REMark®

Volume 9, Issue 2 • February 1988
P/N 885-2097 Issue 97

$2.50

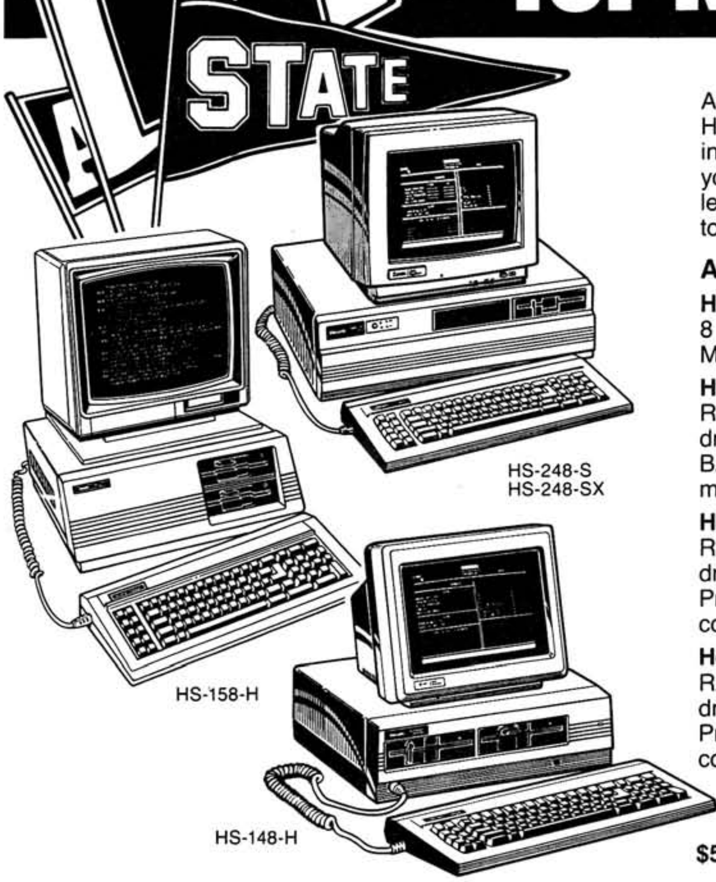
It's Contest Time!!
See Terry Perdue's Article
On Page 39

A Publication of Heath/Zenith

"Why Haven't You Called?"

"I've stayed up 24 hours a day waiting to hear from you. I know that for most of you, this may be your first time, but I promise, I'm very gentle! If you're still unsure, let me tell you what I have to offer. First, there's my message base. Here's where my users exchange ideas, receive assistance, and sell things. Next, is my database. Most of my users will tell you, it's second to none! With over 30 megabytes of hand-picked software it caters to all Heath/Zenith computers, but mostly PC compatibles. Finally, I have something new for you, my 'Bargain Centre'. Here you can buy surplus software and hardware, at unheard of prices. Interested? I hope so! Set your modem to either 300, 1200, or 2400 baud, and call me right now at (616) 982-3956, and register today. If you're still a bit shy, you can still call my human at (616) 982-3837 and register with him. Although he talks at 150 baud, he's gentle, too!"

BACK-TO-SCHOOL prices for kit computers



At Heath/Zenith Computers & Electronics, we have the Heathkit computer you need to start computing or to increase your computer performance. No matter what you want to build, we have it here. From our performance leader, the HS-248-SX, to our expandable 158 model, to our most affordable computer, the 148.

And look what you get!

Heathkit HS-248-S – AT-compatible. Includes 512K RAM, 8 MHz operating speed, zero wait states, 5¼" 1.2MB drive, MS-DOS, Diagnostics, GW BASIC. **ONLY $1599.95**

Heathkit HS-248-SX – AT-compatible. Includes 512K RAM, 8 MHz operating speed, zero wait states, 5¼" 1.2MB drive, 20MB Winchester kit, MS-DOS, Diagnostics, GW BASIC, Spreadsheet, Word Processor, Data Base, Communications, Winchester controller card. **ONLY $1999.95**

Heathkit HS-158-H – PC/XT-compatible. Includes 512K RAM, switchable 8 or 5 MHz operating speed, 5¼" 360K drive, MS-DOS 3.2, Diagnostics, Spreadsheet, Word Processor, Data Base, Communications, mono and color monitor capability. **ONLY $699.95**

Heathkit HS-148-H – PC-compatible. Includes 512K RAM, switchable 8 or 5 MHz operating speed, 5¼" 360K drive, MS-DOS, Diagnostics, Spreadsheet, Word Processor, Data Base, Communications, mono and color monitor capability. **ONLY $499.95**

FREE Heath/Zenith Partnership Pack support package available with computer purchase – a $500 Value. See your Heathkit Catalog for details.

HUG Discount Does *NOT* Apply!

Available NOW at Heath/Zenith Computers & Electronics Centers in the U.S.

ARIZONA – Phoenix, 85017
2727 W. Indian School Rd.
602-279-6247

Tucson, 85711
5616 E. Broadway
602-745-0744

CALIFORNIA – Anaheim, 92805
330 E. Ball Rd.
714-776-9420

San Jose (Campbell), 95008
2350 S. Bascom Ave.
408-377-8920

El Cerrito, 94530
6000 Potrero Ave.
415-236-8870

San Diego (La Mesa), 92041
8363 Center Dr.
619-461-0110

Los Angeles, 90007
2309 S. Flower St.
213-749-0261

Pomona, 91767
1555 N. Orange Grove Ave.
714-623-3543

Redwood City, 94063
2001 Middlefield Rd.
415-365-8155

Sacramento, 95825
1860 Fulton Ave.
916-486-1575

Woodland Hills, 91364
22504 Ventura Blvd.
818-883-0531

COLORADO – Denver (Westminster), 80003
8725 Sheridan Blvd.
303-429-2292

FLORIDA – Hialeah, 33012
4705 W. 16th Ave.
305-823-2280

Jacksonville, 32211
9426 Arlington Expressway
904-725-4554

Plantation, 33317
7173 W. Broward Blvd.
305-791-7300

Tampa, 33614
4019 W. Hillsborough Ave.
813-886-2541

GEORGIA – Atlanta, 30342
5285 Roswell Rd.
404-252-4341

HAWAII – Honolulu (Pearl City), 96782
98-1254 Kaahumanu St.
808-487-0029

ILLINOIS – Chicago, 60645
3466 W. Devon Ave.
312-583-3920

Downers Grove, 60515
224 Ogden Ave.
312-852-1304

INDIANA – Indianapolis, 46220
2112 E. 62nd. St.
317-257-4321

KANSAS – Kansas City (Mission), 66202
5960 Lamar Ave.
913-362-4486

KENTUCKY – Louisville, 40243
12401 Shelbyville Rd.
502-245-7811

LOUISIANA – New Orleans (Kenner), 70062
1900 Veterans Memorial Hwy.
504-467-6321

MARYLAND – Baltimore, 21234
1713 E. Joppa Rd.
301-661-4446

Rockville, 20852
5542 Nicholson Lane
301-881-5420

MASSACHUSETTS – Peabody, 01960
242 Andover St. (Rt. 114)
617-531-9330

Wellesley, 02181
165 Worcester Ave. (Rt. 9)
617-237-1510

MICHIGAN – Farmington Hills, 48018
29433 Orchard Lake Rd.
313-553-4171

East Detroit, 48021
18149 E. Eight Mile Rd.
313-772-0416

St. Joseph, 49085
2987 Lake Shore Drive
616-982-3215

MINNESOTA – Minneapolis (Hopkins), 55343
101 Shady Oak Rd.
612-938-6371

St. Paul, 55106
1645 White Bear Ave.
612-778-1211

MISSOURI – St. Louis (Bridgeton), 63044
3794 McKelvey Rd.
314-291-1850

NEBRASKA – Omaha, 68134
2311 N. 90th St.
402-391-2071

NEW JERSEY – Ocean, 07712
1013 State Hwy. 35
201-775-1231

Fair Lawn, 07410
35-07 Broadway (Rt. 4)
201-791-6935

NEW YORK – Amherst, 14226
3476 Sheridan Dr.
716-835-3090

Jericho, L.I. 11753
15 Jericho Turnpike
516-334-8181

Rochester, 14623
937 Jefferson Rd.
716-424-2560

N. White Plains, 10603
7 Reservoir Rd.
914-761-7690

NORTH CAROLINA – Greensboro, 27407
4620-C W. Market St.
919-299-5390

OHIO – Cincinnati (Springdale), 45246
131 West Kemper Rd.
513-671-1115

Cleveland, 44122
28100 Chagrin Blvd.
216-292-7553

Columbus, 43229
2500 Morse Rd.
614-475-7200

Toledo, 43615
48 S. Byrne Rd.
419-537-1887

OKLAHOMA – Oklahoma City, 73139
7409 South Western
405-632-6418

OREGON – Portland (Tigard), 97223
10115 S.W. Nimbus Ave.
Nimbus Center
503-684-1074

PENNSYLVANIA – Frazer, 19355
630 Lancaster Pike (Rt. 30)
215-647-5555

Philadelphia, 19149
6318 Roosevelt Blvd.
215-288-0180

Pittsburgh, 15235
3482 Wm. Penn Hwy.
412-824-3564

RHODE ISLAND – Warwick, 02886
558 Greenwich Ave.
401-738-5150

TEXAS – Dallas, 75218
12022C Garland Rd.
214-327-4835

Fort Worth, 76116
6825-A Green Oaks Rd.
817-737-8822

Houston, 77008
1704 W. Loop N.
713-869-5263

San Antonio, 78216
7111 Blanco Rd.
512-341-8876

UTAH – Salt Lake City (Midvale), 84047
58 East 7200 South
801-566-4626

VIRGINIA – Alexandria, 22303
6201 Richmond Hwy.
703-765-5515

Virginia Beach, 23455
1055 Independence Blvd.
804-460-0997

WASHINGTON – Seattle, 98109
505 8th Ave. N.
206-682-2172

WISCONSIN – Milwaukee (Wauwatosa), 53226
845 N. Mayfair
414-453-1161

Phone orders accepted.

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

Heath® ZENITH®

Units of Veritechnology Electronics Corporation

Computers & Electronics

HZC-341

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 M. 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
P.O. Box 217
Benton Harbor, MI 49022
(616) 982-3838

REMark®

Official magazine for users of HEATH ZENITH computer equipment.

Volume 9, Issue 2 • February 1988

Index of Advertisers

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

Reader Service No.		Page No.
150	Dante Bencivengo	52
***	EZPLOT	83
104	FBE Research Co., Inc.	52
105	First Capitol Computer	24
137	Jay Gold Software	46
107	Paul F. Herman	37
***	HUG PBBS	2
***	HUG Authors	62
***	Heath Company	8,9
***	Heath Education	50
***	HEPCAT	76
111	KEA Systems, Ltd.	32,61
114	Micronics Technology	52
117	Payload Computer Services	20
132	RAM Technology	38
119	S&K Technology, Inc.	23
122	Secured Computer Systems	68
123	Software Applic. of Wichita	32
105	Software Wizardry	19
161	Kirk L. Thompson	16
***	Veritechnology Elec. Corp.	3

On The Cover: *In February of 1978, the first Issue of REMark came out. This being the February 1988 Issue, we are celebrating ten years of REMark. We hope we've met our objectives over the last ten years of fostering the exchange of ideas to enhance the usage of Heath/Zenith equipment.*

BUGGIN' HUG

From The Staunch 8/89'er

Dear HUG:

The Staunch 8/89'er, the quality quarterly newsletter for H-8/89/90 computer users announces four changes for the coming year. First, subscriptions are available at $8 for the 4-issue year, effective December 1, 1987; overseas, please add $2.

Second, beginning January 1, 1988, Staunch will pay authors $50 for each article submission between 1,000 and 2,000 words. Write to the address below for a writer's quide. Enclose a self-addressed, stamped envelope.

Third, Staunch will accept paid commercial advertising from 8-bit-oriented vendors. Advertising will be a 2- or 4-page insert, bringing Staunch's size to 12 pages, maximum. Ad rates will be $50 per full-page, $25 per half-page. Please send inquiries and camera-ready copy to the address below.

Fourth, Hank Lotz and Kirk Thompson, who have been putting out Staunch for the past year, will swap roles; Kirk will assume the editorship with Hank as contributing editor. The Jan/Feb/Mar issue, #6, will be mailed first-class in February. All subscriptions remain aligned with the calendar year.

Send orders and inquiries to:

Kirk L. Thompson
#6 West Branch Mobile Home Village
West Branch, IA 52358

Make checks payable to "Kirk L. Thompson".

WORD, FANSI-CONSOLE Follow-up

Dear HUG:

Shortly after I wrote a letter to HUG concerning an incompatibility between Microsoft WORD 3.1 and FANSI-console 2.00H, I upgraded to WORD 4.0. The incompatibility is gone. While running WORD 4.0 in the graphics mode, you can call Library Spell, which comes up in the color mode, but returns you to WORD in the graphics mode, just as you left it. WORD 3.1 would be returned in the color mode, and no way to change it without restarting the program.

WORD 4.0 is truly a marvel. It is *much* faster than ever before; has built-in macro capability; can generate forms; omit the window around the text; toggle between graphics and text modes at will; change the colors of window background and text; retrieve documents by keywords, titles, date of creation or revision; allows replacement of character formats; creates style sheets by example; and has a *slew* of "hot" keys to shortcut menus.

WORD 4.0 does have some minor conflicts with Superkey that did not exist with WORD 3.1, and is still not as convenient as, say, PFS:Professional Write for editing ASCII files, but for really serious document handling, printer control, and formatting strength, WORD 4.0 is pretty near unbeatable.

Yours truly,

Robert E. Hawkins
Consulting Engineer
P.O. Box 4533
811 Highway #1 South
Greenville, MS 38704-4533

TREECOPY Bug

Dear HUG:

I wish to report a bug in the TREECOPY program supplied in the MS-DOS Version 3 Programmer's Utility Pack (catalog no. CB-3163-30).

In my copy of this program (Version 3.00; the file is dated 9-19-85 at 8:40 am, and is 6599 bytes long), the code at locations 100C to 1027 is incorrect. To see if you have the incorrect version, install a diskette containing copies (NOT the originals) of DEBUG.COM and TREECOPY .COM, then type DEBUG TREECOPY .COM and return; now type d100c 1027 and return. If the display shows the following:

```
          06 52 B0 23 B4 35 CD 21 89 1E 7E
0F-8C 06 80 0F BA 2E 15 B4 25 CD 21 5A 07
 A3 6C 0E
```

Then you have the bad version. Alternatively, you can type u100c 1027 and return; if the display shows:

```
PUSH    ES
PUSH    DX
MOV     AL,23
MOV     AH,35
INT     21
MOV     [0F7E],BX
MOV     [0F80],ES
MOV     DX,152E
MOV     AH,25
INT     21
POP     DX
POP     ES
MOV     [0E6C],AX
```

Then your version is incorrect.

To fix your working copy (DON'T do this to your original distribution copy), type:

```
a1006
mov [0e6c],ax
push es
push bx
mov al,23
mov ah,35
int 21
mov [07fe],bx
mov [0f80],es
mov dx,152e
mov ah,25
int 21
pop bx
pop es

w
q
```

(On the third to the last line, simply type return; note that I have not shown what you see on the screen, only what you should actually type — each line ends with a return).

The effect of this patch is to move the "mov [0e6c],ax" from its original position at the end of this code fragment to the beginning, and to change the "push dx/ pop dx" pair to "push bx/pop bx". This corrects errors in both the location and the size of TREECOPY's internal read/ write buffer. Depending on the amount of memory you have and your software configuration, you may or may not have noticed the effects of the errors, but if you have installed VDISK or MDISK, the uncorrected version of TREECOPY will write over unpredictable locations on the memory disk, even if you are copying from A: to B:. In my case, I was using the memory disk to store executable programs (editors, compilers, etc.), and the resulting system crashes when I tried to run those programs had me worried for a while!

Yours sincerely,

Richard L. Ferch
1267 Marygrove Circle
Ottawa, Ontario
CANADA K2C 2E1

Changing Directories In WordStar

Dear HUG:

I've seen several letters in REMark that offered solutions to the problem of changing directories in WordStar. I didn't study these recommendations very seriously at first, because I didn't have a fixed disk.

Now that I have a fixed disk and have set up directories, I see quite clearly that there is a problem with keeping WordStar program files in one directory and the text files in another.

After going back through all the old issues and trying the suggestions and programs to correct this inexcusable problem with WordStar, I found that none of the "solutions" was allowing me to run the program the way I wanted to run it.

My goal was to be able to boot my computer (a '148), and from the root directory, be able to type "WS <filename>" and have WordStar load from its subdirectory and log onto the text files in their subdirectory so they would be displayed in the opening WordStar menu without any further "logging" or other foolishness. If I specified a <filename> I wanted it open just as if I had typed the request in the WordStar directory itself. When my WordStar run was over, I wanted to end up back in the root directory with everything restored to its initial condition. Additionally, if I entered WSP <filename> from the root directory, I wanted to run my version of "parallel-printer-configured" WordStar in exactly the same manner.

Additionally, I wanted the performance of CorrectStar and StarIndex unchanged by the environment I set up. For example, I wanted to be able to run CorrectStar from the WordStar menu and have it be able to find not only the target file, but its overlays, as well.

I set up my directories like this:

```
ROOT
  :
  :
WORDSTAR <DIR>  Contains all .EXE,
                .COM, .OVL, etc.
                files
    DICT <DIR>  Contains the WS
                Professional dict files
TEXT <DIR>      Contains all text files
                for WordStar
  :
ALL OTHER DIRECTORIES
```

WORDSTAR and TEXT are first level subdirectories of the root directory. DICT is a subdirectory of the WORDSTAR directory. For this method to consistently work properly, the TEXT <DIR> must NOT be a subdirectory in the WORDSTAR <DIR>. This is unfortunate, but the method doesn't work well any other way.

I only keep COMMAND.COM, .SYS, and .BAT files in my root directory. I rely on the PATH command to allow me to access most of my transient commands (programs) from the root directory.

This short file is in my root directory under the name "WS.BAT". It gets me into the WORDSTAR directory and invokes SWS .BAT which resides there and passes any <filename> I may have typed in.

```
REM WS.BAT
CD WORDSTAR
SES %1
```

This next file, SWS.BAT (stands for Serial WordStar, in my case), is in the WORDSTAR directory and is called by the previous batch program. Explanations of what this file does are below the command line.

```
REM SWS.BAT
```

Always a good idea to title your files.

```
mode com1:9600,n,8,1,P
```

This invokes the mode program to set DOS for my printer. In this and the next two lines make sure you have a path to MODE and SUBST or those files won't be found. If you don't know how to do that, just put the programs MODE and SUBST in your WORDSTAR directory.

```
mode lpt1:=com1:
```

Tells DOS that I want to direct my output to the com port.

```
subst d: c:\text
```

Invokes the SUBST program to fool MS-DOS into thinking my text files are on drive D:.

```
path c:\wordstar
```

Makes sure that we can find all the necessary .OVL files in the WORDSTAR directory.

```
d:
```

Log onto drive D:.

```
ws %1
```

Call WordStar, this is why we used the PATH command above.

```
c:
```

This executes when we LEAVE WordStar, it logs us back onto drive C:.

```
path \bin;\utility;\maint;\copy;
  \norton;\wordstar
```

This restores the PATH that I use in my AUTOEXEC.BAT file that executes when I boot up. Your paths are probably different, if you don't use this command at all, just say "pathX" here.

```
subst d: /d
```

Deletes my substitution of drive "D:".

```
mode lpt1:
```

Gets me back to using my parallel printer; my default setup.

```
cd\
```

Puts me back in the root directory where all this started.

```
REM Batch file terminated successfully
```

I like to know when batch files end properly.

