | 20.00 |
| DUNGEONS & DRAGONS | 885-1093-[37] | HDOS | GAME | 20.00 |
| FLOATING POINT PACKAGE | 885-1063 | HDOS | UTILITY | 18.00 |
| GALACTIC WARRIORS | 885-8009-[37] | HDOS | GAME | 20.00 |
| GALACTIC WARRIORS | 885-8009-[37] | CPM | GAME | 20.00 |
| GAMES 1 | 885-1029-[37] | HDOS | GAMES | 18.00 |
| HARD SECTOR SUPPORT PACKAGE | 885-1121 | HDOS | UTILITY | 20.00 |
| HDOS PROGRAMMERS HELPER | 885-8017 | HDOS | UTILITY | 16.00 |
| HOME FINANCE | 885-1070 | HDOS | BUSINESS | 18.00 |
| HUG DISK DUPLICATION UTILITIES | 885-1217-[37] | CPM | UTILITY | 20.00 |
| HUG SOFTWARE CATALOG | 885-4500 | VARIOUS | PRODUCTS THRU 1982 | 9.75 |
| HUGMAN & MOVIE ANIMATION | 885-1124 | HDOS | ENTERTAINMENT | 20.00 |
| INFO. SYSTEM AND TEL. & MAIL SYSTEM | 885-1108-[37] | HDOS | DBMS | 30.00 |
| LOGBOOK | 885-1107-[37] | HDOS | AMATEUR RADIO | 30.00 |
| MAPLE | 885-8005 | HDOS | COMMUNICATION | 35.00 |
| MAPLE | 885-8012-[37] | CPM | COMMUNICATION | 35.00 |
| MICRONET CONNECTION | 885-1122-[37] | HDOS | COMMUNICATION | 20.00 |
| MISCELLANEOUS UTILITIES | 885-1089-[37] | HDOS | UTILITY | 20.00 |
| MORSE CODE TRANSCEIVER | 885-8016 | HDOS | AMATEUR RADIO | 20.00 |
| MORSE CODE TRANSCEIVER | 885-8031-[37] | CPM | AMATEUR RADIO | 20.00 |
| PAGE EDITOR | 885-1079-[37] | HDOS | UTILITY | 25.00 |
| PROGRAMS FOR PRINTERS | 885-1082 | HDOS | UTILITY | 20.00 |
| REMARK VOL 1 ISSUES 1-13 | 885-4001 | N/A | 1978 TO DECEMBER 1980 | 20.00 |
| RUNOFF | 885-1025 | HDOS | TEXT PROCESSOR | 35.00 |
| SCICALC | 885-8027 | HDOS | UTILITY | 20.00 |
| SMALL BUSINESS PACKAGE | 885-1071-[37] | HDOS | BUSINESS | 75.00 |
| SMALL-C COMPILER | 885-1134 | HDOS | LANGUAGE | 30.00 |
| SOFT SECTOR SUPPORT PACKAGE | 885-1127-[37] | HDOS | UTILITY | 20.00 |
| STUDENT'S STATISTICS PACKAGE | 885-8021 | HDOS | EDUCATION | 20.00 |
| SUBMIT (Z80 ONLY) | 885-8006 | HDOS | UTILITY | 20.00 |
| TERM & HTOC | 885-1207-[37] | CPM | COMMUNICATION & UTILITY | 20.00 |
| TINY BASIC COMPILER | 885-1132-[37] | HDOS | LANGUAGE | 25.00 |
| TINY PASCAL | 885-1086-[37] | HDOS | LANGUAGE | 20.00 |
| UDUMP | 885-8004 | HDOS | UTILITY | 35.00 |
| UTILITIES | 885-1212-[37] | CPM | UTILITY | 20.00 |
| UTILITIES BY PS | 885-1126 | HDOS | UTILITY | 20.00 |
| VARIETY PACKAGE | 885-1135-[37] | HDOS | UTILITY & GAMES | 20.00 |
| VOLUME I | 885-1008 | N/A | SOFTWARE LISTINGS | 9.00 |
| VOLUME II | 885-1013 | N/A | SOFTWARE LISTINGS | 12.00 |
| VOLUME III | 885-1015 | N/A | SOFTWARE LISTINGS | 9.00 |
| VOLUME IV | 885-1037 | N/A | SOFTWARE LISTINGS | 12.00 |
| WATZMAN ROM SOURCE & DOC | 885-1221-[37] | CPM | H19 FIRMWARE | 30.00 |
| WATZMAN ROM | 885-4600 | N/A | H19 FIRMWARE | 45.00 |
| WHEW UTILITIES | 885-1120-[37] | HDOS | UTILITY | 20.00 |
| XMET ROBOT X-ASSEMBLER | 885-1229-[37] | CPM | UTILITY | 20.00 |
| Z80 ASSEMBLER | 885-1078-[37] | HDOS | UTILITY | 25.00 |
| Z80 DEBUGGING TOOL (ALDT) | 885-1116 | HDOS | UTILITY | 20.00 |
| **H8 — H/Z-89/90 — H/Z-100 (Not PC)** | | | | |
| ADVENTURE | 885-1222-[37] | CPM | GAME | 10.00 |
| BASIC-E | 885-1215-[37] | CPM | LANGUAGE | 20.00 |
| CASSINO GAMES | 885-1227-[37] | CPM | GAME | 20.00 |
| CHEAPCALC | 885-1233-[37] | CPM | SPREADSHEET | 20.00 |
| CHECKOFF | 885-8011 | CPM | CHECKBOOK SOFTWARE | 25.00 |
| COPYDOS | 885-1235-37 | CPM | UTILITY | 20.00 |
| DISK DUMP & EDIT UTILITY | 885-1225-[37] | CPM | UTILITY | 30.00 |
| DOCUMAT & DOCULIST | 885-8019-[37] | CPM | TEXT PROCESSOR | 20.00 |
| DUNGEONS & DRAGONS | 885-1209-[37] | CPM | GAMES | 20.00 |
| FAST ACTION GAMES | 885-1228-[37] | CPM | GAME | 20.00 |
| FAST EDDY & BIG EDDY | 885-8018-[37] | CPM | TEXT PROCESSOR | 20.00 |
| FUN DISK I | 885-1236-[37] | CPM | GAMES | 20.00 |
| FUN DISK II | 885-1248-[37] | CPM | GAMES | 35.00 |
| GAMES DISK | 885-1206-[37] | CPM | GAMES | 20.00 |
| GRADE | 885-8036-[37] | CPM | GRADE BOOK | 20.00 |
| HRUN | 885-1223-[37] | CPM | HDOS EMULATOR | 40.00 |
| HUG BINDER | 885-0004 | N/A | REMARK BINDER | 5.75 |
| HUG FILE MANAGER & UTILITIES | 885-1246-[37] | CPM | UTILITY | 20.00 |
| HUG SOFTWARE CATALOG UPDATE #1 | 885-4501 | VARIOUS | PRODUCTS 1983 THRU 1985 | 9.75 |
| KEYMAP CPM-80 | 885-1230-[37] | CPM | UTILITY | 20.00 |
| MBASIC PAYROLL | 885-1218-[37] | CPM | BUSINESS | 60.00 |
| MICRONET CONNECTION | 885-1224-[37] | CPM | COMMUNICATION | 16.00 |
| NAVPROGSEVEN | 885-1219-[37] | CPM | FLIGHT UTILITY | 20.00 |
| REMARK VOL 3 ISSUES 24-35 | 885-4003 | N/A | 1982 | 20.00 |
| REMARK VOL 4 ISSUES 36-47 | 885-4004 | N/A | 1983 | 20.00 |
| REMARK VOL 5 ISSUES 48-59 | 885-4005 | N/A | 1984 | 25.00 |
| REMARK VOL 6 ISSUES 60-71 | 885-4006 | N/A | 1985 | 25.00 |

| PRODUCT NAME | PART NUMBER | OPERATING SYSTEM | DESCRIPTION | PRICE |
|---|---|---|---|---|
| REMARK VOL 7 ISSUES 72-83 | 885-4007 | N/A | 1986 | 25.00 |
| RF CAD | 885-8020-[37] | CPM | UTILITY | 30.00 |
| SEA BATTLE | 885-1211-[37] | CPM | GAME | 20.00 |
| UTILITIES BY PS | 885-1226-[37] | CPM | UTILITY | 20.00 |
| UTILITIES | 885-1237-[37] | CPM | UTILITY | 20.00 |
| X-REFERENCE UTILITIES FOR MBASIC | 885-1231-[37] | CPM | UTILITY | 20.00 |
| ZTERM | 885-3003 | CPM | COMMUNICATION | 20.00 |

### H/Z-100 (Not PC) Only

| PRODUCT NAME | PART NUMBER | OPERATING SYSTEM | DESCRIPTION | PRICE |
|---|---|---|---|---|
| ACCOUNTING SYSTEM | 885-8048-37 | MSDOS | BUSINESS | 20.00 |
| CALC | 885-8043-37 | MSDOS | UTILITY | 20.00 |
| CARDCAT | 885-3021-37 | MSDOS | BUSINESS | 20.00 |
| CHEAPCALC | 885-3005-37 | MSDOS | SPREADSHEET | 20.00 |
| CHECKBOOK MANAGER | 885-3013-37 | MSDOS | BUSINESS | 20.00 |
| CP/EMULATOR | 885-3007-37 | MSDOS | CPM EMULATOR | 20.00 |
| DBZ | 885-8034-37 | MSDOS | DBMS | 25.00 |
| ETCHDUMP | 885-3005-37 | MSDOS | UTILITY | 20.00 |
| EZPLOT | 885-3023-37 | MSDOS | PRINTER PLOTTING UTILITY | 20.00 |
| FAST EDDY | 885-8025-37 | CPM | TEXT PROCESSOR | 20.00 |
| FAST EDDY | 885-8029-37 | MSDOS | TEXT PROCESSOR | 20.00 |
| GAMES CONTEST PACKAGE | 885-3017-37 | MSDOS | GAMES | 25.00 |
| GRAPHICS | 885-3031-37 | MSDOS | ENTERTAINMENT | 20.00 |
| HELPSCREEN | 885-3039-37 | MSDOS | UTILITY | 20.00 |
| HUG BACKGROUND PRINT SPOOLER | 885-1247-[37] | CPM | UTILITY | 20.00 |
| HUG BACKGROUND PRINT SPOOLER | 885-5009-37 | CPM86 | UTILITY | 20.00 |
| HUGPBBS | 885-5006-37 | CPM86 | COMMUNICATION | 40.00 |
| HUGPBBS SOURCE LISTING | 885-5007-37 | CPM86 | COMMUNICATION | 60.00 |
| ICT 8080 TO 8088 TRANSLATOR & HFM | 885-5008-37 | CPM86 | UTILITY | 20.00 |
| KEYMAP | 885-3010-37 | MSDOS | UTILITY | 20.00 |
| KEYMAP | 885-5001-37 | CPM86 | UTILITY | 20.00 |
| KEYMAP CPM-85 | 885-1245-37 | CPM | UTILITY | 20.00 |
| MAPLE | 885-8023-37 | CPM | COMMUNICATION | 35.00 |
| MATHFLASH | 885-8030-37 | MSDOS | EDUCATION | 20.00 |
| MATT | 885-8045-37 | MSDOS | MATRIX UTILITY | 20.00 |
| ORBITS | 885-8041-37 | MSDOS | EDUCATION | 25.00 |
| POKER PARTY | 885-8042-37 | MSDOS | ENTERTAINMENT | 20.00 |
| SCICALC | 885-8028-37 | MSDOS | UTILITY | 20.00 |
| SKYVIEWS | 885-3015-37 | MSDOS | ASTRONOMY UTILITY | 20.00 |
| SMALL-C COMPILER | 885-3026-37 | MSDOS | LANGUAGE | 30.00 |
| SPELL5 | 885-3035-37 | MSDOS | SPELLING CHECKER | 20.00 |
| SPREADSHEET CONTEST PACKAGE | 885-3017-37 | MSDOS | VARIOUS SPREADSHEETS | 25.00 |
| TERM86 & DSKED | 885-5004-37 | CPM86 | COMMUNICATION & UTILITIES | 20.00 |
| TREE-ID | 885-3036-37 | MSDOS | TREE INDENTIFIER | 20.00 |
| USEFUL PROGRAMS I | 885-3022-37 | MSDOS | UTILITIES | 30.00 |
| UTILITIES BY PS | 885-5003-37 | CPM86 | UTILITY | 20.00 |
| UTILITIES | 885-3008-37 | MSDOS | UTILITY | 20.00 |
| ZBASIC DUNGEONS & DRAGONS | 885-3009-37 | MSDOS | GAME | 20.00 |
| ZBASIC GRAPHIC GAMES | 885-3004-37 | MSDOS | GAMES | 20.00 |
| ZBASIC GAMES | 885-3011-37 | MSDOS | GAMES | 20.00 |
| ZPC II | 885-3037-37 | MSDOS | PC EMULATOR | 60.00 |
| ZPC UPGRADE DISK | 885-3042-37 | MSDOS | UTILITY | 20.00 |

### H/Z-100 — PC Compatibles

| PRODUCT NAME | PART NUMBER | OPERATING SYSTEM | DESCRIPTION | PRICE |
|---|---|---|---|---|
| ADVENTURE | 885-3016-37 | MSDOS | GAME | 10.00 |
| ASSEMBLY LANGUAGE UTILITIES | 885-8046-37 | MSDOS | UTILITY | 20.00 |
| DEBUG SUPPORT UTILITIES | 885-3038-37 | MSDOS | UTILITY | 20.00 |
| DOCUMAT & DOCULIST | 885-8035-37 | MSDOS | TEXT PROCESSOR | 20.00 |
| DPATH | 885-8039-37 | MSDOS | UTILITY | 20.00 |
| HADES | 885-3040-37 | MSDOS | UTILITY | 40.00 |
| HELP | 885-8040-37 | MSDOS | CAI | 20.00 |
| HUG BACKGROUND PRINT SPOOLER | 885-3029-37 | MSDOS | UTILITY | 20.00 |
| HUG BINDER | 885-0004 | N/A | REMARK BINDER | 5.75 |
| HUG EDITOR | 885-3012-37 | MSDOS | TEXT PROCESSOR | 20.00 |
| HUG MENU SYSTEM | 885-3020-37 | MSDOS | UTILITY | 20.00 |
| HUG SOFTWARE CATALOG UPDATE #1 | 885-4501 | VARIOUS | PROD 1983 THRU 1985 | 9.75 |
| HUGMCP | 885-3033-37 | MSDOS | COMMUNICATION | 40.00 |
| HUGPBBS SOURCE LISTING | 885-3028-37 | MSDOS | COMMUNICATION | 60.00 |
| HUGPBBS | 885-3027-37 | MSDOS | COMMUNICATION | 40.00 |
| ICT 8080 TO 8088 TRANSLATOR | 885-3024-37 | MSDOS | UTILITY | 20.00 |
| MISCELLANEOUS UTILITIES | 885-3025-37 | MSDOS | UTILITIES | 20.00 |
| REMARK VOL 5 ISSUES 48-59 | 885-4005 | N/A | 1984 | 25.00 |
| REMARK VOL 6 ISSUES 60-71 | 885-4006 | N/A | 1985 | 25.00 |
| REMARK VOL 7 ISSUES 72-83 | 885-4007 | N/Z | 1986 | 25.00 |
| SCREEN DUMP | 885-3043-37 | MSDOS | UTILITY | 30.00 |
| UTILITIES II | 885-3014-37 | MSDOS | UTILITY | 20.00 |

### PC Compatibles

| PRODUCT NAME | PART NUMBER | OPERATING SYSTEM | DESCRIPTION | PRICE |
|---|---|---|---|---|
| ACCOUNTING SYSTEM | 885-8049-37 | MSDOS | BUSINESS | 20.00 |
| CARDCAT | 885-6006-37 | MSDOS | CATALOGING SYSTEM | 20.00 |
| CHEAPCALC | 885-6004-37 | MSDOS | SPREADSHEET | 20.00 |
| CP/EMULATOR II & ZEMULATOR | 885-6002-37 | MSDOS | CPM & Z100 EMULATORS | 20.00 |
| DUNGEONS & DRAGONS | 885-6007-37 | MSDOS | GAME | 20.00 |
| EZPLOT | 885-6003-37 | MSDOS | PRINTER PLOTTING UTILITY | 20.00 |
| FAST EDIT | 885-8033-37 | MSDOS | TEXT PROCESSOR | 20.00 |
| GRADE | 885-8037-37 | MSDOS | GRADE BOOK | 20.00 |
| HAM HELP | 885-6010-37 | MSDOS | AMATEUR RADIO | 20.00 |
| KEYMAP | 885-6001-37 | MSDOS | UTILITY | 20.00 |
| RF CAD | 885-8038-37 | MSDOS | UTILITY | 30.00 |
| SCREEN SAVER PLUS | 885-6009-37 | MSDOS | UTILITIES | 20.00 |
| SKYVIEWS | 885-6005-37 | MSDOS | ASTRONOMY UTILITY | 20.00 |
| TCSPELL | 885-8044-37 | MSDOS | SPELLING CHECKER | 20.00 |

You've got a screen full of important technical data that would be nearly impossible to memorize, and you already have writer's cramps from the last screen full. With ***SCREENDUMP*** from HUG, you can reproduce a complete video screen on a dot matrix printer, including both text and graphics without having to exit the current program. *SCREENDUMP* supports most of the more popular dot matrix printers, including the newer 24-pin and laser jet models. The latest version of *SCREENDUMP* is ***HUG P/N 885-3043-37***.

"Thank Heaven for *HADES*!" That's what a lot of MS-DOS users are saying when ***HADES*** rescues a file that just got accidentally erased. Erased file recovery is only a small part of the capabilities of this program. *HADES* is *HUG's Absolute Disk Editing System*. Within the realms of MS-DOS, *HADES* allows you to directly edit any part of any disk. Directories, files, file attributes. FATS: nothing can hide from you when you use *HADES*. *HADES* works on *ANY* computer that can run MS-DOS version 2 or greater. Order ***HUG P/N 885-3040-37*** today!

Want to keep your H/Z-100? Want to run a lot of that good PC compatible software out there? Don't want to buy a PC compatible though? Then get ***ZPC II, HUG P/N 885-3037-37***, and the ***ZPC II upgrade disk, HUG P/N 885-3042-37***.

***HEPCAT*** is coming! *HEPCAT* is coming! *HEPCAT* is coming! So what is *HEPCAT* you may ask? Why it's just another Pat Swayne SUPER-UTILITY. *HEPCAT* is an acronym for *HUG Engineer's and Programmer's CAlculation Tool*. Just what we don't need, another memory resident calculator, right? Wrong! With *HEPCAT*, you can throw away the rest and use the best. *HEPCAT* only uses two partial lines on your screen, and best of all, does NOT cause existing programs to stop executing! That means while your computer is grinding numbers internally, you can be grinding them externally. Watch for Pat's write-up in an upcoming issue of REMark.

10 – Very Good
9 – Good
8 – Average

## TABLE C
**Product Rating**

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

- **7 – 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/Z-89) 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., 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. UPS Blue Label is $4.00 minimum. No maximum on UPS.), 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.

## NOTES

**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).**

All special update offers announced in REMark (i.e., ZPC II update) must be paid by check or money order, payable to the Heath Users' Group. **NO CREDIT CARDS ACCEPTED.** ZPC II contains only one disk. It is a combination of ZPC I and the ZPC Support disk, plus added improvements. Thank you.

## HUG P/N 885-3044-37 Games Package II ... $25.00

**Introduction:** This two disk set contains 6 different games for the H/Z-100 (not PC) computer system. These games include two version of Backgammon, two casino type games, and two version of a card game. One of the Backgammon games, and one of the versions of BRIDGE was specifically written for a light pen! The rest of these games use the standard keyboard for input.

**Requirements:** In order to play these games, an H/Z-100 (not PC) system with at least 192k of RAM is needed. MS-DOS version 2.0 or greater must also be used. The two casino games, GOBE-SOU, and BLACK-SPOT, require the ZBASIC interpreter; GWBASIC will NOT work. The Backgammon game, BG, and the Bridge game, LPBR, requires the use of a light pen, such as the one mentioned in the September 1986 issue of REMark. It is definitely to your advantage to play these games on a color monitor, however, a monochrome monitor can be used, and in either case, all three video planes must be installed. According to the author, the two light pen programs LPBR and BG require between 256k and 512k of system RAM.

The following files are included on the Games Package II disk:

**Disk A**

| | | | |
|---|---|---|---|
| GAMMON | .BAS | GAMMON | .EXE |
| GAMMON | .CHR | GAMMON | .DOC |
| BG | .EXE | BG1 | .ASM |
| BG2 | .ASM | BKSPOTB | .BAS |
| RANDOM# | | BKSPOTC | .BAS |
| BKSPOT | .BRL | INSTALBF | .BAT |
| INSTALCE | .BAT | BKSPOT | .BAT |
| INSTALCF | .BAT | INSTALBE | .BAT |
| INSTALCB | .BAT | INSTALBB | .BAT |
| BKSPOT | .DOC | README | .DOC |

**Disk B**

| | | | |
|---|---|---|---|
| HBR | .EXE | HBR1 | .ASM |
| HBR2 | .ASM | HBR3 | .ASM |
| HBR4 | .ASM | HBR5 | .ASM |
| HBR | .DOC | GOBE-SOU | .ASM |
| GOBE-SOU | .BLD | GOBE-SOU | .FIG |
| GOBE-SOU | .BAS | GOBE-SOU | .DOC |
| LPBR | .EXE | LPBR1 | .ASM |
| LPBR2 | .ASM | LPBR3 | .ASM |
| LPBR4 | .ASM | LPBR5 | .ASM |

**Authors:**

GAMMON — Michael Scott
BG — Robert F. Hassard
BLACKSPOT — William G. Nabor
HBR — Robert F. Hassard
GOBE-SOU — Lucien Dumas
LPBR — Robert F. Hassard

**Program Content:**

**GAMMON** — This is a game of Backgammon. It was written in ZBASIC and compiled. The source is included for anyone wishing to make modifications. It was written for a color monitor, however, can be played in monochrome. The four files named "GAMMON" make up this game.

**BG** — This is also the game of Backgammon. It was written in assembly language, and the source is also included. This version of Backgammon requires a light pen to play. The pen described in the September 1986 issue of REMark works very nicely. This game also plays in full color.

**BLACKSPOT** — This is a casino-style gambling game that combines some of the features of Blackjack, Roulette, and Faro. It is played with a deck of 100 cards, divided into ten suits of ten cards each. For the Z-100 color version, the suits are blue, green, cyan, red, magenta, yellow, white, redgreen, blueyellow, and black. For the monochrome version of the game, suits are numbered 1 to 9 plus black. There is no ranking of cards within a suit. Each game consists of three (sometimes only two) hands. Each hand consists of a series of draws terminating when a black card is drawn. The player bets on which suit will be declared the "winner" at the end of each hand.

**HBR** — This is Bob Hassard's version of the ever popular card game, BRIDGE. Originally released for both HDOS and CP/M, this version was written for MS-DOS on the H/Z-100 and is suitable for the novice BRIDGE player.

**GOBE-SOU** — This is the best real-time action slot machine simulation I've ever seen on a computer. It plays in full color, you can bet from $1 to $5, and the 'wheel action' is the most life-like ever seen. Source code is included so you can see how it was done.

**LPBR** — This assembly language program plays the game of Bridge with one human using a light pen. After calling up LPBR, the keyboard is used once only to enter the player's name. From then on, the light pen is used until the program is exited by touching the pen to the word 'EXIT'. Touching the word HELP will produce a full screen of instructions for playing the game in detail sufficient for a novice. Touching the screen will cause return to the game.

**TABLE C Rating:** (9), (5), (2), (1) * ZENITH | data systems

## COMPUTERS

RAM Technology carries a full line of Zenith computers, peripherals, and software. Call us toll-free for competitive prices on systems and accessories.