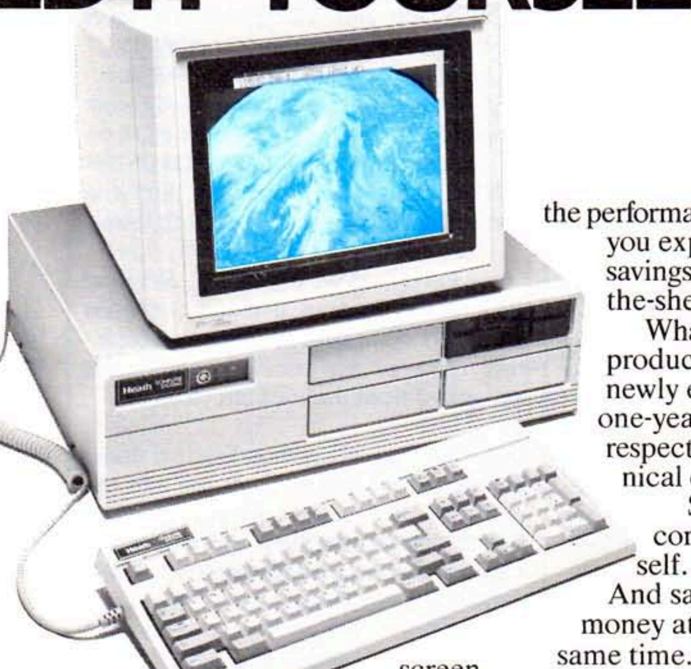
If from the root directory I should type WSP <filename>, the following batch files execute to run my parallel-printer-configured version of WordStar. These are offered without comment, since they are similar to the ones already explained.

```
REM WSP.BAT
CD WORDSTAR
PWS %1
```

Once again, this next .BAT file is in the WORDSTAR directory.

```
REM PWS.BAT
mode lpt1:
SUBST D: C:\TEXT
path c:\wordstar
D:
C:wsp %1
C:
path \bin;\utility;\maint;\copy;
  \norton;\wordstar
```

IF YOU WANT A BETTER COMPUTER BUILD IT YOURSELF.



Heath makes building the powerful 386 computer quick, fun, easy... and a great value.

Now you can own the fastest, most powerful home computer available today - at an affordable cost. Just by building it yourself.

Introducing the Heathkit H-386 Desktop Computer. With its powerful 32-bit processor, 16 MHz computing speed and "zero wait" technology,

the H-386 can breeze through complex calculations in seconds.

Every INTEL 80386 microprocessor used in our H-386 is 100% tested for all functions. And you get superb graphics because one video port provides dazzling 640-by-480 color on Zenith's new flat screen monitor and another port drives EGA, CGA and TTL monochrome monitors. Both ports automatically emulate common video formats for easy system configuration.

Designed for people using large spreadsheets, CAD/CAM or other computation-intensive applications, or anyone who simply wants to own the newest, most powerful hardware on the market, the Heathkit H-386 can be assembled easily and quickly. One or two evenings is all it takes, and no special tools or equipment are required.

In the bargain, you get the satisfaction of having built a powerful computer system all by yourself, and the confidence that this exciting product will deliver all the performance and dependability you expect. At a significant savings over comparable off-the-shelf brands.

What's more, all Heathkit products are backed by a newly extended, limited one-year warranty, highly respected manuals and technical consultation service.

So if you want a better computer, build it yourself. Impress your friends. And save money at the same time.

To order the new Heathkit H-386 Desktop Computer, simply call toll-free **1-800-253-0570. Ask for operator 611.** Use your Visa, MasterCard or Heath Revolving Charge. Or call **616-982-3614** for the Heath/Zenith Computers and Electronics store location nearest you.

For more information on all our quality kits, send now for your free four-color Heathkit catalog. **Write Heath Company, Dept. 016-594, Benton Harbor, MI 49022.**

Prices, product availability and specifications are subject to change without notice.

Heathkit®
Heath Company

```
subst D: /D
cd\
REM Batch file terminated successfully
```

I hope that you can adapt these files to your particular situation. They have eliminated the frustration I initially felt when first running WordStar on a hard disk.

Sincerely,

Jim Meyers
2010 Culpepper Drive
Brandon, FL 33511

CP/EMulator II/ZEMulator Bug

Dear HUG:

Several weeks ago, I purchased a copy of CP/EMulator II and ZEMulator (HUG p/n 885-6002-37) and have uncovered a bug in the program ZEM.COM. I'd observed that pressing the keypad 0 and 1 keys both issued the same code sequence. Suspecting a key mapping problem, I inspected the ZEM.ASM file and indeed found the cause in one of the mapping tables, as shown here:

```
;       ALTERNATE KEYPAD TABLE

ALTPTBL LABEL   BYTE
        DB      1BH,'?n',0      ;.
        DB      '/',0,0,0       ;/
        DB      1BH,'?p',0      ;0
        DB      1BH,'?p',0      ;1
```

The fix was to replace the second '?p' with '?q'. This can be done by patching the ZEM.COM file using MSDOS' DEBUG. Enter the following keystrokes:

```
debug zem.com
carriage return
e0798
carriage return
space bar
space bar
space bar
space bar
71
carriage return
w
q
```

With the above fix in place, everything seems to work well. The only exception I have encountered is MYCHESS, which signs on and then locks up. I want to thank Pat Swayne and Robert Metz for this emulator package. In my particular case, it has turned out to be a most useful piece of software.

Sincerely,

Ray Kilmanas
P.O. Box 262
Trenton, MI 48183

Need A Beginner's Column

Dear HUG:

I would like to share a problem I have with REMark.

I have time in my busy practice to use a computer only 1-2 hours a week, and obviously in that brief time, I cannot become proficient quickly. Even after many months of such limited use, I experience enormous frustration because of my inability to understand and perform simple operations. I skim the magazine month after month to find articles which may help the novice user, but find few or frequently none in a given issue. Often I cannot even understand what the more complex articles are about.

I know I'm not stupid (having two graduate degrees), but the world of computers is strange and alien to me, though I try hard to learn about it.

I need help, for example, with simple things, like what are the different ways the BACKUP command can be used, what happens to the data, and why can't you always get RESTORE to function properly.

In short, we need a Novice Column — there has to be other people like me.

I know the standard advice REMark might give to me — reread a book or MSDOS — I have read three of them and they are usefully inadequate in explaining things simply or clearly relative to things like BACKUP. The poverty of differing examples to cover different situations is particularly frustrating.

I would be most interested in any help you can give to the thousands of us who are non-expert, but interested future computer users.

P.S. I have a Z-151.

ED: Any takers on a Beginner's Column?

Raymond Moon's Helpful Tip

Dear HUG:

Raymond Moon's letter in the September 1987 REMark contained a very helpful tip

TALK IS CHEAP.

Have you heard? For less than $90 your AT or XT-compatible computer can talk! All it needs is the HV-2000 Computer Voice Kit from Heathkit.

Reading letters, transcriptions and computerized instruction can be easier and quicker than you ever thought possible. Computer games gain a new dimension. Your computer can even entertain children with stories and songs.

If you have a modem, the HV-2000 Computer Voice will allow your computer to recite reference and research information from time-sharing services. Or, speak radio transmitted ASCII information.

The HV-2000 Computer Voice Card, containing speech synthesizer and audio amplifier, plugs into any AT or XT-compatible computer's expansion slot. An external speaker is also included. Versatile, Heath-developed software gives you a wide variety of voices and easy interface to high and low level languages.

The HV-2000 Computer Voice. At less than $90, talk IS cheap. To order, call toll-free **1-800-253-0570. Ask for operator 611.** Use your Visa, MasterCard or Heath Revolving Charge card. Or call **616-982-3614** for the nearest store location.

Heathkit®
Heath Company
Benton Harbor, MI 49022

Prices, product availability and specifications are subject to change without notice.

c 1987, Heath Company

for expanding the DOS environment space using the SHELL command in the CONFIG.SYS file. I have used the suggestion successfully under Zenith MS-DOS version 3.10 and, as a result, have been able to expand my PATH to include several additional subdirectories.

I am writing to advise reader's about two qualifications to the tip: First, the tip does not run under all versions of DOS 3.X. Specifically, I could not get it to work on an IBM AT with PC-DOS 3.0. I don't know if this is a quirk of PC-DOS 3.X or of the 3.0 version and, since the tip is based on an undocumented feature, don't know how to check.

Second, the tip does not work under Zenith MS-DOS 3.10 if the SHELL command is the first line in the CONFIG.SYS file, but will work when the SHELL command is at the end of CONFIG.SYS. This information may save other users some time, since the MS-DOS manual does not indicate that the SHELL command should be last.

Sincerely,

Anthony Rodrigues
2255 Barker Avenue, Apt. 4G
Bronx, NY 10467

Checking The Validity Of Dates

Dear HUG:

Here is a subroutine I have found very useful in checking the validity of dates. I first developed this for use with dBASEII, and then made a version for use with BASIC. The routine will check the date for the maximum number of days for that month and if the year is in the 1900s, it will give the day of the week of the first day of the month. An error will be returned if an invalid date is entered.

```
10 ' FIRSTMON < By Pete Crossman 1985 >
20 '
30 '  A SUBROUTINE TO CHECK FOR A VALID
40 '  MONTH AND CALCULATE THE FIRST DAY
50 '  OF THE MONTH AS WELL AS THE MAXI-
60 '  MUM NUMBER OF DAYS IN THE MONTH
70 ' INPUT: M9$-YEAR AND MONTH yy/mm/dd
72 '    (Note: year must be in the 1900s)
80 ' OUTPUT: W9-WEEKDAY OF FIRST DAY OF
      THE MONTH (1-7)
90 '    (SUNDAY = 1) (0 IF INVALID DATE)
100 '    M9-MAXIMUM DAYS IN MONTH
110 '
120 '    VARIABLES:
130 '
140 '    B9-BASE YEAR (84)
150 '    C9-COUNTER
160 '    F9-FIRST DAY OF BASE YEAR
170 '    L9-LEAP YEAR (0=NO, 1=YES)
180 '     M9-MAX DAYS IN MONTH (OUTPUT
            VARIABLE)
190 '     M9-YEAR AND MONTH (yy/mm/dd)
            (INPUT VARIABLE)
200 '     N8-NUMERIC VALUE OF DAY
210 '     N9-NUMERIC VALUE OF MONTH
220 '     W9-NUMERIC FIRST DAY OF MONTH
            (OUTPUT VARIABLE)
230 '     Y9-NUMERIC VALUE OF YEAR
240 '
250 ' ----------------------
260 Y9 = VAL(MID$(M9$,1,2))
270 IF Y9 < 0 GOTO 780 ' INVALID YEAR
280 N8 = VAL(MID$(M9$,7,2))
290 IF INT(Y9/4) = Y9/4 THEN L9 = 1
      ELSE L9 = 0 ' TEST FOR LEAP YEAR
300 ' USE 1984 AS THE BASE YEAR
310 '
320 F9 = 1  ' FIRST DAY OF 1984 WAS A
          SUNDAY
330 B9 = 84 ' THE BASE YEAR IS 84
340 '
350 IF Y9 = B9 GOTO 500
360 IF Y9 < B9 GOTO 420
370    IF INT(B9/4) = B9/4 THEN C9 = 2
         ELSE C9 = 1
380    F9 = F9 + C9 ' CORRECT FOR
         CURRENT YEAR
390    IF F9 > 7 THEN F9 = F9 - 7 ' FIRST
         DAY OF THE NEXT YEAR
400    B9 = B9 + 1
410    GOTO 350
420 ' USE THIS CODE IF THE YEAR IS PRIOR
      TO THE BASE YEAR
430 '
440 B9 = B9 -1 ' MOVE BACK ONE YEAR
450 IF INT(B9/4) = B9/4 THEN C9 = 2
      ELSE C9 = 1
460 F9 = F9 - C9 ' CORRECT FOR CURRENT
      YEAR
470 IF F9 < 1 THEN F9 = F9 + 7 ' FIRST
      DAY OF THE NEXT YEAR
480 GOTO 350
490 '
500 ' NOW WE HAVE THE FIRST DAY TO THE
      TARGET YEAR
510 '
520 N9 = VAL(MID$(M9$,4,2)) ' GET
      NUMERIC VALUE OF THE MONTH
530 IF N9 < 1 OR N9 > 12 GOTO 780 '
      INVALID MONTH
540 C9 = 1 ' SET TO JANUARY
550 W9 = F9 ' INITIALIZE DAY OF MONTH
      TO THE FIRST DAY OF YEAR
560 IF C9 >= N9 GOTO 630 ' ADD IN THE
      NUMBER OF DAYS FOR EACH MONTH
570    IF C9 = 1 OR C9 = 3 OR C9 = 5
         OR C9 = 7 OR C9 = 8
572    IF C9 = 10 OR C9 =12 THEN W9 = W9
         + 31
580    IF C9 = 2 AND L9 = 1 THEN W9 = W9
         + 29
590    IF C9 = 2 AND L9 = 0 THEN W9 = W9
         + 28
600    IF C9 = 4 OR C9 = 6 OR C9 = 9 OR
         C9 = 11 THEN W9 = W9 + 30
610    C9 = C9 + 1 ' BUMP TO NEXT MONTH
620    GOTO 560
630 '
640 W9 = W9 -(INT((W9-1)/7)*7)
650 '
660 ' NOW SET THE MAX NUMBER OF DAYS
      FOR THE MONTH
670 '
680 IF C9 = 1 OR C9 = 3 OR C9 = 5 OR
      C9 = 7 OR C9 = 8
682 IF C9 = 10 OR C9 = 12 THEN M9 = 31
690 IF C9 = 2 AND L9 = 1 THEN M9 = 29
700 IF C9 = 2 AND L9 = 0 THEN M9 = 28
710 IF C9 = 4 OR C9 = 6 OR C9 = 9 OR
      C9 = 11 THEN M9 = 30
720 '
730 IF N8 < 1 OR N8 > M9 GOTO 780 '
      INVALID DATE
740 '
750 ' THAT'S ALL NOW JUST RETURN
760 RETURN
770 '
780 ' COME TO HERE IF INVALID DATE
790 '
800 W9 = 0 ' SET ERROR FLAG
810 RETURN
```

Pete Crossman
RR1, Box 106AA
Trenton, IL 62293

Bug In WordPerfect Vers. 4.2

Dear HUG:

I was amazed to find a bug with WordPerfect Version 4.2 on my Zenith '148 after using WordPerfect V4.1 without problems. The solution may be of interest to other users.

The clock always reset to 2.00 am as soon as WordPerfect was loaded, although the date stayed correct. This is great if you want people to think that you do all your work at night. The bug occurred on several Z-148s and all versions of DOS, but never on other PCs or compatibles, including a Z-158. Neither Zenith nor WordPerfect Corporation could help. Zenith could not reproduce the problem on their Z-148 which had V2.9 bootup PROM, but it showed up on the local Heathkit dealer's demonstration Z-148s. The only difference seemed to be that the Z-148s with the bug had version 2.2 boot up PROM. You can tell which version you have by booting up using Ctl-Alt-Ins and reading the version number in the MFM-140 control monitor's message.

The local dealer sold me a V2.9 PROM for under $20 (Canadian) which cured the bug. Fitting the new PROM was quite easy for anyone with electronics experience. It involved no soldering, but the disk drives had to be removed to reach the socket. These come out as a unit, but mark the ribbon cables, as some can be reconnected backwards. Remember to use SHIP first if you have a hard disk. The only problem was retightening that stupid little screw deep down in the narrow chasm between the power supply and the disk frame. Zenith, how could anyone have approved that design!

Continued on Page 44

On The Leading Edge

William M. Adney
P. O. Box 531655
Grand Prairie, TX 75053-1655

Saving Money On Hard Disks, Disk Interleave, Subdirectory Names

One of the hazards of making written plans and lists is that, if you lose the plan or list, you are in deep trouble. I usually plan various articles about six months to a year in advance, and if I lose that master plan, it's difficult to remember exactly where I was. Well, I did that. My master plan disappeared for a couple of months, and I could not remember all of the items that were planned for future articles. Although I usually keep my plans in a folder specifically for that purpose, I somehow misplaced the latest list. Many of you have been asking about my comment on saving money on a hard disk. The idea is to revisit some information that was presented last year and carefully study the implications of it. It begins with how DOS allocates disk space for files

How DOS Allocates File Space

Virtually all computer systems, mainframe to micros, allocate space on magnetic media in terms of a block. A block can have any number of specific definitions depending on the type of media (i.e. tape or disk). Because of that, a BLOCK is most generally defined as a group of bytes that are handled as a single unit. Although there are ways to get around the handling of blocks in a mainframe computer system (e.g. an unblocked tape), all operating systems write and read data on magnetic media in blocks.

To complicate matters somewhat, another term is used in DOS-based systems called a sector. Despite some protestations to the contrary, the standard SECTOR is defined as 512 bytes in DOS. Although it is true that you can have other sector sizes in DOS, 512 bytes is the standard. In case you are wondering why you would want a different sector size, one reason is that you can do cute things with copy-protection by using different sector sizes. Odd sector sizes (1024 and 128 bytes) are also used in Z-100 systems for 8" disks. And if you use the VDISK device driver to create an electronic virtual disk, you can also specify the sector size if you wish.

DOS allocates disk space in "blocks" of sectors, and these blocks are called CLUSTERS. In a standard 5.25" 9 sector, 360 KB floppy disk that you probably use, there are two sectors (1024 bytes) per cluster. The real point is that, for any file less than 1024 bytes in this specific format, the file still requires the two sectors, which can "waste" a considerable amount of disk space. For example, you may have a small CONFIG.SYS file (or any small file) that contains 34 bytes as displayed by the DIR command. That 34 byte file occupies one cluster (2 sectors -- 1024 bytes) so that 990 bytes of space in the cluster is unused. Similarly, a file that exceeds one cluster (1024 bytes), like 1025 bytes, requires another cluster to store the file even though only one byte of the second cluster may be used.

In summary, DOS allocates file space on a disk in groups of sectors called clusters. The number of sectors per cluster, sometimes called the cluster factor, is strictly dependent on the specific media and format. There is even a difference among the various types of 5.25" disks. The standard 5.25", DS/DD 360 KB floppy has two sectors per cluster; and a SS/DD floppy has only one sector per cluster. The 5.25", 1.2 MB high density disk used in the Z-200 and Z-300 also has one sector per cluster. For practical purposes, all of this formatting is handled by the FORMAT command for floppy disks. If you have a hard disk, there are a few differences in the way the formatting is done.

Low-Level Hard Disk Formatting

When you have a hard disk, you must do two types of formatting: low-level formatting and high-level formatting. The LOW-LEVEL format is required to add the special "stuff" to a hard disk that is not necessary on a floppy. For most hard disks, this special "stuff" is essentially a partition table and some other things which are added to a hard disk in the form of a Boot Record (or a SuperBlock on the Z-100). The Boot Record is actually created by the low-level format program on the hard disk, but you can change it with the Zenith PART command.

If you use Zenith MS-DOS, the PREP command is used to create the Boot Record or SuperBlock on the hard disk. In addition, PREP also performs a multi-pass media test to check for any bad sectors, and a Bad Sector Table will be created if any bad sectors are found. If you have a Z-100, you must use PREP for the low-level format. If you have a PC compatible, you may have another choice depending on which hard disk controller you have in your system.