## VIDEO

| | |
|---|---|
| Paradise Auto Switch EGA | $429.00 |
| Video 7 EGA Deluxe | $429.00 |
| Video 7 EGA High-Resolution Color Card | $369.00 |
| Video 7 MGA High Res. Monochrome Graphics | $189.00 |
| NEC Multisync High Res. Color Monitor | $599.00 |
| C. ITOH CM1000 Color Monitor | $399.00 |
| Mitsubishi EGA High Res. Color Monitor | $429.00 |
| Premier Technologies Z148 1½-Slot Board | $119.00 |
| ZVM-1230-A: Non-Glare, Green Monitor | $ 99.00 |
| ZVM-1240: Non-Glare, Amber Monitor, TTL | $159.00 |
| ZVM-1200-1: Tilt Swivel Base | $ 17.00 |
| ZVM-1330: High-Resolution Color, RGB | $459.00 |

## DYNAMIC RAM MEMORY

| | | |
|---|---|---|
| 4164 64K 150 NS.DRAM | QTY: 1-50: | $1.40 |
| 41256 256K 150 NS.DRAM | QTY: 1-50: | $2.90 |
| 41256 256K 120 NS. DRAM | QTY: 1-50: | $3.25 |
| 41256 256K 100 NS. DRAM | QTY: 1-50: | 3.95 |

## MODEMS

| | |
|---|---|
| US ROBOTICS SPORTSTER 1200 (EXT) | $136.00 |
| US ROBOTICS PASSWORD 300/1200 | $199.00 |
| US ROBOTICS MICROLINK 1200 W/Telpac | $239.00 |
| US ROBOTICS MICROLINK 2400 W/Telpac | $389.00 |
| US ROBOTICS COURIER 1200 | $239.00 |
| US ROBOTICS VAR MODEM 1200 (INT.) | $136.00 |
| US ROBOTICS VAR MODEM 2400 (INT.) | $198.00 |
| US ROBOTICS COURIER 2400e | $439.00 |
| US ROBOTICS COURIER HST (9600 baud) | $699.00 |

## PC COMPATIBLE UPGRADE ACCESSORIES

| | |
|---|---|
| INTEL ABOVE BOARD PC, 64K Expandable to 2 MB | $299.00 |
| EVEREX MAGICCARD | $159.00 |
| NEC V20 8 MHz Processor | $ 14.95 |
| INTEL 8087-3 5 MHz. Co-Processor | $109.00 |
| INTEL 8087-2 8 MHz. Co-Processor | $169.00 |
| SMARTWATCH Clock/Cal. Module | $ 44.00 |
| PRINTER CABLES, STANDARD PARALLEL, 6 Ft. | $ 22.00 |
| SWITCHBOXES: 2 or 4 Position, 5 Yr. Warranty | $ 99.00 |
| AST RAMPAGE EMS/EEMS w/256k, Software | $337.00 |
| AST ADVANTAGE EMS/EEMS w/128k Serial, Parallel Ports, Software | $398.00 |

## Z100 UPGRADE ACCESSORIES

| | |
|---|---|
| C.D.R. Z100 Speed Module | $ 39.95 |
| FBE RESEARCH ZMF-100A: Install Up To 768K On Z100 Motherboards #181-4917 Or Lower | $ 59.95 |
| UCI Ramboard for Z-100, OK Expandable to 2MB | $ 299.00 |
| SEAGATE 20 MB Winchester Kit, COMPLETE | |
| W/Zenith Contr., Hardware & Software | $849.00 |
| W/CDR Contr., Hardware & Software | $ 949.00 |
| SYQUEST 10MB Removeable Media HD w/CDR Contr., Hardware, Software, and One Cartridge | $1248.00 |

## PRINTERS

| | |
|---|---|
| EPSON DX10 Daisy Wheel | $ 189.00 |
| CANON LBP-8A1 Laser Prinier | $2290.00 |
| CANON A40 NLQ Dot Matrix, 140 cps. QUIET | $ 279.00 |
| OKIDATA ML182 Dot Matrix, 120 cps | $ 269.00 |
| OKIDATA ML192 Dot Matrix, 160 cps | $ 399.00 |
| C. ITOH Pro Writer Jr. | $ 259.00 |

## Z200 UPGRADE ACCESSORIES

| | |
|---|---|
| INTEL ABOVE BOARD AT, 128K, Expandable to 2 MB | $459.00 |
| INTEL ABOVE BOARD AT PIGGYBACK, 128K; Expandable to 2 MB; Expands AB AT to 4 MB | $229.00 |
| INTEL 80287 6 MHz. Co-Processor | $195.00 |
| INTEL 80287 8 MHz. Co-Processor | $318.00 |
| INTEL 80287 10 MHz. Co-Processor | $397.00 |
| SEAGATE ST4038 30MB, 40 ms. avg. | $669.00 |
| SEAGATE ST4051 40MB, 40 ms. avg. | $799.00 |
| NEWBURY NDR 1085 85 MB, 30 ms. avg | $1399.00 |
| NEWBURY NDR 2190 190 MB, 30 ms. avg | $3569.00 |
| MINISCRIBE 6053 53 MB, 28 ms. avg | $849.00 |

Nashua

**Box of 10 With Lifetime Guarantee**

| | |
|---|---|
| Nashua Diskettes: DS/DD, 48 TPI, Soft-Sectored or Hard-Sectored (Specify) | $\$9^{50}$ |
| Nashua Diskettes: DS/DD, 96 TPI, Soft-Sectored or Hard-Sectored (Specify) | $21.50 |
| Nashua Diskettes: DS/HD, SS, 600 Oersted Media for AT and Compatible 1.2 MB Drives | $27.95 |

## HARD DISK DRIVES

With One-Year Replacement Warranty

| | |
|---|---|
| SEAGATE ST 225 20MB Hard Disk | $299.00 |
| SEAGATE 20 MB HD w/OMTI | $399.00 |
| CMTI/LAPINES 20 MB Hardcard | $449.00 |
| SEAGATE ST238 30 MB w/OMTI RLL Controller | $499.00 |
| SEAGATE ST251 50 MB Half HT HD w/OMTI Controller | $899.00 |
| SMS 25 MB Tape Backup w/One Tape | 599.00 |
| SMS 60 MB Tape Backup w/One Tape | 949.00 |

## FLOPPY DISK DRIVES

With One-Year Replacement Warranty

| | |
|---|---|
| FUJITSU M2551A: 5¼, Halt Ht, 48tpi, DS/DD | $112.50 |
| MF501A: 5¼" HalfHeight, 48 TPI, DS/DD | $119.00 |
| MF504A: 5¼" Half Height, 96 TPI, 1.2 MB | $159.00 |
| 5¼" 360K External Floppy Drive for ZF-181 | $199.00 |

## SOFTWARE

| | |
|---|---|
| Hilgraeve HyperAccess for H/Z100 and PC Compatibles | CALL |
| FASTBACK ver 5.13 | $134.00 |
| MICROSOFT QUICK BASIC COMPILER | $69.00 |
| PETER NORTON UTILITIES | $79.00 |
| The Norton Commander | $59.00 |

## OTHER HARDWARE

| | |
|---|---|
| RAM Technology's Z-MAX 150 640/704K Upgrade w/RAMDSK SOFTWARE | $24.95 |
| RAM Technology's Z-MEG 150 1.2 MB Upgrade w/RAMDSK SOFTWARE | $49.95 |
| RAM Technology's Z-MEG 171 1.0 MB Upgrade w/RAMDSK SOFTWARE | CALL |
| MICROSPEED's FAST 88 SPEED Upgrade for 4.7 MHz PC's and Compatibles | $124.00 |
| LOGITECH SERIAL MOUSE | $89.00 |

## TERMS

Prices and specifications subject to change. Minimum 90-day warranty on all hardware. Personal checks held 10 working days; money orders accepted as cash. Please add 2% (minimum $2.00) for shipping; any surplus will be refunded. COD orders accepted; cash or cashier's check only. VISA/MC accepted; please add 2%. Purchase Orders accepted from businesses and educational institutions. User group inquiries invited. We appreciate your business!

# RAM Technology

**427-3 AMHERST STREET, SUITE 265**
**NASHUA, NEW HAMPSHIRE 03063**

**ORDER LINE: 1-800-662-0070**
**INFORMATION: (603) 889-0633**

HOURS OF OPERATION: Monday-Friday, 9:00 A.M. - 5:00 P.M. EST

MasterCard

VISA

# A High-Level Fortran Graphics Subroutine For The Z-100

***Arnold R. Miller and Ann M. Johnson***
*Department of Medical Information Science*
*University of Illinois*
*1408 West University Avenue*
*Urbana, IL 61801*

## *About The Authors*

***Arnold R. Miller*** *is Research Scientist and Visiting Assistant Professor in the Department of Medical Information Science, University of Illinois, Urbana. His research concerns a new branch of reliability theory that has applications to the biology of aging. When the work of this paper was performed,* ***Ann M. Johnson*** *was Research Programmer in the Department of Medical Information Science, University of Illinois. She is presently a graduate student of bio-statistics in the Department of Preventive Medicine and Biometrics, University of Colorado Health Science Center, Denver, where she also serves as computer consultant in the Medical Computing Center.*

### Introduction

As a workstation, the Z-100 computer plays an important role in our mathematical and computer modeling research. All codes (in Fortran) are developed locally on the Z-100. Those for number crunching are uploaded to one of two mainframes, a Cyber 175 or Cray X-MP, where they are recompiled and executed. Generally, some form of the data generated by the model is then downloaded to the Z-100 for analysis and storage. Many other programs, e.g., those for data analysis, are developed and used solely on the workstation. Graphics is essential in our work, and the bit-mapped graphics capability of the Z-100 contributes to its usefulness. Our primary display device for mainframe graphics is the Z-100 in the form of a Tektronix 4010 emulator. However, a useful, but commercially unavailable, function of the machine is the capability of doing high-level Fortran graphics locally, i.e., with the Z-100 as a stand-alone graphics workstation. For this purpose, we have written a Fortran graphics subroutine, called ZGRAF, that nicely accomplishes this objective and which we are placing in the public domain.

By "high-level Fortran graphics" we mean the capability, common in mainframe graphics, of creating a complete graph with axis labels from unscaled data arrays and a single subroutine CALL statement. We are unaware of commercially available software for the Z-100 that will accomplish this objective. There are, however, lower-level libraries of graphics primitives (routines that draw simple graphics figures, such as lines and circles), and indeed ZGRAF is based on one of these, the Clarkson University Graphics Library, whose routines are modeled after the graphics commands of Z-BASIC. There is also higher-level stand-alone graphics software that requires no Fortran (or other language) programming at all. However, for our work, the middle ground is best: software that allows the creation of a graphics display from a single Fortran statement from within the same program that created the graphics data in the first place.

In this article, we describe the use and give some technical details of subroutine ZGRAF. With ZGRAF, all information necessary to create a graph is contained in the subroutine argument list, which includes the names of the data arrays, various format control parameters, and character strings to label or annotate the graph. Depending on the format control parameters, ZGRAF plots the data in various forms and colors, draws a box around it, and labels it. It can subsequently overlay any number of graphs on the initial coordinate system and box, and it has a simple zoom capability.

### Use Of ZGRAF

In its simplest form, use of ZGRAF entails only a single CALL statement. Of course the data arrays must be dimensioned and the screen may need to be cleared, but otherwise no additional Fortran statements are required. The following example illustrates the simplicity of its use:

```
CALL ZGRAF( X, Y, 100, 'CONTINUOUS', 'NO',
:  'NORMAL', 'X  ', 'Y  ', 'CUBIC POLYNOMIAL',
:    IERR )
```

Real arrays X and Y contain the coordinates of the data points; the third argument specifies that there are 100 such points. The string 'CONTINUOUS', a format control parameter, specifies that a continuous curve will be drawn through the 100 points. 'NO' indicates that the graph is not an overlay, and hence, the coordinate system will be scaled with reference to the given arrays. 'NORMAL' describes the size ana orientation of the box on the screen. Following this are three character strings that are labels for the x- and y-axes and a title for the graph. The final argument, IERR, is an integer error code returned by ZGRAF. With the ZGRAF object module in a disk library accessible to the linker, the statement CALL ZGRAF(...) becomes an extension of the Fortran language: you simply use it where you please, without any additional fuss. Figure 1 illustrates the display (dumped to a dot-matrix printer) as generated by the above sample statement.

More generally, all ZGRAF arguments may be variables, which can therefore be read from a file, entered from the keyboard, or

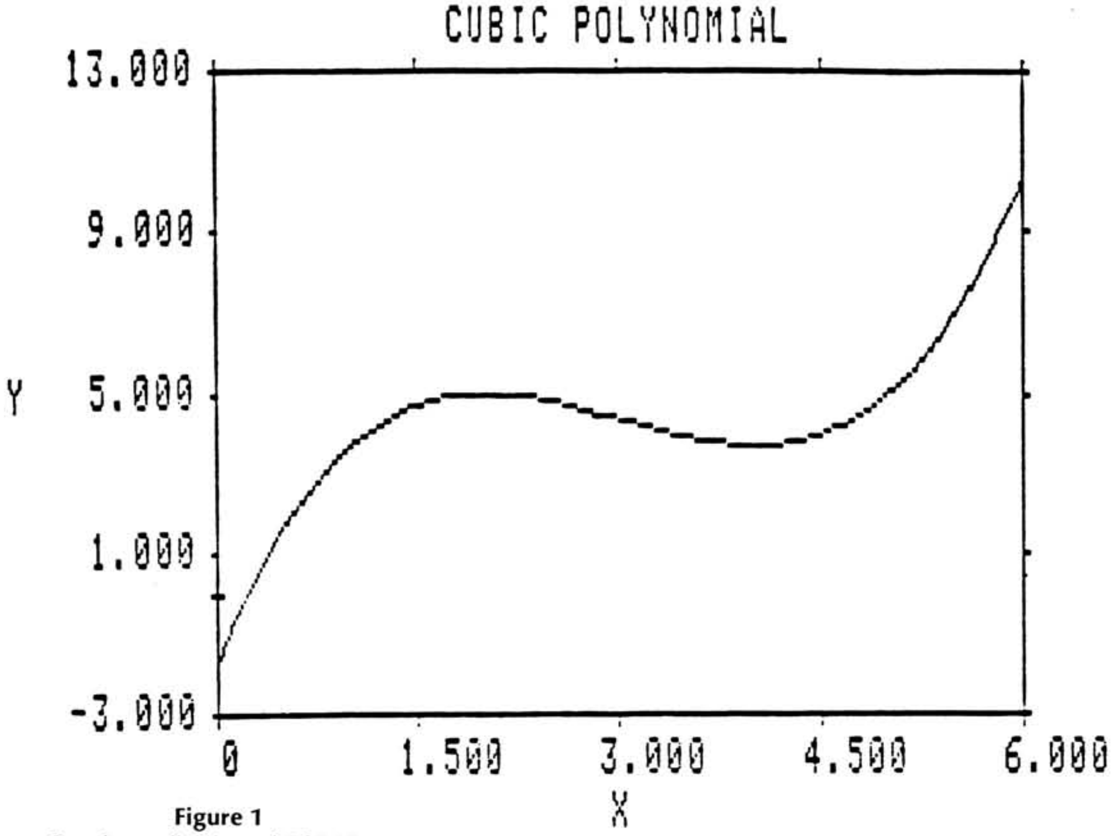


**Figure 1**
Simple application of ZGRAF —
The plotting of a cubic polynomial.

computed by the calling program. In this case, at least the character variables must be specified by a Fortran type statement. The general form is

```
CALL ZGRAF( X, Y, N, OPT, OVRLAY, PLACE,
  XLABEL,
:    YLABEL, TITLE, IERR )
```

where the arguments are defined as follows:

X and Y are single-precision real arrays as described above. N, an integer, is the dimension size of X and Y and is limited only by the compiler and memory size of the computer.

OPT is a character variable of length three or more that specifies the style and color of the graph. The possible values of the variable and corresponding graphic characteristics are as follows. (Note: character substrings in parentheses are optional.) The value 'NUL(L)' specifies a "null graph"; it simply establishes a coordinate system and draws and labels the box. 'CON (TINUOUS)', the default value, draws a continuous curve, in the foreground color of the screen, through the data points. 'DIS (CRETE)' draws vertical line segments from the bottom or top of the box to the data points (see Figure 2). 'POI(NTS)' simply turns on the pixels corresponding to the data points (see Figure 3). The OPT values 'BLA(CK)', 'BLU(E)', 'RED', '(MAG)ENTA', '(GRE)EN,' 'CYA(N)', 'YEL(LOW)', AND 'WHI(TE)' specify a continuous curve of the indicated color through the data points. The value 'BLACK', which is similar to 'NULL', draws an invisible graph, or more usefully, erases an existing one; 'WHITE' is equivalent to 'CONTINUOUS'. On a monochrome monitor the colors will be displayed as shades of gray that increase in intensity from black (the background color) to white (the foreground color) in the order given.

OVRLAY is a character variable of length one or more that specifies whether the graph is an overlay. It assumes only two values, 'N(O)' and 'Y(ES)'. The value 'NO' indicates that the graph is not to be an overlay, and hence, the coordinate system is established with respect to the arrays in the same CALL statement. 'YES' specifies that the graph is to be an overlay on an existing coordinate system, which is assumed to have been established by a previous CALL statement.

PLACE is a character variable of length one or more that specifies the size and orientation of the box. The value 'N(ORMAL)', the default, places the box on the screen such that space (for writing text or annotations from the calling program) remains above and to the left. Similarly, the values 'S(MALL)', 'C(ENTER)', 'V(ERTICAL)', and 'F(ILL SCREEN)' control the size and orientation of the box.

Character variables XLABEL (of maximum length 30), YLABEL (maximum length 18), and TITLE (maximum length 80) specify the x-axis label, the y-axis label, and the title, respectively. The strings are limited to characters from the upper-case alphabet, the digits 0–9, and the 22 special characters *'^ ><[]()-=%+"=/=\=__,=.:;==%. Labels are terminated by two consecutive blanks, but may have leading blanks. The labels are automatically centered.

IERR is a three-digit integer that reports the error status of ZGRAF. ZGRAF attempts to recover from most error conditions, and there are only two fatal errors: failure to specify either OVRLAY == 'NO' or OVRLAY == 'YES', and specifying OPT == 'NULL' when OVRLAY == 'YES'.

Listings 1 and 2 illustrate the general use of ZGRAF. Listing 1 and the resulting display in Figure 3 illustrate the use of OPT = 'POINTS' and OVRLAY = 'YES' — a regression function is overlaid on sample data points. Note that the arguments PLACE, XLABEL, YLABEL, and TITLE are ignored when OVRLAY = 'YES'. In order that Fortran WRITE statements may annotate the dis-

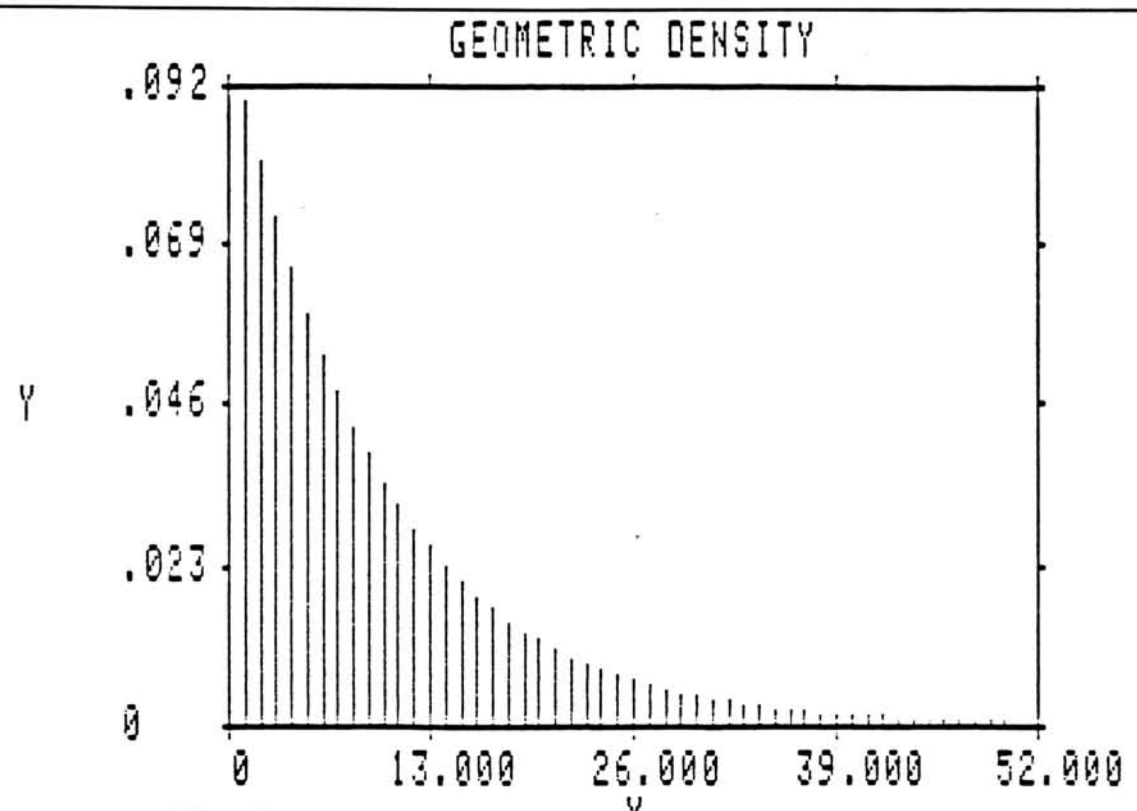


**Figure 2**
Application of OPT = 'DISCRETE' —
The plotting of a discrete probability density function.

play before ZGRAF is called, ZGRAF does not automatically clear the screen (the Clarkson routine CLS is used); also, it does not automatically pause after graphing the data (the READ statement accomplishes this).

Listing 2 and the display in Figure 4 illustrate the use of OPT = 'NULL' in conjunction with OVRLAY = 'YES'. This option allows the possibility of forcing the initial coordinate system to the programmer's specifications, which allows, for example, a simple zoom capability — the magnification of a portion of a graph. Figure 4 shows a detail from Figure 1. To use the 'NULL' option for this purpose, one first creates two null data arrays, NULLX and NULLY in the listing, containing only two elements each, which together are sufficient to determine the boundaries of the box. A coordinate system is established and the box is drawn, but the data points are not plotted. Subsequent calls to ZGRAF using OVRLAY = 'YES' overlay the desired graphs on the null graph.

The Z-100 software requirements for ZGRAF are dictated by the Clarkson graphics primitives. Presently, the routines, for the Z-100 only, are available for versions 3.1 and 3.2 of Microsoft Fortran; if they become available for other computers or other Fortran compilers, then our code, written in standard Fortran 77, should be portable. With the Microsoft linker, presumably our object code can also be linked to Pascal programs. The Clarkson primitives are written in 8088 assembly language and reference the non-interlaced video of the Z-100. We use the 8087 coprocessor, which accelerates floating-point operations (there are no transcendental functions in our code) but has no effect on the actual drawing operations performed by the primitives. In our machines, ZGRAF is quite fast: in the program that created Figure 1, execution of the ZGRAF subroutine itself required =.02 second. However, the subroutine requires substantial memory. Our Fortran object code is 56K in size, and the Clarkson routines add about another 15K. Thus, the simplest executable Fortran program, one that simply calls ZGRAF, requires more than 70K of memory.

### The ZGRAF Code

Subroutine ZGRAF is actually the executive module of a collection of nine subroutines. ZGRAF itself calls seven others — ZLOW, ZUP, ZTIC, ZAXES, ZLINE, ZDISC, and ZPOINT — and ZAXES calls an eighth — ZTRING. Of these, only ZAXES, ZLINE, ZDISC, ZPOINT, and ZTRING directly call the Clarkson routines. Our code is written in structured, modular, readable, standard Fortran 77; the copiously commented source is 103K in size. It is easily maintainable, and since the display format is controlled by easily changed program parameters, ZGRAF is highly customizable.

ZLOW, ZUP, and ZTIC are purely numerical routines that create a coordinate system for the graph. ZLOW and ZUP find "nice" lower and upper bounds, respectively, for the data. A "nice" bound is a real number that is a bound (upper or lower) and contains only two significant digits. For example, if the X array has data extending from x = 3.151 to x = 24.75, nice bounds are 3.1 and 25. Starting with nice bounds for each axis, ZTIC finds a tic-interval length such that four tic intervals cover the data along each axis and such that each of the resulting five tic marks occurs at nice values of x or y. In the above example, ZTIC will compute a tic interval length of six units with tic marks at x = 2, 8, 14, 20, and 26. When the data are graphed, it will of course extend (to the nearest pixel) from 3.151 to 24.75 along this scale. From the endpoints of the scale, returned by ZTIC, control module ZGRAF computes linear functions that map the coordinate system and data to pixel locations on the screen.

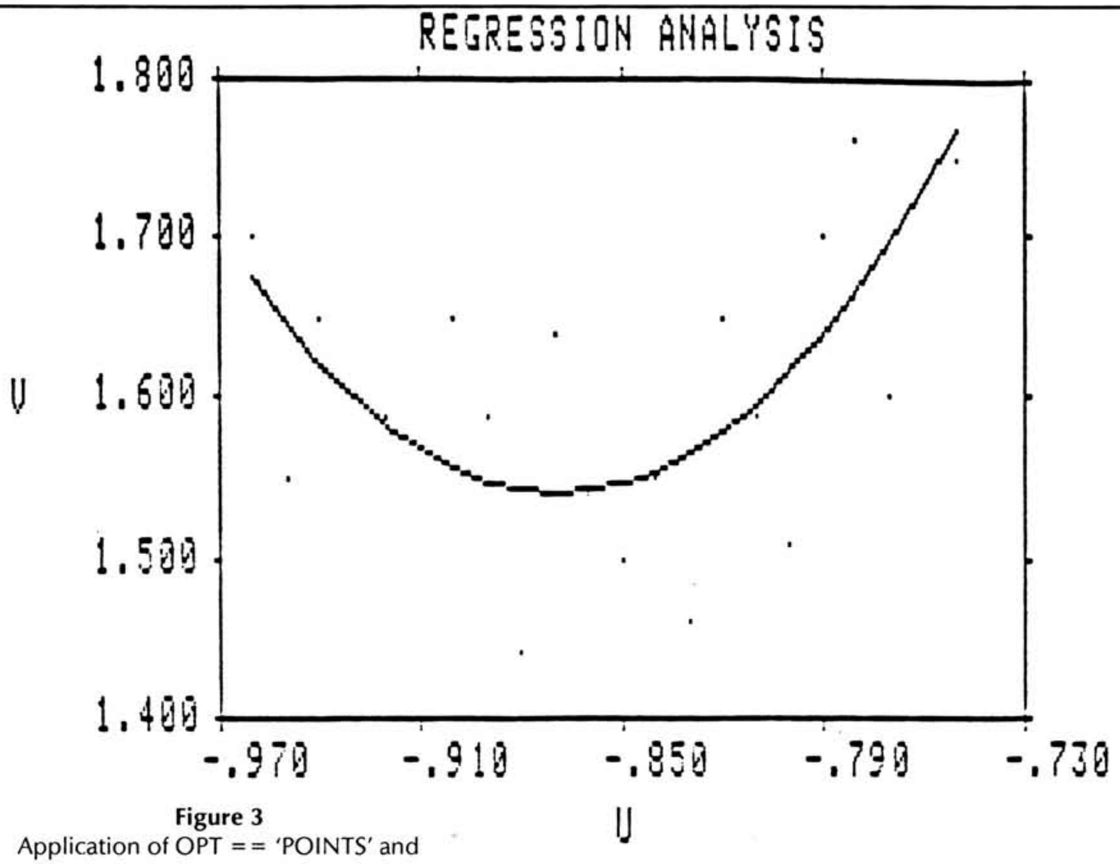


**Figure 3**
Application of OPT == 'POINTS' and
OVRLAY == 'YES' to regression analysis

Subroutine ZAXES is straightforward in function but rather involved in implementation. It draws the box enclosing the graph; the box is marked with tic marks on all four sides and labeled. Numerical labels are centered at each tic mark along the bottom and left side, the x- and y-axis labels are centered below the bottom and to the left, and the title is centered above the box. For the actual labeling operations, ZAXES calls subroutine ZTRING.