Many PC compatible hard disk controllers include a low-level formatting program. It is commonly found in the controller ROM, and you may need to use the DEBUG command to run it. As I recall, both the Omti and Western Digital hard disk controllers have the low-level formatting program in ROM. In most cases, it is better to run the Zenith PREP command to perform the low-level formatting because of the media testing. One of the functions of the low-level formatting program is to create something called the "interleave" byte in the boot record. We'll talk about the interleave byte later since its value can have a significant impact on the performance of your system. Regardless of how you do the low-level formatting, the next step is to partition the hard disk with the Zenith PART command.

A word of caution before you even attempt to start "fixing" your hard disk that applies to the PC Series computers only. If you even remotely suspect that you have a "non-IBM standard" hard disk (such as a hard disk on a card), you MUST have Zenith MS-DOS version 3.20 or later so that you can use the PREP/Q (Query) command to ask for the hard disk characteristics. In addition, you MUST also have the detailed technical characteristics of the hard disk so that you can answer the following PREP questions:

- Will this hard disk require servo track information (Y/N)?
- Enter maximum number of cylinders in dec:
- Enter maximum number of heads in dec:
- Enter the starting write precompensation cylinder in dec:
- Enter the landing (shipping) zone in dec: (Z-200/300)
- Enter the starting reduced write current cyl in dec: (Z-150)

My experience is that very few hard disk manufacturers supply the answers to these questions with their units, so you will need to write to the manufacturer to get the information. In many cases, it is quite unlikely that the dealer will have this information, although you will probably find that you must start with the dealer in order to get the manufacturer's address. If you decide to go that far, I also recommend that you just get the technical manual from the manufacturer for your specific hard disk, but be aware that some manufacturers charge as much as $50 for that manual. Regardless of how you do the low-level formatting, the next step is to partition the hard disk with the Zenith PART command.

Partitioning A Hard Disk

Running PART is generally the next step in installing a hard disk. Even if you decide not to use the multiple partition features of Zenith MS-DOS, you will probably want to run the PART command. The PART command allows you to specify the bootable partition on your hard disk in addition to creating multiple partitions. PART actually changes the Boot Record (or Z-100 SuperBlock) so that DOS can find the bootable partition. If you have created multiple partitions with PART, the boot record contains beginning and ending addresses for each partition. If you are using an entire hard disk as a single partition, that information is also part of the Boot Record. Now that you have partitioned the hard disk, it's time to do the high-level formatting.

High-Level Formatting on a Hard Disk

A hard disk must be formatted with the FORMAT command just like a floppy disk. You will need to use the /S (System) switch to make the partition bootable that was specified with the PART command. If you specified multiple partitions, you will also need to FORMAT each one. For a PC Series computer, you may want to change the partition assignment flag with CONFIGUR or DSKSETUP (depending on your MS-DOS version -- 3.10 or 3.20) so that you can use the ASGNPART command. For a Z-100 MS-DOS system, you will need to use the ASGNPART command before you can FORMAT the partitions.

At this point, your hard disk is completely set up and ready for use. You have performed the low-level formatting with PREP or the program that was supplied with your hard disk controller. Then you ran the PART command to specify the bootable partition and perhaps identify additional partitions. Next is the use of the ASGNPART command to assign a drive letter to a partition which may be omitted if you use the automatic partition assignment feature of Zenith MS-DOS version 3.10 and later. And finally, you performed the high-level formatting on each partition with the FORMAT command. I have digressed from the discussion of sectors and clusters in order to describe what is happening in each step for the preparation of a hard disk. Let's now return to a discussion of sectors and clusters on a hard disk.

Sectors and Clusters on a Hard Disk

With the exception of the Z-100 MS-DOS, the hard disk will be "divided" into 512 byte sectors just like a floppy. The Z-100 exception is that you can specify the /K (Kilobyte) switch with PREP to create 1024 byte sectors. Regardless of the size of each sector, it is an interesting fact that the number of sectors in each cluster is dependent on the size of the partition that was defined in the PART command. But before we get too far along, I would like to be sure that we all understand what software I am talking about.

All of the following information about sectors and clusters is specific to Zenith MS-DOS version (BIOS version 3.04) 3.10 and later. It may or may not apply to any version 2 release, and I have not tested any of this information with Zenith MS-DOS version 2, although I suspect that you might be able to do some of this. In addition, you MUST use the Zenith version 3 MS-DOS commands to set up your hard disk, and this includes PREP, PART, and FORMAT. Otherwise, you may find that this information does not apply to your system. If, for example, you used the low-level formatting program provided with the Omti and Western Digital hard disk controllers, you may find that you have 16 sectors per cluster. The point is that you must start with the Zenith version 3 PREP command and follow the steps previously mentioned in order to take advantage of these features.

Do You REALLY Need a Larger Hard Disk?

In quite a few cases, the answer is NO. As a matter of fact, there is an easy way to effectively increase the capacity of your hard disk if you use Zenith version 3 MS-DOS. In extreme cases, you may be able to quadruple the effective capacity of the hard disk with a little work. It's nice to be able to get four times the current capacity of your hard disk by doing a little work. Let's see how this is possible.

You can change the number of sectors per cluster on a hard disk by adjusting the partition size with the Zenith PART command. In the most extreme case, you may find that your hard disk has 16 sectors per cluster (8,192 bytes), and if you have a lot of small files, that will result in an amazing amount of wasted disk space. Consider the 34 byte CONFIG.SYS file mentioned earlier. When you have 16 sectors per cluster, that is an incredible 8,158 bytes (8,192 - 34) of wasted space for that small file. If you could reduce the cluster factor to 4 sectors per cluster (2,048 bytes), you could theoretically get four times as many small files on the disk, assuming each file was less that 2,048 bytes long. In practice, many files are not that small, but it IS realistic to assume that you really could effectively double the capacity of your hard disk in the extreme case. The obvious result is that you may not REALLY need a larger hard disk which will save you some M-O-N-E-Y. If that appeals to you, read on.

Adjusting Cluster Sizes with PART

Perhaps one of the nicest features of the hard disk commands in the Zenith MS-DOS is the capability to adjust the cluster sizes. You cannot do this directly, but you can change cluster sizes by adjusting the partition size with the PART command. If you have the Programmer's Utility Package for MS-DOS Version 3, you can find the basic information for my tables on pages 5.7 (PC Series) and 5.8 (Z-100).

For the PC Series computers, cluster factors are listed in Table 1. If your partition size is between 4096.5 kilobytes (about 4 megabytes) and 16,340 kilobytes (about 16 megabytes), your system is using 8 sectors per cluster or 4,096 bytes as a minimum for each file. If you have a 10 megabyte partition, you may be wasting a lot of space on the hard disk. The solution is to reduce the size of each partition so that it is less than about 4 megabytes so that you can cut that cluster size to 4 sectors per cluster. That could theoretically double the space available on your hard disk.

4 and 16 megabytes since that requires eight sectors per cluster.

If you have a 10 megabyte hard disk, you can also see from the table that just under a 4 megabyte partition will also allow you to have better efficiency. My suggestion is to define two 4 MB partitions and one 2 MB partition for a 10 MB hard disk. That will give you drives C, D, and E. Be sure to run the CONFIGUR (for version 3.10) or DSKSETUP (version 3.20) command to use manual partition assignment before you assign the partitions with the ASGNPART command.

Table 1 -- Cluster Sizes for PC Series MS-DOS Version 3

Partition Size	Sectors/ Partition*	Sectors/ Cluster (Bytes)
32K - 256K	64 - 512	1 (512)
256.5K - 1024K	513 - 2048	2 (1024)
1024.5K - 4096K	2049 - 8192	4 (2048)
4096.5K - 16340K	8193 - 32680	8 (4096)
16340.5K - 32768K	32681 - 65536	4 (2048)

* Less boot, FAT and directory sectors

If you look at Table 1 carefully, it is obvious that you should avoid a partition size in the range of about 4 megabytes to 16 megabytes because it requires 8 sectors (4096 bytes) per cluster. Partition sizes below and above that range cut the cluster factor in half, so you will probably want to create partitions in the 1-4 megabyte range or the 16-32 megabyte range, depending on the size of your hard disk.

Notice that the cluster size drops from 8 sectors per cluster (4096 bytes) to 4 sectors per cluster (2048 bytes) when the partition size is just over 16,340 KB. Without going into a lot of technical details, suffice it to say that the reason for the decrease in the cluster factor is a result of using a 16-bit FAT (instead of a 12-bit FAT) when the partition size exceeds 16,340 KB. In my '241 40 MB hard disk, I have two partitions that are defined at just over 16 megabytes so that I could reduce the cluster sizes. For best efficiency, you should not define partition sizes between

For the Z-100, the cluster factors are slightly different, as shown in Table 2. This table applies to hard disks that have been prepared with the PREP command using the standard 512 byte sectors.

As you can see, it's best to avoid a partition size between 8 and 16 megabytes because of the cluster size. If you have used the PREP/K command to define 1 kilobyte sectors, partition size breakpoints are the same, but the cluster factor increases in a 2, 4, 8 order since only a 12-bit File Allocation Table entry is used.

Since the cluster size is strictly dependent on the SIZE of the partition, you will need to play with the PART command to get an "ideal" cluster size in your system. The

Table 2 -- Cluster Sizes for Z-100 MS-DOS Version 3

Partition Size	Sectors/ Partition*	Sectors Cluster (Bytes)
OK - 8 MB	0 - 16384	4 (2048)
8 MB - 16 MB	16385 - 32768	8 (4096)
16 MB - 32 MB	32769 - 65536	4 (2048)

* Less boot, FAT and directory sectors (512 bytes)

PART command for the PC Series requires that you enter partition sizes with beginning and ending cylinder sizes. The PART command for the Z-100 requires that partition sizes be entered as a percentage of the total disk space. In both cases, you will need to experiment with PART to get an ideal cluster size based on the information shown in Table 1 or Table 2.

When you use these tables, you need to know that all values are shown in kilobytes. Remember that a KILOBYTE is 1,024 bytes, and that definition applies to both memory and magnetic media such as disks. To complicate matters, a MEGABYTE is defined as 1024 kilobytes or 1,048,576 bytes. You may find these conversion factors useful when working with Table 1 and Table 2.

After running PART, you will, of course, have to FORMAT the new partition before you can check the cluster size. Be sure that you backup everything on the hard disk before you try any of this since the PART/FORMAT combination will destroy all data on the hard disk.

Checking Cluster Size

There are probably a zillion ways to check on the cluster size. My personal preference is to use the FRAGCHK program that is included in the Mace Utilities that I discussed last November. The FRAGCHK program displays all kinds of information about your hard disk partition, and its particular function is to tell you how many files are fragmented on your hard disk. Then you can run the UNFRAG program to unfragment all files if necessary.

But there is another way to determine cluster size that is reasonably obvious. This method relies on the fact that DOS allocates a single cluster for any small file that I mentioned earlier. First, use the DIR (or CHKDSK) command to find out how many "free bytes" you currently have in a partition, and write down that number. Then, pick a very small file (less than 512 bytes) like AUTOEXEC.BAT, and COPY it to a file name like CLUSTER. Rerun DIR (or CHKDSK) to again find out how many free bytes you now have in the partition, and subtract that from the original value found in the first step. The difference is the number of bytes in a cluster and will always be an even multiple of 512 such as 2048, 4096, etc. The last step is to divide the difference by 512 to determine the number of sectors in the cluster. This trick works on any disk, and you can determine any cluster factor for any standard DOS disk using this technique. I used this trick to figure out the cluster factor on my Z-100 since I did not have any utilities that displayed the value.

There are a couple of other things that can have a significant impact on the performance of your hard disk system. One of the least obvious, and most important, is the interleave factor.

What is Interleave?

One of the facts of using a computer is that input and output (I/O) of all kinds are slow. Humans are the slowest of all input "devices" which is the major reason that I have been so critical of keyboards used in the newest microcomputers. Printers are also slow devices compared to computer speeds. And, of course, disk I/O is slow too. The speed of disk I/O is substantially improved when you get a hard disk, but let's see what is happening inside the system.

Reading from and writing to a disk is not instantaneous -- it takes a finite amount of time to read or write a sector of data. If a disk spins fast enough, the system may not have enough time to read the next sector because it has already passed under the read/write head. Therefore, the system must wait for one disk rotation and get the next sector on that pass. Since it is easy to calculate the time that a sector passes under a read/write head for a hard disk, I think you may find it interesting.

A standard PC compatible hard disk rotates at 3600 RPM and has 17 sectors per track. Therefore, a single rotation of the hard disk takes 60/3600 seconds or about 16.7 milliseconds. Since there are 17 tracks, it takes 16.7/17 milliseconds or about 0.98 ms for the read/write heads to pass over one sector. The hardware/software only has about one thousandth of a second to process the data from one sector before the next one passes under the heads. Unfortunately, most microcomputer systems are not generally capable of processing data quite that fast. Therefore, the interleave factor is introduced to make sure that the system has time to process data from each sector so that it will not have to wait for a complete rotation to read or write each one.

Most standard 5.25" floppy disks rotate at 300 RPM except for the high-density drives that rotate at 360 RPM. At those speeds, there is no need to have the system wait since it can process data with no problems. Physical sectors can easily be read in the following order:

01 02 03 04 05 06 07 08 . . .

Since these sectors are sequential, we can say that we have a "1" interleave factor.

Now we'll take a look at the interleave factor for a "fast" hard disk system, specifically the Z-200. My system has an interleave factor of "2" so that sectors are arranged in the following order:

01 xx 02 xx 03 xx 04 xx . . .

Notice that every second sector is sequential, and the "xx" means that we don't care about that sector.

It's difficult to pick a "standard" PC compatible system for an interleave example because values can be anywhere in the 3-6 range, depending on the low-level formatting program. As an example, an interleave factor of "4" will result in sectors being arranged in the following order:

01 xx xx xx 02 xx xx xx 03 xx . . .

As you can see, every fourth sector is sequential. In general, an interleave factor of "n" means that every "nth" sector is sequential.

The whole point of using interleave is simply to allow a system enough time to read or write sectors without having to wait for a complete disk rotation for each one. The actual value of the interleave factor depends specifically on which low-level formatting program (e.g. PREP) you used on your hard disk. An interleave of "2" is quite common on fast systems such as the Z- 200 or IBM AT. Slower systems, like the original IBM XT, may have an interleave in the 3-6 range which is a result of system clock speed as well as the hard disk controller and low-level formatting program.

The interleave value can have a significant effect on the performance of your system. The objective is to set the interleave at the smallest possible value that your system can handle. If the interleave is too large, your system may be running slower than necessary, and you may be able to improve your system performance for hard disk I/O by reducing that value. That may be important to you if you have added some kind of a "speed-up" kit to your system. The good news is that there is some software that can help you optimize the interleave for your system. The bad news is that it does not run on the Z-100 since it is for PC compatible systems only.

The HTEST/HFORMAT Software

There are six programs included in this software package. Before I start, it is important to note that you need to backup your entire hard disk before using most of these programs. Since you know about the interleave factor, I'll start with the program that will help you "optimize" the interleave factor for your system: HOPTIMUM.

When you run the HOPTIMUM program, the idea is to test your system to determine the best interleave factor. It is not necessary to test the entire hard disk, but the manual recommends that a minimum of 10 cylinders be used for the test. If you use a larger number of cylinders, the test will take longer. When the program has completed, it displays the best interleave factor for your system.

Once you know the best interleave factor, the next task is to change it on your hard disk with the HFORMAT program. HFORMAT is a low-level formatting program like PREP, and it will totally destroy all data on your hard disk. It is similar to PREP in many ways, and it also does media testing too. If the interleave factor is not specified in the command line, it is set at "2" for AT compatible and "6" for XT compatible (e.g., Z-150 series) systems. Be absolutely sure that you have a good hard disk backup before using this command, because changing the interleave factor may be hazardous to your sanity, since it effectively destroys all data on a hard disk. As you can see, HOPTIMUM and HFORMAT are used together to test and change the interleave factor on your hard disk. When the low-level formatting has been completed with HFORMAT, you must also use the standard FORMAT command for high-level formatting of the hard disk as usual.

HTEST is designed to perform a non-destructive read test or a destructive write test on a hard disk at the user's option. It is a heavy-duty program that can be used to check out a wide variety of hard disk errors and defects including some controller problems. This program reminds me of the Zenith DETECT command that is used to isolate bad sectors, but HTEST performs more hardware testing than DETECT does.

The RELOCATE program performs the same function as the Zenith SHIP command. It is used to move the hard disk read/write heads to the center of the disk so that the heads will not bounce on the media (and your data) if your computer is moved or bumped.

The last two programs are intended to work together. The GETSEC program performs an absolute sector read on a disk and writes the result to a specified file name. The PUTSEC program takes the data from a specified file name and writes it to a specified sector on a disk. Because of the nature of these programs, they are not recommended for beginners, although you might find some interesting things while working with the GETSEC program.

I found this to be a very interesting software package, but it clearly is not intended for everyone. For example, if you have a Z-200 or Z-300 system, you won't be able to do much with the interleave since it is about the best it can be considering the system capabilities. On the other hand, owners of the Z- 157/158/159 systems might find it helpful since they might be able to improve system performance by changing the interleave factor because of the 8 megahertz clock speed. I haven't tried these programs on those machines, so I can't say for sure.

Another interesting point is that this software includes those utilities that IBM forgot to include with PC-DOS. For example, HFORMAT and HTEST perform functions that are similar to the Zenith PREP and DETECT commands. And of course, the RELOCATE program performs the same function as the Zenith SHIP command.

The programs seem to perform their intended functions quite well, and I found no problems. Perhaps my biggest criticism is that the manual is somewhat cryptic for beginners, and it needs a little work on organization, but everything you need to know is there. If you are interested in working with your system on this level, the HTEST/HFORMAT package is absolutely recommended if you are using PC-DOS. If you have Zenith MS-DOS, I don't think the need is quite so critical since you already have some of the same (but not all) of the test capabilities and commands; however, you may find that you can improve your system performance if you have a "fast" system like the Z-157, Z-158, Z-159 or Z-148.

Microsoft C - Just Say No

There has been a lot of discussion in the popular press lately that Microsoft has had some very significant problems with the latest versions of both their C language products: Microsoft C 5.0 and QuickC. It seems that the Microsoft C compiler does not run properly (or at all) if you have an AUTOEXEC.BAT or CONFIG.SYS file on the disk. Isn't that incredible? Even more incredible is the report that QuickC does not work in a hard disk system that has a Western Digital controller. Absolutely amazing! Although I have seen these problems reported in several places, the most recent is on page 98 of the November 30, 1987, InfoWorld. Reports on the problems with QuickC vary depending on what you read, but at least one report (InfoWorld) says that QuickC will scramble the File Allocation Table on your hard disk. One other report says that all you need to do is reboot the system since the FAT gets scrambled in memory only, not on the hard disk.

I guess even Microsoft makes a real whopper of a mistake (or two) once in a while. However, it's difficult to believe that these problems were not found during beta testing of the products. I suspect that someone sent the wrong master disks for duplication since that is really the most reasonable explanation for the problem. It will be interesting to see if the project manager keeps his job as a result of these problems.

My reason for mentioning this is to note that you should use extreme care when checking out new software. If you have a hard disk, the best advice is to backup everything before you install a new software package. Although I always back up my hard disk partitions before I make any hardware changes, I occasionally don't do that when I am updating software. I would not expect catastrophic bugs in programming languages in particular. It never occurred to me that compiler software might have a bug that would really cause a loss of data.

Another point to be made about this problem is to be very careful with all new and updated software. That is particularly important with operating systems such as the new OS/2, and it even applies if you are getting a new DOS version. In some cases, you will find that commands have been changed slightly so they may not work like you expect.