**Listing 1**
Use of the OPT = 'POINTS' and OVRLAY = 'YES' options.

```
      PROGRAM REG
      REAL  U(5Ø), V(5Ø), X(5Ø), Y(5Ø)
      CHARACTER  ANY, TITLE*8Ø
C
      TITLE == 'REGRESSION ANALYSIS  '
            =.
            =.
            =.
C   ARRAYS U, V, X, AND Y ARE LOADED IN THIS PART OF THE PROGRAM.
C   U AND V CONTAIN THE COORDINATES OF 22 EMPIRICAL DATA POINTS AND
C   X AND Y CONTAIN THE COORDINATES OF 22 POINTS FOR THE REGRESSION
C   CURVE.
            =.
            =.
            =.
C   CLEAR THE SCREEN:
      CALL CLS
C   GRAPH THE EMPIRICAL DATA:
      CALL ZGRAF( U, V, 22, 'POI', 'NO', 'NORM', 'U  ', 'V  ',
     :    TITLE, IERR )
C   OVERLAY THE REGRESSION FUNCTION:
      CALL ZGRAF( X, Y, 22, 'CONT', 'YES', '  ', '  ', '  ',
     :    '  ', IERR)
C
C   PAUSE TO VIEW THE DISPLAY:
      READ(*,'(A1)') ANY
C
      END
```

ZTRING is based on the Clarkson DRAW routine, analogous to the command by the same name in Z-BASIC, which has arguments that allow the angle of orientation and scale factor (size) of a drawn figure to be easily varied. Subroutine ZTRING contains a library of 58 characters (see above), each drawn within an 8 × 9 pixel block and having a similar font to the Z-100's standard character set. (The actual character is generally 5 × 7 pixels within a border.) We originally intended to use the rotation capability of DRAW to place characters sideways within vertical labels, but the small size of the Z-100's standard characters and the low pixel density in the vertical screen direction (225 vertically versus 640 horizontally) presented legibility problems. Because of the different pixel densities along the two axes, an 8 × 9 pixel block (9 pixels vertical) is approximately twice as tall as it is wide. If the block is rotated 90°, the DRAW routine preserves its proportions so that the resulting block is about twice as wide as it is tall. However, because of the low pixel density in the vertical direction, the block may now contain only four pixels along the y-axis. This is generally too few to form a legible character. Larger characters, however, may successfully be rotated. We chose not to increase the character size but to orient the characters of vertical strings

upright, one character below another. However, ZGRAF is easily modified so that vertical strings can be drawn sideways. It is also easily modified so that larger characters are drawn. Even without exploiting the rotation and scale-factor capabilities of DRAW, subroutine ZTRING is much more flexible and convenient than using Fortran WRITE statements to annotate the graph. ZTRING can begin a string at any of 225 vertical and 640 horizontal pixel locations; Fortran can begin a string at only 25 vertical and 80 horizontal locations. ZTRING draws a string without removing existing information from the screen; Fortran writes a line by first obliterating an existing line (for this reason WRITE statements should generally precede a call to ZGRAF). ZGRAF can be used separately in the calling program to annotate the screen with, e.g., enlarged or rotated characters; Fortran is restricted to the standard character set of the Z-100.

Subroutine ZLINE, ZDISC and ZPOINT do the actual plotting of the data once the coordinate system, box, and labels are in place. They use the linear functions computed by control module ZGRAF to map the data points to pixel locations. They also have the important function of properly truncating an overlay whose data exceeds the bounds of the box. ZLINE, based on the Clarkson routine LINE, implements the 'CONTINUOUS' and color options specified by OPT. ZDISC, also based on LINE, draws the vertical line segments for graphs using the 'DISCRETE' option. ZPOINT, based on the Clarkson PSET routine, merely turns on the pixel corresponding to each data point when the 'POINTS' option is used.

We have been using the ZGRAF subroutines in various stages of refinement for more than a year in our research, and we are not aware of any bugs in our Fortran code. Although we naturally cannot vouch for the general robustness of the Clarkson library or the Microsoft compiler, we can say that we have experienced no trouble with them in debugged versions of ZGRAF and with our applications.

## Discussion

We have found ZGRAF to be an indispensable tool. We have intensively used it in our research, and it fully meets our expectations (up to the character orientation of vertical labels) from the beginning of this project. As with most microcomputer software, ZGRAF is easier to use than mainframe graphics (but of course it is much less powerful). To be able to do off-line, high-level Fortran graphics, especially graphics involving experimentation, without the contention and unreliability of a mainframe environment is a great convenience. As we have shown in this paper, ZGRAF provides quite usable moderate-resolution (225 × 640 pixels), two-dimensional graphics and — in conjunction with a screen-dump program — hardcopy plots. It is starkly simple, fast, easily customizable, and robust. In summary, ZGRAF performs a limited number of graphics functions, but it performs them efficiently and very well.

We believe that others also will find ZGRAF a useful tool, and to this end we are placing the source code in the public domain. Since the Clarkson graphics primitives are a prerequisite to using the code, the Clarkson Educational Computing System has agreed to provide ZGRAF and its brief documentation to purchasers of the Graphics Library. Neither Clarkson University nor we can provide support for ZGRAF itself. The authors would, however, appreciate being notified of any bugs attributable to ZGRAF.

## Acknowledgement

This work was supported by Grant AG03331 awarded by the National Institute on Aging.

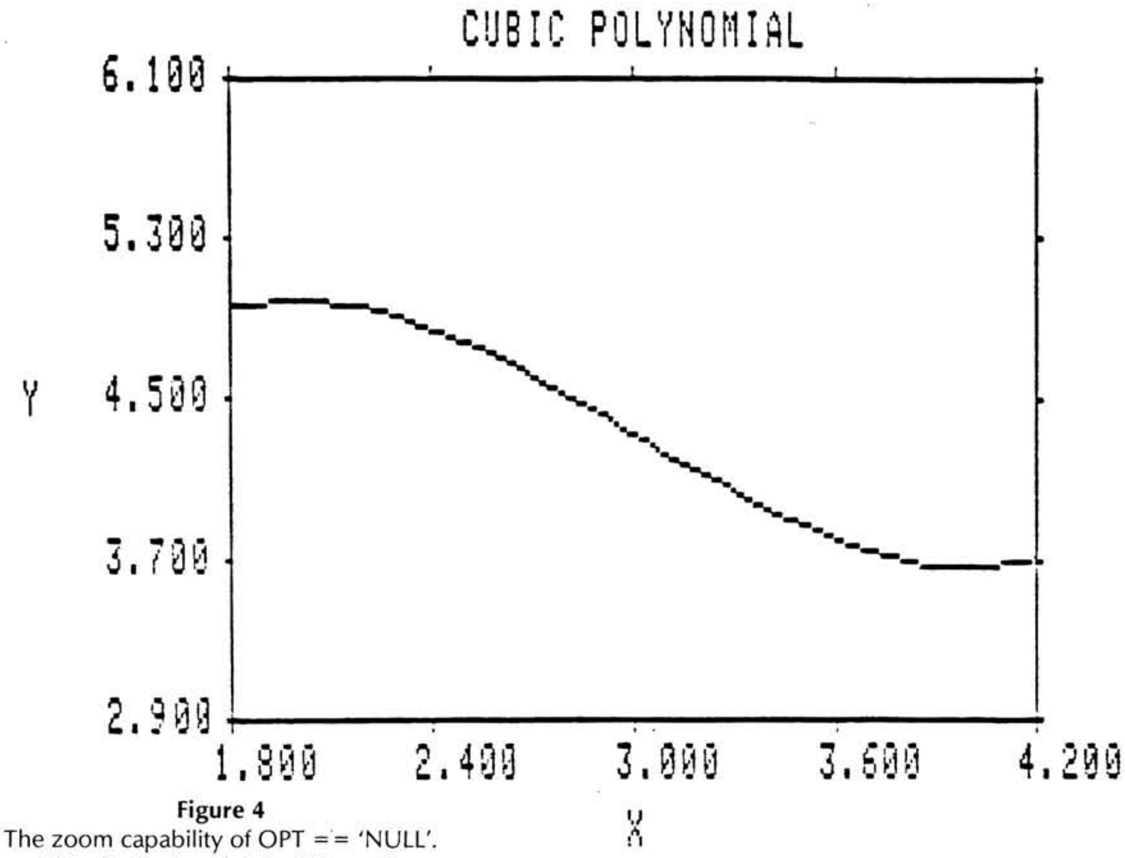


**Figure 4**
The zoom capability of OPT == 'NULL'.
This display is a detail of Figure 1.

### Listing 2

Use of the zoom capability of OPT = 'NULL'.

```
      PROGRAM ZOOM
      REAL  NULLX(2), NULLY(2), X(100), Y(100)
      CHARACTER  ANY, TITLE*80
C
      TITLE == 'CUBIC POLYNOMIAL  '
C
C   SET ARRAYS NULLX AND NULLY:
      READ(*,*) NULLX, NULLY
          =.
          =.
          =.
C   ARRAYS X AND Y ARE LOADED WITH THE SAME DATA AS USED FOR
C   FIGURE 1 IN THIS PART OF THE PROGRAM.
          =.
          =.
          =.
C   CLEAR THE SCREEN:
      CALL CLS
C   GRAPH THE NULL GRAPH:
      CALL ZGRAF( NULLX, NULLY, 2, 'NULL', 'NO', 'NORM', 'X  ',
     :      'Y  ', TITLE, IERR )
C   OVERLAY THE FUNCTION:
      CALL ZGRAF( X, Y, 100, 'CONT', 'YES', '  ', '  ', '  ',
     :      '  ', IERR)
C
C   PAUSE TO VIEW THE DISPLAY:
      READ(*,'(A1)') ANY
C
      END
```

### Ordering Information

Graphics Library, $49
(substantial discounts for volume purchases).
Educational Computing System
Clarkson University
Potsdam, NY 13676
(315) 268-6455

# PW
# Password Protection For Your MS-DOS Computer

*Jeff Kalis*
*1920 Sylvan SE*
*Grand Rapids, MI 49506*

There are times when I wish MS-DOS had some sort of file security. There was a time when I was doing some programming on a TRS-80 Model II from Radio Shack and found their dual level password protection effective and useful. MS-DOS has no file protection except for the READ ONLY and HIDDEN attributes, but these are easily removed by a number of attribute utilities. Data encryption can be employed, but each time the file must be decrypted before it is usable.

I figured the best way around this was to limit the access to the whole entire computer! If the system would ask for a password before the computer was fully booted, unauthorized use could be controlled. Writing a password program was no big deal. I stuck it in a batch file and it worked great. There was just one problem. While the computer is busy waking up and just after COMMAND.COM finds AUTOEXEC, a Control-C will break the batch processor and ask if the batch file should be terminated. Answering yes will stop the AUTOEXEC batch from running and stop the password program before it even starts.

The only place where a program can be installed where it can't be defeated is in CONFIG.SYS. CONFIG.SYS is a text file that contains configuration information for MS-DOS. One use of the CONFIG.SYS file is to install additional device drivers for peripheral equipment. Included with MS-DOS version 2 and 3 are two such drivers, ANSI .SYS and VDISK.SYS. PW is a device driver although there is really no device to drive. The program is set up to handle any commands that may be sent to it, but the only one implemented is the initialization command. PW should run on any computer running MS-DOS version 2.0 or higher. I have checked it out on a Z-100, Z-150 and a Tandy 3000 AT compatible.

There are two types of device drivers, character devices and block devices. Block devices are used to talk to things like disk drives. A block device will normally have a drive letter for a name. These types of device drivers are rather difficult to implement and I won't even attempt a discussion here. VDISK.SYS is an example of a block device driver.

Character device drivers operate on one character at a time. Names of some standard character device drivers are CON, AUX, COM1, etc. These drivers are contained in IO.SYS and installed during boot up. PW.SYS is a character device driver.

Up until this project, device drivers were a little confusing to me. I looked into them briefly before, but could not make much sense out of the information. I also had no application, which made my attention span a little shorter than was needed. I find that without an application, there is not much chance I will retain much. It's like trying to learn to program in C by just reading a book. Good luck.

Character device drivers are really no big thing. There are three parts to a device driver. MS-DOS provides a fourth part which is used to pass information. The three parts of the device driver are (1) the Device Header, (2) the Strategy Routine and (3) the Interrupt Routine. The part which MS-DOS provides is called the Request Header. Of these four parts, the Request Header and the Device Header are just data tables. The Interrupt Routine and Strategy Routine contain the executable code for the driver.

Device drivers are nothing more than a special type of COM file. A COM file has an origin of 100H where a device driver must have an origin of 0. In a COM file, execution will begin at location 100H. In a device driver, location 0 is the start of the Device Header and execution can begin at any location in the driver. The specific location is contained in the Device Header.

A Device Header consists of 5 fields with a total length of 18 bytes. The fields break down like this:

| Field | Length | Use |
|---|---|---|
| 1 | DWORD | Pointer to Next Device |
| 2 | WORD | Attribute Field |
| 3 | WORD | Pointer to Strategy entry point |
| 4 | WORD | Pointer to Interrpt entry point |
| 5 | 8 bytes | Device Name |

MS-DOS can and does have many device drivers installed when the system is running. Each device follows another in a chain. These devices do not follow each other directly though, there is code in between the drivers. To allow MS-DOS to traverse the chain easily, the first double word in the Device Header will point to the next device in the chain. This field must contain 0FFFFFFFFH when it is loaded. MS-DOS will modify this pointer when the next device is installed.

The attribute field is used to inform MS-DOS what type of device driver it is. If bit 15 is clear, the device driver is a block device. If bit 15 is set, the driver is a character device. PW is a character device, so bit 15 will be set. Other bits in the attribute byte have no use in PW and a discussion of them will not be continued here.

The third and fourth fields contain pointers to the entry points of the Strategy and Interrupt routines, respectively. These pointers are words so they are only offsets from the same segment that is used to point to the Device Header.

The fifth field is an 8 byte field containing the name of the device driver. In a block device, this field takes on a different meaning. In PW.SYS, the name of the driver is PW$. The remaining 5 bytes are spaces.

MS-DOS provides a second table called the Request Header. This is a variable length data table somewhere in memory. When the device driver's Strategy Routine is called, the location of the Request Header is contained in the ES:BX register pair. More on the Strategy Routine later, but now we'll look into the Request Header data table.

The Request Header is always at least 13 bytes long. These first 13 bytes are called the Static Request Header. The third byte in the Static Request Header will contain a command code. There are 13 defined commands for a device driver under MS-DOS version 2 and 16 commands for version 3. PW only implements function 0, the initialization function. All the other functions simply return a done flag in the status word.

The status word is the fourth and fifth bytes in the Request Header. This status word is used to tell MS-DOS how things went when the device driver returns control back to MS-DOS. The length of the entire Request Block will change depending on the command. For a more complete discussion of the Request Header and device drivers in general, I would direct you to the Programmers Utility Pack available from ZDS.

When a device driver is installed, the first thing to happen is the Strategy Routine gets called. The Strategy Routine is called only once and only when the driver is installed. It's function is to get the pointer to the MS-DOS Request Header and save it in a place where the Interrupt Routine can get at it. The pointer to the Request Header is contained in the ES:BX register pair. ES contains the segment and BX contains the offset. The Strategy Routine in PW saves the ES:BX pointer in REQHDRS and REQHDRO. When PW's Interrupt Routine is called, it will know right where to find the Request Header.

Immediately after the Strategy Routine finishes up and returns to MS-DOS, the Interrupt Routine is called. The Interrupt Routine is then going to get the pointer to the Request Header which was saved by the Strategy Routine. After the pointer is retrieved, the command byte is examined. There is a command dispatcher in the Interrupt Routine which directs control to the proper place within the Interrupt Routine. This first call to the Interrupt Routine will have a command byte of 0. Command 0 is the initialization command. Command 0 will only be called once and only immediately after the Strategy Routine is called. The initialization command is the only one implemented in PW. All the others simply return a done flag in the status byte of the Request Header.

The Interrupt Routine is where PW does its work. The first thing that is done is to save all the processor registers. Next, PW determines which version of MS-DOS is running and adjusts the maximum number of commands allowed the device driver. The pointer to the Request Header is retrieved and loaded in DS:BX. PW looks at the command byte to see if it is a good command. The first command will be the initialization command 0, but there has to be a mechanism for handling all the commands that may be sent to the device driver. If the command is not known by PW, an unknown command status is returned to DOS.

When the command is verified, it is then used to create an offset into the command dispatch table. Because each table entry is 2 bytes long and the commands start at 0, it is a simple matter to double the command value and add it to the first location of the table. Now there is a pointer to the command routine that was requested. This is loaded in SI then the DS register is restored. An indirect jump to where SI is pointing will give control to the proper routine.

Even though PW only uses command 0, another command could be received by the device driver. Like COM and LPT, PW could be written to, read from, etc. The device name is specified in the 8 byte name field of the Device Header. PW's device name is PW$. If the device name was the same as the program name, PW, there would be no way to get to the program. When a command line is entered, the device drivers are searched first by DOS. If a driver matches, then DOS will try to perform the function using the device driver. Because of this, PW has to be able to handle any command that may be thrown at it. Any command other than 0 will return directly with the done flag set in the status word.

Once PW's initialization command is called, PW sets up a local stack. The stack space provided by DOS when a device driver is called is small. PW uses more space so a local stack is created. The original stack pointer is saved in the new local stack.

The only reason the initialization command is called immediately after loading the device driver is so the device driver can tell DOS how much memory it uses. This information is needed by DOS so it knows where to load the next driver in the chain. This is what PW does next. The Request Block for an initialization command is 22 bytes long. The first 13 bytes make up the Static Request Header and the remaining 9 bytes are specific to the initialization command. Of these 9, only four bytes are used by character devices. This is where the driver stores the next free byte of memory after the device driver. It is similar to the terminate and stay resident function. These 4 bytes are called the Break Address. They will also be returned in DS:DX when the driver terminates. PW stores the address of the next byte after the local stack as the Break Address. In doing this, PW will stay resident while the computer is running. This takes up some RAM space but not enough to worry about.

It is here that PW takes a deviation from the standard device driver. Before returning to DOS, a call is made to the password program. Now we finally get to the real program. The initial sign-on message is printed on the screen. PW then inputs a string from the keyboard. As each character is typed, an asterisk is echoed on the screen. Every keystroke is accepted as an input, except a backspace and a return. This means that any keyboard character can be used as a character in the password, even control characters. The backspace key works like one would expect. When the return key is pushed, the string of characters entered is compared to the password stored in the program. If they match, the 'Access Granted' message is displayed then the program returns to the initialization routine of the device driver. If the password entered does not match, the 'Access Denied' message is displayed and the processor enters an endless loop. The only way out of the loop is to reset the computer.

The initial password in PW is 'password'. It must be entered in lower case letters. The maximum length for a password in PW is 10 characters. Any characters after the last one up to the tenth character must be spaces. Here are some examples of other passwords that may be used:

```
PW_WORDDB   'Apple     '
; The A must be upper case, the rest lower
PW_WORDDB   'spa    ces'
; 3 char's then 4 spaces then 3 char's
PW_WORDDB   1,'control  '
; Control A char then the word control
```

In every case, the password is 10 characters long. If the word is shorter than 10 characters, spaces are assumed. In the third example, the first character entered is a control A. Push the control key and then the A key. One asterisk will be echoed on the screen. Next enter the letters in the word 'control', in lower case of course. Even the function keys could be used by putting the 2 character key codes into the password.

The only MS-DOS function call used by the password part of PW.SYS is function 6, Direct Console I/O. This is used because there is no check for Control-C with function 6. If function 9, Display String, was used to display the messages, a Control-C typed in at just the right time would break out of the program.

Type in the program using your favorite editor, then assemble the code into PW .SYS using the following commands:

```
MASM PW;
LINK PW;
EXE2BIN PW PW.SYS
```

When checking the PW.SYS driver, it is a good idea to format a new floppy disk and install the MS-DOS system files on it. Next create CONFIG.SYS on the new floppy. CONFIG.SYS should contain the line 'device=pw.sys'. This is easily done by the following commands:

```
COPY CON d:CONFIG.SYS
device=pw.sys
^Z[RETURN]
```

d: is the drive where the new floppy is located. ^Z is a Control-Z character. COPY the PW.SYS file just created to the new floppy. To test out the PW.SYS driver, reboot the computer using the new floppy disk. This is done because if there is a problem with the program, you will still be able to boot the computer normally with your regular disk or hard disk partition.

Install PW.SYS and CONFIG.SYS on each bootable floppy disk you would like protected. Putting PW.SYS on a hard disk will set up password protection there. The only drawback to PW is anybody with a system disk that does not contain PW can still boot the computer.

As always, if typing is not your bag, send me a buck and your disk and I'll return the disk with the PW files on it. Sending me six dollars will get you a new DSDD disk with the PW files on it. Happy computing.

```
        PAGE    60,132

        TITLE   PW.SYS  Password Entry Driver

FAR_RET MACRO
        DB      0CBH            ; Define far return
        ENDM

BK_SPC     EQU  8          ; ASCII back space char
LF         EQU  10         ; ASCII line feed char
CR         EQU  13         ; ASCII carriage return char
ESC        EQU  27         ; ASCII escape char
MAX_PW_LEN EQU  10         ; Length of password
SYS_CALL   EQU  21H        ; Function call interrupt
GET_DOS    EQU  30H        ; Get DOS version function
GET_BYTE   EQU  -1         ; FF for direct I/O input
DIRIO      EQU  6          ; Direct I/O function

                           ; Offset into Static Request Header for
CMD     EQU     2          ;   command byte
STAT    EQU     3          ;   status byte
_ENDOFF EQU     14         ;   end of code offset
_ENDSEG EQU     16         ;   end of code segment

DON     EQU     01H        ; Status bit for function DONE
ERR     EQU     80H        ; Status bit for function ERROR
_?CMD   EQU     03H        ; Error code for unknown command
ATTRIB  EQU     8000h      ; Attribute indicating character device

DUMMY   SEGMENT STACK      ; Eliminates Stack warning in LINK
DUMMY   ENDS

CODE    SEGMENT
        ASSUME  DS:CODE,CS:CODE,ES:CODE,SS:CODE

BEGIN:

; Define Device Header

DEV_HDR DW      -1          ; Pointer to Next Device Header
        DW      -1
        DW      ATTRIB      ; Attribute for this device
        DW      STRAT       ; Pointer to Strategy entry point
        DW      INT         ; Pointer to Interrupt entry point
        DB      'PW$     ';Name of this device, 8 bytes long

; Strategy Routine - All that's done here is to save the
;                    pointer to the Static Request Header
;                    (SRH)

STRAT:  MOV     CS:REQHDRO,BX ; Save SRH Offset
        MOV     CS:REQHDRS,ES ; Save SRH Segment
        FAR_RET               ; Back to DOS

PSRH    LABEL   DWORD         ; Pointer to static request header
REQHDRO DW      0             ; Offset pointer to SRH
REQHDRS DW      0             ; Segment pointer to SRH

; Command Dispatch Table - The function passed to this
; device in the SRH is converted to an offset into this
; table so the correct function will be called.  All
; functions not implemented just return a done status
; flag.

FUNCTBL DW  PW_INIT    ; Initialization Routine
        DW  PW_DONE    ;  Media Check    - not implemented
        DW   PW_DONE   ;  Build BPB       - not implemented
        DW   PW_DONE   ;  I/O Control Input - not implemented
        DW   PW_DONE   ;  Input           - not implemented
        DW   PW_DONE   ;  No Wait Input - not implemented
        DW   PW_DONE   ;  Input Status  - not implemented
        DW   PW_DONE   ;  Input Flush   - not implemented
        DW   PW_DONE   ;  Output Routine - not implemented
        DW   PW_DONE   ;  Output w/verify - not implemented
        DW   PW_DONE   ;  Output Status - not implemented
        DW   PW_DONE   ;  Output Flush  - not implemented
        DW   PW_DONE   ;  I/O Control Out - not implemented
        DW   PW_DONE   ;  Device Open   - not implemented
        DW   PW_DONE   ;  Device Close  - not implemented
        DW   PW_DONE   ;  Removable Media - not implemented
        DW   PW_DONE   ;  Output Till Busy - not implemented

; Interrupt Routine - This routine is called directly after
;                     the strategy routine.

INT:    PUSHF                   ; Save all the registers
        CLD
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    SI
        PUSH    DI
        PUSH    DS
        PUSH    ES
        PUSH    BP

        MOV     AH,GET_DOS ; Get the DOS version number
        INT     SYS_CALL
        CMP     AL,3        ; Adjust number of functions if < V3.x
        JAE     DOS3
        SUB     BYTE PTR LAST_FNC,3
DOS3:   LDS     BX,CS:PSRH       ; Load DS:BX with the SRH pointers
        MOV     AL,[BX+CMD]      ;  Get the command
        MOV     BYTE PTR SCMD,AL;  Save it
        CMP     AL,LAST_FNC      ;   check for legal function
        JA      BADFUNC          ;    error return if not
        XOR     AH,AH
        SHL     AX,1             ;    else convert command byte to table
        LEA     SI,FUNCTBL       ;    offset and get function address
        ADD     SI,AX            ;    in SI
        PUSH    CS
        POP     DS               ;    Reset DS
        JMP     WORD PTR [SI]    ;  Jump to the function

BADFUNC:MOV     AL,_?CMD         ; Set error code to unknown command

ERROUT: MOV     AH,ERR+DON       ; Set error and done bits
        JMP     SHORT EXIT

PW_DONE:MOV     AH,DON           ; Set done bit
        XOR     AL,AL

EXIT:   LDS     BX,CS:PSRH       ; Get the pointers to the SRH
        MOV     [BX+STAT],AX     ;   Store the status word
```