One case in point is the FORMAT command that is available in the current 3.20 version of Zenith MS-DOS. As far as I know, all previous versions of Zenith MS-DOS, including the Z-100 version, used the /V switch to verify the format was done correctly. In some releases, the verify is done automatically and the /V switch was included for compatibility with previous releases. In the very latest version, the /V switch tells FORMAT to ask for a Volume label which was not always requested in some versions. Personally, I thought the Volume label prompt in FORMAT was a pain anyway, and I used the LABEL command to add the label to disks. This gets particularly tricky if you have batch files that contain DOS commands, so be sure to check EVERYTHING when you install a new DOS version from any manufacturer. The new version of the command may not work the way you are used to.

In the Mail

I recently received a postcard from a Huggie named Bob who asked why DOS subdirectory names are limited to eight characters. I'm not sure where he got that notion, but the answer is that subdirectory names are NOT limited to eight characters by DOS. I have seen at least one application program — I think it was a word processor, but I can't remember for sure — that only accepts eight character subdirectory names.

In general, subdirectory names are in the usual DOS format of "filename.typ". After all, a subdirectory name is nothing more than a DOS file name with a special file attribute. As you might expect, a subdirectory name must also follow the same rules as any file name in DOS. Although you should be able to use any valid DOS subdirectory name in a current application program, be sure to check just in case you have one that only accepts up to eight characters.

PC Hardware and the Z-100

I recently received a letter from Huggie Hugh Kenner (Baltimore, MD) asking if I knew of any way to add a PC-type hardware board, like a bus mouse, to the Z-100. Unfortunately, I don't. Has anyone heard of an "adapter" board that can be used with the Z- 100's S-100 bus so that a PC-type hardware board can be added? Hugh has the Gemini Emulator, so that PC emulation should not be a problem. I suspect that the technical problem is more related to having the "adapter" work correctly with PC emulation. In any case, please let me know if you have found such a beast. Also be sure to tell me what hardware you are using so that I can help other Huggies with a similar interest.

Powering Down

If things continue as scheduled, I hope to have a copy of the Zenith OS/2 operating system within the next month. Because of its capabilities, it looks like I will be writing several articles on OS/2 in general. I have already seen IBM's version of OS/2 at a local computer store, and a quick look at the manuals indicates that there are a lot of things that have NOT been adequately described. Perhaps Zenith will be able to do a better job with their OS/2 documentation.

From my notes, I see that I had planned to discuss disk access time last month, but the disappearance of my notes did not help that much. We'll take a look at that next time because it will possibly help you answer the question: "Are the high speed (e.g. 25 ms) hard disks really worth the extra cost?"

Based on the letters that you have sent, there is still a considerable interest in hard disks. From the technical perspective, there isn't much more to say about a hard disk after you know about the implications of disk access time. There are still a number of helpful hints and such that I will mention from time to time. And I just received a new version of DSBACKUP+ from Design Software, so we'll take a look at that next time. If you have any specific questions about hard disks and MS-DOS, be sure to let me know. I appreciate your letting me know about your questions and interests.

If you have any questions about anything in this column, be sure to include a self-addressed, stamped envelope (business size preferred) if you would like a personal reply to your question, suggestion or comment.

Products Discussed

DSBACKUP+ $79.95
Design Software
1275 W. Roosevelt Road
Chicago, IL 60185
(800) 231-3088 (Orders only)

IBM OS/2 $325.00
IBM Product Centers

HTEST/HFORMAT $89.95
MACE Utilities version 4.10 $99.00
Paul Mace Software, Inc.
400 Williamson Way
Ashland, OR 97520
(800) 523-0258 (Orders only)

*

KEYBOARD OVERLAYS

for

-19/89, -100, & PC-compats.

$5 per half dozen postpaid

Kirk L. Thompson

6 W. Branch Mob Hom Vil

W. Branch, IA 52358

Desktop Publishing With AutoCAD™

Patrick Swayne
HUG Software Engineer

One of the more popular uses of personal computers these days is desktop publishing. There are a large number of publishing programs available, and many of them are quite inexpensive. The biggest problem in terms of cost is not the software, but the output device. Laser printers, which produce the best results, are beyond the budget of many home computer users.

Many desktop publishing programs support dot matrix printers, but the results obtained with them are often not as good as they could be. One reason is that the printer may not be operated at its maximum graphic resolution. Another reason is that the font characters provided for dot matrix use may be pixel oriented, which means that if the characters are enlarged (which is not always permitted), the curved lines of the characters will become rough ("stair stepped").

Publish with AutoCAD™

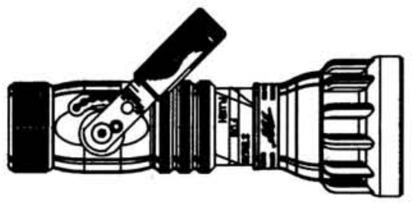
One possible solution to the problem of getting good output from an inexpensive device is to use AutoCAD (a Computer Aided Design program) for desktop publishing. The "bare bones" version of AutoCAD sold for $300 in the Heathkit catalog can be used for publishing. Unlike other CAD programs, the character sets in AutoCAD are proportionally spaced, and text handling is versatile enough to make creative page layouts possible. AutoCAD also offers these advantages:

1. AutoCAD is object oriented. That means that shapes (including letters and numbers) can be enlarged or shrunk to any size, and curved lines will remain relatively smooth. (Text characters in Autocad are actually composed of short straight segments, and you will see these in large characters, but you will not see the "stair stepping" found in pixel oriented characters.)

2. AutoCAD supports a less expensive alternative to the laser printer: the plotter. A Heath 5208 (Sweet P) plotter, when equipped with the proper paper and pen, can produce results that look very good.

3. A completed page can be printed at any size, scaled to precise dimensions.

4. With AutoCAD, you can put drawings of unparalleled preciseness on your page.


You can put complex drawings anywhere on your page!

5. Recent versions of AutoCAD (version 2.5 or higher) support user supplied output drivers. If AutoCAD cannot drive your printer at its maximum graphic resolution, you can write a driver that can.

Publishing Problems

Of course, AutoCAD was not designed for desktop publishing, and there are some drawbacks to using it for that purpose. Some of the problems you will encounter are:

1. AutoCAD cannot justify both margins of text. You can have left justification, right justification, or you can center the text. Since jagged right (or left) margins seem to be popular now, this deficiency may not be a problem for you.

2. You can only enter text using the Text command of AutoCAD. You will have to guess at how many words to put on each line, and re-do lines when you guess wrong. The U (Undo) command in AutoCAD versions 2.5 and above makes correcting lines easy, but it can still be frustrating until you get a feel for how many words will fit on a line.

3. A plotter is a very slow device to use for printing pages, and a printer driven at its maximum graphic resolution is also very slow. You may want to "test plot" your pages on a printer at a low resolution (if you have that option) before you do the final plot. If you intend to produce only a few pages and then prepare copies using a copy machine or print shop, slow output is less of an objection.

4. The fonts provided with AutoCAD are all line drawings, which means that large letters used in headlines, etc., will not be solid characters. You can design your own fonts with AutoCAD, but I doubt that it is possible to design a solid character font. You could, however, design some solid letters as "blocks" that would have to be inserted into the drawing one at at time.

Page Layout Procedure

To layout a page with AutoCAD, you first need to decide how big the page will be, and how big your letters will be on the page. If you are going to produce a page that looks like a page you already have, you can measure the page you have precisely and decide how many drawing units to assign per inch, centimeter, etc. The first page of this article, which as you have probably guessed by now, was produced using AutoCAD. I measured a typical REMark page, then decided to let two drawing units equal one inch. Since the AutoCAD drawing screen, at the default zoom value, is 9 drawing units high and 14 drawing units wide, I could view a 4.5-inch high section of a page at nearly the full width. Since the zoom is virtually infinitely variable with AutoCAD, you can set your view window to any size you are comfortable with. You should select a window size so that you can read the text size you have chosen and still see and work with several lines of text.

If there are to be any lines on your page, such as dividing lines between columns of text, you should draw those lines first. You can also mark off parts of the page for illustrations, or you can fit one in later after you have added text, as I did. It helps to have a ruler graduated in 10ths of an inch if you are measuring another page and using it as a guide for placement of lines, etc. Such rulers are used by "real" publishers, so most of the measurements will fall on an even tenth, which makes using a conventional fractional ruler difficult. Of course, if you live in a metric country, you will not have to worry about fractional rulers.

I set the normal text on my page to .2 drawing units high, which means that the capital letters are .1 inch high on the printed page. I used the "simplex" font for the normal text, the "complex" font for the headline and subtitles, and the "italic" font for the by-lines. These fonts are all supplied with AutoCAD. AutoCAD has an automatic command repeat feature and an automatic text positioning feature that make it easy to enter lines of text into your page. The automatic command repeat feature works this way: If you hit Return in response to the "Command:" prompt, the previous command you entered will be automatically repeated. When you automatically repeat the Text command, the automatic text positioning feature can be used to govern where the text will be placed. If you are left justifying your text, the automatically positioned line will be right below the previous line, with its left edge lined up with the previous line. The feature works similarly for centered or right justified text. The automatic text positioning feature works differently for versions of AutoCAD below 2.5 than it does for versions 2.5 and above. With versions below 2.5, it works only when you repeat the Text command automatically, which means that if you have to use another command, such as Erase to correct a mistake, you will have to position the next line of text manually. If you have version 2.5 or above, automatic positioning happens whenever you hit return in response to the text position prompt, regardless of what you have done since the last text command. With any version, you can automatically leave a blank line between paragraphs just by entering Return by itself as the text on the blank line.

To make it easy to position text and pictures precisely on your page, you should have Snap turned on and set to .1 drawing unit (assuming you are using the settings I previously described). The text automatically positioned by AutoCAD will not be spaced in even tenths, however, so you should follow this procedure whenever you need to position a line manually just below another line. First, move the cross hairs as closely as possible to the line above with snap turned on. Then turn snap off and adjust the vertical position (the horizontal position should always be on an even tenth) so that the horizontal cross hair is even with the bottom of the line of text. Then enter the Text command and enter Return as the line of text. Press Return again, to repeat the Text command, and if you have version 2.5 or above, press Return once more in response to the position prompt. Now you can start entering text again, and the new text will be positioned correctly below the last line.

If you make a mistake in a line of text with AutoCAD versions below 2.5, you will have to use the Erase command to remove the bad line and retype it. You will then have to manually position the next line. If you have version 2.5 or above, you can use the U command to remove the bad line. Then enter the text command again (type it — do not press Return, or U will be repeated) and press Return in response to the position prompt to correctly position the new line.

Illustrations

Putting illustrations on your page is the easiest part of using AutoCAD as a publishing tool, because it is, after all, a drawing program. Any drawing can be inserted as a part of another drawing as long as the Base command has been used to establish a base point in the drawing to be inserted. The base point determines where the drawing will be inserted, and it is usually a good idea to make the lower left corner of a drawing the base point. You can even put a base point in your completed page, and then use it as an illustration on the page! That may not be a good idea, because the plotted results may not turn out too well, especially if you use a printer for plotting. The printer driver may "greek" lines of text that are very small, which means that the text is not printed as text, but as straight lines.

You can scale the inserted drawing so that it will fit into any box you may have drawn on the screen. If you are guessing at the scale size and guess wrong, you can erase the inserted drawing and try again, without messing up anything else on your page. AutoCAD considers an inserted drawing as a single "entity" so it takes just one Erase command to remove it, no matter how complex it is. And with versions 2.5 or above, you can also use the U command to remove the drawing. After you have inserted a drawing, you can use the Move command to re-position it if you did not get it just right. You can also use the MOVE command to reposition anything else, including dividing lines or blocks of text. Remember that as you work on your page, you can zoom or pan as required to get the best view of what you are doing.

"Printing" The Page

When you have completed a page, you will need to "print" or plot it to see what it really looks like. It is at this point where you specify the inch/drawing unit (or millimeter/drawing unit) scale, so you can make a "test page" smaller or larger than the final page is intended to be. If you have both a fast graphics printer and a plotter, you may want to use the printer for test plots, and then plot the final copy on your plotter.

As you can see from the first page of this article, a page can be produced that looks quite good. We used a Hewlett Packard plotter to plot the first page, but you can get results almost as good with a Heath 5208 (Sweet P) plotter IF you use a good pen. You can install a pen designed for use with a Hewlett Packard 75xx series plotter in the Sweet P plotter by wrapping a few layers of tape around the small end of the ink cartridge so that it fits tightly into the Sweet P pen holder. The fiber tipped pens normally used with the Sweet P plotter (such as Heath Part No. 406-684) are just not good enough for small text, because they become dull (draw wider lines) with use.

I do not suggest that you rush out and get AutoCAD if all you want to do is desktop publishing. But if you already own AutoCAD, or if you intend to get it for its intended purpose, you can put it to double use, and do desktop publishing, as well as precision drawings.

*

New

Software Wizardry's

TimePiece

Date and Time of Day Clock

1. Fully Compatible
TimePiece is fully compatible with all Zenith MSDOS-based computer systems, from the Z-100 through the Z-386. Includes everything you need!

2. Easy to Install
You don't have to be a technician in order to install TimePiece. You simply remove an IC from a socket, plug this IC into TimePiece, then reinstall TimePiece into the socket. All required instructions are included.

3. All Software is Included
TimePiece comes with a program to automatically set the time and date on any MSDOS system. Any programs using the system time will be updated with no user intervention.

Only $39.95!

Plus$2 Shipping & Handling

From the same people who brought you RAMPAL, WildFire, and many other fine products...

Software Wizardry
THE MAGIC TOUCH

Software Wizardry, Inc.
#8 Cherokee Drive
St. Peters, MO 63376
(314) 447-7737

FINAL Z-100 SOFTWARE CLOSE OUT

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	$10.00
MS-463-7	Multiplan	$195.00	$10.00
CB-463-11	Z-Chart	$150.00	$10.00
CD-463-2	Condor File Manager	$299.00	$10.00
PK-100-4	All 4 listed above	$819.00	$38.00
MS-253-1	Microsoft BASIC-80 (8-bit)	$175.00	$10.00
OS-53-2	CP/M-85 (8 bit)	$150.00	$15.00
OS-63-4	Z-DOS	$150.00	$20.00
CB-463-9	PECON Peachtree to Condor	$99.00	SOLD
RS-463-5	Peachtree Inventory Management	$499.00	SOLD
WI-463-1	Remote Batch Terminal Emulator	$899.00	SOLD

*** 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. $269.00 Dual Drive Unit $399.00

COLOR GRAPHICS UPGRADE. All memory chips (16 pieces 150 ns) required to update Z-100 Series computer for color. Installation instructions included. Order Memory Kit #100-64-16 ....$29.50

MEMORY UPGRADE. All memory chips (9 pieces 150 ns) required to upgrade from 128K to 192K RAM. Installation instructions included. Order Memory Kit #100-64-9 ....$16.50

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 $159.00 with 27 256K RAM chips included.

UCI MEMORY UPGRADE PAL CHIP SET. For Z-100's with the newer motherboard (181-4918 or greater). This set permits installation of 256 K RAM chips on the motherboard. PAL set $64.00 or $153.00 with 27 256K chips required for 768K of memory on the motherboard.

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 $392.00, One MEG $456.00, Two MEG $579.00. Add $35.00 for EasyDrive RAM Drive Software if desired.

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 $578.00, 31 MEG $598.00
System without Drive and Controller $239.00

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

*** PANASONIC DOT MATRIX PRINTERS ***

KPX-10801 120 cps 10"	$219.00
KPX-10911 160 cps 10" 29cps NLQ	$272.00
KPX-10921 180 cps 10" 33 cps NLQ	$338.00
KPX-1592 180 cps 16.5" 38 cps NLQ	$421.00
KPX-1595 240 cps 16.5"	$462.00

*** 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 $339.00 30 MEG $359.00. 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. $118.00

Winchester Hard Disk Drive External Set-up. Includes Winchester drive, controller/interface card, power supply and case with fan.
With 20 MEG (formated) drive $478.00
With 30 MEG (formated) drive $498.00

ZFL-181-93 LAPTOP PORTABLE with the amazing Supertwist LCD backlit screen 640K RAM, two 720K 3.5" disk drives, clock, P & S Ports, 8 MEG. Retail $2399.00. Payload $1660.00
With 1 floppy and 20 MEG hand disk $2545.00

*** HALF HEIGHT FLOPPY *** DISK DRIVES

MITSUBISHI M501	5.25"	48 TPI DS/DD	320K/360K	$98.00
MITSUBISHI M504	5.25"	96 TPI DS/DD	360K/1.2 MEG	$139.00

*** SEAGATE HARD DISK DRIVES ***

ST-225 20 MEG Winchester Hard Disk	$295.00
With Western Digital Controller & Cables	$339.00
ST-238 30 MEG, Requires RLL type controller	$309.00
With RLL Controller & Cables	$359.00
ST-4038 30 MEG High Speed for Z-200	$544.00
ST-4051 40 MEG High Speed for Z-200	$629.00
ST-4096 80 MEG High Speed with Software	$929.00
ST-251 40 MEG High Speed Z-150/200	$489.00

*** POWER SUPPLIES AND CASES FOR DISK DRIVES ***

Rugged steel construction with heavy duty power supplies. Purchase with drives and we will install drives in case.

Single 5.25" unit for Full Height Drive	$68.00
Dual 5.25" unit, for Half Height Drives	$92.00
Dual 8" unit with fan, for Half Height Drives	$188.00
Face plate for single drive in dual case	$8.00
For WINCHESTER Drive, with fan	$157.00

*** 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	$1.85 each
256K Dynamic RAM, 150 ns	$3.88 each
256K Dynamic RAM, 120 ns	$4.10 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

MasterCard

PAYLOAD
COMPUTER SERVICES

VISA

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 or more day warranty. Add $5.00 to all prepaid orders for handling and shipping, we pay the balance. Texas Residents please add 7.25% sales tax. We accept purchase orders from schools, government and approved accounts.

Two for the Show

by Joseph Katz

Computer clip art from Micrografx and PC Quik-Art can save you manual labor.

Quality desktop publishing software like PageMaker and Ventura Publisher is a natural for producing microcomputer instruction manuals. You do the text in your trusty word processing program, the illustrations in a draw or paint program, and use one of those professional page makeup programs to integrate words and pictures into camera ready copy for the finished pages. Words, as we all know from reading computer magazines and manuals, are easy. Pictures, though, are hard. Even people who can't recognize a participle dangling in front of their noses can sort of sense that what was supposed to illustrate a floppy diskette looks more like a clock melted by Salvador Dali. But who wants to Dali in a technical manual?

Take heart. If you have no Great Art Talent, or only a little and don't yet trust your skills to manipulate draw or paint programs, or quite a bit but don't have the time to whip up original illustrations, you should rejoice over computer clip art packages from Micrografx and PC Quik-Art. They're Micrografx's computer ClipArt and PC Quik-Art's Volume SP 7. Owners of Zenith computers have special reason to rejoice because, despite their emphasis on IBM computers, those two computer clip art packages include illustrations of some Zenith products.

Those illustrations can be "clipped"—copied—into a graphics program for editing. Micrografx's Computer ClipArt

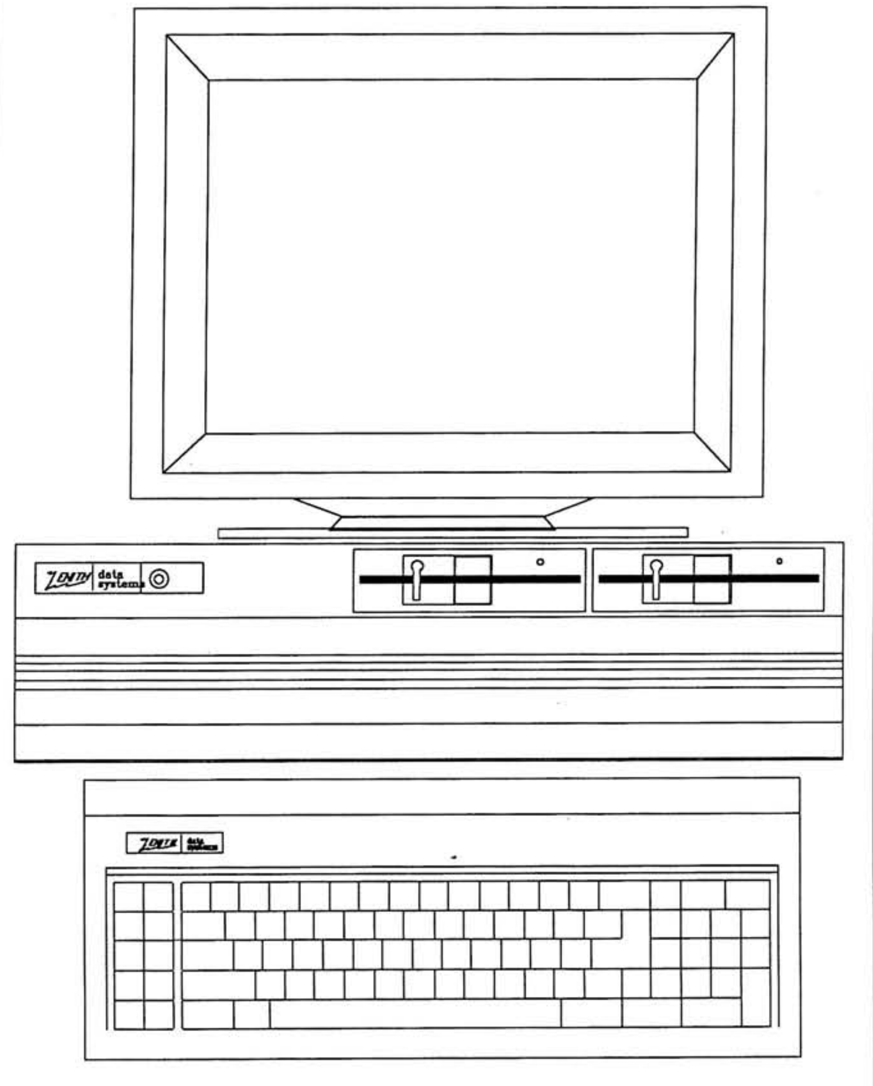


Figure 1. Zenith's Z-200 style from Micrografx's COMPUTR2 file.

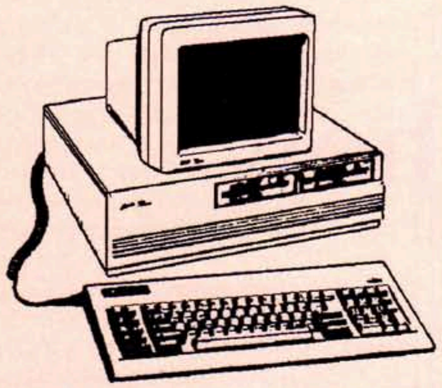


Figure 2. Zenith's Z-200 style from PC Quik-Art's file S7-01-05.

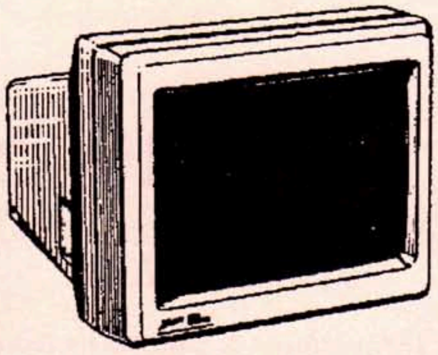


Figure 3. A Zenith monitor from PC Quik-Art's file S7-02-01.

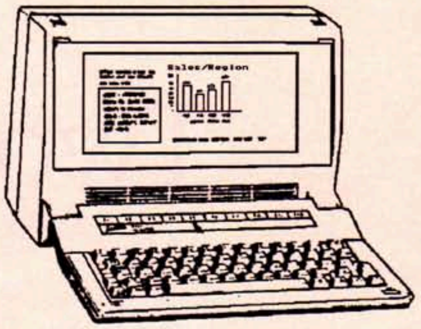


Figure 4. Zenith's Z-171 from PC Quik-Art's file S7-02-03.

is object oriented files for Micrografx's own draw programs, Windows Draw and In*A*Vision. Because these *are* object oriented pictures you can use either of the Micrografx programs to change their sizes without distorting the images and you even can decompose and rearrange their elements as building blocks for your own drawings. For example you can Break Apart the Z-200, pull out Zenith's logo from the computer, and use it elsewhere, as I did in Figure 6. PC Quik-Art's clip art is bit mapped images in the PCC format used by the Frieze utility program supplied with Z-Soft's popular Paintbrush series of paint programs. You can Paste these images into a Z-Soft program such as Publisher's Paintbrush and massage them there to get more, less, or different detail. Size changes, however, are less satisfactory with bit mapped images than with object oriented images because they stretch or shrink the little dots that comprise each picture: at least some distortion is inevitable.

As for the relative merits of object oriented and bit mapped images, it depends on whether you want crisp lines or pointillistic rendering. Your choice should take into account just which page makeup program you'll use. Because Micrografx's ClipArt is in the format required by its own programs, and because they and PageMaker are Microsoft Windows applications, they're not readily usable in Ventura Publisher. Ventura Publisher runs in a subset of Digital Research's GEM environment and does not accept Micrografx's draw format. If you have Micrografx's Convert utility, a separate program that also requires Windows, you can convert a PIC file to an AutoCAD DFX file, then use the DXFTOGEM utility supplied on Disk 11 of Ventura Publisher to convert the DFX file to a GEM file and import that file into Ventura Publisher. Expect to lose details in the translation. Micrografx doesn't mention the procedure and I don't really recommend it. But it can be done. Because both PageMaker and Ventura Publisher can import files in Z-Soft's Frieze format, you can use PC Quik-Art's clip art collections in either page makeup program.

Clip art is packaged illustration in a

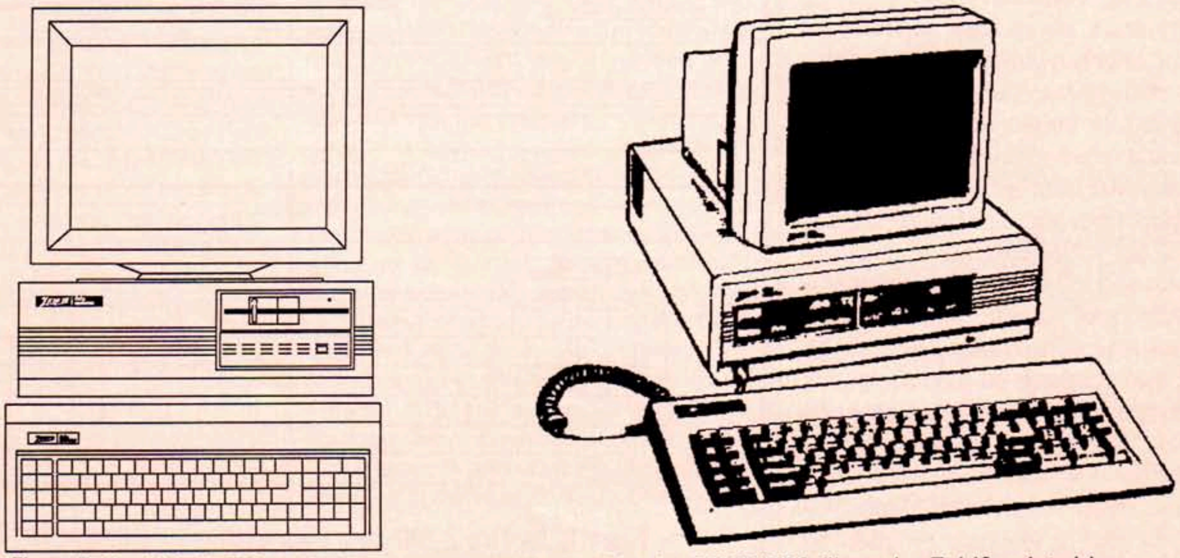


Figure 5. Zenith's Z-150 style drawn by Micrografx (left) in the COMPUTR2 file and a Z-148 painted by PC Quik-Art (right) in file S7-02-04.

style generalized enough so it can be used in different contexts and more or less fit them all. Usually it's in the form of sketches. You see clip art used heavily in the yellow pages of your phone book, in newspaper ads, and in other places where a picture may be worth a thousand words but apparently not the few hundred dollars it might cost to have an artist do an original work. When you see the same sketch used over and over in ads run by different businesses, chances are that it's clip art. There's nothing new about clip art: bibliographers specializing in the study of early printed books have found in different incunabula the same block prints identified as depicting entirely different subjects. What Micrografx and PC Quik-Art have done is digitize the form. Of course what they've done is not limited either to illustrations of Zenith computers or even to illustrations of computers only. The pictures I've reproduced here indicate the style of offerings from these two companies and might suggest possibilities to you.

I prepared this article with Micrografx's ClipArt and Micrografx's In*A*Vision, PC Quik-Art and Z-Soft's Publisher's Paintbrush, and PageMaker. The text was done with XyQuest's XyWrite III Plus. The computer was a Z-248 linked to an Apple LaserWriter at 19200 Baud with Blitz.

Products Discussed

ClipArt (Computers). $49.95.
Micrografx
1820 N. Greenville Avenue
Richardson, TX 75081
214/234-1769

Quik-Art Volume SP7. $59.95.
PC Quik-Art
394 S. Milledge Avenue
Athens, GA 30606
800/523-1796

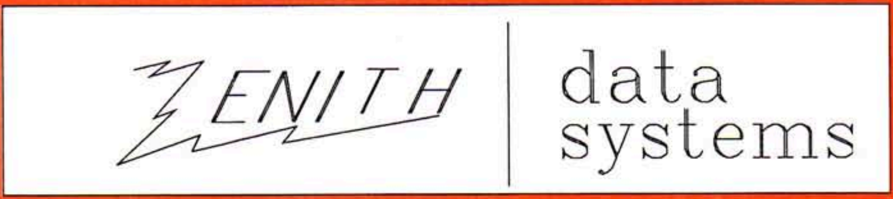


Figure 6. The Zenith Data Systems logo, extracted and enlarged from the keyboard of Figure 1 — Monografx's Z-200.

✱

S & K Technology, Inc.
Quality Software for Heath/Zenith Microcomputers

Announcing PC WatchWord® Version 3

New features:

- **Mail merge and automatic indexing**
- **DOS command execution with screen capture**
- **User defined display fonts (EGA only)**
- **Proportional font support for the HP LaserJet**
- **Arithmetic with user-settable precision and base**
- **Expanded manual loaded with new commands**

Plus all of the features of WatchWord Version 2 including formatting, centering, macros, horizontal scrolling on long lines, large file capability, split screen, color, and an extensive configuration facility.

PC WatchWord	**$129.95**	**PC Resident Speller™**	
PC WatchWord Demo Disk	**$2.00**	**(includes STRIKE™)**	**$99.95**
Resident Speller Demo Disk	**$2.00**	**STRIKE**	**$49.95**

SPECIAL OFFER: PC WatchWord - PC Resident Speller Combination $199.95

Texas Residents add state sales tax.

S & K Technology, Inc., 4610 Spotted Oak Woods, San Antonio, TX 78249, (512) 492-3384

"I hear First Capitol Computer is a Pretty Good Place to buy stuff from, but I can save $100 by buying from Joe's Computer City..."

Let's Give Joe a Hand!

Good old Joe - he's done a lot for our business! You see Joe's very good at making our prices look high. Providing, of course, you don't look too closely!

You see, Joe always has the lowest price in his ad. It doesn't matter what he sells or what it costs him to buy it. He spends hours each week clipping ads from his competitors in any publication he can lay his hands on. Every month you see another new ad from Joe, and every month it's the same thing. Column after column of prices just *that much* lower than the previous month.

Problem is that Joe has little to sell except price. His warehouse is a garage (literally), his technician is the kid next door that's "Going to school to learn about computers", he stocks what he sells - after he sells it.

So, you place your order with Joe and you think him a pretty nice guy - until you discover that your charge card has been billed, and Joe doesn't know when he'll *really* be able to ship the product. For a very good reason. Joe isn't factory authorized to carry what he sells.

We won't even talk about the nasty prospect of you actually needing Joe's help to fix a problem!

You see, Joe is actually using your money to get his business off the ground. He can't afford to hire knowledgeable employees (or any employee, for that matter). He has no inventory, no assets, and is no help at all.

The First Capitol Difference

First Capitol Computer has been carrying Zenith Data Systems products ever since Zenith started selling through dealers. Through this entire period we have maintained factory authorization for the entire line, Z89 through Z386.

On full-time staff we have engineering, computer science, and technical personnel dedicated to making Zenith's computers scream with power! For you see, we are a division of Software Wizardry, a major manufacturer of add-on boards and software for Zenith computer systems. We handle, and resolve, more technical questions in a day than the average computer dealer does in a month!

...And we're good at it! So much so that First Capitol Computer was listed in the December Inc. Magazine issue as "one of the fastest growing companies in the nation". As proud as this makes us, we know that YOU put us there, and we won't forget that fact!

Our pledge to you!

First Capitol Computer won't insult your intelligence by claiming to ALWAYS have the lowest price in the country - but we WILL promise you this:

First Capitol Computer will meet or beat your best price on any product we carry that isn't below our cost!

So, read Joe's ads...

But when it comes time to spend your money - spend it where you won't get shortchanged. We guarantee prompt, courteous answers to your questions, rapid shipment (normally from stock), no charge card billings until the product leaves our warehouse, and the best price you can find!

We wish Joe the best! Many of our customers have bought from him - but *only* once!

First Capitol Computer

16 Algana, St. Peters, MO 63376
1-800-TO-BUY-IT Orders Only
1-314-447-8697 Technical Line

REMIND

Larry Conklin
105 Riverglen Road
Liverpool, NY 13090

An Automatic Memory Jogger For MS-DOS Or CP/M

Have you ever become so engrossed in a programming project that you forgot about something important in the real world? Something like your wife's birthday? Do you occasionally forget to make a car payment, and end up having to pay a late penalty? My nemesis is the power company. Every other month they send a man around to read the meters. Since my wife and I both work, there is seldom anyone home to let the meter reader into the basement where the meters are located. You can get around this by taking the readings yourself on the day they are supposed to be read, and phoning them in. Failing that, you get an estimated bill. Needless to say, the estimate is not likely to be lower than your actual useage. My problem was that I would invariably forget to read the meters until the bill arrived, and then, of course, it was too late.

I use my computer nearly every day, and it occurred to me that it would be nice to have a computer based memory jogger. To be really useful, the operation of such a utility should be fully automatic. After all, if you're absent minded enough to forget to read the gas meter, you probably won't remember to run a program that is supposed to remind you to do it. REMIND is a program that will remind you of all of those forgettable dates every time you use your computer. When you boot up, the program reads the system date and scans the contents of a file of dated memo records. Any records that have start and end dates that bracket the current date are displayed on the console. This article describes two versions of the program which will run on MS-DOS or CP/M systems, respectively.

REMIND is designed to be invoked automaticly everytime the system is cold booted. In an MS-DOS system, this is easily done by including a call to the program in your AUTOEXEC.BAT file. In a CP/M system, the program is invoked as an auto start command. The method for setting up a CP/M auto start command is straight forward, and has been described in the January/February 1985 issue of Microsystems in an article by Kelly Smith, "Using CP/M's Undocumented Autoload Feature". If you are running ZCPR3 on your CP/M system, the STARTUP facility makes the installation of even a fairly elaborate startup command sequences a very simple procedure.

In addition to the automatic mode, REMIND can also be run interactively to add new reminder records or to review and selectively delete records from the data file. The command, REMIND A, invokes a procedure to add new records. The program prompts for the starting and ending dates for the period during which a reminder is to be displayed, and a single line of text, which may be up to 80 characters in length. Dates are entered in the conventional mm/dd/yy format. It is not necessary to include leading zeros when a field can be represented as a single digit; 5/9/87 or 05/09/87 are equally acceptable. If either or both of the corresponding fields in the start and end dates of a record are zero, the field will not be considered in the determination of whether the record should be displayed. For example, if you have a car payment that is due at the end of each month, the start and end dates could be specified as 0/10/87 and 0/15/89, respectively. This would cause the record to be displayed from the 10th to the 15th of each month until the end of 1989. Or, if you want to be reminded that your wife's birthday is coming up each year, you can specify the start and end dates as 7/10/0 and 7/20/0. The program continues to prompt for additional records until a single slash character (/) is entered instead of a start date.

The command, REMIND E, allows you to review and edit the contents of the memo file. Each record is displayed with its start and end dates, and a prompt asking whether the record should be deleted from the file. Responding with 'N' or 'n' will cause the record to be dropped. If you have already bought your wife's birthday present, there is no need for further reminders. It is normally not necessary to use this procedure to delete records whose end dates are older than the current date. When run in the automatic display mode, any records which have an end date that is earlier than the current date, and which do not contain any "don't care" fields, are automaticly dropped. Therefore, it should only be necessary to use the edit procedure to delete reminders that are still eligible for display now or in the future.

REMIND was originally written in Assembly Language to run on a CP/M system. It worked fairly well, but had at least one subtle bug that I never did get around to tracking down. Also, the Assembly Language version had a very rigid and unforgiving data entry format, and it did not include the house cleaning function to delete past records. I had been reading a lot about C, and decided to rewrite the program in C as a vehicle for learning something about the language. The only way to really learn a programming language is to use it, so I acquired an inexpensive C compiler and a good tutorial book and plowed ahead.

I have been very happy with my choices of both the compiler and the book. The CP/M version of the program was written in C/80 from The Software Toolworks. They seem to have an excellent track record for producing well written software at an affordable price, and their C compiler is a good example. The book I used was the "C Programming Guide", written by Jack Purdum and published by Que. Purdum's treatment is at just the right level for someone who has done some programming, but is new to C. The book has many example programs that you are encouraged to compile and run. This is probably a very good way to learn the language, but I would never have had the patience to type in and run all of the examples. So, I decided to rewrite my REMIND program in C. In doing so, I would exercise most of the important features of the language, and would end up with a useful program. The exercise more than fulfilled my expectations as an educational experience.

When I retired my CP/M system and upgraded to a Heathkit H-151, I updated the program again to run under MS-DOS. The MS-DOS version uses the DeSmet C compiler. Notwithstanding the alleged portability of programs written in a high level language like C, there were enough differences between the two compilers that the conversion was not entirely straight forward. The most exasperating problems resulted from differences in the implementation of standard library functions for console and file I/O. Although the function libraries for both compilers included ostensibly identical functions, subtle differences between them produced some very obscure bugs.

The CP/M and MS-DOS versions of the program are essentially identical, except for the means used to determine the current date. In the MS-DOS version, the date is obtained from the system with a DeSmet library function which reads the system date. Ideally, your computer should have a hardware clock which initializes the system date everytime the system is booted. Otherwise, you will have to include the DATE command in your AUTOEXEC.BAT file, which will prompt you for the correct date and set the system date accordingly. In either case, the procedure which initializes the system date must precede the invocation of the REMIND program in the AUTOEXEC file. Since CP/M does not keep track of the date, the CP/M version of REMIND must read a hardware clock directly. If your system does not have a hardware clock, the get_date() function could be rewritten to prompt you to enter the current date from the console.