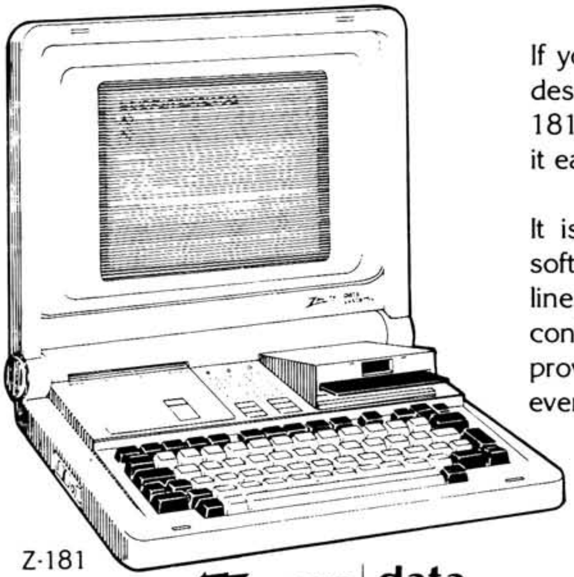
```
        POP     BP                  ; Restore all the registers
        POP     ES
        POP     DS
        POP     DI
        POP     SI
        POP     DX
        POP     CX
        POP     BX
        CMP     BYTE PTR SCMD,Ø
        JZ      NOTØ
        MOV     DS,BRKOFF           ; Break address in DS:DX if command INIT
        MOV     DX,BRKSEG
NOTØ:   POP     AX
        POPF
        FAR_RET                     ; Back to DOS


PW_INIT:MOV     AX,SS               ; Get the stack address in AX:BP
        MOV     BP,SP
        MOV     DX,CS               ; Set up a new local stack
        MOV     SS,DX
        MOV     SP,OFFSET STACK
        PUSH    AX                  ; Save the old stack address
        PUSH    BP                  ;  in to our new stack

        PUSH    CS                  ; Make ES=CS
        POP     ES
        LEA     AX,CS:WORD PTR LAST_BYTE+1; Put the offset address of the
        MOV     DX,CS               ;  last byte of our code into AX
        LDS     BX,CS:PSRH          ; Get the address of the SRH
        MOV     [BX+_ENDSEG],DX ;Load the address of the end
        MOV     [BX+_ENDOFF],AX ;   of our code into the SRH
        MOV     BRKSEG,DX           ; Save Segment
        MOV     BRKOFF,AX           ; Save Offset
        PUSH    CS
        POP     DS                  ; Make DS=CS

        CALL    PWORD               ; Run the program

        POP     BP                  ; Restore the original stack
        POP     SS
        MOV     SP,BP
        JMP     PW_DONE             ;  Go back to DOS


PWORD:  MOV     BX,OFFSET INIT_MSG   ; Print initial message
        CALL    PSTRNG
        MOV     BX,OFFSET PW_BUF    ; Point to input buffer
        CALL    ISTRNG              ;  input a string
        MOV     SI,OFFSET PW_BUF    ; Point to input buffer
        MOV     DI,OFFSET PW_WORD   ; Point to password
        MOV     CL,MAX_PW_LEN       ;  count in CX
        XOR     CH,CH
        REPZ    CMPSB                 ; Compare input to password
        JZ      IS_OK                 ;  branch if match

        MOV     BX,OFFSET NOGO_MSG ; Print no match message
        CALL    PSTRNG
LOCKUP: JMP     LOCKUP
```

```
IS_OK:  MOV     BX,OFFSET PASS_MSG ; Print OK message
        CALL    PSTRNG
        RET

; Input String - Input a string of characters from the
;                keyboard. Store the string in a buffer
;                pointed to by BX.

ISTRNG: XOR     CL,CL          ; Clear byte count
W4KEY:  MOV     DL,GET_BYTE    ;   Set up for CON input
        CALL    IOBYTE         ;    get a byte if one is ready
        JZ      W4KEY
        CMP     AL,BK_SPC      ; Is key backspace
        JZ      BACKUP         ;  if so, back up one time
        CMP     AL,CR          ; Is key carriage return
        JZ      CRET           ;  if so, end input routine
        CMP     CL,MAX_PW_LEN  ; Is the buffer full
        JZ      W4KEY          ;  if so, ignore key
        MOV     [BX],AL        ; else store the char in the buffer
        MOV     DL,'*'
        CALL    IOBYTE         ; Print *
        INC     BX
        INC     CL             ; Bump the pointer and counter
        JMP     SHORT W4KEY    ;  then go back to get the next char

BACKUP: OR      CL,CL        ; If no char yet, return
        JZ      W4KEY
        MOV     DL,BK_SPC    ;  else send backspace to CON
        CALL    IOBYTE
        DEC     CL           ;       subtract 1 from byte count
        DEC     BX           ;       subtract 1 from buffer pointer
        MOV     DL,' '       ;       put a space in the buffer
        MOV     [BX],DL      ;       print space on CON
        CALL    IOBYTE
        MOV     DL,BK_SPC    ;       send backspace to CON
        CALL    IOBYTE
        JMP     SHORT W4KEY  ;        Get next char

CRET:   RET
```

```
; Print String - Send the string pointed to by BX out to
; the CON. String ends with NUL. Calls IOBYTE.

PSTRNG: MOV     DL,[BX]    ; Get a byte from the string
        OR      DL,DL      ;  If it's NUL then we're finished
        JZ      P_OUT
        CALL    IOBYTE     ;   else display the byte
        INC     BX
        JMP     SHORT PSTRNG

; Input/Output Byte - Send the byte in DL out to the CON
; with no ^C check. If DL = FF then AL returns with char
; if typed.

IOBYTE: MOV     AH,DIRIO ; Load AH for direct I/O function
        INT     SYS_CALL ;  Perform the function
P_OUT:  RET

; String storage space

INIT_MSG DB     'PW.SYS  Ver. 1.Ø      By J. Kalis'
         DB     CR,LF,LF,'Enter Password ',Ø
NOGO_MSG DB     CR,LF,LF,'<<< Access Denied >>> ',Ø
PASS_MSG DB     CR,LF,LF,'Access Granted',CR,LF,Ø
LAST_FNC DB     15
SCMD     DB     Ø
BRKSEG   DW     Ø
BRKOFF   DW     Ø

PW_WORD  DB     'password  ' ; password - 1Ø spaces
PW_BUF   DB     '          ' ; password buffer - 1Ø spaces

         DW     64 DUP(Ø)    ; Stack space
STACK    EQU    $            ; Top of local stack
LAST_BYTE =     $
CODE     ENDS
         END    BEGIN
```

*

# Portable Computing from Zenith and FCC



Z-181

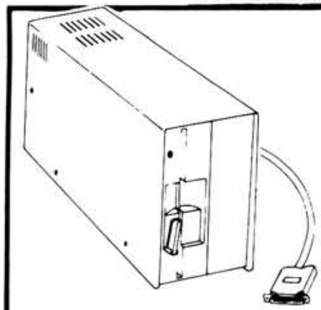
ZENITH | data systems

AUTHORIZED SALES AND SERVICE

If you're a person on the go, you'll never find a more desirable portable computer to take with you than the Z-181! This computer weighs less than 12 pounds, making it easy to carry.

It is a **true** PC-compatible, and runs virtually all PC software. It has a special supertwisted LCD display with 25 lines of 80 columns each, a 60 degree viewing angle and a contrast ratio higher than 8! A built-in illumination tube is provided which lights the display from the rear, for use even in dark rooms.

Standard features are 1 serial port, 1 parallel port, a rechargeable battery with built-in recharger, and an external RGB monitor card. The standard unit comes with 2 built-in 3.5" floppy drives (720k per drive) plus an external floppy expansion port suitable for connecting our FCC external 5" floppy drive.

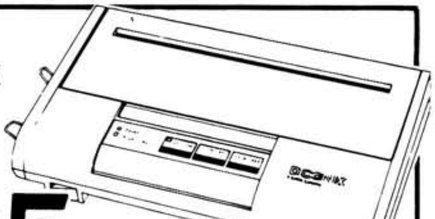


## FCC External 5" Drive

**$250**

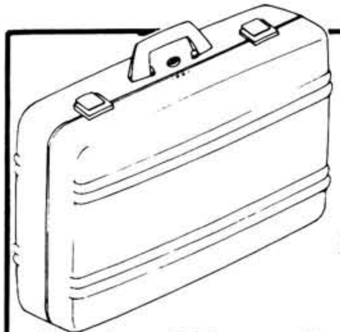
Since the Z-181 computer comes with a 3.5" drive, and 5" drives are more common, First Capitol built an add-on 5" drive. Our drive is enclosed in an external cabinet with 110VAC power supply and a shielded cable to attach to the Z-181 itself! (Does not use a battery).

## Diconyx Printer



**$395**

The Diconyx printer is a **portable** inkjet that runs off its own internal rechargeable battery. The carriage is 9" wide, and able to accommodate both friction and tractor feed paper without buying any options.



## Ruggedized Carrying Case

**$295**

To make portable computing even more convenient, we offer a sturdy carrying case. This hardside case has a foam interior to protect the Z-181, Diconyx printer, and power packs. What an easy way to transport your portable computer system!

**First Capitol Computer**

1106 First Capitol Dr.
St. Charles, MO 63301

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

1-314-724-2336 Technical support and order status

Before you buy . . . . . .

**Call Us First. And Last.**

# BetterBASIC:

# A New Approach To BASIC Programming

***Fred Kent***
*1057 Lake Road*
*Conneaut, OH 44030*

### Dilemma

I am in a dilemma. My trusty Z-100 with UCI upgrade for IBM compatibility and loaded with a full house of 768K RAM runs out of memory during BASIC sort routines of 4000 items and up. In order to take advantage of the Quiksort routines, it is required that the list to be sorted be loaded into RAM memory.

To combat this lack of memory access, I have tried several schemes, such as sorting on only the first four or five bytes of each record and dividing the list into two lists, the first from A to L and the second from M to Z. Following this, each smaller file is separately loaded into a dimension and Quiksorted, followed by appending the second list to the first.

The maddening thing is all that unused memory is sitting out there inaccessible by ordinary BASIC means, unless one is clever enough to be able to machine code a subroutine to take advantage of the extra memory available.

In consultation with son, Donald, an Electrical Engineer with Kodak, we came up with several comparisons of other available languages such as 'C', Pascal, Fortran, etc. and finally came up with a little known product called BetterBASIC which purported to use all available memory in the computer up to several gigabytes. The decision was further influenced by my age (62) and the necessity in my advanced years of learning the syntax of a new language from scratch.

### Solution

BetterBASIC was ordered directly from:

Summit Software Technology Inc.
106 Access Road
Norwood, MA 02062

The advertised price was $195.00. The COD charge upon arrival, however, was only $160.00. This was only the first good thing that happened. Upon first examination of the documentation we found a very complete description of all features, functions, keywords, modules and a list of additional modules prepared at extra cost for any who need them.

Additional modules available from the company:

1. Virtual Memory Manager (To more than 4 gigabytes)
2. C-Language Link Module
3. Decimal Math Module (Preferred for business applications)
4. 8087/80287 Math Module (For math co-processor chips)
5. Btrieve(tm) Module (For use with ISAM File Manager)
6. Runtime System (Produces a true standalone .exe file)

BetterBASIC is a completely modular system in which the programmer can build procedures with any and all known GW-BASIC commands plus about 50 new commands, including direct DOS and BIOS calls. Line numbers are unknown outside of each procedure. Procedures may be called simply by their procedure names or they may be written to require an argument list, if necessary. Functions are similarly called.

Any procedure or function may be made into a module and added to the BetterBASIC configuration file. The result of this is that one can define a function or procedure and add it to his personal BetterBASIC. An example which I have already used and placed into my BetterBASIC is a function to calculate the current Julian date from the date$. Any time I enter 'date' into the computer, either in a program or not, the proper Julian date is printed and furnished to BetterBASIC for subsequent use throughout the program. The procedure 'date' then needs be entered only once. It has now become a keyword for all practical purposes.

At sign-on, BetterBASIC in my computer states that 400 Kbytes is available for programming use. This may be modified if the programmer modifies his configuration file to allot some of the memory for use as extended memory. One specifies the use of extended memory for arrays and strings by including /X at the end of a module line declaration.

All graphic functions of GW and IBM BASIC are supported unchanged in both screen mode 1 and screen mode 2.

### General Concepts Of Modular Programming In BetterBASIC

Fixed and dynamic variables are declared by first use in a procedure or function and will be unknown outside that function unless specific direction by the programmer is given such as:

External: `a$.b(10.10)`, etc.
Integer argument: `c.q`, etc.
Real argument: `e.h`, etc.
String argument: `b$.sum$`, etc.

Declaring variables without a predetermined size sets a dynamic process in which an array or string can grow at runtime as necessary. Variables with declared sizes, such as string a$[10] will set up static variables which do not grow, and therefore, do not eat up available memory as will dynamic variables. This is the preferred programming method for BetterBASIC when at all possible.

When at all feasible, the main program is kept short with the bulk of processing done by various modules. If a module is of frequent use, it may be stored on disk and added to any new program by the use of load "modulename". BetterBASIC then advises that the module will be blended into your existing program and asks if you desire to do so or not.

Of particular interest is a feature lacking in all 8088 BASICs, that being when a program is saved to disk and another program of the same name is already on disk, BetterBASIC will ask if you wish to overwrite the existing file. I cannot count the times I have wiped out a saved program by forgetting what names were already on my disk and using the same name again. This is one feature I would like to see added to Microsoft-produced BASICs. A similar safeguard was available in my first Heath computer (H-11) which used Digital Equipment Company programming and I have sorely missed this simple safeguard in MSDOS style BASICs.

BetterBASIC also features a 'structure' mode in which a structure can be predefined. Some special techniques are used to access a structure, such as structure pointers, etc. This technique offers unique ways to handle IO to and from disk, and renders unnecessary, the fielding statement normally used with random files. Strings may be defined up to 32567 bytes in length. Fields within a structure, as well as record length, array size and length are all defined prior to writing code for the procedure. Defining a pointer to the structure name within the structure allows a very sophisticated method of linking various structures and arrays together. Very large databases can be built and linked up.

BetterBASIC does not care for the 'goto' statement, but will allow it within a procedure or function. You cannot jump outside the current procedure or function with a 'goto'. The processing words of BetterBASIC available are:

```
for . . . next
do
do if
do x times
do until
end do
repeat
repeat if
repeat until
on . . . goto
on . . . gosub
goto (within procedure only)
```

The trace mode (tron) can be rapid, slow, stepwise or to hard copy. The trace is displayed only on line 25 and does not otherwise clutter the screen.

The only disappointment I have so far noticed is that errors are only given as numbers and I must scramble through the books to determine what number stands for what error. I plan to remedy that by adding a module to print the error messages in ASCII as they occur and making it a part of BetterBASIC.

All existing GW and IBM tokenized BASIC files can be converted by BetterBASIC to BetterBASIC type files by entering Convert "oldfile" to "newfile". Program source codes can be saved either in compiled form or ASCII form. As usual, the short form loads much faster.

Probably the best news of all is the fact that BetterBASIC is not just an interpreter. BetterBASIC will always make a dummy run through a newly loaded program to initialize all its offsets to various lines, procedures, functions and structures and produces, therefore, an already compiled program. While BetterBASIC seems faster than other BASICs tested, there seems little doubt that a compiler (available through Summit Software, $250.00), will probably triple the execution speed. *(Actually the interpreted code is compiled and the 'compiler' makes it a runable module. The interpreted code and 'compiled' code executes at the same speed. -ed.)*

A Save "programname" always produces a compiled program. Listing is available as needed once the program is loaded. All values are available at any time for debugging purposes and do not disappear once the program has run through. If values are known only within a module, you just enter Edit "modulename" check for values and bugs inside that module.

Summit Software offers a 90 day warranty on this product and furnishes a hotline number for technical questions and support. Note that BetterBASIC is not copy protected, however, the usual copyright and usage agreements must be observed.

### Summary

BetterBASIC is a remarkable upgrade from its roots (IBM BASICA). I highly recommend its use as a primary programming language. Containing all the features of BASICA and GWBASIC, it need be no more difficult to learn than BASICA or GWBASIC. It vastly upgrades the uses to which a computer with extra memory may be put. With the addition of large capacity memory cards and the virtual or extended memory module, it can well render the desktop computer capable of rivaling the power of a main frame system.

Note here that BetterBASIC requires a minimum 250 K RAM to utilize all of its power, although it can be scaled down somewhat by reducing some of the unneeded modules. Certainly, to make full use of it one ought to have a fully populated memory bank.

A serious critique should mention drawbacks, as well as good features of the product under review. I found one feature that just does not work in my copy of BetterBASIC and have returned the disk to the company with my findings. I have no doubt that it is a flaw which the company will correct. The flaw is that the 'put' statement to disk simply crashes every time it comes up. *(The 'put' function works properly in the latest version -ed.)*

Files may be opened in the usual manner for random access and a 'get # x,x' will produce the proper fields. Processing these fields presented no problems until the time to write back to disk with the 'put # x,x' statement. Up pops an error code number matching nothing in the BetterBASIC document books. No data is destroyed or altered. It just simply fails to go past a 'put' statement.

I have fiddled with this problem at length, thinking it may be my fault or I did not read enough from the books. There is an alternative method of IO available by forming fixed length structures with appropriate pointers and using a routine called write record # channel, fileposition, ptr.data. This technique has no problem in writing to disk, however, in checking dumps of data files produced with this command, I found that the fields are placed to disk in reverse order (i.e. the string "field" becomes "dleif". A companion command 'read record' will bring the fields back in correct order. This technique, if used, would require that all my existing datafiles be transposed. It seems much simpler, therefore, to report to Summit that the 'put' command is faulty. Most probably, one byte is weak or faulty on the master disk as the demo disk which I first used had no problem with normal 'put' and 'get' routines.

One other very pleasant feature of BetterBASIC is worthy of mention. Up to five windows may be defined. Each defined window may be boxed in with a double line box with the command 'frame window #x, and finally 'header x' "string" will produce a header line for that window which is centered. The active window is chosen with a 'select #x' command. All screen output will remain within the window boundaries. 'clw' will clear only the active window and will not destroy the box frame or the header line. 'cls' and 'cld' are also available meaning clear the entire screen or clear the entire display. Use of this feature does not require graphics mode as does 'window' and 'view' in GWBASIC or IBM BASIC. I have been able to produce some unique screen presentations for my business customers popping up various routines in selected windows. The sales people especially appreciate having the bottom line 'price salestax price+salestax' neatly displayed in its own little box. Separating bottom line charges from the mass of preceding data entered to produce a price has already reduced the number of clerical errors.

'xmem' command will produce the amount of extra memory in use and the amount of extra memory remaining unused in paragraphs which I believe are 8 byte increments.

**'mem' Command Produces On Screen The Following Information**

| **Current Memory Usage:** | **(Bytes)** |
|---|---|
| Descriptors and Constants: | x |
| Current program space: | x |
| Current symbol space: | x |
| Current source space: | x |
| Total program space: | x |
| Static variable space: | x |
| Dynamic variable space: | x |
| Total memory left: | xxxxxx |
| xmem: paragraphs used: | xxxxx |
| xmem: paragraphs left: | xxxxx |

**Listing 1**

```
source
precision= 7
autodef=on
option base=0
erl=on
errormode=global
resume=statement
formode=gw
printmode=gw
scope=on
procs=1

procedure: date
end procedure


procedure: date
   real: m,x,k,z
      10 data 0,31,59,90,120,151,181,212,243,273,304,334
      20 m=val(left$(date$,2))
      30 for x=1 to m
      40   read k
      50 next
      60 z=val(mid$(date$,4,2))+k
      70 if m>2 and val(right$(date$,1)) mod 4=0 then z=z+1
      80 print z
end procedure

'main program:


endfile
```

To illustrate the techniques of programming with BetterBASIC I reproduced several small procedures.

The first 10 listed lines show the current status of BetterBASIC and are always printed in any list all request. Line 12 declares that the program contain 1 procedure. Line 13 declares an integer named 'dat' (2 bytes).

In the procedure listing header, 'dat' is declared to be external to that procedure thereby making its value available to the main program. The next line declares all variables are in the procedure as integer values (2 bytes each).

The data line contains values of all months commencing with January in terms of elapsed days. Line 20 determines the current month from the current date. Lines 30 to 50 will loop until x%=m%. k% then will be the number of days in past months. Line 60 obtains the current day of the month and adds it to the elapsed days, placing the result in 'dat'. 'Dat' now contains the exact Julian date for the current year. Line 70 corrects the 'dat' for leap year which is any year evenly divisible by 4. Line 80 outputs 'dat' to the screen, and procedure ends note that since it was declared external to procedure date, the value for 'dat' is now available at any time to the main program.

**Listing 2**

```
source
precision= 7
autodef=on
option base=0
erl=on
errormode=global
resume=statement
formode=gw
printmode=gw
scope=off
procs=1
real: x,sum

real function: double
   real arg: sum
end function

real function: double
      10 result=sum*2
end function

'main program:


   10 input "enter a number: ";sum
   20 print double(sum)

endfile
```

This is a complete, though short, BetterBASIC program utilizing a user defined function named 'double'. Most functions require an argument to accompany the call. In function, 'double' declaration notes that the function is designed to require an argument 'value' passed to it with the function call. The function then calculates 2 times the value of 'value' and ends with the result=B statement.

The main program, line 10, asks for an input 'value'. Line 20 calls the function 'double' and passes the required argument (value) with the call. Subsequent calls can be made at any time as long as a proper number value is defined in variable 'value' by the method used in line 20.

I have just finished reading "MegaBASIC" in the December '86 issue of REMark by Richard Tilden. Mr. Tilden's chief criticism of MegaBASIC seemed to be its rather high price. Although I am not nearly as versed as Mr. Tilden, I have run speed tests similar to those described by him. I noted the following comparisons using the Quiksort routine described in REMark about two years ago. Quiksort is said to be faster than a Heapsort.

Data used was a random file of 4196 names with record length of 20 bytes, total filesize=83920.

| File Used | Time<br>1K Records | Time<br>4196 Rec. |
|---|---|---|
| GW-BASIC | 155 seconds | |
| Z-BASIC | 227 | |
| Mega(r) | 53 | |
| Mega | 71 | |
| BB | 33 | 161 |
| Bascom | 7 | 17 |

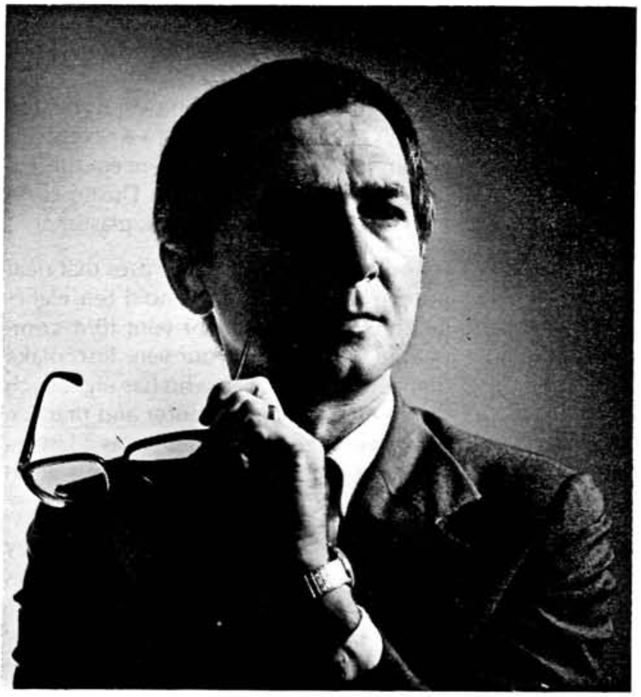
**Note:** Files of this size are not normally capable of sort by GWBASIC, ZBASIC or Bascom. Having need for something that would handle this size is what brought me to look into BetterBASIC or BB. I handle files of this size by cheating and inputting into memory only the first four bytes of each record. But what happens when you have five people named Smith?

Using BetterBASIC, I loaded the entire 4196 by 20 byte records into one dimension, their starting file position in a second and used two 2000 by two dimensions for intermediate sorting as required by Quiksort. I had about 200 Kbytes of unused memory remaining.

By contrast with MegaBASIC, BetterBASIC is apparently a combination interpreter compiler rolled into one. It costs less than half of the MegaBASIC price. All GWBASIC statements and BASICA statements are compatible, unchanged. It seems to be faster than any other BASIC I have ever tried, including 'MegaBASIC' which I have not tried but only read about. The ability to declare fixed sizes of strings greatly reduces the need for garbage collection as fixed data does not grow at runtime.

Any more detailed discussion by me would have to await some months of programming experience. I have had BetterBASIC for only two weeks.

*



# Mainstream Computing

*Joseph Katz*
*103 South Edisto Avenue*
*Columbia, SC 29205*

I've just recovered from one of those rare explosive laughing fits that make my abdomen hurt. You know the kind. What ignited it was a sudden shift in point-of-view from my own to what I think is my friend Bert Dillon's. Bert, Janet, and I take lunch together most Tuesdays and Thursdays every semester and talk about nearly everything in the world. We've done it for years and will continue to do so despite Bert's disconcerting habit of leaning forward across the table from time to time and saying earnestly, "You're really quite mad, you know." He's wrong, of course. Janet is the sanest person I know. But for one brief instant a few moments ago I glimpsed things his way and had the wild thought that he might mean me. I burst out laughing uncontrollably.

The story concerns a color television set I use as a computer monitor and a computer monitor . . . Well, to put the situation in an instructive light, here's what you can do with a surplus composite color monitor. Just don't have lunch with our friend Bert afterwards.

A few months ago my father, Louis, decided I should have a nice, new color television set to replace the eighteen-year-old monochrome I kept in my office. Dad always has had a thing about electronic gadgets. Soon after the end of World War II, for example, he bought the first television set in our neighborhood and we all sat around watching test patterns until 5:30 in the afternoon when programming suddenly declined.

"It's also a computer monitor," he said in the phone call telling me he had just shipped my birthday present, "and it's small enough so it can fit on your work counter." What arrived was a Zenith System 3 Receiver/Monitor. One thing I really didn't need right then was another computer monitor, but since Dad has an instinct for snazzy gadgets I connected this dual-personality television set to my *Z-241* for a workout. It's a decent composite color monitor for the times I need to use the CGA (Color Graphics Adaptor) standard in color on my '241, really nice as a monitor for the VCR I'd been using to experiment with computer graphics, and super as a television set. I would be able to replace two boxes on the counter with one box. Dad had a good idea.

So I retired my old Admiral, shifted the Zenith System 3 into its place, and was left with the problem of what to do with the *ZVM-134* (a "real" color computer monitor) I had been using when I needed to see the computer display in color instead of high-resolution TTL monochrome — which as you recall I have been using for text work to save my eyes from further degradation. Put that problem on hold for a moment while I introduce another complication into this saga.

A few minutes after I saw the marvellous printing I could do with PostScript on my Apple LaserWriter, I began to yearn for a monitor that would at least approximately display it. Addison-Wesley had sent me ArborText's *TeX Preview/PC* package for seeing *TeX* output on a monitor, but the catch is you need a high-resolution graphics display for it. The CGA board that is standard in most Heath/Zenith compatibles offers a good compromise for general-purpose computing and it saves you from having to buy a video board (the way you must with other computer brands). It's just not a high-resolution graphics board. *TeX Preview/PC* (and most other good graphics programs) requires one like the Hercules Monochrome Graphics Card, Tecmar Graphics Master, or some kind of EGA ("Enhanced Graphics Adaptor," introduced by IBM in 1984 to supercede the CGA) board.

Every time I looked at the marvellous things ArborText's *DVIPS* PostScript driver for *TeX* (which Addison-Wesley also sent me) was producing on my Apple *LaserWriter* — simply the best-looking printing I've ever seen from a microcomputer system — I glanced at the *TeX Preview/PC* box with yearning. Janet says I can resist anything but temptation. That's why, she says, vendors with nifty products send me their wares. The word is out that I enjoy spreading the word about stuff I find valuable in my work.

I really resent her implication: I am notoriously a stolid person of impregnable demeanor. Everyone knows that. "Harumph," as middle-aged gentlemen used to grunt in Victorian novels.

At any rate, I of course had to have a NEC *MultiSync* monitor. It supports so many different display standards, including the EGA, that I could get both the *ZVM-1230* CGA monitor and the *ZVM-1240* high resolution TTL monochrome monitor off my counter. You should understand that my goal was to make Janet happy, so she could walk into my office without being stunned by a battery of three monitors and a television set often running at the same time. She is, if you follow my reasoning, a visually-oriented person whose sensibilities could be permanently affected by so confusing an array. I love my wife and — I know you really understand — will do anything for her. Now all I have on the counter is the Zenith System 3 Dad sent me and the NEC *MultiSync*.