Figure 1 depicts the structure of the program. Each box represents a single C function. The portion of the diagram drawn with solid lines represents the functions which are common to both versions. The dotted lines represent the additional functions required in the CP/M version to read the date from a hardware clock.

If REMIND is run with no command line arguments, the dump() routine is called to display any reminder records that qualify for display on the current date. This is the normal mode of operation from the AUTOEXEC batch file. The program may also be run interactively and passed a single character argument from the command line. The argument is interpreted to determine which housekeeping function to perform. If the argument is an 'A' or 'a' the add_rec() function is called to add new data records to the reminder file. An 'E' or 'e' invokes the edit() function, which allows review and selective deletion of data records from the reminder file. Any other character is invalid and results in a warning message and a return to the operating system.

After opening the reminder data file and creating a temporary file which will be used to receive an updated copy of the data, dump() calls get_date() to determine the current date. In the MS-DOS version, get_date() is simply #defined to the DeSmet dates() library function, which returns the MS-DOS system date. In the CP/M version, get_date() calls several routines to read the real time clock hardware. The clk_cmd() function is used to suspend updates to the clock registers while the clock is being read, and to resume the updates afterward. The clock registers are read by repeated calls to clk_digit(), which returns BCD encoded digits for the month, day and year. Finally, the to_str() function is called, which con-

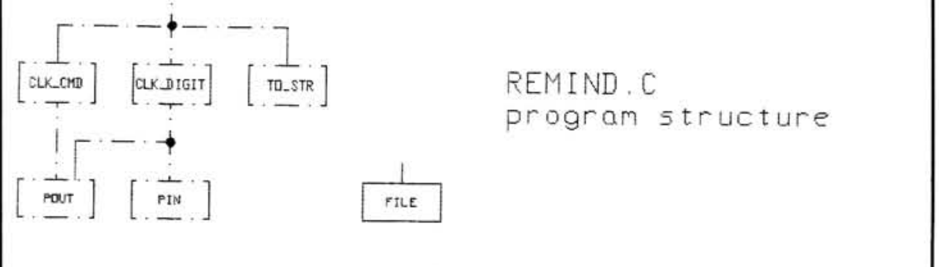


Figure 1

verts the array of BCD digits to the date, encoded as an ASCII string in mm/dd/yy format. The clock control functions depend on two additional functions for reading and writing to the clock ports. The get_date() function and all of its subordinates are specific to my particular hardware and will have to be rewritten for a different system. The conversion from BCD to mm/dd/yy format could have been avoided, but I wanted a get_date() function that I could put in a library for use in other programs.

The next step in the dump() processing is to call parse() to convert the date string for the current date to integer codes for the month, day, and year. The routine now enters a loop to evaluate the individual reminder records from the data file. The start and end dates from the record are processed by parse() to convert them to integer representations, and then the integer arrays for the start, current and end dates are passed to test_date() which determines whether to display the record, and whether the record is out of date and should be deleted.

The test_date() function looks a little strange, even to me. The test is complicated by the requirement to provide for recurring intervals based on "don't care" fields in the dates. First, two flags, 'save' and 'dspl' are initialized to FALSE. These flags will be evaluated later to determine the fate of the record under consideration. Next, if the month, day or year code for either the start or end dates is zero, it is replaced with the corresponding value for the current date, which effectively removes that field from consideration when the end dates are compared with the current date. If either of the end dates contain a "don't care" field, the 'save' flag is set to TRUE, since recurring records are never automatically deleted. The date codes are now converted to a sort of poor man's Julian date scheme which is sufficient to represent the dates in their chronological order, but doesn't accurately represent time differences. Finally, the date code for the current date is compared with the codes for the end dates and if it is within the interval, the 'dspl' flag is set TRUE. The 'save' flag is also set TRUE, if today's date is not older than the end date from the record. The flags are then evaluated to obtain a single digit code which indicates whether the record should be displayed, whether it should be retained, or neither.

The dump() routine completes processing for the record by displaying the message text, if necessary, and by copying the record into the output file unless the return code from test_date() indicates that the record should be dropped. When all records have been processed, the original data file is deleted and the new file is renamed as REMIND.DAT and closed.

The add_rec() and edit() functions are similar to one another and very straight forward. The add_rec() function repeatedly prompts for new reminder records and appends them to the data file. The loop is terminated when a slash character (/) is entered instead of a starting date. The edit() function reads each data record in turn and asks whether it should be deleted.

The display() is called from dump() and edit() to display each record on the console. It takes a single argument which defines whether the dates are to be included in the display. The dates are shown only when the routine is called from edit().

Like the dump() function, edit() creates an updated data file by processing the data into a temporary file, and then renaming it when the processing has been completed. Any record marked for deletion is simply bypassed, instead of copying it into the output file. The low level operations for opening files are in the file() function, which exits directly back to the operating system if the specified file can not be opened. There is a minor inconvenience in the CP/M version of the program due to the fact that the C/80 fopen() function will not create a file if it does not already exist. I get around this by using my text editor to create the REMIND.DAT file containing a single blank. This only has to be done once. The problem does not occur in the MS-DOS version.

Listing 1 is a complete listing for the MS-DOS version of the program. Listing 2 contains the CP/M version of the get_date() function and the subordinate functions required for reading the hardware clock. These functions are necessarily dependent on your particular hardware configuration and will probably have to be changed accordingly. My CP/M system used a QT Systems S-100 clock/calendar board. If you use my routines as an example, it should not be difficult to write comparable functions for reading whatever clock hardware you have.

There are a few changes to the main program listing that are required due to differences between the DeSmet and C/80 compilers. The DeSmet compiler provides a header file (stdio.h) which contains #defines for the logical TRUE, FALSE and EOF. The C/80 compiler does not provide one, so it is necessary either to create one or to include #defines for these constants in the program source file. Second, the MS-DOS listing includes two #defines for the getline() and get_date() functions, which should be deleted if the C/80 compiler is used. The line input routine used in the MS-DOS version uses the DeSmet gets() function (#defined as getline), which requires only one argument. The C/80 getline() function requires a second argument which specifies the maximum number of characters which will be accepted. Therefore, everywhere that getline appears in Listing 1, getline(arg) should be replaced with getline(arg, sizeof(arg)). This should really be done with a #define statement at the beginning of the program, but the C/80 compiler does not allow arguments in #define statements. Finally, in the C/80 version of the file() function, the statement i = fopen(fname,"r") in case 1 must be changed to i = fopen(fname,"rb"). With these changes and by replacing the get_date function in Listing 1 with something equivalent to Listing 2 which supports your particular real-time clock hardware, you will have a version which will run on a CP/M system.

My total immersion approach to learning C may not have been the most painless method, but it was certainly interesting. And this year, when my wife's birthday rolls around, you can bet that I won't forget it.

Are you reading
a borrowed copy of REMark?
Subscribe now!

Listing 1

```
/**************************************************************************
 *                                                                 *
 *            REMIND - A Log-on Reminder Utility                        *
 *                                                                 *
 *                  Larry Conklin                                   *
 *                  105 Riverglen Rd.                                 *
 *                  Liverpool, N.Y. 13090                          *
 *                                                                 *
 *               MS-DOS Version 2.1 - 6/19/87                          *
 *                                                                 *
 * This memory jogger utility is intended to be called following a cold  *
 * boot.  It reads the system clock, and then compares the today's date  *
 * with dated memo records stored in a file named REMIND.DAT. A record  *
 * whose start and end dates bracket today's date is displayed on the    *
 * console.  The routine can also be run interactively to add new memos  *
 * (REMIND A) or review and selectively delete records from the file     *
 * (REMIND E).                                                     *
 *                                                                 *
 **************************************************************************/

#include <stdio.h>

#define BELL 7

#define getline(s) gets(s)           /* DeSmet line input function */
#define get_date(t) dates(t)         /* DeSmet date routine */

struct record
{
   char sdate[9];
   char edate[9];
   char text[81];
} memo;



main(argc, argv)

int argc;
char *argv[];
{
   if (argc == 2)                    /* If there is a command argument */
      switch (*argv[1])
      {
         case 'A':
         case 'a':                   /* If command argument = "A" */
            add_rec();               /* add records to the file */
            break;
         case 'E':
         case 'e':                   /* If command argument = "E" */
            edit();                  /* review and edit the file */
            break;
         default:                    /* Anything else is invalid */
         {
            printf("Invalid command\n");
            return;
         }
      }
   else                                   /* If no command line argument  */
      dump();                           /* dump today's memos*/
}



/**************************************************************************
 * add_rec() - This function is invoked to add new memo records to     *
 *             the REMIND.DAT file.                                  *
 **************************************************************************/

add_rec()
{
   int file1;

   file1 = file("REMIND.DAT",3);

   printf("\n\nEnter dates in mm/dd/yy format.  A '0' entered for any\n");
   printf("field will be interperted as a don't care.  To quit entry\n");
   printf("loop, enter '/' at the start date prompt.\n\n\n");

   do
   {
      printf("\n\nEnter first date memo is to be displayed - ");
      getline(memo.sdate);
      if (memo.sdate[0] != '/')
   {
         printf("\nEnter last date memo is to be displayed - ");
         getline(memo.edate);
         printf("\nEnter text of the memo -\n");
         getline(memo.text);
         putrec(file1);
      }
   } while (memo.sdate[0] != '/');

   fclose(file1);
}



/**************************************************************************
 * edit() - Review and edit the contents of REMIND.DAT. All entries not   *
 *          marked for deletion are copied into a temporary file, which    *
 *          becomes the new REMIND.DAT at the completion of the procedure  *
 **************************************************************************/

edit()
{
   int file1, file2, flag;
   char ans[4];

   file1 = file("REMIND.DAT",1);
   file2 = file("REMIND.$$$",2);

   do
   {
      flag = getrec(file1);          /* Get a record from memo file */
      if (flag != EOF)
      {
         display(1);                     /* Display on console */
         printf("\nDelete this record? ");
         getline(ans);
         if (ans[0] != 'Y' && ans[0] != 'y') /* If not told to delete */
            putrec(file2);               /* copy to output file */
      }
   } while (flag != EOF);           /* Repeat for all records in file */

   fclose(file1);
   fclose(file2);
   unlink("REMIND.DAT");
   rename("REMIND.$$$","REMIND.DAT");
}



/************************************************************************
 * dump() - This routine examines the records in REMIND.DAT and displays  *
 *          any record whose start and end dates bracket the date read    *
 *          from the system clock. In addition, a file house-keeping     *
 *          function is performed. Records read from REMIND.DAT are      *
 *          copied into a new file, but only if the end date of the      *
 *          record is later than today's date. At the end of the        *
 *          procedure the temporary file becomes the new MEMO.DAT.       *
 ************************************************************************/

dump()
{
   int i, file1, file2, flag, test, scode[3], tcode[3], ecode[3];
   char today[9];

   file1 = file("REMIND.DAT",1);
   file2 = file("REMIND.$$$",2);

   get_date(today);           /* get system date, formatted as mm/dd/yy */

   for (i = 0; i < 9; i++)    /* replace any blanks in date string with */
      if (today[i] == ' ')    /* zeros (the blanks mess up parse() */
         today[i] = '0';

   parse(today,tcode);        /* convert date string to m, d, and y codes */

   do
   {
      flag = getrec(file1);          /* get a record from the memo file */
      if (flag == EOF)
         break;

      parse(memo.sdate,scode);       /* decode start and end dates */
      parse(memo.edate,ecode);

      test = test_dat(scode, ecode, tcode);  /* test to see if today */
                                             /* falls between start */
                                             /* and end dates */

      if (test == 2)                /* Dispaly record if */
      {                           /* sdate <= today <= edate */
         display(0);
         putchar(BELL);
      }

      if (test != 0)                /* Keep record only if */
         putrec(file2);             /* end date >= today */
   } while (flag != EOF);             /* repeat for all records */

   fclose(file1);
   fclose(file2);
   unlink("REMIND.DAT");
   rename("REMIND.$$$","REMIND.DAT");
}



/*****************************************************************
 * display(flag) - This routine displays the contents of a memo   *
 *                 record on the console. Expects address of a    *
 *                 structure of type 'record' and a flag that     *
 *                 determines whether the date fields are to be   *
 *                 included in the display (0 = no, 1 = yes).     *
 *****************************************************************/

display(flag)
int flag;
{

   printf("\n\n");
   if (flag)          /* if flag is set, print start and end dates */
   {
      printf("%s",memo.sdate);
      printf(" - ");
      printf("%s\n",memo.edate);
   }
   printf("%s\n",memo.text);          /* print memo line regardless */
}



/**********************************************************************
 * test_dat(s,e,t) - Compare today's date with sdate and edate and return *
 *                   0 if record should be dropped                      *
 *                   1 if record should be saved but not displayed    *
 *                   2 if record should be saved and displayed        *
 *                                                                     *
 *                Expects three three-digit arrays for sdate, edate    *
 *                and tdate                                            *
 **********************************************************************/

test_dat(s, e, t)
int s[], e[], t[];
{
   int scode, ecode, tcode, da, mo, yr, dspl, save;

   s[0] = s[0] - 83;                /* Normalize the year to 1984 */
   e[0] = e[0] - 83;
     yr = t[0] - 83;                /* and initialize mo, da & */
     mo = t[1];                    /* for today's date */
   da = t[2];
  save = FALSE;                 /* Preset flags */
  dspl = FALSE;

  if (s[0] <= 0 || e[0] <= 0)        /* If any field is a don't care */
  {                         /* set corresponding field in */
     s[0] = e[0] = yr;          /* both sdate and edate equal */
     save = TRUE;             /* to today's value, and set */
  }                         /* 'save' flag. */
  if (s[1] == 0 || e[1] == 0)
  {
     s[1] = e[1] = mo;
     save = TRUE;
  }
  if (s[2] == 0 || e[2] == 0)
  {
     s[2] = e[2] = da;
     save = TRUE;
  }

  scode = s[0]*365 + s[1]*31 + s[2];          /* convert all three dates */
  ecode = e[0]*365 + e[1]*31 + e[2];          /* to integer codes */
  tcode =   yr*365 +   mo*31 +   da;

  if ((scode <= tcode) && (tcode <= ecode))   /* set display flag if */
     dspl = TRUE;                         /* today is between start */
                                        /* and end dates */

  if (tcode <= ecode)                      /* save records with end */
     save = TRUE;                          /* dates later than today */

  if ((dspl == FALSE) && (save == FALSE))     /* set return codes based */
     return (0);                          /* on save and dspl flags */
  if ((dspl == FALSE) && (save == TRUE))
     return (1);
  if (dspl == TRUE)
     return (2);
}


/***********************************************************************
 * parse() - Convert mm/dd/yy string into a three digit code: y,m,d.    *
 *          Receives array containing the string and array to receive  *
 *          the 3 digit result.                                       *
 ***********************************************************************/

parse(date_str,buf)
char date_str[];
int buf[];
{
   int i;

   buf[1] = atoi(&date_str[0]);                /* Convert mm field */
   for (i=0; isdigit(date_str[i]); i++) ;      /* Find '/' character */
   buf[2] = atoi(&date_str[++i]);              /* Convert dd field */
   for (   ; isdigit(date_str[i]); i++) ;      /* Find '/' character */
   buf[0] = atoi(&date_str[++i]);              /* Converrt yy field */
}
```

```
/***********************************************************************
 * getrec(file) - Read a record from file into a structure of type      *
 *              'record'.  Arguments are address of the structure     *
 *              and the file channel number.                        *
 ***********************************************************************/

getrec(file)
int file;
{
   int c, i;

   for (i=0; i<=8; i++)
   {
      if ((c = getc(file)) == EOF)
         return (EOF);
      else
         memo.sdate[i] = c;
   }
   for (i=0; i<=8; i++)
   {
      if ((c = getc(file)) == EOF)
         return (EOF);
      else
         memo.edate[i] = c;
   }
   for (i=0; i<=80; i++)
   {
      if ((c = getc(file)) == EOF)
         return (EOF);
      else
         memo.text[i] = c;
   }
   return (0);
}

/***********************************************************************
 * putrec(file) - Write a record into 'file' from a structure of        *
 *              type 'record'. Arguments are address of the          *
 *              structure and the file channel number.               *
 ***********************************************************************/

putrec(file)
int file;
{
   int i;

   for (i=0; i<=8; i++)
      putc(memo.sdate[i],file);
   for (i=0; i<=8; i++)
      putc(memo.edate[i],file);
   for (i=0; i<=80; i++)
      putc(memo.text[i],file);
}

/***********************************************************************
 * file(fname,mode) - Open the file 'fname' and return channel number or, *
 *                  if unsuccessful, print an error message and exit    *
 *                  to MS-DOS.                                        *
 ***********************************************************************/
```

```
file(fname,mode)
char *fname;
int mode;
{
   int i;

   switch (mode)
   {
      case 1:
         i = fopen(fname,"r");
         break;
      case 2:
         i = fopen(fname,"w");
         break;
      case 3:
         i = fopen(fname,"a");
         break;
      default:
         i = -1;
         break;
   }
   if (i > 0)
      return (i);
   else
   {
      printf("Unable to open %s\n",fname);
      exit(1);
   }
}
```

Listing 2

```
/***************************************************************************
 *                                                                        *
 *  This file contains the get_date() routine and its subordinates, for   *
 *  reading the system date from a QT Systems S-100 clock/calendar board.  *
 *  These routines will have to be changed to support another board, but   *
 *  they provide an example of what is needed.                             *
 *                                                                        *
 ***************************************************************************/

*/

The following definitions should be included at the start of the program,
in addition to the definitions given in the REMIND.C listing.

*/

#define CLKDAT 1              /* clock board port addresses */
#define CLKADR 2

/***************************************************************************
 * get_date() - This function reads the real-time clock and returns the   *
 *              date in a character array, formatted as mm/dd/yy. Expects *
 *              the name of the character array that is to receive the    *
 *              date string.                                              *
 ***************************************************************************/

get_date(dptr)
char dptr[];
{
   int i, buf[6], reg;

   clk_cmd(0);                                /* suspend clock updates */
   for (reg = 7, i = 0; reg <= 12; reg++, i++)
      buf[i] = clk_digit(reg);                  /* read a clock register */
   clk_cmd(1);                                /* resume clock updates */
   to_str();                                  /* convert to mm/dd/yy */
}

/***************************************************************************
 * to_str() - Converts date encoded in BCD digits to a string in mm/dd/yy  *
 *            format. buf contains the BCD data, and dptr points to an    *
 *            array to receive the string.                                *
 ***************************************************************************/

to_str(buf, dptr)
int buf[];
char dptr[];
{
   dptr[0] = buf[3] + 0x30;
   dptr[1] = buf[2] + 0x30;
   dptr[2] = '/';
   dptr[3] = (buf[1] & 3) + 0x30;           /* mask off 'leap bit' */
   dptr[4] = buf[0] + 0x30;
   dptr[5] = '/';
   dptr[6] = buf[5] + 0x30;
   dptr[7] = buf[4] + 0x30;
   dptr[8] = 0;
}


/***************************************************************************
 * clk_cmd() - send a command to enable (n = 1) or disable (n = 0) clock  *
 * updates.                                                               *
 ***************************************************************************/

clk_cmd(n)
int n;
{
   in (n == 0)
      pout(0x20,CLKDAT);       /* disable updates if n = 0 */
   else
      pout(0,CLKDAT);          /* enable if n <> 0 */
}


/***************************************************************************
 * clk_digit() - Read the contents of register 'n' from the real-time     *
 * clock.                                                                 *
 ***************************************************************************/

clk_digit(n)
int n;
```

```
{
    int digit;

    pout(n + 0x20, CLKADR);
    digit = pin(CLKDAT);
    return (digit);
}

/**************************************************************************
 * pin() and pout() - routines to read and write data to I/O ports. The  *
 *                    arguments are passed as integers, but only the low- *
 *                    order bits are used.                                *
 *************************************************************************/

pin(port)
int port;
{
#asm
        POP     D
        POP     B
        DB      0EDH,68H
        MVI     H,0
        PUSH    B
        PUSH    D
#endasm
}

pout(data,port)
int data, port;
{
#asm
        POP     D
        POP     B
        POP     H
        DB      0EDH,69H
        PUSH    H
        PUSH    B
        PUSH    D
#endasm
}
```

*

TaxAide

Income tax preparation program for use with your Lotus 1-2-3.