But now on what had been the only free floor space in my office I had an old Admiral black-and-white television set, a *ZVM-1240* high-resolution TTL monochrome monitor (with the requisite monochrome adaptor board), a *ZVM-1230* monochrome CGA monitor, and a *ZVM-134* color CGA monitor. It's a very small office and the horde of plastic boxes with picture tubes had taken over. I was getting nervous. My theory of the universe includes a recognition of its cyclical nature. This cycle evidently had begun The Year of the Monitor, which rolls around every so often. That meant I surely would be sent another monitor, then another and another until I would not be able to get into my office to turn on any of them. Janet already had two monitors of her own in her small studio, so she simply snickered at my suggestions that maybe she could use one or two more.

"Aha!" I exclaimed, "I have the perfect solution! I shall turn the *ZVM-134* monitor into a color television set and move it into the dining room." This idea really was quite sensible. After all, what computer company also produces great television sets — one of which I had just received? And, after all, what is a *ZVM-134* but (as Tom Jorgenson says) an excellent television set without a tuner? And, finally, did not Jim Buszkiewicz tell me that the market was flooded with used *ZVM-134*s so no rational person would buy mine?

For some reason she never explained, Janet prefers that I not go to a computer or electronics store without being escorted by some other member of the family. Since she could not be found at the moment for departure (later she remarked that she was in her studio, where she said she would be), I hustled Matthew into the car and we searched Columbia for a video tuner. We found a Sony *ACU-270* — with all sorts of great features for Sony's breathtakingly-expensive component television and stereo system — at a price I couldn't turn down. So I didn't turn it down. I put an indoor UHF/VHF antenna on the tuner, ran a cable from the tuner to the *ZVM-134*, and turned the thing on.

Gloriosky! It is an absolutely incredible-looking component television system. It's a studio-quality display, albeit smallish. When I connected the system to feed our four-foot-high Tannoy speakers, it was also an absolutely incredibly-sounding component television system. It's better than studio-quality sound. Janet was impressed. She is not easily impressed.

A few minutes after I got the thing working, one of Janet's assistants came to call. "Oooo!," she oooed, "that must be one of those new television systems I've heard about," she said. Then, after a thoughtful pause, "But it looks just like an old computer monitor. Interesting." I put on my most humble face, smiled deprecatingly, shuffled my feet a few times, and actually said "Aw, shucks." Matthew fled to his room and Janet gave me a filthy look.

I don't care. I have a superb component television system in my dining room, and if I can get a couple more *ZVM-134*s or — even better — some *ZVM-135*s at close-out prices, I'll buy some more tuners and blanket our house with bargain-priced component color television systems. If you didn't know they were computer monitors, you'd think they looked just like computer monitors.

So that's the story of how I come to have a color television set employed occasionally as a color computer monitor in my office and a color computer monitor employed exclusively as a color television set in my dining room. For a brief moment, I saw the situation as Bert likely would when I explain things to him, and I burst out laughing. Our friend, Bert Dillon, has an interesting sense of humor.

Nevertheless, if you own one of Zenith's CGA monitors and plan on moving to another graphics standard, think "component television" and consider shopping for a tuner. Now I'm thinking about writing an article on how to convert a computer into a coffee table. Some day you'll be reading my stuff in *Good Housekeeping*.

Until then, I'm sure you'll welcome a couple of Dr. Katz's handy household hints.

### Dr. Katz's Handy Household Hint: Number 1

If you use an indoor antenna when you turn your CGA color monitor into a color television set, you'll want to conceal the antenna. As the fellow said to Dustin Hoffman in *The Graduate*, "Think plastics."

I used one of those plastic cases that neat people and I employ to hold ten eight-inch floppy diskettes. (If your IBM-compatible computer is your very first, make friends with someone who has eight-inch drives on a CP/M computer and practice your sweet talking. Of course, as a last resort, you can always buy a case at a good computer store.)

Remove and discard the tilting doohickey that holds the diskettes inside the case. Then drill a hole on the back of the case — the back, not the spine — about the size of the cable that connects the antenna to the tuner. (If you don't have a drill, hold the box next to this column and I'll try to bore it for you.) Pull a little of that cable through that hole and make your connections. The last stage is to sort of cram the antenna stuff into the plastic box and snap the lid closed. You now have a decorator indoor television antenna.

Place it strategically near the tuner, or monitor, or wherever you get good reception, and invite your friends to see your new rig. I can hear them now. "Hey. He has a plastic diskette box next to that television set that looks like a computer monitor. Wow."

### Dr. Katz's Handy Household Hint: Number 2

I use one of Logitech's wonderful *Logimouse C7* mice. I like it. It's thoroughly compatible with the Microsoft mouse and is much nicer to use, I think. But it kept getting lost.

In fact one of the nuisances that taints life for many people is the mouse when it is not in use. The electronic rodent hides, falls on the floor, or otherwise behaves like a rat. It becomes clutter.

I have solved the problem.

From a fabric store or other source, buy a strip of self-adhesive Velcro hook-and-loop fastener. Make sure you get a "set"; not only the Velcro hook-and-loop part, but also the felt part that the hook-and-loop part hooks into. Radio Shack sells four sets in a package (Catalog Number 64-2345) for $1.89.

Find a place on the *bottom* of the mouse where you can stick the *felt* strip without interfering with the mouse ball, or sensor, or whatever it is that allows your mouse to make tracks. Then trim the set so it's the width of the mouse, stick the *felt* strip to the place you've located on the bottom of the mouse, and stick the *hook-and-loop* strip to some place near your computer where you can conveniently hang the mouse. Then hang the little rascal. Mine dangles conveniently from the right side of my laminate covered work counter.

As a bonus, the felt strip ought to make the mouse easier to move and track more accurately. It serves as a guide and glide — at least on my Logitech *Logimouse C7* mouse.

Don't forget where you read this Handy Household Hint. I want the credit for what surely is a major contribution to making contemporary life sweeter.

Of course if it doesn't work, don't blame me.

### *The Final Word II* Word Processor

I'm really glad I followed separate suggestions by Mark Huth and Jim Hardin. You agree, I know, that when a really knowledgeable computer user recommends a product it's worth the time to take a look. I'm glad I did.

Mark of the Unicorn's *The Final Word II* is an exciting word processing program with useful features I haven't seen in any other single program. Some of those features are only in *The Final Word II*, so far as I know. Unfortunately, Mark of the Unicorn doesn't advertise much in magazines: "Too expensive," a spokesman told me, "so we rely mostly on word-of-mouth advertising." My impression is that Mark of the Unicorn is a low key, low profile company. But what a word processing program is *The Final Word II*. If you're in the market for a good, solid, general word processing program with a special feature you can't find elsewhere, this one might well have it.

Unfortunately, I can't pretend to know all those features. One reason is that *The Final Word II* package has so many, tucked modestly away on the distribution disk and manual. Another reason is that most of those features are the kind that don't make headlines. In fact, they're the kind that I, at least, don't recognize as anything special until I need them. For example, when I needed to find as many programs of every kind that output to a PostScript printer, Mark and Jim independently — from different parts of the country — said I had to look at *The Final Word II*.

It has a superb PostScript driver — much more sophisticated than even that in Version 3.0 of Microsoft *Word*, which until now I thought the best word processing program for use with a PostScript printer. I'm not detracting from *Word* in the least when I say that you can do much more and much more easily in PostScript with *The Final Word II* than you can with Microsoft *Word*. The two programs are directed at different audiences anyway.

For example, once I began working with *The Final Word II*, I stumbled across a few other features I needed but didn't know I needed. The program can be installed for the *Z-100*, as well as for IBM-compatibles. All you do is specify the *Z-100* driver when you install *The Final Word II* for your terminal.

As another example, it has an Intergraphics driver. Intergraphics, Inc., is a typesetting service company in Alexandria, Virginia, that specializes in real typesetting — not "near typesetting" or "desktop publishing" — from microcomputer output. We have had an Intergraphics account for several years. So, because of Janet, has the University. She takes an ordinary ASCII or *WordStar* file, codes it according to Intergraphics' requirements, dials up one of its Merganthaler 202 typesetters from her modem on a toll-free line, and dumps the coded file to the remote typesetter. The next day or day after (depending upon my beloved's schedule), what comes back is type in galleys ready to be pasted into camera-ready copy. We recommend Intergraphics highly. The necessary coding is tedious, however, so I was delighted to see *The Final Word II* Intergraphics driver. It's configured for the Century typeface (designed for use by that great Nineteenth-Century magazine, *The Century*) but — as with all the drivers I've examined for *The Final Word II* — you get source code introduced with brief, but knowledgeable, notes on how to modify it. You can modify this driver for any of the type in use at Intergraphics.

The concept of using terminal and printer drivers for I/O isn't new, of course, but I haven't seen many programs that apply it as well as *The Final Word II*. If there's presently no driver for your own eccentric — excuse me, I mean "distinctive" — printer or terminal, you should be able to work one out for yourself, or maybe try to persuade the amenable support staff at Mark of the Unicorn to do it for you.

Oh yes, the program itself. I got so carried away with its intelligent use of drivers that I almost forgot to say anything about using *The Final Word II*. I had a sense of *deja vu* while I was exploring the program. It reminded me of *Perfect Writer*, a program I detested on my CP/M machines, except that it couldn't be *Perfect Writer* because I found *The Final Word II* fast, likable, easy to learn, and easy to use. None of that was true of *Perfect Writer*. The feeling was so persistent, though, that I telephoned Mark of the Unicorn to put the question direct. It turns out that the same people conceived both programs. If you know *Perfect Writer*, therefore, imagine its assets with none of its liabilities and you'll wind up somewhere near *The Final Word II*: multiple windows, multiple columns, infinitely-flexible print formatting, but *fast*.

You have to understand power before you can fully appreciate *The Final Word II*. It's most definitely not a WYSIWYG ("What You See is What You Get") program: you insert print formatting codes that are on the order of "@i{Parmesan}" to print italic *Parmesan* and "@b{Limburger}" for bold **Limburger**, and so on. It's, therefore, not a WYSISYG program ("What You See is All You Get") either: those print formatting codes, and others you can design for your own applications and equipment and prejudices, let you print text any way you want. There are pop-up menus that will insert the formatting codes for you, or you can turn them off and work with a nearly blank sheet of paper on the screen. One sweet touch if you (like me) own an EGA monitor is a utility that lets you work with forty-three lines — legible, stable, non-flickering lines — on the screen at a time. You can set an EGA monitor to do that with other word processing programs, but with *The Final Word II* you get the program thrown in. You get a spelling checker too, but I haven't tried it yet. There's also an optional mail-merge program, which I haven't seen.

Perhaps the best way I can summarize my feelings about *The Final Word II* is to say it's a serious program for serious writers. It's now most definitely in my arsenal of word processing programs, in the select company on which I rely.

### *A-Plus —*
### Integrated Software For School Kids

Speaking of word processing, if you have a child in high school, junior high, or even the advanced grades of elementary school (assuming your bairn has the wherewithal to float a computer or a parent sufficiently enlightened to provide it), you ought to know about *A-Plus* from Savant Corporation. It's billed as "The Home Work Solution" and I think it may be just that. *A-Plus* is the first integrated software package that I think appropriate to its audience.

You remember integrated software as the stuff that does so much so badly and so slowly. A few years ago everyone was producing or promising integrated software would revolutionize the way we use computers. Run one program and you got word processing, apple slicing, car polishing, behavior modification, and anything else you might think you want. Integrated software of that kind was the toaster oven of computerdom. Well, none of the parts in such programs ever turned out to be nearly as useful as their standalone competitors. Inevitably, we returned to a specialized program like *WordStar* for word processing and added on to it other powerful programs that would do well what we really needed done well.

Savant Corporation, however, seems to have put software integration to appropriate use for an appropriate audience. It's primarily a word processing program with integrated graphics, or maybe the other way 'round, plus a clock, calendar, calculator, and utilities. Everything works together nicely and reliably in an interface that uses the metaphor of binders and folders. There's a thorough and well-written manual, with sufficient detail and illustrations that strike me as being right on target for its audience. Those are young people who quite rightly resent anyone who tries to talk down to them. This manual doesn't. If you have a kid who knows a little about computers and can read plain English, give that young person the *A-Plus* diskette and manual, and go listen to Mozart.

*A-Plus* is priced at a respectable $89.95 and the package is respectable enough to justify it. It is not a toy or placebo software, nor is it the kind of software cobbled together in the spare time of a programming parent who wants to do "something educational" for the kids. It is a high quality software package, professionally designed and produced. There's a demo disk available if you want to try before you buy. It's a nice demo that truly represents the real thing because it is the real thing, except it won't let you print or save files.

No, I won't be using *A-Plus* myself. It's not for me or people like me, or like you either: its strength is the way it meets the needs of the homework set. I have no hesitation at all about recommending it for them. I consider *A-Plus* a real find. I'd love to see more well-defined and well-executed programs like it, and fewer toaster ovens.

### Happy Birthday, *Sextant*

The Heath/Zenith community — and Heath Company and Zenith Data Systems — is lucky to have two such useful, interesting, and quite different magazines as *REMark* and *Sextant*. I subscribe to both. (Yes, since I write for both magazines I do get free copies. But I also have paid for a subscription to each, and I began subscribing long before I was asked to write for either.) *Sextant* has just published its Fifth Anniversary issue. Happy Birthday Charlie, John, and the rest of the crew.

### The Photo At The Head Of The Column

It's the work of Jim Huff, a talented photographer at the University of South Carolina's Instructional Services Center.

### Products Mentioned

*MicroTeX* (Version 1.5). $295.00
*TeX Preview/PC* $250.00
*DVIPS*. $300.00
Addison-Wesley Publishing Company, Inc.
Educational Media Systems Division
Reading, MA 01867
(617) 944-6795

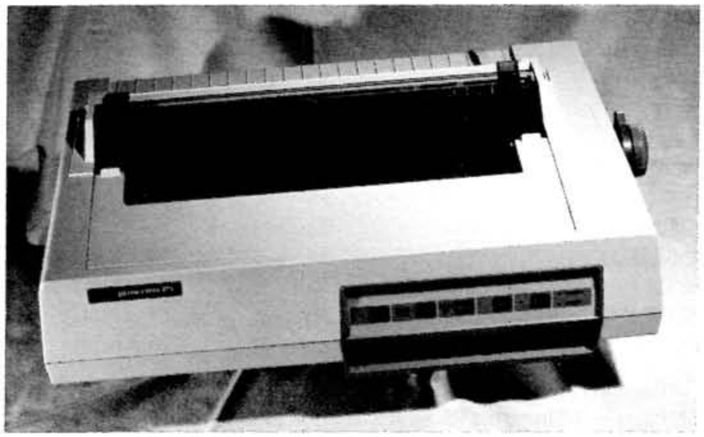
*Logimouse C7*. $99.00
Logitech
805 Veterans Boulevard
Redwood City, CA 94063
(415) 365-9852

*The Final Word II*. $
Mark of the Unicorn, Inc.
222 Third Street
Cambridge, MA 02142
(617) 489-5078
Intergraphics, Inc.
106-A South Columbus Street
Alexandria, VA 22314
(703) 683-9414

*A-Plus* (Version 1.0). $ 89.95
Savant Corporation
P.O. Box 1077
Waltham, MA 02254
(617) 891-0638

*

# 24-Pin Printing



## Road Tests Of The Toshiba P351 And NEC P5 Pinwriter

*Donald S. Alspach*
*2101 Dawnlight Court*
*Anchorage, Alaska 99501*

### Looking Through The Want Ads

A rare opportunity for a non-computer professional product reviewer came my way while I was searching for a new printer to speed up printing operations from an H-120. The 120 is used in a word processing and mailing list production business. The current setup includes the H-120 with two 360K disk drives, 768K of memory, an old motherboard and no speed packages. I am using MS-DOS Version 3.0 as the operating system. On the other end of the ribbon cable is a Diablo 620 printer. The 620 is daisywheel letter quality printer with a tractor feed which has a top speed of 20 cps. The speed of the printer is not really important until you begin to print a mailing list containing some 2000 names. There is nothing like listening to a printer pound away for two hours even inside a sound box. However, it does turn out nice looking labels and documents for the clients.

Now you know the reason for the quest for speed. I dove into the computer mags looking for a dot matrix printer that would give me the speed I thought I needed, and as an added bonus the ability to do dot matrix graphics. The printer I zeroed in on was the Toshiba P351. It had the speed of 288 cps in draft mode, graphics, and an NLQ mode at 100 cps. It also has a list price of $1699 to $1749 depending on a parallel or serial version, plus another $199 for the tractor feed. A bit steep for the budget. The dream of a new printer was put on hold.

Then it happened. An ad appeared in the local want ads asking for bids on a P351 and an NEC P560. Curious, I called the number and arranged to see the P351. I discovered both printers were brand new in the box and the sale was the result of a claim from a truck accident. The insurance company owned the printers and wished to recover as much of the cost as possible. The printers were offered "as is" although the insurance company did state the printers did operate. I bid on the P351 because I did not know anything about the NEC. As it turned out I was either the high bidder or the only bidder because my offer on the P351 was accepted. Not only that, but they wanted me to take the NEC for the same bid. I could not resist. I took a chance and bought the NEC as well. Now I have two near equal printers at less than the cost of just one and the rare opportunity to take them each for a test drive at home.

### The Road Test

I want to approach the review of these two printers from the standpoint of providing all the data necessary to make an informed decision before purchasing either one. At least when I go looking for a piece of hardware like this I want to know as much about it as I can so I can compare it with others to see which best fits my needs. I did the same thing when I chose the H-120. The NEC is a P5 Pinwriter model P560 which means it is a parallel only version with a retail value of $1440. The P351 is a parallel/serial version with a bottom feed feature. So the two are not exactly equal. My intent was to use the dot matrix printer as a parallel printer and maintain the 620 as the serial printer. That way I can jump between the two depending upon the clients requirements without cable changes or using an expensive switching box. For instance I can make up a WordStar disk for each printer. Although that may not be necessary for disks other than WordStar now that Jeff Kalis has developed a Configuration Utility for the H-100s that will allow the jumping between printers without changing disks (HUG April 1986, Page 45).

Each of the printers has a remarkable list of features. For the sake of comparison, a listing of the features according to manufacturer manuals is the best way to present this information.

| | P560 | P351 |
|---|---|---|
| Standard equipment | | |
| Ribbon | yes | yes |
| Power cord | yes | yes |
| Cut sheet guide | yes | yes |

| | | |
|---|---|---|
| .iensions | | |
| Width | 22.8″ | 22″ |
| Depth | 14.7″ | 15″ |
| Height | 6.1″ | 5.9″ |
| Weight | 35.2 lbs | 33 lbs |
| Print buffer | 8K | 4K |
| Print speeds | | |
| High speed draft 12 cpi | 290 cps | 288 cps |
| Normal draft 10 cpi | 240 cps | 240 cps |
| Letter quality 10 cpi | 100 cps | 83 cps |
| Print head | 24 pin | 24 pin |
| Ribbon | | |
| Type cartridge | fabric | fabric |
| Life in characters | 3.1 million | 1.6 million |
| Local cost per ribbon | $16.00 | $9.00 |
| Paper | | |
| Minimum width | 5″ | 4″ |
| Maximum width | 16″ | 15″ |
| Copy capacity original plus | 3 copies | 4 copies |
| Front panel controls | | |
| Power | LED | LED |
| Status | LED Alert | LED Alarm |
| Paper End | Error LED | LED Paper end |
| Offline | Select | Select/Desel. |
| Line feed | Feed | Paper feed |
| Form feed | Feed hold | Top of page |
| Draft/Letter Quality | Mode | |
| Special feature 48 dBA/53 dBA | Quiet | |
| CPI's built in | | |
| 10 draft | yes | yes |
| 10 letter quality Courier | yes | yes |
| 12 draft | yes | yes |
| 12 letter quality Courier | yes | no |
| 12 letter quality Prestige Elite | no | yes |
| 16.7 draft | no | yes |
| 17 draft | yes | no |
| 20 draft | yes | no |
| Proportional space | yes | yes |
| Optional fonts | yes | yes |
| Location | front | back |
| Number of cartridges | 2 | 1 |
| Optional equipment | | |
| Tractor feed bi-directional | yes | yes |
| Single bin sheet feeder | yes | yes |
| Dual bin sheet feeder | yes | no |
| Expandable print buffer | yes | no |

### Left Face, Right Face, Parallel Interface

Each printer uses a Centronics female connector on the printer. I chose a ribbon cable with a Centronics male connector for the printer and a male RS232C for the H-120 parallel port J3. I prefer the ribbon cable because it is much easier to route neatly around the computer room. Another plus is solderless connectors. I find the solderless connector eliminates the problem of bridges and cold solder connections that sometimes takes forever to find.

The key to any cable connection is reasonably good instructions from the manufacturer on the pin layout for his printer so they can be matched to the computer. NEC did a fine job on outlining the interface pins description. Toshiba did well on the serial layout, but failed to mention the parallel layout. At any rate the P351 did work just by matching the H-120 pins to the P351 pins, and it worked fine with the P560 pin arrangement. The real arrangement as obtained from Toshiba later is as shown.

| PIN | H120* | P560** | P351*** |
|---|---|---|---|
| STROBE | 1 | 1 | 1 |
| PDATA1 | 2 | 2 | 2 |
| PDATA2 | 3 | 3 | 3 |
| PDATA3 | 4 | 4 | 4 |
| PDATA4 | 5 | 5 | 5 |
| PDATA5 | 6 | 6 | 6 |
| PDATA6 | 7 | 7 | 7 |
| PDATA7 | 8 | 8 | 8 |
| PDATA8 | 9 | 9 | 9 |
| ACKNLG | 10 | 10 | 10 |
| BUSY | 11 | 11 | 11 |
| GND | 12 | 17 | 17 |
| UNUSED | 13 | 15 | 14 |
| UNUSED | 14 | 16 | 15 |
| ERROR | 15 | 32 | 32 |
| INIT | 16 | 31 | 31 |
| GROUND | 17-25 | 19-30 | 19-30 |
| AUTO FEED XT | | 14 | |
| +5V | | 15 | 18 |
| SIGNAL GND | | 33 | |
| UNUSED | | 34 | 34,35,36 |
| +5V | | 35 | |
| SLCT IN | | 36 | 13 |
| 0V | | | 16,33 |

*Page 2.138 Z-100 Series Technical Manual Hardware

**Page 1-1 Pinwriter P5 Series Technical Reference Guide

***Page 4-1 Section 4. Interface 4.1 Centronics Parallel Interface. Toshiba Reference Unknown

### You Know What I Like

Now that I have played with each of the printers, I have found some features I enjoy in each and others that are annoying. Starting with the good things on the P560. I like being able to change the fonts from the front panel. The selection is displayed with an LED so that at a glance you know where you are. The P351 requires the change to be made with DIP switches on the back of the printer. Therefore, you must either know where you are or look at the DIP switches or do a print test to find out. To change the fonts on the P351 you must trip the DIP switches to the right position, turn off the printer and then on again to reset the printer. The DIP switches are located on the back of the printer, check the manual, get up and set the switches, turn off the printer, turn on the printer and return to the computer. Of course the fonts can be changed through software which is easy in BASIC, but with my present knowledge it is difficult in WordStar and Multiplan.

Likewise, the changing of font cartridges is out in front with the NEC. A glance tells where you are. The cost of these cartridges is rather expensive so I doubt that I would ever invest in them for either printer. Another out in front option is the addition print buffer. It is possible with the P560 to add to the print buffer. The standard buffer is 8K. Print buffer cartridges are available in 16K increments that can increase the buffer to 40K. The drawback is the buffer cartridges take up the space of the font cartridges. Otherwise, big print buffer, no added fonts; 24K buffer one font cartridge. Anyway this is a nice feature to avoid buying a separate print buffer. The HUG print spooler program (P/N 885-3029-37) somewhat negates this feature if your computer has ample RAM.

I like the NEC emphasis on controlling noise. It is very obvious the printer is enclosed to control noise. There is a big difference in the noise level with the P351 printing. The P5 tractor feed is even enclosed on all sides to reduce noise. I know when I added the tractor feed to the Diablo 620 the noise levels went way up. The Quiet feature is also a demonstration by NEC to restrict noise. The Quiet feature when selected will slow down the print speed to reduce the noise. The manual claims the reduction is from 53 dBA to 48 dBA. It does make a difference. When I talked to a local NEC salesman I was informed that the quiet mode reduces the force of the pins impact to cut the noise while maintaining the speed. He had not read the manual.

### The Gripes

I have my gripes about the P560 as well. The first is the platen control. There is only one and it is located on the right side of the printer. It is a personal problem but my setup places that side of the printer a long way from the operator. Anyway, it is a point to consider. Another gripe is the paper traveling. What I mean is that without a tractor feed you might as well forget printing any long documents. For some reason all printers seem to have this problem. The P560, P351 and the P620 all like to run everything to the right eventually destroying the right side of the paper. When I finally decide on which printer to keep, a tractor feed is in my future.

Another problem is the interface with WordStar. The P560 printer seems to be able to handle all the features of WordStar, but I am unable to get the subscript/superscript to work correctly or at least what I am use to seeing with the 620. Also, the character spacing and line height commands are ignored by the P560. If I work with it enough, it may be possible to get it to function. On the other hand, the P351 has the same problems with WordStar unless I tell WordStar to think the P351 is a P620, then everything works just right. Line height, character width, and superscript and subscript work just like the P620 — interesting. If I try that with the P560, we just get each character separated by a form feed. The bold print ends up as four separate characters and then a form feed. Of course, in both cases MS-DOS is CONFIGUR'd for a parallel printer; only the INSTALL.COM on WordStar is changed. The P560 also kept double spacing everything until I figured out the DIP switch for Auto feed XT must be disabled. No such problems with the 351.

### The Great Race

To compare the speed of the two printers I did some unscientific timing of various printing jobs with a stop watch. First I put H-120 in to typing mode with CTRL P. I chose to use a file that many readers would have access to for test purposes. I chose the assembly version of SCRNCLK by Pat Swayne on the HUG print spooler disk. The second test was a LLISTing of a BASIC program entitled BASMAPER from REMARK February 1984 by Ted Miller Jr. and third I used PRINT.TST from Wordstar. The timing was done at the point where I hit the enter button after either Type SCRNCLK.ASM or LLIST or ESC in Wordstar.