PRODUCES IRS APPROVED PRINTOUTS - NO SPECIAL FORMS TO BUY

LOAD AND USE

Does not require spreadsheet building knowledge

Screen Displays Resemble Actual IRS forms

Current — Contains Latest Tax Reform Changes

Automatically Calculates Income Tax

Easy to Use — Menu Driven - Integrated

Full Technical Support — Satisfaction Guaranteed

Provides worksheets for Forms 1040, 1040A, 2106, 2119, 2210, 2441, 3468, 3800, 3903, 4562, 6251, and 8598; Schedules 1, A, B, C, D, E, F, R, and SE; plus IRS worksheets. Multiple Forms 2106 and 4562 and Schedules C and SE.

$49.95

REQUIRES ONLY 192K of RAM

for Lotus 1-2-3 Release 1A

- Requires 256K for Release 2 using DOS 2.xx.
- Requires 320K for Release 2 using DOS 3.xx.

Requires Lotus 1-2-3 Release 1A or later, and DOS 2.0 or later.

To order, send check to:

Software Applications of Wichita
2204 Winstead Circle
Wichita, KS 67226-1122

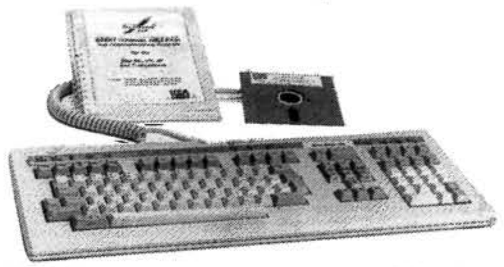
For VISA or MasterCard orders, call (316) 684-0304

(Kansas residents add 5% sales tax, 3½" disks add $5.00)

TaxAide is a trademark of Software Applications of Wichita. Lotus and 1-2-3 are registered trademarks of Lotus Development Corp.

***PowerStation*™**

A Complete VT220 / VT241 Work Station Upgrade for Zenith PC's, AT's and Compatibles



"Now true 800 pixel wide VT241 display and VT220/VT241 support for VMS Services for MS-DOS."

PowerStation™ 240 — **$435**
VT240 style keyboard and ZSTEM VT240 Emulation Software. Optional WPS labelled keys (GOLD KEY MODEL) add $30.

ZSTEM pc™ - VT240 Emulator Emulation Software only. — **$295**
VT240/241 Emulation software with all the features of ZSTEM VT220 plus ZSTEM 4014, sixel and ReGIS graphics.

PowerStation™ 220 — **$289**
VT220 style keyboard and ZSTEM VT220 emulation Software. Optional WPS labelled keys (GOLD KEY MODEL) add $30.

ZSTEMpc™-VT220 Emulator Emulation Software only. — **$150**
All the features of ZSTEM VT100 plus 8-bit mode, downloadable fonts, user defined keys, full national/multi-national modes. Extended macros-/script language. True 132 columns on Hercules, VGAs, Super EGAs, and standard EGAs using the EGAmate option. 128 columns on CGAs. 43 line support on EGAs. Enhanced keyboard support. Ungermann Bass Net/One and VMS services for MS-DOS support.

EGAmate™ — **$39**
Daughterboard option for 132 columns and 800 pixel ReGIS on standard EGA adaptors.

PS220/2 — **$19**
Keyboard adaptor cable for PS200 on PS/2 systems.

ZSTEMpc™-4014 Emulator — **$99**
Use with ZSTEM VT100, VT220, or stand-alone. Interactive zoom and pan. Save/recall images from disk. Keypad, mouse, digitizer, printer, plotter, and TIFF support. 4100 color and line style color mapping. 640 x 400 and 640 x 480 on some adaptor/monitors.

ZSTEMpc™-VT100 Emulator — **$99**
High performance COLOR VT100. True double high/wide, smooth scrolling. ISO and attribute mapped color. XMODEM and KERMIT, softkey/MACROS, DOS access.

KEA Systems Ltd.
#412 - 2150 West Broadway, Vancouver, B.C. Canada V6K 4L9
Support (604) 732-7411 **TELEX** 04-352848 VCR **FAX** (604) 732-0715
Order Desk (800) 663-8702 Toll Free
30 day money back guarantee AMEX/MC/VISA

EXPLORE
NEW WORLDS
WITH
HUG
GAME
SOFTWARE

C_Power Part 10

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

I feel a little like the proverbial "Cat on a Hot Tin Roof" must have felt; due in part, to my attempt at a dual role. I have been writing the program that forms the nucleus for this part of the series, trying to get it ready for this month's article. In the past, I've been able to draw on an existing resource that only needed a little polishing to become suitable for publication. That is no longer the case. One benefit derived from this method of operation is an accelerated learning rate (for me, not you). This is a phenomena enjoyed by all those who attempt to instruct others. In my case, it is particularly acute since my experience with this type of programming is very limited.

My first effort at a word processing program was, like many others, written in BASIC. After finding that it was rather slow (no surprise) and quite limited, I created the next editor using Assembly Language. Of course, I first had to learn the language, which is no small task, but I was aided by a Heath course that proved to be quite enjoyable and very helpful. Of course, this was for the H-89 and CP/M. The new editor was primitive, but quite fast and very helpful, since I lacked a commercial word processor at that time. Probably its biggest success was in providing a vehicle for learning. I soon felt that I could improve on some of the facets that lacked polish in the Assembly Language version and this was about the time I discovered "C". I felt that, with the new tool, I could create a really useful editor.

As those of you who have followed this series from its inception know, "C" is array oriented. I felt that this would be a good basis for my proposed design (I was very naive). Besides, I was a little leery of getting too deep in structures, pointers, and linked lists. Not to mention that I didn't begin to understand them. The end result was an editor that worked, but was a little clumsy and slow, especially when deleting a line near the beginning of a large file. Another disadvantage to this methodology is the need to reserve vast quantities of memory for the array. My dissatisfaction led me to attempt to utilize structures, pointers and linked lists for the construction of the editor featured in this article. Don't close the magazine and shudder! It's not that bad!

One of the major considerations when designing a program to manipulate strings, delete lines, insert lines and create a rather large file in the process, is that of dynamic storage. We need to be able to create storage on the fly, allocating memory as required and freeing memory when not needed. Another requirement is the need to link lines together to form a contiguous file after line deletion or insertion. Further, no screen editor could function without a means of placing the cursor at the desired point within the document and enable character deletion, insertion or even the creation of a new line. The need for this kind of flexibility causes some unique problems that require a certain amount of sophistication to accomplish. Now we are going to look at a way to achieve this.

My concept of a structure in "C" is a conglomeration of parts (variables) that make up a whole, allowing reference to any individual member through the use of pointers and member names. Remember, a pointer is an object which points to (or contains the address of) a particular place in the deep dark recesses of the computer's memory. If we design a structure to hold our string array, with pointers to hold the address of the lines before and after it in memory, as well as a pointer to the first element; then we will be able to locate any individual line through a traversal of the intermediate lines. Think about that for a moment and then examine the following.

The code fragment . . .

```
typedef struct link
          {
          char string[80];
          struct link *next, *prev;
          } list;
struct list *ptr, *base, *runptr, *last;
```

will provide us with a storage location for our string array, a pointer to the address of its predecessor and its successor, as well as a pointer to an individual element. Each of these members is a part of a structure which may be referenced by using a pointer followed by the member name as in: ptr->string or ptr->next. The second example is actually a pointer to a pointer, since "next" is a pointer too (as indicated by the asterisk preceding the variable in the declaration). About now, you are probably thinking that redundancy is fine when it concerns airplane engines, but why do we need a multiplicity of pointers when all we are dealing with is a simple little structure? The answer to that is the key to our problem when dealing with the need to traverse through our document for the purpose of editing.

Each line of the text being edited (or created) is an entity in itself and in order to be manipulated must be addressed. If we employ the code fragment:

```
create()
{
ptr = (list*)malloc(sizeof(list));
}
```

Several things happen at once. First, "ptr" holds the address of a structure which contains the elements shown above. Secondly, the structure created was of a size commensurate with our needs (dictated by the structure declaration). The memory allocation for our newly created entity is from the "heap" (an area of memory not used for program code). Next, we must consider whether this is the first line of our file or not. Let's assume for the moment that it is the beginning of what will be a masterpiece of literary genius. You wouldn't want to lose such a gem, would you? OK, let's use a unique pointer to hold its address, one that will be used for this purpose only throughout the life of the program. We'll use:

```
base=ptr;
```

There, that should do it. Remember, this base pointer must not be disturbed unless the first line of our file is to be replaced. Well, now we can find the first line of our file, big deal. What if the file is to be many lines? I was hoping you would ask that. We'll let runptr = ptr invoke our create() function to provide room for the next line of our masterpiece and link them together with the following code:

```
runptr->next = ptr;
   /* runptr holds the address of first
      line */
   /* runptr->next holds the address of
      second line */
ptr->prev = runptr;
   /* a backward link from the just
      created */
   /* second line to the first
      (runptr) */
runptr = ptr;
   /* runptr now holds the address of
      the 2nd line */
```

This sequence is repeated for each line created. You will notice that we lose direct reference to any given line (excepting the first) after moving on to the next. However, we can access a particular line by stepping from the first through the intermediate lines to the one we wish to occupy. By employing the pointer "last" to hold the address of our last line, we will be able to reach any element of our database by stepping through the individual lines either forward or backward. A very powerful technique which enables the programmer to derive maximum utilitization of the computer's memory.

Now we have the nuts and bolts of the program skeleton designed. One more very important step remains before we get into the actual writing of the source code. Since this is to be a screen oriented editor (is there any other kind worth considering?), we must be able to move the cursor around the screen to any point we wish, while keeping a reference to its location relative to the file being edited. This action presupposes an ability to recognize the cursor control (arrow) keys on the keyboard.

If we could create an input function which would recognize the arrow keys as being unique and then utilize the already created locate(row,col) function, we would have the solution to this immediate problem.

When any of the keypad keys (or function keys) on a PC or compatible are struck, the computer returns an integer. The first byte contains a zero and the second byte a specific character (code). Other keys return the character in byte zero. By testing byte zero for the presence of a 0, we can decode the arrow keys and, consequently, accomplish our objective.

The code fragment:

```
go()
{
union REGS regs;        /* see Turbo C "dos.h". File, must be  */
regs.h.ah=0;                /* "included" with this source code */
int86(0x16, &regs, &regs);  /* interrupt #16 returns an integer */
}
get_ch()
{
int flag;    /* flag to be used in conjunction with a "switch */
               /* case" statement for further decoding */
union scan {
      int c;        /* union is implemented to convert int */
      char ch[2];     /* to two byte array */
         } sc;flag=0;
      sc.c=go();
      if(sc.ch[0]==0)
          /* character returned is arrow */
      { chr=sc.ch[1];flag=1; }
      else chr=sc.ch[0];
          /* character is alpha/numeric */
```

will provide us with just the tool we need for decoding the arrow and function keys. OK, have you got a handle on the above? Alright, let's move along.

Now, we are ready to get on with the job at hand. Bear in mind that the program presented will not be a substitute for WordStar or anything remotely like it. What you will have by the final installment is a good serviceable editor which can be modified by you and included in a larger program or used as a stand alone utility.

This article will include the code to provide the program input module, as well as a linking mechanism which will subsequently enable the building of a linked list. The list will be doubly linked, both in a forward direction, as well as backwards. Why?

We may be working on line twenty-three of a literary masterpiece and discover an error on line one. Moving the cursor up to line one will cause the program to traverse the file, a line at a time, from the current line to the first line in a backwards direction. Of course, the reverse is true when the cursor is moving down. In short, the design of the program dictates that the cursor be on the current line, and since a move backwards (up) will call for the address of the PREVious line (use ptr->prev to obtain address of line moved to). Likewise, a move downwards will generate a need for the address of the NEXT line (obtained through the use of ptr->next). OK, I think you are ready for the source code. Examine it carefully and try to get a thorough understanding of the techniques employed.

Only three include files are listed (more will be added in the next article). The "scrnlib.c" file has been derived from "pclib.c" with a few small changes. The dos.h file is needed to complement the union declarations. You will also note a list of global variables below the define statements.

The "go()" and "get__ch()" functions have already been covered in the previous text down to the "switch-case" statement preceded by an if(flag==1) gate to the body of our arrow decoding routine. Notice the large number of choices available to the routine through the use of a combination of the "if" statement followed by a "switch-case" statement followed, in turn, by an else statement.

Case 75 and 77 correspond to the left and right arrow keys, respectively. The cursor is moved to its new location (thanks to locate(row,col) and the string subscript (i) is modified to agree with the cursor's location. The variable "in" is made to equal zero, acting as a flag for the insert function which will be implemented in the next C__Power article.

Case 72 and 80 are the row modifiers, moving the cursor up or down, respectively. A new pointer is obtained through the use of an algorithm familiar to us by now. If the cursor direction is up, the new pointer is the product of: ptr=ptr->prev and, of course, the reverse is true when moving the cursor down the screen.

The delete character, delete line, insert character and insert line options have not been implemented as yet. They, along with the file I/O, will be covered in C__ Power — Part Eleven.

Moving down to the "else" statement within the switch-case-chr routine, we look for a carriage return (13). When encountered, a newline ('Xn') character is printed, the row variable is incremented if it is currently less than twenty-four, and col is made to equal one. A test is performed to determine if the line occupied is the last line (if(ptr==last)). If the test is successful; "i" is made equal to eighty (forcing an exit from input loop), line is incremented, and ptr->lineno is made equal to the new value of line. If the (ptr==last) test is unsuccessful, then a line other than the last is occupied; hence the pointer is updated (ptr=ptr->next), the skip flag is made equal to TRUE and the character is made equal to zero (VACANT).

When a backspace character (case 8) is encountered, it is printed. A space is printed in the cursor position (to blank out the character previously in that position) and another backspace is printed to force the cursor back to its position prior to the printing of the space. The "col" variable is decremented, a null character is substituted for the character to be eliminated and "i" is decremented by one, allowing a new character to take the place of its predecessor.

Like a bucket at the bottom of this switch-case, function is the "default" statement. Here is the routine which is employed to add the current character to "string". First, "i" is made equal to col-1, the current character is added, then printed, and then "col" is incremented.

The F10 key has been chosen to cause an exit from the input mode of the editor. Pressing F10 will cause the flag to be set to one and a "D" (68) to be input; hence, the if statement below our switch-case decoding module which prevents the inclusion of the characters associated with the arrow and function keys in our text file.

As mentioned previously, a means must be provided to address the first line of our creation; a line requiring a unique link with the lines following it. A function to accomplish this task bears the name "first()" (how imaginative). After furnishing this one-time service, the variable "init" is set to TRUE; providing a means of avoiding a repeat visit to this function. Both of the pointers, next and prev are set to 0 and "runptr" is given the value of "ptr". Look carefully at this function and its running mate: "rest()". Here is the code we discussed earlier in the text, providing a link between the last line input and the current line. Remember, "runptr" holds the address of the previous line when this function is called and given the value of the current pointer before exiting.

OK, hold your hand up if you have a question . . . I don't see any hands, so I guess everybody understands the linking function.

The "fix__line()" routine is a very important adjunct to this program. It enables the placement of characters anywhere on any (existing) line with no thought given to the intervening characters. Spaces are inserted wherever a gap is found between existing text and added characters. Each line is parsed, looking for characters not equal to zero. Len is made equal to the loop counter whenever this occurs. At the exit of this loop, we have an accurate length counter of the line occupied. The line is parsed a second time inserting spaces wherever a zero is found within the line. The variable len is used to determine the number of character positions to be parsed; thus creating an end of line marker (0) by default.

The "build__line()" function begins to put all this together by calling the other functions. It, in turn, is called by means of a loop which tests the value of "end" before each iteration by "main()". The "work horse" of this routine is the for loop which is incrementing i, calling get__ch() and testing the value of the character returned for the presence of "D" (68) while flag is equal to one. The logic of this loop requires that both (D=68 and flag=1) conditions are met or (i=80) before an exit is executed. Examine this code carefully before continuing if you have any questions about how it works. If c=68 and flag=1, then we set "end" equal to TRUE and terminate the current string. If F10 has not been pressed, then the "end" condition is not met and the function returns to its calling routine, "main()".

Besides providing a home for the loop which calls our "build__line()" routine, "main()" incorporates a mechanism for printing our linked list, both from beginning to end and from end to beginning; thus proving a linked list exists in both directions.

Please don't regard this listing as a complete program, it is only a skeleton upon which we will build something quite useful. Make a thorough test of the editor by inputing text and then making changes. Using the arrow keys, you may move the cursor around the screen without regard for spaces or existing text. The cursor is confined to existing lines and is in the "overstrike" mode. We will add an insert character, insert line, delete character, delete line, and file I/O capability to this program in "C__Power — Part Eleven. When testing this module, type only a few lines before pressing the F10 key. A copy of your text input will be displayed in a front to back, back to front manner.

By the time you get to this point in the article, you should have learned a great deal about linked lists. We are going to exploit your new knowledge in the coming articles so stay tuned. I will try and make your efforts worthwhile. "C" you soon.