**Speed Chart**

| Item | SCRNCLK.ASM | BASMAPER.BAS | PRINT.TST |
|---|---|---|---|
| Lines | 268 | 31 | 121 |
| Characters | 7051 | 3743 | 4093 |
| Time Trials (sec.) | | | |
| P351 | | | |
| Hi speed | 98 = 72cps | 25 = 150cps | 82 = 50cps |
| Letter quality | 292 = 24cps | 73 = 51cps | 151 = 27cps |
| P560 | | | |
| Hi speed | 92 = 77cps | 27 = 138cps | 76 = 54cps |
| Letter quality | 224 = 31cps | 81 = 46cps | 112 = 36cps |
| P620 | | | |
| Letter quality | 632 = 11cps | 291 = 13cps | 354 = 12cps |

As you can see the two printers are very closely matched for speed.

### About Face, Serial Interface

The P351 as I said above is also a serial printer. The manual lists the printers that should be selected if the P351 does not appear on the computer's install list. The printers that match the P351 setup most closely are Qume Sprint 11, Qume Sprint 5, Diablo 1610, and Diablo 1620. There are other serial printers mentioned, but the manual states the match is not perfect. The printer seems to have great flexibility to be setup for serial operation. Baud rates are from 110 to 19200 with 9600 being the default setting. Baud is selected by DIP switch. In addition, there are jumpers that further configure the printer for the computer. The instructions seem to be very complete, although I did not need to explore any of these features. I did hook the P351 up as a serial printer using the P620 cable. Everything was fine except I had to change the baud rate at the computer end.

Speaking of baud rates, there are a number of other controls operated through DIP switched. I believe a comparison table will help to show the differences.

| DIP switches | P560 | P351 |
|---|---|---|
| 6 line/8line | yes | no |
| 1 inch margin | yes | no |
| form feed 72 lines/66 lines | yes | yes |
| auto line feed | yes | no |
| line feed CR on/off | yes | yes |
| CR with line feed on/off | yes | yes |
| shape of 0 with without slash | yes | no |
| set left margin on/off | yes | no |
| draft or LQ 10 CPI | yes | yes |
| 7 or 8 bit data | yes | yes |
| input buffer on/off | yes | no |
| code sequence | 12 countries | 8 countries |
| character set IBM/Italic | yes | no |
| SLCT in signal(low) | yes | no |
| Auto feed XT signal(low) | yes | no |
| Interface and protocol | NA | yes |
| baud rate 110,300,600,1200,2400, 4800,9600,19200 | NA | yes |
| parity even,odd,ignore,no | NA | yes |
| stop bit 1 or 2 | NA | yes |
| character pitch 10,12,PS,16.7 | no | yes |
| image print selection | no | yes |
| font selection | no | yes |
| bidirectional on/off | no | yes |
| auto character wraparound | no | yes |
| emulation mode Qume/IBM | no | yes |
| select on/off at power up | no | yes |
| line buffer 4K/256K | no | yes |

### 24-Pin Graphics

I think a word about the 24-pin graphics technology is needed. I am by no means an expert, but I have been able to print graphics using the 24-pin technique. Both printers use the Image Data Transfer Graphics for addressing each pin. The pins are 8 mils or 2 mm in diameter or about twice the diameter of a hair.

The graphics are available in either eight or 24 pins. However, operating the individual pins is not that difficult. The pins are arranged in a column so that they may be addressed by position. Each position has been assigned a value so the sum of the values determines which pins will fire. For eight pin graphics, the dot column values are:

```
Position              Column
128                     +
64                      +
32                      +
16                      +
8                       +
4                       +
2                       +
1                       +

Thus if the 8 bit data byte were
sent such as 69,  or 155, or 42
the result would be:

   Firing pin = *            Pin       Columns
                             128           +    128 *       +
                             64         64 *        +       +
                             32            +        +    32 *
                             16            +     16 *       +
                             8             +      8 *     8 *
                             4           4 *        +       +
                             2             +      2 *     2 *
                             1          _1_*    __1_*     ___+
                                Sum     69      155       42
```

Since the eight pin graphics is every third pin to use the 24-pin graphics, we need to send three bytes of data and think of the graphics as three eight bit bytes stacked one above the other. Thus, if we send the same data bytes to the printer in the 24-pin mode, we would have a dot column that looks like this:

```
128          +
64        64 *     Data bytes to printer: 69,155,42
32           +
16           +
8            +
4          4 *
2            +
1          1 *
128      128 *
64           +
32           +
16        16 *
8          8 *
4            +
2          2 *
1          1 *
128          +
64           +
32        32 *
16           +
8          8 *
4            +
2          2 *
1            +
```

The only item left then is the density of the dots. Dot density is controlled by ESC codes. There are four ESC codes to control density:

ESC K = single density or 60 dots per inch
ESC L = double density or 120 dots per inch
ESC Y = high speed double density or 120 dots per inch
ESC Z = quadruple density or 240 dots per inch

All this is set up by the formula n1+(n2*256) where n2 = number of dot columns /256 and n1 = number of dot columns - (n2 * 256). The printers will expect the amount of graphics data it will be receiving based upon this formula. If the the program sends more data than is expected, the result will be regular characters. If there is not enough data, then the printer will stop until it receives additional data. Both manuals go into enough detail to allow you to figure the entire process out. It works. I tried a screen dump program in BASIC that got the results, although it was very slow. Not the printers fault, mine in the program.

### The Choice

As I close out this article, I am still not sure which printer I will finally keep. The P351 seems to be much friendlier with the H-120. By that I mean there is less programming and changes required on my part to make all the features work as I have become accustom. The drawbacks are the noise and font changes. The P560 has the opposite drawbacks. It seems to be harder to make it interface with the H-120, but it is quiet and it easy to change fonts. I am convinced that both printers were made with PCs in mind, rather than an H-120.

Let's see who heads it's . . .

### Printer Manufacture Addresses:

Toshiba 3-IN-ONE Printer P351
Toshiba America,Inc
Information Systems Division
2441 Michelle Drive
Tustin, CA 92680

NEC Pinwriter P5 Series Model P560
NEC Information System, Inc
1414 Massachusetts Avenue
Boxborough, MA 01719

Diablo 620
Diablo Systems, Inc
901 Page Avenue
P.O. Box 5030
Fremont, CA 94537

## *About The Author*

*Donald S. Alspach is married and has a daughter in high school. His family and he have lived in Alaska for 16 years. He works for the Municipality of Anchorage as Division Manager in the Department of Community Planning. His job requires him to work with an IBM mainframe computer using TIF, GDDM, PM, PS/370 and ADRS2. His H-120 is his only home computer and has been in service since 1983. The H-120 is used primarily for a word processing and mailing label production business.*

# Building An 8 MHz H-151 And Installing An EGA Video Board

*Robert Maskasky*
*3220 Carlton Court*
*Green Cove Springs, FL 32043*

Recently, I assembled the H-150 Speed-up Modification, REMark, June 1986, D. Bencivengo. I was excited about the prospect of increasing the speed of my H-151 computer. The modification worked exactly as written in the June REMark magazine. I was unhappy with the prospect of adding a switch to the front panel of my H-151. I also found myself using the [Ctrl]-[Alt]-[Del] key combination to reboot the computer, only to encounter the "Timer Interrupt Failure". I began investigating possible solutions to these problems.

Evidently, the MFM-150 ROM tests the system timer during the boot-up procedure. The CPU programs the system timer to interrupt the processor after a certain time delay, the CPU then executes a software timing loop, and waits for the system timer interrupt. When the CPU is running at a speed faster than 4.77 MHz, the CPU completes the software timing loop before the system timer provides the required interrupt, thus the CPU reports a "Timer Interrupt Failure".

Normally, resolution of this problem would require modification of the MFM-150 ROM program. There are several ways to work around this problem. The circuit designed by D. Bencivengo uses an R-C timer network to ensure the H-151 system timer runs at 7.38 MHz for the first few seconds after the hardware RESET. During this time, the MFM-150 ROM completes the system timer tests. After the system timer passes the test, the H-150 Speed-up circuit, returns the system timer to the normal 4.77 MHz, this is important to ensure accuracy of the DOS clock. Since this R-C timer requires a hardware RESET to initialize the circuit, the [Ctrl]-[Alt]-[Del] key combination, which does not provide a hardware RESET, will not allow the CPU and the system timer to start at the same speed.

The H-151 is provided with diagnostic LEDs on the CPU board, which indicate the status of the system self-test. By observing these LEDs during the boot-up, it is evident that the MFM-150 ROM uses these LEDs to indicate the status of the self-test any time the MFM-150 ROM is testing the computer. Specifically, the "INT" lamp indicates when the system timer is being tested. By using the "INT" lamp to control the speed of the H-150 Speed-up circuit, the problem with the [Ctrl]-[Alt]-[Del] reboot is solved.

The "INT" LED is driven by integrated circuit U200 on the CPU board (U201 on some versions of the CPU board). This Integrated circuit is a write only port used exclusively to drive the diagnostic LEDs. Therefore, any software running on the computer is unlikely to write to this port, and software cannot read the status of the LEDs. Therefore, a short program can be written to change the status of the "INT" LED, and thus the speed of the CPU without any worry of interfering with other software which may be used on the computer. Following are the details of the necessary modification to the H-150 CPU Speed-up circuit (refer to REMark, June 1986).

The MFM-150 ROM uses the diagnostic LEDs to report the self-test status of the machine. By coupling the speed switch to these LEDs, the MFM-150 will shift into slower speed whenever the Timer interrupt is being tested. For the schematic and the circuit layout diagram of the modified H-150 Speed-up board, see the June 1986 REMark magazine. The modification to the H-150 CPU Speed-up circuit board is as follows:

1. Remove R3, R4, CR1, R5, and C3.
2. Cut the following circuit runs:
   a. From U3-10 to U3-5
   b. From U3-4 to U3-9 (U3-4 still connects to U1-13)
3. Remove J1 and connect U2-13 to ground.
4. Connect U3-8 to U3-6 (This is the speed input).
5. Connect U3-9 to U3-5 to Ground.
6. Connect U3-10 to a 330 Ohm resistor.
7. Connect the other end of the 330 Ohm resistor to a wire to the cathode of an LED (The Turbo Speed indicator).

8. Connect the Anode of the Turbo Speed indicator LED to +5 Volts (P2 on the H-150 CPU Speed-up board).
9. Using a piece of 22ga wire, carefully solder one end to the cathode of CR4 (the "INT" LED system CPU board). Connect the other end of the wire to the speed input (P1 on the H-150 CPU Speed-up board).

The Speed indicator LED will light when the CPU is running at the fast speed. This LED may be mounted in the empty space below the disk drives with a drop of super glue with no permanent modification to the computer case. A rectangular style LED works particularly well.

To test the circuit modification, boot-up the computer, the Speed indicator LED should remain off for about two seconds, then the LED should come on indicating the CPU is running at fast speed. The Speed indicator should be illuminated (on) when the "INT" LED on the CPU board is off and vice versa. Now try the [Ctrl]-[Alt]-[Del].

The Turbo board will now shift into lower gear when the interrupts are being tested by the MFM-150 (including the notorious system timer interrupt) and shift into fast speed after the test is completed. The hardware RESET button is optional, depending on your needs.

Now to control the speed of the computer from software. The diagnostic LEDs are controlled by port 00C0 hex. Specifically, Bit 08 controls the "INT" LED and thus the CPU speed. If Bit 08 is set to a "1", the "INT" LED is off and the CPU runs at fast speed. If Bit 08 is reset to "0", the "INT" LED is on, and the CPU runs at 4.77 MHz. For those people familiar with assembly language programming, following is the assembly language for two short programs, SLOW.COM which sets the CPU to run at 4.77 MHz, and FAST.COM which sets the CPU to run at fast speed.

```
CODE SEGMENT

ASSUME CS:CODE

TITLE  ASSEMBLY LANGUAGE PROGRAM FOR SETTING FAST CPU SPEED (FAST.COM)

ORG 100H

START:  MOV AX,4CFFh
        OUT 0C0h,AL                 ;OUTPUT FF TO PORT C0 hex
        INT 21H                     ;FUNCTION CODE 4C (TERMINATE)

CODE ENDS

END START
```

```
CODE SEGMENT

ASSUME CS:CODE

TITLE  ASSEMBLY LANGUAGE PROGRAM FOR SETTING SLOW CPU SPEED (SLOW.COM)

ORG 100H

START:  MOV AX,4CF7h
        OUT 0C0h,AL                 ;OUTPUT F7 TO PORT C0 hex
        INT 21H                     ;FUNCTION CODE 4C (TERMINATE)

CODE ENDS

END START
```

If the reader does not have experience with assembly language programming, the programs can also be created using the MS-DOS DEBUG utility. To create the SLOW .COM program, type the following items as written (x represents don't care response):

```
DEBUG
-NSLOW.COM
-E100
xxxx:0100  xx.B8 xx.F7 xx.4C xx.E6 xx.C0
     xx.CD xx.21
-RCX
cx 0000
:0007
-W
writing 0007 bytes
-Q
```

To create the FAST.COM program, type the following items as written (x represents don't care response):

```
DEBUG
-NFAST.COM
-E100
xxxx:0100  xx.B8 xx.FF xx.4C xx.E6 xx.C0
     xx.CD xx.21
-RCX
cx 0000
:0007
-W
writing 0007 bytes
-Q
```

Now that the FAST.COM and the SLOW .COM programs have been created, try typing "SLOW" and the Speed indicator LED should go off indicating the CPU is running at slow speed (4.77 MHz). Try typing "FAST" and the CPU should shift back to fast speed.

Now that we have software control of the CPU speed, one more problem remains unresolved. The "FORMAT" program will not work at fast speed. This problem is more than a problem with the FORMAT .COM routine. Some programs will not run at the fast speed due to the same reason, and the machine may display a message that the floppy disk is defective or unusable. This is due to an incompatibility in the ROM BIOS in the H-151 computer. Specifically, the interrupt handler for interrupt 13H function code 4. This function is used to verify that a sector on the floppy disk is readable, without actually transferring any data into memory. When software attempts to perform this operation, the BIOS will usually return an error code 08H, indicating a DMA error occurred. The software usually interprets this error as a bad floppy disk and displays an appropriate message. My studies indicate that the no error has actually occurred, and any data transferred to or from the floppy disk by one of the other function codes is correct. Function code 4 is used by the FORMAT.COM routine to verify that the directory track on the floppy disk is usable prior to formatting the disk. A few other programs will use this function code to verify the disk.

The Read function (function code 04h) also verifies the disk data, and transfers the data to memory. The read function works normally, so by transferring the data to a non-existent or read-only address, the equivalent of the verify function can be performed. The following program "DISKFIX.COM" will trap the verify function and change it to a read function with the data transferred to a read-only address within the computer. Once loaded, the program will remain resident in the computer memory until the machine is rebooted. I suggest installing "DISKFIX" in the AUTOEXEC.BAT file on the boot-up disk. After DISKFIX is installed in the machine, the FORMAT program will operate at either fast or slow speed, and I have not found any other programs which will not run at the fast speed. Although playing some games at the fast speed requires super-human coordination.

```
PAGE 60,132

;*******************************************************
; INTERRUPT 13 TURBO SPEED DISK FIX-UP ROUTINE
;    -DISKFIX-
; BY  Robert J Maskasky
;*******************************************************

CODE SEGMENT

ASSUME CS:CODE,DS:CODE,ES:CODE

ORG 100H

START:
      MOV AX,3513H
      INT 21H                          ;GET Floppy INTERRUPT VECTOR
      MOV WORD PTR NEXT_SEGMENT,ES     ;SAVE THE TRANSFER ADDRESS
      MOV WORD PTR NEXT_OFFSET,BX
      MOV WORD PTR NEXT_SEGMENT1,ES
      MOV WORD PTR NEXT_OFFSET1,BX

      MOV DX,OFFSET BEGIN              ;SET THE TRANSFER ADDRESS
      MOV AX,2513H
      INT 21H

      MOV DX,OFFSET LAST_BYTE          ;EXIT AND STAY RESIDENT
      INT 27H

;**********************************************************************
; FLOPPY DISK (INT 13h) HANDLER
;
; Function is to intercept INT 13H, Function 04h and return a verify
; good (no error occurred)
;**********************************************************************

BEGIN:
      CMP AH,04h                       ;TEST FOR VERIFY FUNCTION
      JNE INT_13_CONTINUE              ;IF NOT VERIFY FUNCTION

      CMP DL,04h                       ;TEST FOR WINCHESTER OPERATION
      JA INT_13_CONTINUE               ;IF NOT FLOPPY FUNCTION

      MOV AH,02h                       ;CHANGE TO A READ OPERATION

      MOV WORD PTR BX_SAVE,BX          ;SAVE BX VALUE
      MOV WORD PTR ES_SAVE,ES          ;SAVE ES VALUE

      MOV BX,0F800h
      MOV ES,BX
      XOR BX,BX

      PUSHF
      DB 09AH                          ;CALL TO INTERRUPT HANDLER
NEXT_OFFSET:  DW 0
NEXT_SEGMENT: DW 0
      MOV BX,WORD PTR ES_SAVE          ;GET ES VALUE
      MOV ES,BX
      MOV BX,WORD PTR BX_SAVE          ;GET BX VALUE

      DB 0CAh                          ;IRET AND SAVE FLAGS
      DW 02h

INT_13_CONTINUE:
      DB 0EAH                          ;JUMP TO INTERRUPT HANDLER
NEXT_OFFSET1:  DW 0
NEXT_SEGMENT1: DW 0

BX_SAVE DW 0
ES_SAVE DW 0

LAST_BYTE: DB 0

CODE ENDS

END START
```

For those people not familiar with assembly language programming, the DISKFIX program can be created using the DEBUG program as follows. To create the DISKFIX .COM program type the following items as written in Italic letters (x represents don't care response):

```
DEBUG
-NDISKFIX.COM
-E100
xxxx:0100  xx.B8 xx.13 xx.35 xx.CD xx.21 xx.8C xx.06 xx.41
xxxx:0108  xx.01 xx.89 xx.1E xx.3F xx.01 xx.8C xx.06 xx.53
xxxx:0110  xx.01 xx.89 xx.1E xx.51 xx.01 xx.BA xx.22 xx.01
xxxx:0118  xx.B8 xx.13 xx.25 xx.CD xx.21 xx.BA xx.59 xx.01
xxxx:0120  xx.CD xx.27 xx.80 xx.FC xx.04 xx.75 xx.29 xx.80
xxxx:0128  xx.FA xx.04 xx.77 xx.24 xx.B4 xx.02 xx.89 xx.1E
xxxx:0130  xx.55 xx.01 xx.8C xx.06 xx.57 xx.01 xx.BB xx.00
xxxx:0138  xx.F8 xx.8E xx.C3 xx.33 xx.DB xx.9C xx.9A xx.00
xxxx:0140  xx.00 xx.00 xx.00 xx.8B xx.1E xx.57 xx.01 xx.8E
xxxx:0148  xx.C3 xx.8B xx.1E xx.55 xx.01 xx.CA xx.02 xx.00
xxxx:0150  xx.EA xx.00 xx.00 xx.00 xx.00 xx.00 xx.00 xx.00
xxxx:0158  xx.00 xx.00 <Return>
-RCX
cx 0000
:005A
-W
writing 005A bytes
-Q
```

### Breaking The 8 MHz Barrier!

During my studies of several H-151 computers, I have found that almost all of the modified H-151 computers will run at 6.6 MHz (19.8 MHz crystal) using the Zenith provided integrated circuits. Some machines will run at 7.3 MHz (22 MHz crystal) with little or no modification. I have succeeded in running my H-151 computer at 8.0 MHz (24 MHZ crystal) with no wait states. With no wait states, the 7.0 MHz H-151 will run even with a H-158 computer running at 8 MHz (1 wait state). At 8 MHz the H-151 will easily pass the H-158 computer in speed. My computer may even exceed 8 MHz if I could find the necessary crystal.

There are four areas in the H-151 computer which usually give trouble at the faster speeds.

The DMA Controller chip (8237) is the most frequent problem area. Most 8237 chips are manufactured to operate at 5 MHz, and operating them at faster speeds is strictly trial and error. However, some manufacturer chips run better at the faster speeds than others. The Intel 8237A-5 (denoted by an "I 8237" on the chip) will frequently run at 6.6 MHz. The Intel 8237 chip will probably carry an AMD copyright symbol. If it specifies an I 8237, it is probably an Intel part. Harris manufactures an 8 MHz 82C37A chip and is a must for anyone wanting to break the 7 MHz barrier. The Harris part number is "CP82C37A" (do not purchase the CP82C37A-5 version, as this is only a 5 MHz part).

I have found no problems with the Bus Controller (8288). This chip is in most cases a 10 MHz rated part, and probably will not give troubles at 8 MHz. Since the 8288 chip controls most of the operations of the computer, problems with other chips not running at the faster speeds may appear to be related to the 8288 chip. The Harris CP82C88 is preferred because of the lower power consumption.

The Interrupt Controller (8259) is another troublesome area. The AMD 8259 frequently will work at 7 MHz. The Harris "CP82C59" is once again the ultimate since it is also rated at 8 MHz.

The 8088 is a chip which should not be overlooked. The 8088-2 is rated at 8 MHz and is necessary for speeds in excess of 7 MHz. The NEC V20-2 is also a good alternative for the faster speeds. Despite many fantastic claims of the advantages of the NEC V20 over the 8088, I have experienced an average speed improvement of only about 10% when using the V20. The Norton utilities will rate the V20 at about an 80% speed improvement over the 8088. This is due to the way Norton uses to test the machine speed. Norton loops on a small set of instructions to test the machine speed. These instructions are infrequently used in programs, but execute significantly faster on the V20 chip than on the 8088. For the same reason, Norton Utilities will slightly exaggerate the speed of the 80286 based machines (Z-241 and 248).

The memory should be rated at at least 150 nanoseconds. It is preferred to use 120 nanoseconds memory chips to ensure no memory speed problems are encountered. This is especially critical due to the no wait state hardware configuration of the H-151. In order to realize the full benefit of the faster memory chips, the dynamic RAM delay line U456 should be jumpered to the 60 nanosecond position. This can be accomplished by making a socket adapter out of a regular IC socket. The socket adapter will fit between U456 and the socket on the Heath/Zenith memory board. Pins 4 and 8 should be removed from the socket so that these pins on the delay line will not make contact with the memory board. Then, carefully solder a jumper wire from pin 12 on the socket adapter to a lead which is epoxyed into the pin 4 position on the memory board side of the socket adapter. Connect another jumper wire from pin 10 to a lead which is epoxyed into the pin 8 position on the memory side of the socket adapter. This effectively connects pin 10 of the delay line (60 nS) to the memory board socket position 8 (CASTIM) and connects pin 12 of the delay line (20 nS) to the memory board socket position 4 (MUXTIM).

**Wire List For U456 Socket Adapter**

| from | to |
|---|---|
|  | Memory Board |
| Delay Line U456 | Socket |
| Pin # | Pin # |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | - |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | - |
| 9 | 9 |
| 10 | 8 |
| 11 | 11 |
| 12 | 4 |
| 13 | 13 |
| 14 | 14 |

This modification will permit the H-151 to realize the full benefit of the 120 nanosecond RAM, and is necessary for 8 MHz speed.

The last item is the crystal on the turbo board. Remember, the crystal frequency required is three times the speed of the machine. Thus 8 MHz speed requires a 24 MHz crystal. I recommend progressively trying several crystals between 16 MHz and 24 MHz to determine the maximum machine speed. Once the maximum crystal frequency is found, remember, the machine still may not work 100% at that frequency. If a program bombs at the faster speed, try it at the slow speed. You may discover that an infrequently used instruction, or hardware function does not work at the faster speed and you may need to reselect the high speed crystal or look for the individual chips causing the problem.

### Installing An EGA Board

The H-151 computer utilizes a peculiar hardware configuration on the video board. The MFM-150 ROM requires 16 kilobytes of RAM in order to operate. Unfortunately, this RAM is located on the Video board, making removal of the video board impossible. An EGA board will not operate properly with the Zenith Video board installed. The simplest solution to this problem is to purchase a special address decoder ROM from your Heath/Zenith dealer. This ROM disables the Video portion of the Video board and still allows the system RAM to operate on the Heath/Zenith video board. This solution uses an extra expansion slot in the computer since the Heath/Zenith video board is still required solely to provide 16 K of RAM. Zenith designed the CPU board to hold the extra RAM required, but this requires some reorganization of the chips on the CPU board.

The following modification requires some special equipment such as a ROM burner capable of burning a 27256 EPROM and a 74S287 ROM.

First, we must free a 128 K memory socket. This can be accomplished by copying the two 128 K system ROMS U207 and U208 into a single 27256 EPROM (this EPROM must be 150 nanoseconds or better if the machine is to be used at 8 MHz). This 27256 chip will eventually replace U207. Note: The program from U208 should be placed in the lower half of the 27256 and the program from U207 should be placed in the upper half of the 27256 chip.

The address decoder chip U236 must be replaced with another address decoder to allow access to the full 256 K of ROM as U207. The address decoder chip required is a Harris 7611 or a 74S287. This chip is a high speed ROM and requires the following pattern be burned into it using a Bipolar ROM Burner.

```
Address Data
0000    0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
0010    0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
0020    0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
0030    0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
0040    0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
0050    0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
0060    0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
0070    0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
0080    0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
0090    0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
00A0    0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
00B0    0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
00C0    0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
00D0    0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
00E0    0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
00F0    09 09 09 09 0F 0F 0F 0F 0A 0A 0A 0A 0A 0A 0A 0A
```

Next, we will install a 16 kilobyte RAM in socket U208 on the CPU board. The NEC 43256 Static RAM is recommended. This chip is actually 32 kilobytes, but is pin compatible, and can be used as a 16 kilobyte RAM. This chip sells for about $15.00 (mail-order) for a 120 nanosecond part.

Now we are only but a few jumpers away from eliminating the Video Board. Remove all jumpers from P203 and install the following jumpers:

P203 Pins 1 to 6 (ROM 2 = RAM)
P203 Pins 5 to 10 (ROM 1 = 256K)

Many H-151 owners have expanded their memory by installing 256 kilobyte memory chips on the main memory board and replacing the decoder PAL (U455) with one that decodes either 640 kilobytes or 704 kilobytes of memory. The EGA video board uses the memory between A0000 hex and B0000 hex for video memory. This is the same area of memory used by the additional RAM memory (the memory between 640 k and 704 k). The simplest solution to this problem is to replace the PAL with one which only decodes 640 k memory. A second alternative which will work with some 704 k PALs, is to remove U455 from its socket, and reinstall it allowing pin 4 to fit outside the socket (not making contact with the memory board). This solution will only work with some PALs, (generally the ones which use all 41256 memory chips and no 4164 chips. If this does not work with your PAL, the problem will become evident when you run a memory hungry program and discover that the same memory location is mapped in two different areas of the machine.

## Miscellaneous

The components listed in the above article are available from:

| | |
|---|---|
| CP82C37A (8 MHz) | Harris Semiconductor |
| CP82C59 (8 MHz) | Harris Semiconductor |
| CP82C88 | Harris Semiconductor |
| 8088-2 | I.C. Express, Jameco Elec. |
| V-20 (8 MHz) | I.C. Express, Jameco Elec. |
| 27256* (150 nanosecond) | I.C. Express |
| 43256 (120 nanosecond) | I.C. Express |
| 74S287* | Jameco Electronics |

* Requires Programming

The 74S287 ROM is available pre-programmed at a price of $15.00 from R. J. Maskasky.

## Conclusion

The H-150 Speed-up Modification is an excellent modification for the H-150 computer, making the CPU speed much more acceptable, for a minor expense. This modification surpasses many of the "TURBO" kits costing several hundred dollars. As with most of the "TURBO" modification kits, the H-150 Speed-up modification does not solve the problem of compatibility with the MFM-150 ROM program. With minor hardware changes to the H-150 Speed-up modification circuit board, these non-compatibility problems can be resolved, and now the speed of the H-150 can be changed from the keyboard using software rather than a switch located on the front panel, or worse yet, the rear of the computer. With a little work and minor expense, the H-151 computer can be made to run faster than the H-158 computers. My H-151 computer is running at 8.0 MHz with no problems. I believe it could run faster, if I could find the necessary crystal. One final note: Be cautious of any claims that the H-151 computer can run faster than the Z-240 series computers. I have seen several such claims. Some modification kits available for the H-151 computers lower the compatibility of the computer, or are great exaggerations of the speed capabilities of the product.

## References

H-151 Personal Desktop Computer, Service Data Manual, Zenith Data Systems, 1984.

Bipolar Microcomputer Components Data Book, Texas Instruments, 1977.

CMOS Digital Data Book, Harris Corporation, 1986.

Microsystems Components Handbook, Intel Corporation, 1986.

MOS Microprocessors and Peripherals Data Book, Advanced Micro Devices, 1985.

Remark Magazine, Volume 7 Issue 6, June 1986.

## Products Mentioned

I.C. Express
15358 Valley Boulevard
City of Industry, CA 91746
(818) 369-2688

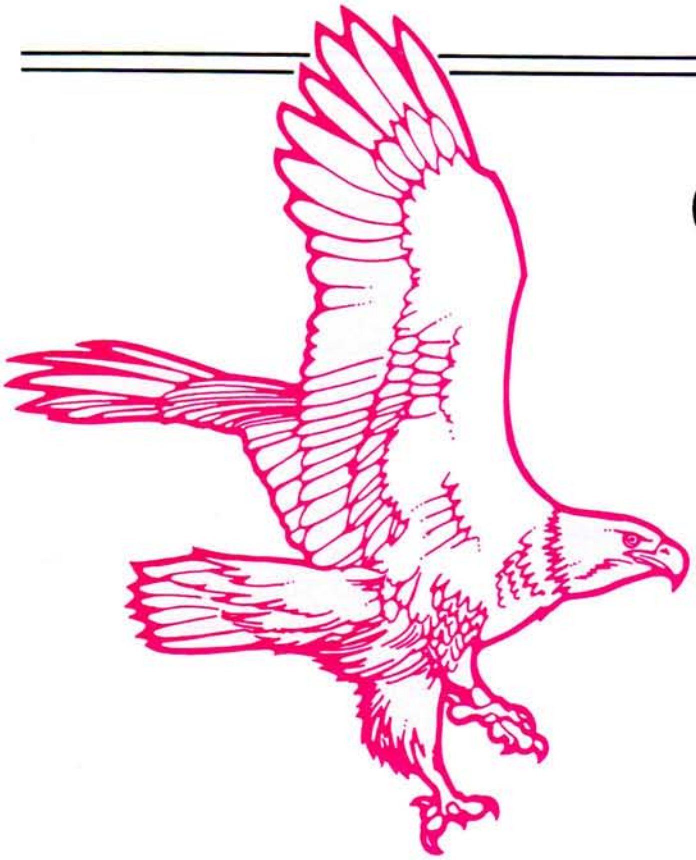
Harris Semiconductor
2410 Palm Bay Road
Palm Bay, FL 32905

Jameco Electronics
1355 Shoreway Road
Belmont, CA 94002
(415) 592-8097

```
PAGE 60,132

;********************************************************
; INTERRUPT 13 TURBO SPEED FIX-UP ROUTINE
; BY  Robert J Maskasky
;********************************************************

CODE SEGMENT

ASSUME CS:CODE,DS:CODE,ES:CODE

ORG 100H

START:
      MOV AX,3513H
      INT 21H                           ;GET Floppy INTERRUPT VECTOR
```

```
        MOV WORD PTR NEXT_SEGMENT,ES        ;SAVE THE TRANSFER ADDRESS
        MOV WORD PTR NEXT_OFFSET,BX
        MOV WORD PTR NEXT_SEGMENT1,ES
        MOV WORD PTR NEXT_OFFSET1,BX

        MOV DX,OFFSET BEGIN                 ;SET THE TRANSFER ADDRESS
        MOV AX,2513H
        INT 21H

        MOV DX,OFFSET LAST_BYTE             ;EXIT AND STAY RESIDENT
        INT 27H

;***************************************************************
; FLOPPY DISK (INT 13h) HANDLER
;
; Function is to intercept INT 13H, Function 04h and return a verify
; good (no error occurred)
;***************************************************************

BEGIN:
        CMP AH,04h                          ;TEST FOR VERIFY FUNCTION
        JNE INT_13_CONTINUE                 ;IF NCT VERIFY FUNCTION

        CMP DL,04h                          ;TEST FOR WINCHESTER OPERATION
        JA INT_13_CONTINUE                  ;IF NCT FLOPPY FUNCTION

        MOV AH,02h                          ;CHANGE TO A READ OPERATION

        MOV WORD PTR BX_SAVE,BX             ;SAVE BX VALUE
        MOV WORD PTR ES_SAVE,ES             ;SAVE ES VALUE

        MOV BX,0F800h
        MOV ES,BX
        XOR BX,BX

        PUSHF
        DB 09AH                             ;CALL TO INTERRUPT HANDLER
NEXT_OFFSET:  DW 0
NEXT_SEGMENT: DW 0

        MOV BX,WORD PTR ES_SAVE             ;GET ES VALUE
        MOV ES,BX
        MOV BX,WORD PTR BX_SAVE             ;GET BX VALUE

        DB 0CAh                             ;IRET AND SAVE FLAGS
        DW 02h
```

```
INT_13_CONTINUE:
        DB 0EAH                 ;JUMP TO INTERRUPT HANDLER
NEXT_OFFSET1:  DW 0
NEXT_SEGMENT1: DW 0

BX_SAVE DW 0
ES_SAVE DW 0

LAST_BYTE: DB 0

CODE ENDS

END START


CODE SEGMENT

ASSUME CS:CODE

ORG 100H

START:  MOV AX,4CFFh
        OUT 0C0h,AL             ;OUTPUT FF TO PORT C0 hex
        INT 21H                 ;FUNCTION CODE 4C (TERMINATE)

CODE ENDS

END START


CODE SEGMENT

ASSUME CS:CODE

ORG 100H

START:  MOV AX,4CF7h
        OUT 0C0h,AL             ;OUTPUT F7 TO PORT C0 hex
        INT 21H                 ;FUNCTION CODE 4C (TERMINATE)

CODE ENDS

END START
```



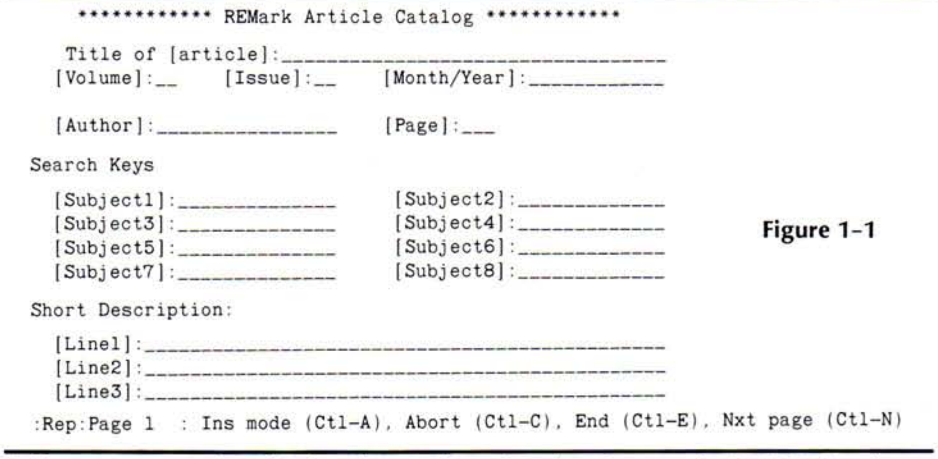
# CONDORizing Your REMarks

**_Don Breslauer_**
*2404 Free State Lane*
*Lawrence, KS 66046*

I just got this great idea for a WordStar patch! Now, if I could only find Pat Swayne's article in REMark. (The search continues for thirty minutes as I rummage through all the past issues of REMark trying to find the article.) Found at last! I turn around, tired and worn out from the search, only to find three years worth of REMarks strewn across the floor of the computer room — there has to be a better way.

Well, as you have probably guessed by now, there is. Two database managers, Condor FMS and Condor rDBMS, enables the creation of a database to store REMark articles. However, unlike most databases created within the Condor environment, our database, which I'll just name REMARK, will require the use of a help menu and command files. Condor command files, although not very well documented in the manual, are very easy to use.

After booting MS-DOS with the Condor system disk, type DBMS <RET>. After the Condor copyright has been displayed, log onto the data disk. The database REMARK begins the way every Condor database is begun, using the FORMAT command. At the "B>>" (or the letter of whatever drive you are using) type FORMAT REMARK <RET>. This starts the creation process. Condor responds with blanking the screen except for a line of key combinations at the bottom. In this stage, the form for the database is designed. As depicted in Figure 1-1, field names are enclosed in brackets and field length is determined by the number of underscores after the name. The first field is ARTICLE. It has 46 underscores after it. The next two fields, VOLUME and ISSUE, have two underscores following them. AUTHOR is twenty underscores in length and PAGE is three. The next eight fields are under the heading of Search Keys. These will contain key words or subjects. For instance, instead of searching every magazine for the article on patching WordStar, I could have instead searched for a subject of "WordStar" or "Patch." Each SUBJECT is sixteen underscores in length. Finally, there are three LINES with 61 underscores used for a brief description of the article. After the last underscore has been typed, press ^E to end format.



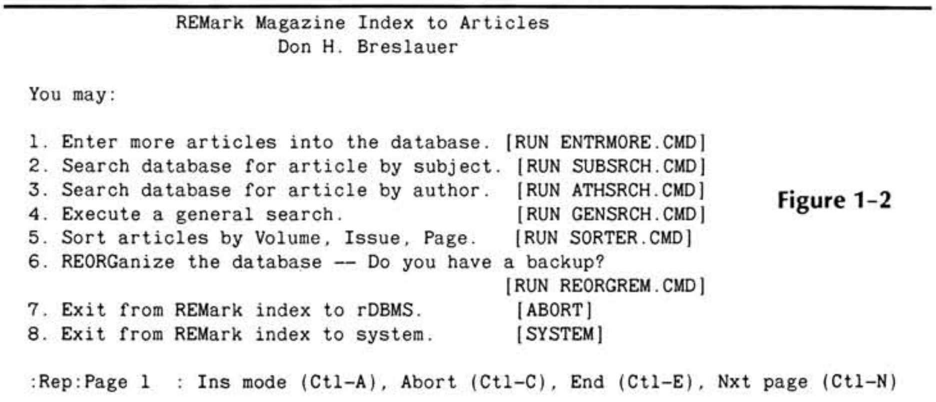
```
     ************ REMark Article Catalog ************

      Title of [article]:______________________________________
     [Volume]:__     [Issue]:__     [Month/Year]:____________

     [Author]:________________      [Page]:___

  Search Keys

     [Subject1]:______________      [Subject2]:______________
     [Subject3]:______________      [Subject4]:______________
     [Subject5]:______________      [Subject6]:______________
     [Subject7]:______________      [Subject8]:______________

  Short Description:

     [Line1]:_____________________________________________
     [Line2]:_____________________________________________
     [Line3]:_____________________________________________
 :Rep:Page 1  : Ins mode (Ctl-A), Abort (Ctl-C), End (Ctl-E), Nxt page (Ctl-N)
```

Figure 1-1

After the format of REMARK has been ended, Condor activates DEFINE, a program that defines the fields of the database. Condor will ask for the type of field. All the fields except VOLUME, ISSUE, and PAGE should be entered as AN (alphanumeric). VOLUME, ISSUE, and PAGE, should be entered as N (numeric). However, there is a problem. The second number in the definition line of the numeric fields is negative. This is the minimum value Condor will accept. Because REMark should not have any negative VOLUME, ISSUE, or PAGE numbers (we hope!), 'N' should be the answered to Definitions OK (Y/N). Press <RET> after ARTICLE is displayed. At VOLUME, bring the cursor over to the middle number and replace it with a positive one. Press <RET> and do the same for ISSUE.

Press <RET> two more times when Condor displays the PAGE definition. Replace the middle number with a positive one and Press <RET> until Condor prompts Definitions OK (Y/N), where 'Y' should be answered. Select NO to have the database indexed and YES if you wish to have the definitions printed. REMARK database defined at last!

Although records could be entered into REMARK at this point, it would be very inefficient to do so. One would have to search through every SUBJECT each time an article was desired. Fortunately, Condor provides a mechanism to make this task much easier. Using Condor's HELP program, a menu can be designed to call upon command files to execute desired functions. To create this menu, FORMAT is again used. At the "B>>" type FORMAT REMARK.HLP <RET>. Figure 1-2 displays a picture of how the help file will be designed. There are eight options. Each option is started by a number and followed by a description of the desired function. Enclosed in brackets, at the end of each description, are the instructions that tell Condor which command file to run. Option number one allows the user to enter more data into the REMark database. At the end of that line, the statement [RUN ENTRMORE.CMD] tells Condor to use that command file if choice one is selected. Option number two executes a search throughout the database for articles dealing with a particular subject. If choice two is selected, the command file, SUBSRCH.CMD, will automatically search for the specified subject in all of the subject keys, ie., SUBJECT1, SUBJECT2, SUBJECT3, ect. The third option will search REMARK to find all articles written by a specified author. The command file for this choice is ATHSRCH.CMD. The fourth choice the user may select executes a "general" search. GENSRCH.CMD, the command file, locates articles distinguished other than by AUTHOR or SUBJECT. For instance, search for all articles in the June 1985 issue of REMark. Choice number five calls on SORTER.CMD to sort all the articles in the database by VOLUME, then ISSUE, and finally PAGE. Option number six begins a REORGanization of the REMARK database. This has been included for the user who would like to change the structure of REMARK. Because changing the structure has the capability of destroying an entire field, this function should be used with caution and only if a backup copy of the database has been made. Due to lack of screen space, the statement [RUN REORGREM.CMD] has been placed below the choice description (Condor doesn't seem to care!). Option seven will exit the user to the Condor system (the B>>). Unlike the other options, a command file is not called upon. Instead, the command [ABORT] instructs Condor to exit. Option eight exits the user from the REMark Index to MS-DOS system. As with option seven, a command file is not needed. The statement [SYSTEM] at the end of the choice description directs Condor. After option eight has been successfully entered, press ^E to save REMARK .HLP.



```
                    REMark Magazine Index to Articles
                           Don H. Breslauer

You may:

1. Enter more articles into the database. [RUN ENTRMORE.CMD]
2. Search database for article by subject. [RUN SUBSRCH.CMD]
3. Search database for article by author.  [RUN ATHSRCH.CMD]
4. Execute a general search.               [RUN GENSRCH.CMD]
5. Sort articles by Volume, Issue, Page.   [RUN SORTER.CMD]
6. REORGanize the database -- Do you have a backup?
                                           [RUN REORGREM.CMD]
7. Exit from REMark index to rDBMS.         [ABORT]
8. Exit from REMark index to system.        [SYSTEM]

:Rep:Page 1  : Ins mode (Ctl-A), Abort (Ctl-C), End (Ctl-E), Nxt page (Ctl-N)
```

**Figure 1-2**

Ok, now that we've created the database and the help menu, let's try it out. At the "B>>" type HELP REMARK <RET>. Suddenly, our help menu appears on the screen. But wait, we haven't written the command files yet. Unfortunately, unlike other database managers such as dBASE II and dBASE III, Condor does not have a built in text editor. Therefore, command files must be written in some text editor outside of the Condor system. Where Condor has displayed the "Enter Choice:" on the help menu type 8 and press <RET> to exit to MS-DOS. Although the Condor manual gives an example using the MS-DOS EDLIN, I prefer using a word processor. If you would like to use a word processor, be sure that the output will be in standard ASCII code (e.g. enter WordStar in the non-document mode). After a text processor has been selected, load it and open the file "ENTRMORE.CMD" — the first command file.

```
; ENTRMORE.CMD
;
enter remark
help remark
```

**Figure 1-3**

The job of ENTRMORE.CMD is simply to allow the user to enter more articles into the database. Therefore, as displayed by Figure 1-3, the file contains only two lines. The first, ENTER REMARK, invokes the entering process. The second, HELP REMARK, will call up the REMARK menu after the user has finished entering data. After ENTRMORE.CMD has been completed, save the file and open "SUBSRCH.CMD".

The command file, SUBSRCH.CMD, must ask the user for the subject to be located, find out if the user wishes just to display the data or update it as well, and finally search all eight keys in all store articles for the desired subject. SUBSRCH, shown in Figure 1-4, is much more complex then ENTRMORE. SUBSRCH requires the use of the Condor language. When functions of this language are used, such as MSG (MESSAGE), GET, and LET, they must be preceded by an asterisk. Variables are preceded by a dollar sign. Because Condor will automatically put a ':' after a message, the first *MSG statement does not need punctuation at the end. Directly below the message to enter the subject, is the *GET statement that stores the string into $1. (When searching for a subject that is more than one word, the subject must be enclosed in quotes.) Although UPDATE has all the functions of DISPLAY, for safety's sake I would only select U(pdate) when wanting to change data, hence reducing accidental loss of information. The answer to the second message is stored in $2. Because Condor does not allow the use of the "OR" statement in command files, all the variables must be uniform. Therefore, the next two sets of statements make use of IF-ENDIF pairs to translate a lowercase response into uppercase. This is accomplished through the LET function. Condor requires that there be a space separating the "=" from both sides of the condition. Now that $2 is in capital letters, SUBSRCH will select to UPDATE REMARK or DISPLAY REMARK, depending on the choice made. (Notice that there should not be an * before the command to UPDATE or DISPLAY because these are system not language commands. In both DISPLAY and UPDATE, where usually the string of whatever was to be searched would be placed in the command line, $2 containing this information is used. The search is broken down into three parts: the search for the specified topic in SUBJECTs one, two, and three; the search in keys four, five, and six; and the search in keys seven and eight. This dividing of searches must occur because of the tremendous length of the command. After the search has been completed, Condor returns to the REMark Index Menu.

Wow! That was a long one, but don't worry, once you get the hang of it, command files are not that hard. After you have finished typing SUBSRCH.CMD, save the file and open ATHSRCH.CMD. ATHSRCH, Figure 1-5, is very similar to the SUBSRCH. Instead of searching for a subject, ATHSRCH searches for articles by a specific author. Because there is only one author, the search can be carried out in a single step. As with the previous command files, after search is completed, Condor will return to the help menu. After ATHSRCH.CMD has been typed in, save it and open GENSRCH .CMD — our fourth command file.

The major difference between GENSRCH and the other two search command files we have written is the addition of a third variable, the one that will store the field name. This, as seen in Figure 1-6, is inputted by the user at the beginning of the command file and stored in $1. The UPDATE and DISPLAY statements, therefore, have two variables: $1 and $2. After the UPDATE or DISPLAY has been completed, GENSRCH returns the user — you guessed it! — to the REMark Index Menu. When finished typing in GENSRCH.CMD, save it and open SORTER.CMD. Both SORTER and REORGREM (the sixth command file), like ENTRMORE, do not make use of the Condor language. The two lines, displayed in Figure 1-7, of SORTER.CMD simply SORTs REMARK BY VOLUME, ISSUE, and PAGE and then returns to HELP REMARK. After completing SORTER, save it and open REORGREM.CMD. This file, depicted in Figure 1-8, also contains two lines. The first, begins the reorganization of the database. The second, invokes the help menu. Save REORGREM and then exit from your text editor — mission accomplished!

We can now take our completed database and return to Condor and try it out. To enter the REMark Index Main Menu, type HELP REMARK at the "B>>" prompt. Although the search commands are functional, I would strongly suggest that you enter a few articles first — searches generally work much better when there is data to be found!

A few closing comments: Due to the tremendous amount of disk access involved with Condor I recommend the use of a hard disk drive or better yet, a memory disk. A memory disk can be created through the use of the device driver MDISK .DVD found on the MS-DOS Version 2 distribution diskette. Documentation on using the memory disk can be found in the MS-DOS manual. When selecting a size for the disk, be sure to allow a minimum of 80k free for the Condor system. In addition, for anyone who is not thrilled with the idea of having to enter in all the past issues of REMark on disk, I have entered 87 articles (mostly dealing the Z-100 and Z-150) from October 1984 to present. If you would like to have my data file send me a disk with a self-addressed stamped envelope and five dollars. HAPPY COMPUTING!

```
;SUBSRCH.CMD
;
;
*MSG Enter subject you wish to search for
*GET $1
*MSG Do you wish to U(pdate) or D(isplay) data?
*GET $2
;
*IF $@ = u
*LET $2 = U
*ENDIF
;
*IF $2 = d
*LET $2 = D
*ENDIF
;
*IF $2 = U
UPDATE REMARK WHERE SUBJECT1 IS $1 OR SUBJECT2 IS $1 OR SUBJECT3 IS $1
UPDATE REMARK WHERE SUBJECT4 IS $1 OR SUBJECT5 IS $1 OR SUBJECT6 IS $1
UPDATE REMARK WHERE SUBJECT7 IS $1 OR SUBJECT8 IS $1
*ENDIF
;
*IF $2 = D
DISPLAY REMARK WHERE SUBJECT1 IS $1 OR SUBJECT2 IS $1 OR SUBJECT3 IS $1
DISPLAY REMARK WHERE SUBJECT4 IS $1 OR SUBJECT5 IS $1 OR SUBJECT6 IS $1
DISPLAY REMARK WHERE SUBJECT7 IS $1 OR SUBJECT8 IS $1
*ENDIF
;
HELP REMARK
```

Figure 1-4

```
;ATHSRCH.CMD
;
;
*MSG Enter name of author you wish to
        search for
*GET $1
*MSG Do you want to U(pdate) or D(isplay)?
*GET $2
;
*IF $2 = u
*LET $2 = U
*ENDIF
;
*IF $2 = d
*LET $2 = D
*ENDIF
;
*IF $2 = U
UPDATE REMARK WHERE AUTHOR IS $1
*ENDIF
;
*IF $2 = D
DISPLAY REMARK WHERE AUTHOR IS $1
*ENDIF
;
HELP REMARK
```

Figure 1-5

```
GENSRCH.CMD
;
;
*MSG Enter name of field to be searched
*GET $1
*MSG Enter data to be located
*GET $2
*MSG Do you want to U(pdate) or D(isplay)?
*GET $3
;
*IF $3 = u
*LET $3 = U
*ENDIF
;
*IF $3 = d
*LET $3 = D
*ENDIF
;
*IF $3 = U
UPDATE REMARK WHERE $1 IS $2
*ENDIF
;
*IF $3 = D
DISPLAY REMARK WHERE $1 IS $2
*ENDIF
;
HELP REMARK
```

Figure 1-6

```
;SORTER.CMD
;
;
SORT REMARK BY VOLUME,ISSUE,PAGE
HELP REMARK
```

Figure 1-7

```
;REORGREM.CMD
;
;
REORG REMARK
HELP REMARK
```

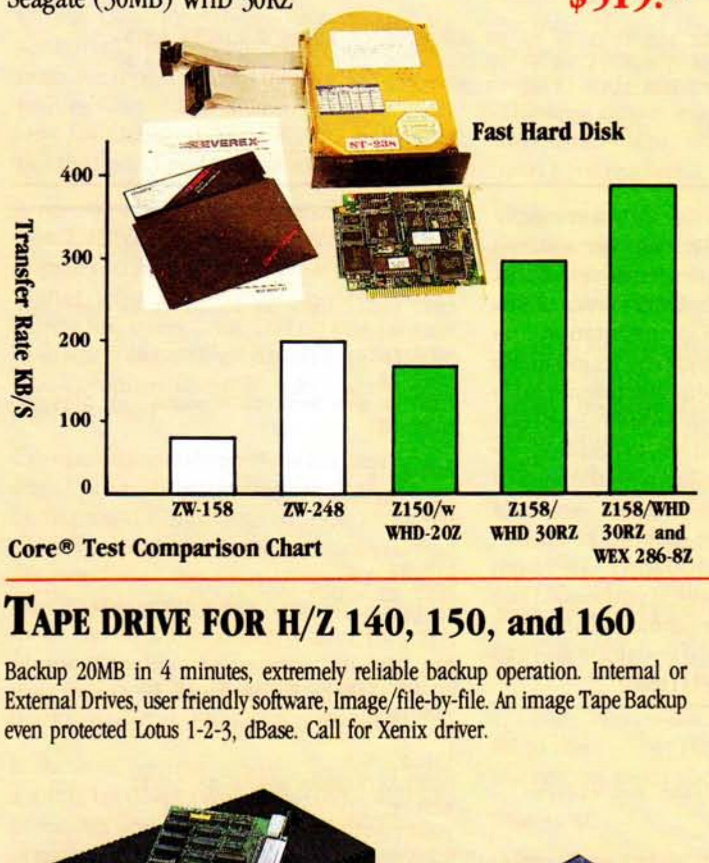
Figure 1-8

# WELCOME TO THE EVEREX WORLD OF H/Z HIGH QUALITY UPGRADE

## FOR H/Z-138, 148, 150, 158, and 160

Fully compatible with HS-317, supports all boot commands, boot directly from Hard Disk without seeking drive A. Extremely fast transfer! Up to 100% faster than ZW-248 and 500% faster than ZW-158.

| | |
|---|---|
| Seagate (20MB) WHD 20Z | $449.50 |
| Seagate (30MB) WHD 30RZ | $519.50 |

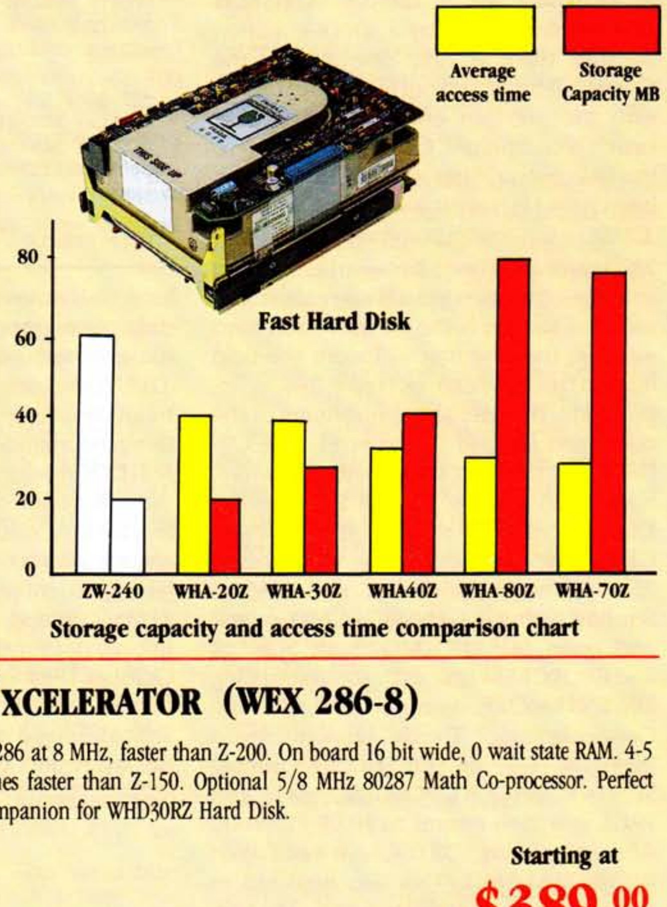


Core® Test Comparison Chart

## FOR H/Z-200, 248, and 300s

| | |
|---|---|
| WHA-20Z | $529.00 |
| WHA-30Z | $619.00 |
| WHA-40Z | $739.00 |
| WHA-80Z | $1199.00 |
| WHA-70Z | $1299.00 |

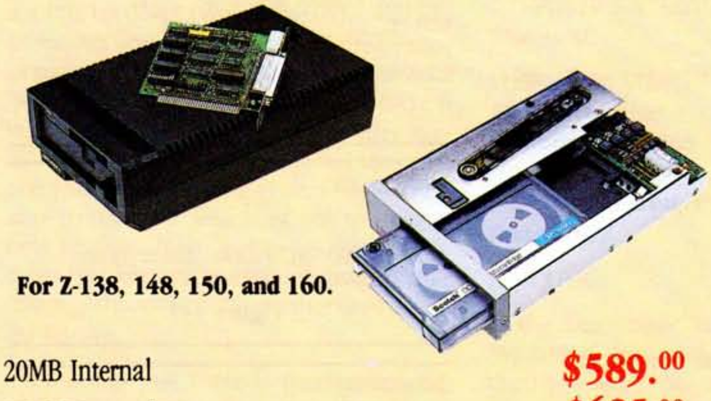
Up to 160% faster than ZW-240 in average access time. Up to 400% more storage (80MB). Runs MS-DOS, Xenix, Novell, and 3Com.



Storage capacity and access time comparison chart

## TAPE DRIVE FOR H/Z 140, 150, and 160

Backup 20MB in 4 minutes, extremely reliable backup operation. Internal or External Drives, user friendly software, Image/file-by-file. An image Tape Backup even protected Lotus 1-2-3, dBase. Call for Xenix driver.



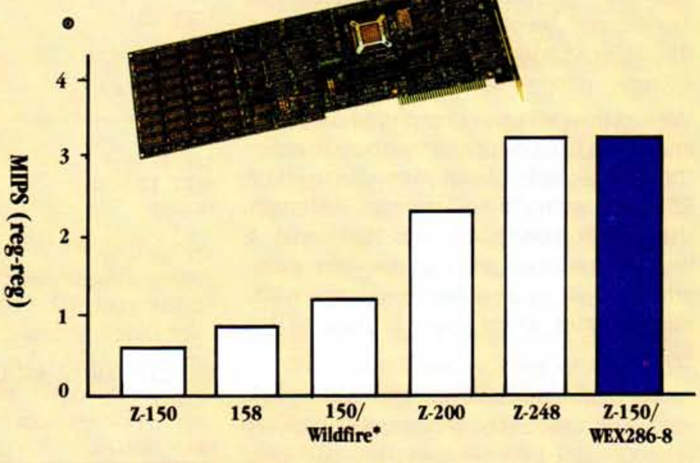
For Z-138, 148, 150, and 160.

| | |
|---|---|
| 20MB Internal | $589.00 |
| 20MB External | $625.00 |
| 60MB Internal | $769.00 |
| 60MB External | $849.00 |
| 60MB External Economical | $689.00 |

## EXCELERATOR (WEX 286-8)

80286 at 8 MHz, faster than Z-200. On board 16 bit wide, 0 wait state RAM. 4-5 times faster than Z-150. Optional 5/8 MHz 80287 Math Co-processor. Perfect companion for WHD30RZ Hard Disk.

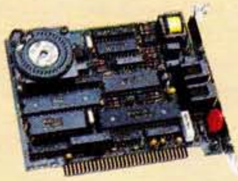
Starting at $389.00



CPU Performance Comparison Chart

★ Software Wizardry

# TOTAL HEATH/ZENITH COMPATIBILITY AND RELIABILITY BY WONG'S ADVANCED TECHNOLOGIES, INC.



## EVERCOM 300/1200

- ★ Half size
- ★ Configurable to COM 1 to COM 4
- ★ Fully Hayes compatible
- ★ BitCom communication software
- ★ Emulates VT-100, IBM 3101, VT-52
- ★ User definable
- ★ Powerful script commands
- ★ Stores up to 32K numbers with different communication parameters
- ★ Reliable even on noisy lines.

**$149**

## EVERCOM 1200/2400

**$259**

**Excellent for long distance call**
**FCC class 'B' approved**

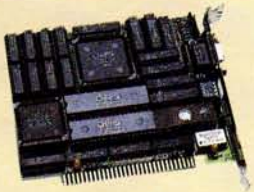
- ★ Half size,
- ★ Configurable to COM 1 to COM 4
- ★ Fully Hayes compatible
- ★ BitCom communication software
- ★ Emulates VT-100, IBM 3101, VT-52
- ★ User definable
- ★ Powerful script commands
- ★ Stores up to 32K numbers with different communication parameters
- ★ Reliable even on noisy lines.

## EVERGRAPHICS

- ★ Hercules 720 x 348 Monochrome graphics/64K
- ★ High Resolution text for W.P./Text editing
- ★ Spread sheet
- ★ High Resolution graphics for Acad, Lotus 1-2-3, and D.P.
- ★ Expandable to EGA

(TTL Monochrome monitor required)

**$125**



## MICRO ENHANCER

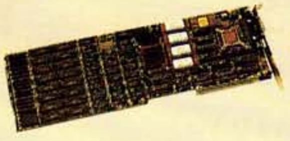
- ★ 640 x 350/16 colors from a pallette of 64
- ★ 256K RAM
- ★ Short card—5" long
- ★ W.P./color
- ★ Lotus 1-2-3, Acad, and M.S. window
- ★ Works with all 3 different monitors (EGA monitor is required to take full advantage of the EGA Mode.)

**$229**

## ENHANCER

- ★ 640 x 350/16 colors from a pallette of 64
- ★ 256K RAM
- ★ W.P./ color
- ★ Lotus 1-2-3, Acad, and window
- ★ Works with all 3 different monitors (EGA monitor is required to take full advantage of the EGA Mode.)
- ★ 2/3 size
- ★ With 1 parallel port—configurable as LPT 1-LPT 3

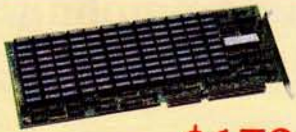
**$239**



## EPGA

- ★ Fully PGA compatible
- ★ 640 x 480/256 colors out of pallette of 4096
- ★ Emulates, HGA & CGA concurrently operating on one monitor
- ★ 8 MHz 80286/512K RAM/TI Graphics controller to perform 2D and 3D graphics commands. For example: lines, viewing transformations, and more at a very high speed.
- ★ For CAD engineering, Scientific and Educational (PGA monitor required)

**$950**

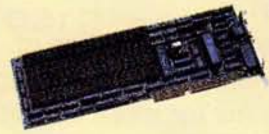


## RAM 3000

**RAM Board, ϕK $179**

- ★ Add up to 3MB of extended memory on 1 board, 5MB with piggy back RAM for Z-200/248
- ★ High speed, up to 12 MHz operation
- ★ Compatible with Novell, MS DOS 5.0, Xenix, and RAM disk for MS DOS 3.1
- ★ Free software

RAM Chips (installed) 1/2MB 150ns **$56.50**
RAM Chips (installed) 1/2MB 120ns **$62.50**
RAM Chips (installed) 1/2MB 100ns **CALL**



## MAGIC 16

**(Multifunction Card for Z-200)**

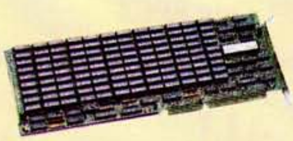
- ★ Add up to 2MB extended memory
- ★ 1 parallel, 1 serial port, optional 2nd serial
- ★ Free software

**With ϕK $195**

## MAXI MAGIC

- ★ Up to 2MB of expanded memory
- ★ For RAM disk
- ★ Print spooling
- ★ Run Lotus 1-2-3, MS Window and more
- ★ Up to 4MB with piggy-back board

**$119**



## EMS

- ★ 16 bit of Z-200

**CALL**

**ORDER TOLL FREE**
**(800) 626-1030**

**3201 Loyola Drive**
**Kenner, Louisiana 70065**

**INSIDE LOUISIANA**
**(504) 464-7379**

# WONG'S ADVANCED TECHNOLOGIES, INC.

**ORDER TOLL FREE**
**(800) 626-1030**

**3201 Loyola Drive**
**Kenner, Louisiana 70065**

**INSIDE LOUISIANA**
**(504) 464-7379**

star® micronics

- ★ Outstanding print quality and performance
- ★ High-speed draft and fine near-letter-quality printing
- ★ Front Control Panel for easy operation
- ★ Tractor and Friction feeds
- ★ IBM™ and Epson™ compatible
- ★ Full one-year Warranty

NB-15

| | |
|---|---|
| 100 cps NP-10 | ***$225*** |
| 100 cps NP-15 | ***$399*** |
| 120 cps NX-10 | ***$279*** |
| 120 cps NX-15 | ***$429*** |
| 180 cps ND-10 | ***$399*** |
| 180 cps ND-15 | ***$569*** |
| 216 cps NB-2415 (24 pin) | ***$849*** |
| 300 cps NB-15 (24 pin) | ***$1199*** |

## TATUNG TTL-MONOCHROME

- ★ Nonglare
- ★ 12" screen
- ★ 800 x 350 resolution
- ★ FCC class 'B' approved

**$129**
Amber or Green

**We also carry a full-line of Mitsubishi and Sony monitors. Please call for more information.**

## TATUNG EGA

- ★ 640 x 350
- ★ 640 x 200
- ★ EGA and CGA
- ★ FCC class 'B' approved

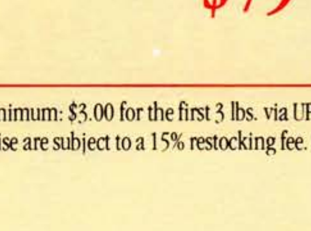
**$489**

## MITSUBISHI DIAMOND SCAN

- ★ 800H x 560V
- ★ Test Mode Switch
- ★ No Installation required
- ★ EGA, PGA, CGA, and HGZ compatible

**Perfect Match for Our EPGA**

**CALL**



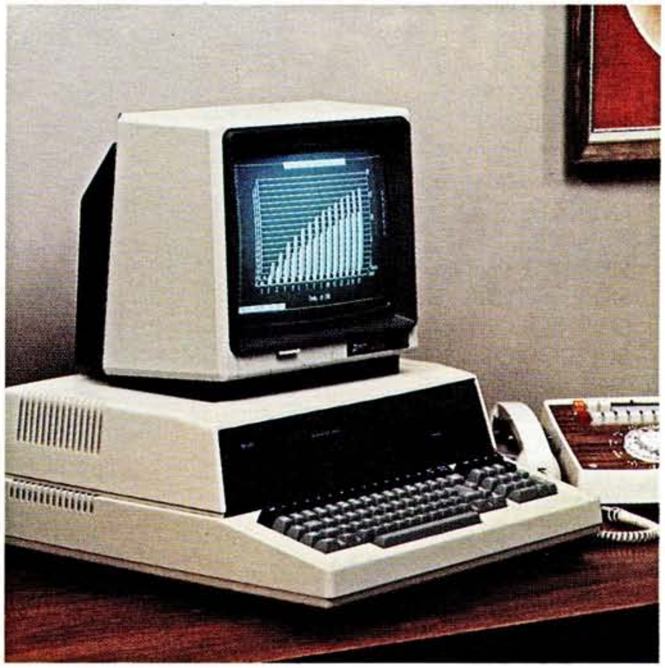
## M2 MOUSE

- ★ No slot needed, work through SIO
- ★ Removable Nylon Ball gives positive response
- ★ Works with all major softwares supporting a mouse
- ★ A must for Lotus, MS window and Acad

**$79**

We accept pre-paid, COD orders. Credit card purchases please call. Shipping charges minimum: $3.00 for the first 3 lbs. via UPS ground. Louisiana residents add 8% sales tax. No return merchandise will be accepted without an RMA number. Return merchandise are subject to a 15% restocking fee. Prices are subject to change without notice.

# ZPC Update #16



*Pat Swayne*
*HUG Software Engineer*

This is the sixteenth in a series of articles in support of ZPC, a program that allows you to run IBM PC software in H/Z-100 (dual processor) computers. ZPC is available from HUG as part no. 885-3037-37. An upgrade disk for ZPC is also available as part no. 885-3042-37.

In this episode of Update, I will present a new patch for QuickBASIC version 2, and patches for Microsoft Chart version 2.02 and Maxthink version 3.3. I will also provide an updated list of programs that I am aware of that can be run under ZPC.

### QuickBASIC Version 2

I made an error with the patch originally released for QuickBASIC version 2 in that I never tested it on a completely unmodified Z-100. Even with the Scottie Board removed from my system, it still has the video board jumper installed, which allows some programs to run that will hang on unmodified systems. So I missed some illegal port accesses in QuickBASIC that, sure enough, caused it to hang when I tried it on Nancy's Z-100 (the only other 768k Z-100 we have here at HUG headquarters). Here is a new patch for QuickBASIC that fixes the problem.

```
QUICKBASIC version 2
Insert the disk containing QB.EXE.
QB.EXE
7C63,CD,90
A6C5,90,90,90,90,90,90,90,90,90,90,90
A6E4,90,90,90,90,90,90,90,90,90,90,90
A74B,90,90,90,90,90,90,90,90,90,90,90
10B54,90,90,90,90,90,90,90,90,90,90,90
1C44B,CD,90
2647A,CD,90
z
```

Replace the patch for QuickBASIC in your PATCHER.DAT file with these lines, and then re-patch QuickBASIC.

### Microsoft Chart Version 2.02

To use Microsoft Chart version 2.02 under ZPC, you must patch it whether you have a ZHS board or not. To patch it, add the following lines to your PATCHER.DAT file.

```
MICROSOFT CHART version 2.02
Insert the disk containing CHART.COM
CHART.COM
65EA,0,0
65F1,B0
7872,90
7875,90
z
```

Patch the file by following the PATCHER instructions in your ZPC manual.

### Maxthink Version 3.3

Maxthink must be patched to run properly under ZPC. Even after it is patched, it does something unusual that I have not been able to figure out, but it is still usable. When it reads in and displays a text file on the screen, it uses DOS in such a way that ZPC is somehow bypassed. Therefore, the text will be in the Z-100 format even though you are in the PC mode. This only happens when Maxthink is displaying a text file, and at all other times it uses a screen output method that ZPC interprets correctly. To patch Maxthink, copy these lines to your PATCHER.DAT file, and then make the patch in the usual way.

```
MAXTHINK version 3.3
Insert the disk containing MAX.EXE and
  WALL.EXE.
MAX.EXE
4F7F,9,FF,75,5,B8,3,0,CD,10,90
4F9A,90
50E7,90,90,90,90,90,90,90,90,90,90,90
50FA,90,90,90,90,90
x
WALL.EXE
D7F,9,FF,75,5,B8,3,0,CD,10,90
D9A,90
E20,90,90,90,90,90,90,90,90,90,90,90
E33,90,90,90,90,90
z
```

### Programs That Run Under ZPC

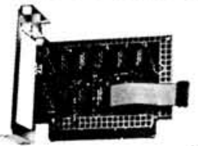
Below is a list of programs that can be run under ZPC Version 2 (with 768k of memory), as of 3-16-87. Below the list are some notes indicating what must be done, if anything, to get the programs to run.

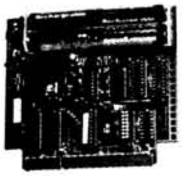
| Program: | See Notes: |
|---|---|
| BENCHMARK Word Processor vers. 4.4 | 1 |
| CHART MASTER v. 6.05 | 1 |
| CORNERSTONE database | 1 |
| DAC EASY ACCOUNTING | 1 |
| DBASE III version 1.1 | 2,3,5,7 |
| DBASE III + version 1.0 | 2,3,7 |
| DBASE III + version 1.1 | 4 |
| EDIX version 2.05 | 2,3,5 |
| EINSTEIN WRITER version 7.2 | 2,3,5 |
| ENABLE version 1.1 (8-8-85) | 2,3 |
| ENABLE version 2.0 (1-9-87) | 2,10 |
| FONTASY | 1 |
| FRAMEWORK version 1.1 | 2,3,7 |
| FRAMEWORK II version 1.0 | 2,3,7 |
| FRAMEWORK II version 1.1 | 2,4,7 |
| GENERIC CADD | 4 or 10 |
| GEM (Digital Research) | 10 |
| GW-BASIC (Zenith PC versions) | 2,3 or 4, 5 |
| Compiled PC GW-BASIC Programs | 1 |
| JAVELIN v. 1.1 (7-14-86 or 11-12-86) | 10 |
| LETTRIX | 1 |
| LOTUS 1-2-3 release 1A | 2,3 |
| LOTUS 1-2-3 release 2 | 7,9 |
| LOTUS SYMPHONY | 2,4,7,9 |
| MAXTHINK version 2.3 | 10 |
| MICROSOFT WORD version 1.1 (Zenith PC) | 2,3 |
| MICROSOFT WORD version 2.0 | 1 |
| MICROSOFT WORD version 3.0 | 4 |
| MICROSOFT CHART version 2.02 | 10 |
| MULTIMATE version 3.3 | 2,3 or 4 |
| MULTIPLAN version 1.2 (Zenith PC) | 2,3 |
| NORTON UTILITIES | 1 |
| PC FILE | 1 |
| PC PALETTE version 1.0 | 3 |
| PC TOOLS R1.03 | 2,11 |
| PC WRITE version 2.4 or 2.55 | 3 |
| PC WRITE version 2.6 | 10 |
| PFS FILE | 2,11 |
| PFS PLAN | 2,11 |
| PRINT MASTER | 1 |
| PRODESIGN II | 1 |
| Compiled QUICKBASIC (v. 1) Programs | 8 |
| QUICKBASIC version 2 | 6,10 |
| QUICK DOS version 1.21 | 10 |
| RUN/C | 1 |
| SIDEWAYS version 2.02, 3 | 1 |
| SIGN MASTER v. 5.05 | 1 |
| SUPERCALC3 version 2.0, 2.1 | 3 or 4,6 |
| SUPERCALC4 v. 1.00 (6-17-86, 10-24-86) | 10 |
| THINKTANK v. 2.10 | 1,7 |
| TURBO PASCAL (see inst. in ZPC manual) | 1 |
| TURBO GRAPHICS | 10 |
| TYPING TUTOR III | 2,11 |
| VOLKSWRITER DELUXE version 2.0 | 3 |
| WORD FINDER version 1 | 1 |
| WORD PERFECT version 4.1 | 2,3 or 4,5 |
| WRITING ASSISTANT | 2,11 |
| ZPAY II version 1.6 | 10 |

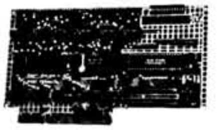
**Notes:**

1. Runs without any patches or hardware support with ZPC in the default mode of operation (color/graphics mode).
2. Runs without any patches or hardware support with ZPC in the monochrome card emulation mode (PC 7).
3. Runs after being patched with patches included with ZPC. Most programs in category 3 or 4 run in the color/graphics mode. Some also run in the monochrome mode.
4. Runs after being patched with patches included with the ZPC Upgrade disk.
5. Runs without patches if the ZPC Hardware Support circuitry is installed. Most programs in category 5 or 6 run in the color/graphics mode. Some also run in the monochrome card mode.
6. Runs without patches if the Improved ZPC Hardware Support circuitry is installed, and ZPC has been upgraded with the ZPC Upgrade disk.
7. Copy protection must be removed before you can run this program.
8. Requires a special patcher, supplied with ZPC.
9. Requires a special patcher, supplied with the ZPC Upgrade disk.
10. Patches for this program were published in REMark after the Upgrade disk came out.
11. This software may run in other configurations, but has not been tested in them.

**Disclaimer:** The above list is not a guarantee that a program will run under ZPC. Programs are subject to change by their publishers without notice, and several releases of a given version number may be produced. Patches are usually specific to a particular release of a particular version. *







International HUGCON87

HUG • Hilltop Road • St. Joseph, MI 49085 • (616) 982-3463

# 10th Anniversary Celebration!

## INTERNATIONAL HEATH/ZENITH USERS' GROUP CONFERENCE

## O'Hare Hyatt Regency
## Rosemont, Illinois
## August 21, 22, 23, 1987

**Name(s):** ______________________________

______________________________

______________________________

**Company:** ______________________________

**Address:** ______________________________

**City:** ______________ **State:** ________ **Zip:** ________

Enclosed is $27.00 for each of the individuals listed above to attend the International HUG Conference being held the weekend of August 21, 22, and 23, 1987. 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.

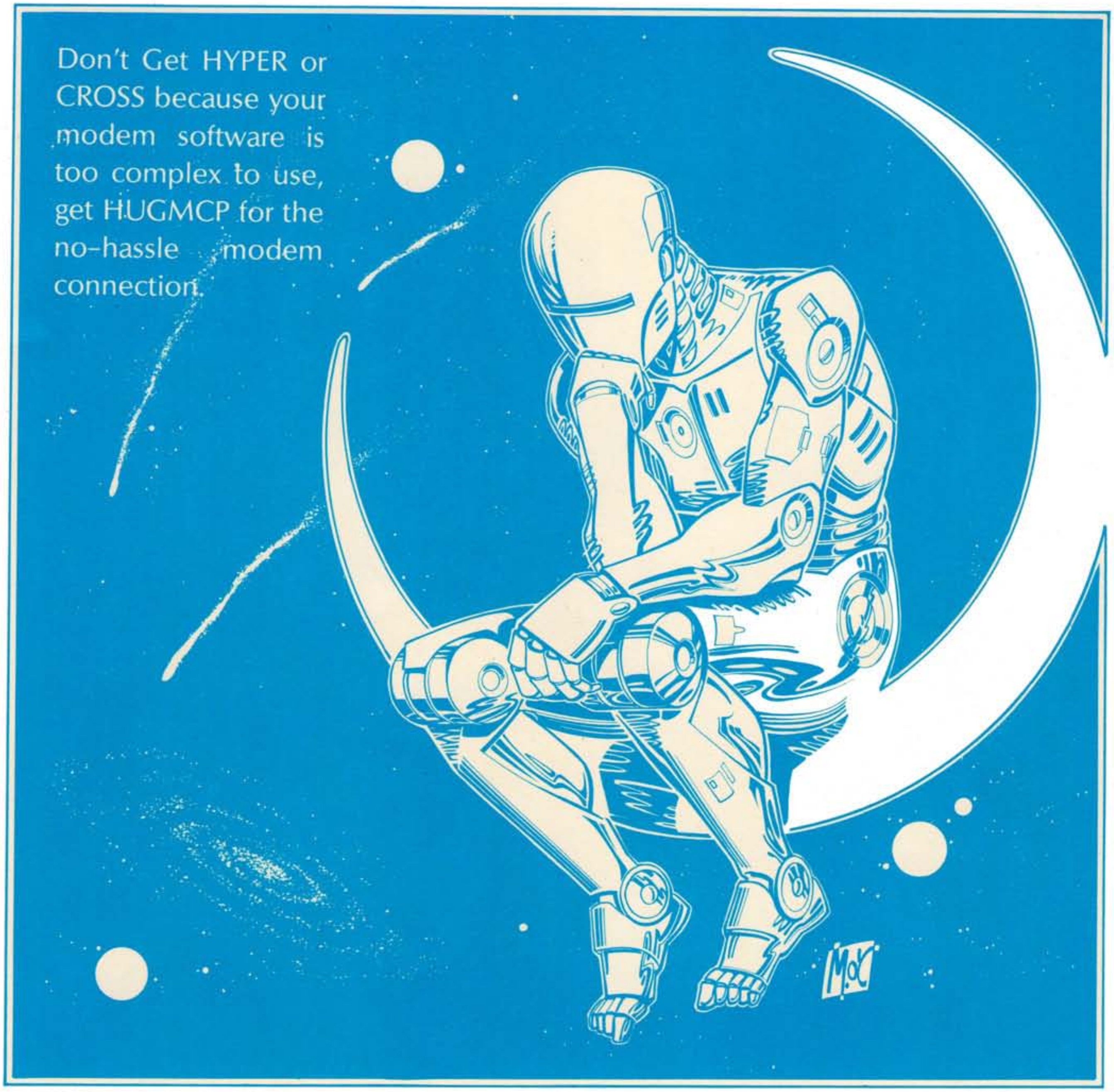
**For Your Information:**

The $27.00 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 $12.00. 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 31,1987. Cancellation will not be accepted after this date.**
**Sorry, We cannot accept purchase orders**


Don't Get HYPER or CROSS because your modem software is too complex to use, get HUGMCP for the no-hassle modem connection.

Heath/Zenith Users' Group

Hilltop Road
Saint Joseph, Michigan 49085