EDTXT Listing

```
/* ************ Edtxt by John P. Lewis ************* */
/* ************ Ver 1.0, last update 11/16/87 ****** */

#include <dos.h>
#include <stdio.h>
#include ''scrnlib.c''
#define VIDEO 0x10
#define TRUE 1
#define FALSE 0
#define VACANT 0
#define SPACE 32

int flag, init, row, col, i, len, end, in, skip, line;
char string[81], chr;

 typedef struct line
    {
    char strng[80];
    struct line *next, *prev;
    int lineno;
    } list;
 list *ptr, *runptr, *base, *last, *nxtptr, *tmptr;

 create()
 {
 ptr = (list*)malloc(sizeof(list));   /* create storage area for above */
 for(i=0;i <= 79; ++i)                /* clean out the garbage */
 ptr->strng[i]=VACANT;
 }

 go()              /* see commented code in text, this code */
 {                 /* is similar  */
 union REGS regs;
 regs.h.ah=0;
 int86(0x16, &regs, &regs);
 }

get_ch()
 {                        /* see above */
 union scan {
       int c;
       char ch[2];
          } sc;skip=in=flag=0;
        sc.c=go();
        if(sc.ch[0]==0)
        { chr=sc.ch[1];flag=1; }
        else chr=sc.ch[0];
        if(flag==1)
        {
        switch(chr)    /* further decoding of input done using */
         {           /* switch-case function */
         case 75:if(col > 1) col--;i=col-1;  /* left arrow, move cursor */
         locate(row,col);in=0;break;   /* left, decrement string subscript */
         case 77:if(col < 80) col++;i=col-1; /* right arrow, move cursor */
         locate(row,col);in=0;break;  /* right, increment string subscript */
         case 72:if(row > 1) {row--;  /* up arrow, move cursor up */
         ptr=ptr->prev;}             /* get new pointer */
         locate(row,col);in=0;break;
         case 80:                              /* down arrow */
         if((row < 24) && ((ptr->lineno) < line))  /* test for file */
         {row++;ptr=ptr->next;}      /* & screen position, get new pointer */
         locate(row,col);in=0;break;
     /*  case 66, 67, 82, 83 not yet implemented */
         }
        }
          else
              switch(chr)
              {
              case 13: putchar('\n');      /* carriage return */
              if(row < 24) row++;
              col=1;if(ptr==last )          /* test for ptr pos in file */
              {i=80;                 /* force loop exit if last line */
              line+=1;locate(row,col);   /* increment no of line(s) */
              ptr->lineno=line;break;}else   /* store line no */
              {skip=TRUE;chr=VACANT;         /* if not last line */
              ptr=ptr->next;break;}   /* get new ptr, skip linker */
              case  8:putchar(8);putchar(32); /* backspace char */
              putchar(8);col--; /* move cursor back, print */
              ptr->strng[col-1]='';/* space, move cursor back, decrement */
              i-=1;break;       /* ptr->strng subscript */
              default: i=col-1;  /* must be alpha/numeric character */
              ptr->strng[i]=chr;           /* add character to string */
              putchar(chr);               /* print it to scrn */
              col++;break;
              }
          if(flag==1 && chr != 68) chr='';  /* null special keys */
          return chr;
        }

first()
 {
 base=ptr;ptr->prev=0;             /* first line, initialize next */
 ptr->next=0;runptr=ptr;
 init=TRUE;last=ptr;       /* & prev to 0, create ptr to 1st line */
 }

rest()
{
runptr->next=ptr;ptr->prev=runptr; /* link previous line (runptr) */
ptr->next=0;runptr=ptr;last=ptr; /* to this one (ptr), link prev ptr to */
skip=FALSE;           /* previous line - see text */
}

fix_line()
{
int i, len;
for(i=0; i <= 79; ++i)
if(ptr->strng[i]!=VACANT) len=i;if(len < 1) len=0;if(len > 80) len=0;
for(i=0; i <= len; ++i)
if(ptr->strng[i]==VACANT) ptr->strng[i]=SPACE;
}

build_line()
{
char c;
int ln;
if(skip==TRUE) goto skipl;
create();        /* create storage for structure (line) */
if(init==0)       /* see if first line */
first();          /* link as first */
else rest(); /* link as any other line */
skipl:
for(i=col-1; i < 80 && ((c=get_ch()) != 68 || flag != 1 ); ++i); /*exit w/F10*/
/* above line of code is input loop as well as a trap for exit char */
if(flag == 1 && c == 68)
  {
  end=TRUE;
  }
}

main()
{
int lev;     /* initialize variables */
row=1;col=1;end=init=FALSE;cls();line=0;
locate(row,col);
while( end != TRUE)
{
build_line();
}
ptr=base;cls();lev=1;  /* point to first line */
do
{
fix_line();
locate(lev,1);
printf("%s",ptr->strng);            /* print current line */
ptr=ptr->next;lev+=1;            /* get pointer for next line */
} while(ptr->next != 0);              /* until next points to 0 */
locate(lev,1);printf("%s",ptr->strng);    /* print last line & */
last=ptr;                          /* create pointer to last line */
puts("\n End");
ptr=last;
while(ptr->prev != 0 )           /* this code prints file from last */
{                               /* line to first, proving a link */
puts(ptr->strng);ptr=ptr->prev;   /* exists in both directions */
}
puts(ptr->strng);
}
```

SCRNLIB.C Listing

```
/* **************** SCRNLIB.C by John P. Lewis ************ */
/* **************** Created 8/1/87 ********************** */

#define VIDEO 0x10

void locate(x,y)    /* positions cursor at row, col */
int x,y;
{
  union REGS regs;
  x-=1;y-=1;  /* enable first = 1 */
  regs.x.ax=(2 << 8)+00; /* set cursor position */
  regs.x.dx=(x << 8)+y;
  regs.h.bh=0;  /* video page = 0 */
  int86(VIDEO, &regs, &regs);
  }

void cls()
{
  union REGS regs;
  regs.h.ah=7;
  regs.x.cx=(00 << 8) + 00;   /* top left of screen */
  regs.x.dx=(23 << 8) + 79;  /* bottom right of screen */
  regs.x.bx=(7 << 8) + 00; /* 80 X 25 monochrome */
  regs.h.al=0;                /* does not clear the 25th line */
  int86(VIDEO, &regs, &regs);
  locate(1,1);
}

void clr_dn(row1,row2)
int row1, row2;
{
  union REGS regs;
  row1-=1;row2-=1;  /* enable usage first row = 1 */
  regs.h.ah=7;
  regs.x.cx=(row1 << 8) + 0;
  regs.x.dx=(row2 << 8) + 79;
  regs.x.bx=(0x0F << 8) + 00;
  regs.h.al=0;
 int86(VIDEO, &regs, &regs);
}
```

*

Z-100 Software

DOODLER–V Graphics Package	**$ 99.00**
with D–MOUSE Mouse Driver	**$109.00**
with Logitech C7 Mouse & Driver	**$189.00**
Font Library Disk for DOODLER–V	**$ 29.95**
44 Additional Ready–to–use fonts	
Texture & Symbol Library Disk	**$ 29.95**
Hundreds of pictures, symbols, and textures.	
ScreenPro Utility	**$ 59.00**
The ultimate screen print/capture utility.	
Word Scramble Game	**$ 29.95**
Crossword style word game.	

Order toll–free...

VISA **800–346–2152** MasterCard

Or in Florida... 813–376–9347
Unconditional money–back guarantee
if your not satisfied!

Paul F. Herman Inc.

3620 Amazon Drive
New Port Richey, FL 34655

RAM Technology Products

All RAM Technology products include Hardware, Installation Instructions, and One Year Warranty. No soldering is required.

SMARTWATCH: Real-time no-slot clock/calendar module with 10 year lithium battery for the Z-100, all Z-100 PC series computers, PC/XT compatibles and most other MS-DOS computers. Includes software with source code. **$39.95**
Spacer Kit For Z-100 Computers. **$2.00**

Z-100 1.2 MB External Floppy System: Utilize your Z-100's built-in 8" controller for high capacity storage and back-up. System includes drive(s), cables, external power supply and cabinet.
One 5 1/4" 1.2 MB High Density Floppy Disk Drive system. **$249.00**
Two 5 1/4" 1.2 MB High Density Floppy Disk Drive system. **$439.00**

Z-MAX 100: 768K Memory Upgrade for Z-100's with new motherboards; #181-4918 or greater, simple plug-in installation. **$59.95**

Z-MAX 148: 704K Memory Upgrade for the H/Z138/148 computer. **$19.95**

Z-MAX 150: 640/704K Memory Upgrade for H/Z 150/160 and Eazy 150 series computers. Installs on your existing memory board. **$24.95**

Z-MEG 150: 1.2 MB. Memory Upgrade for H/Z 150/160 and Eazy 150 series computers. Up to 704K of addressable memory and 512K RAM DISK can be installed on your existing memory board. Please specify 640K EGA Compatible or 704K version when ordering. RAM-DSK software included. **$49.95**

Z-MEG 171: 1 MB. Memory Upgrade for the ZFL-171. Up to 640K of addressable memory and a 384K RAM DISK can be installed on your existing memory board. Software included. **$89.95**

Z-181 External 5 1/4" Disk Drive: 360K DS/DD 48 TPI Disk Drive mounted in half height enclosure. Interface Cable included. Please specify the model number of your computer: ZFL-181-92, ZFL-181-93 or ZWL-183-92. .... **$249.00**

FREE SHIPPING ON ORDERS OVER $100.00

HARD DISK DRIVES & CONTROLLERS

All Hard Disk Drives come with Installation Instructions, hardware, cables and a One Year Warranty. All controllers capable of handling two drives unless otherwise specified.

PC Compatible Hard Disk Systems

ST-125: 20 MB.. 28 ms., Half Height Hard Disk with Controller.	**$359.00**
ST-225: 20 MB., 65 ms., Half Height Hard Disk with Controller.	**$299.00**
ST-238R: 30 MB., 65 ms., Half Height Hard Disk with RLL Controller.	**$319.00**
ST-251: 40 MB., 40 ms., Half Height Hard Disk with Controller.	**$539.00**
ST-251R: 44 MB., 40 ms., Half Height Hard Disk with RLL Controller.	**$569.00**
ST-277R: 65 MB., 40 ms., Half Height Hard Disk with RLL Controller.	**$619.00**

Bare and AT Compatible Hard Disk Drives

ST-125: 20 MB.. 28 ms., Half Height.	**$299.00**
ST-225: 20 MB., 65 ms., Half Height.	**$269.00**
ST-238R: 30 MB., 65 ms., Half Height.	**$299.00**
ST-251: 40 MB., 40 ms., Half Height.	**$479.00**
ST-251R: 44 MB., 40 ms., Half Height.	**$669.00**
ST-277R: 65 MB., 40 ms., Half Height.	**$699.00**
ST-4038: 30 MB., 40 ms., Full Height.	**$529.00**
ST-4051: 40 MB., 40 ms., Full Height.	**$619.00**
ST-4053: 45 MB., 28 ms., Full Height.	**$639.00**
ST-4096: 80 MB., 28 ms., Full Height.	**$1019.00**
ID-40: 40 MB., 25 ms., Full Height.	**$779.00**
ID-45: 44 MB., 25 ms., Half Height.	**$799.00**
ID-100R: 103 MB., 25 ms., Full Height W/Controller.	**$1225.00**
ID-130: 130 MB., 25 ms., Full Height.	**$2279.00**
ID-230R: 230 MB., 25 ms., Full Height W/Controller.	**$2550.00**
ID-230: 230 MB., 25 ms., Full Height.	**$2314.80**

Hard Disk Controllers: We carry Adaptec, DTC, OMTI, Seagate and Western Digital MFM, SCSI and RLL Controllers for PC's , PC/AT's and compatibles in stock. Call for additional information and pricing.

RAM Technology Hard Cards

RAMDISK-20: 20 MB. Hard Card.	**$389.00**
RAMDISK-30: 30 MB. Hard Card.	**$439.00**
RAMDISK-40: 47 MB. Hard Card.	**$599.00**

Z100 UPGRADE ACCESSORIES

RAM Technology carries a complete selection of upgrades and accessories for your Z-100 computer. Call Toll Free for pricing and availability.

C.D.R. ZS-100: 7.67 Mhz. Speed Upgrade, no soldering required................ **$44.95**
CDR 317 Controller Kit: Interface And Controller. **$575.00**
CDR 317-IIB SCSI Interface: For use with SCSI Hard Disk Drives. **$335.00**

SMARTWATCH: See description under RAM Technology products. **$41.95**

Z-100 1.2 MB. External Floppy Disk Drive Sub-systems: See description under RAM Technology Products. ....From **$249.00**

Z-MAX 100: See description under RAM Technology Products. **$59.95**

FBE RESEARCH ZMF-100A: 768K Memory Upgrade for old motherboards; #181-4917 or lower; no soldering required. **$59.95**

UCI EASYRAM: 2 MB. Memory Expansion Board...................................... **$320.00**
UCI EASYDRIVE: RAM Disk Softwarefor UCI EASYRAM. **$45.00**
UCI EASYIO/2: Provides 2 PC Compatible Serial Ports. **$125.00**
UCI EASY PC: PC emulator board. **$480.00**
UCI EASYWIN: Hard Disk Controller Interface. **$245.00**
UCI EASYWIN: Hard Disk Interface And Controller. **$335.00**
UCI EASY87: Math Co-Processor Board. **$85.00**

Z-100 Hard Disk Drive Sub-Systems

Includes Installation Instructions, Hard Disk Controller, Cables, Hardware, Software and One Year Warranty.

20 MB. Internal Hard Disk Drive System With CDR Controller:	**$725.00**
40 MB. Internal Hard Disk Drive System With CDR Controller:	**$925.00**
65 MB. Internal Hard Disk Drive System With CDR Controller:	**$1075.00**

SERVICE ■ SUPPORT ■ QUALITY ■ VALUE

DYNAMIC RAM MEMORY & PROCESSORS

Prices current 12/21/87 and subject to change. Call toll-free for current prices.

4164-150:	64K 150 ns. DRAM.	**$1.99**
41256-150:	256K 150 ns. DRAM.	**$4.20**
41256-120:	256K 120 ns. DRAM.	**$4.35**
41256-100:	256K 100 ns. DRAM.	**$5.25**
41256-80:	256K 80 ns. DRAM.	**$5.50**
8088-2:	8 MHz. Processor.	**$8.95**
8087-3:	5 MHz. Math Co-Processor.	**$119.00**
8087-2:	8 MHz. Math Co-Processor.	**$169.00**
8087-1:	10 MHz. Math Co-Processor.	**$225.00**
80287-6:	6 MHz. Math Co-Processor.	**$195.00**
80287-8:	8 MHz. Math Co-Processor.	**$269.00**
80287-10:	10 MHz. Math Co-Processor.	**$325.00**
80387-16:	16 MHz. Math Co-Processor.	**$595.00**

NEC V20: 8 MHz. 8088 replacement Microprocessor. Increases CPU efficiency 10-30%. Includes Installation Instructions, One Year Warranty. **$14.95**

NO SURCHARGE ON VISA AND MASTERCARD

MISCELLANEOUS UPGRADES

Just because you don't see it doesn't mean we don't carry it. Call Toll Free for pricing and availability on the accessory of your dreams!

AST ADVANTAGE: AT memory expansion board, EMS/EEMS, expandable to 1.5 MB. One Serial and One Parallel Port. 128K Installed **$379.00**

AST RAMPAGE/2: PC memory expansion board, EMS/EEMS, expandable up to 2 MB., 64K Memory Installed. **$269.00**

MICRON MEMORY EXPANSION BOARDS: 2 to 16 MB. memory expansion boards for PC's and PC/AT's. Call for futher information.

MICROSPEED FAST 88: Speed upgrade for PC's and Compatibles. Increases CPU speed up to 8 MHz. Speed is switchable via software. **$99.00**

PTZ-148 DAUGHTERBOARD: Expansion board for Z-148's provides 2 expansion slots and clock/calendar. **$139.00**

3 1/2" Disk Drive Upgrades: For PC's and Compatibles including the Z-100, Z-100 PC, Z-200 PC and Z-300 PC series computers with standard 5 1/4" enclosures.
FDD-3530: 720K, for all systems, requires MS-DOS 3.2. .... **$139.00**
FDD-3550: 1.4 MB., for Z-200, Z-300 systems, requires MS-DOS 3.21. .... **$169.00**

CALL TOLL FREE TO ORDER!

RAM Technology

427-3 Amherst Street ■ Suite 265
Nashua, New Hampshire 03063

1 800 662-0070 ■ 603 889-0633

Hours Of Operation: Moday - Friday 9:00 A.M. - 5:00 P.M.

The Fine Print

Prices and specifications subject to change without notice. Personal checks held 10 working days; money orders accepted as cash. COD orders accepted; cash or cashier's check only. No surcharge for VISA or MASTERCARD orders. Free shipping on orders over $100.00 via UPS in Continental U.S.A, all other orders are charged our cost of shipping (min. $3.00). Purchase orders accepted from qualified businesses, government agencies and educational institutions.

Programming The HV-2000 For Fun . . . And Profit!

Terry Perdue
Heath Company

In the January issue of Remark, I introduced Heath's new HV-2000 PC Voice Card. As an inexpensive kit that goes together easily in an evening and has many novel and useful applications, Marketing expected it to be a popular product, but it's been selling at over twice their forecast!

I suspect that many HUGgies have found or are creating unique applications for it that hadn't even occurred to us. At the end of this article, you'll find information on a contest that will give you a chance to win a prize in return for sharing your applications with others. At the conclusion of the contest, HUG will compile a library of programs and routines, and make it available on disk.

In the meantime, Heath's Technical Correspondence Department has received a number of requests for more information on how the HV-2000 can be made to sing and produce sound effects. So, in the next few sections, I'm going to offer three short BASIC programs to get you started, as well as some programming hints and information on interfacing directly with the hardware, bypassing the device driver. This discussion assumes that you are already familiar with the material in the HV-2000 manual.

Making The HV-2000 Sing

It apparently wasn't clear in the manual that simply specifying a Note attribute (nN) does not cause any sound to be created. Instead, it determines the pitch of any phonemes subsequently spoken. For example, to sing 'DO, RE, MI' from the DOS prompt, you could use the command

```
SPEAK "={N d o w pa 2N r a ie pa 4N m
e ie pa}"
```

Note: The '=' in the string returns all attributes to their default values. This includes defaulting to Mode 0 (transitioned inflection), which is not appropriate for singing. However, when the device driver sees the first note attribute in the string, it automatically switches to Mode 1 (instantaneous inflection), so that the notes will occur at the proper pitch.

If you are writing a program to sing a song, it is usually best to use phonetic spellings of the words (in braces), because you then have control over the timing. You can change the duration of individual phonemes in a word and/or add additional phonemes until the duration of each word is just right.

As an example, Program 1 sings part of Lara's Theme, then plays the tune with two-note 'chords' (actually alternating notes).

To aid in picking note values, Figure 1 is an illustration of a section of a piano key-

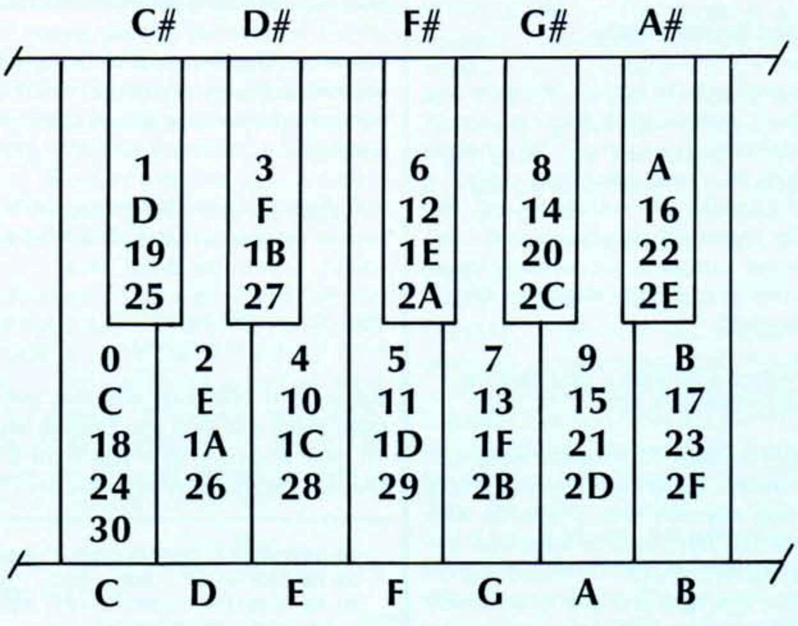


Figure 1
Hex Note Values

```
10 OPEN "O",1,"HV":PRINT#1,"=":CLOSE
20 DATA "10N 1b 3aw 3aw 3aw 3er 13N 3r 3uh 3uh 3pa 1BN m 1ah ie"
30 DATA "1CN 11b 3uh2 3v 3pa 3pa 3pa 3pa 3pa 3pa 17N th 1e 1ng k"
40 DATA "1AN 3uh2 3v 0pa 18N 1m ie pa 13N n ahl 1u w 3pa"
50 DATA "12N 3ae n d 11N thv eh eh n 3pa 3pa 3pa 3pa 3pa 0k g 3aw 3aw d pa"
60 DATA "13N s p 3e 3ie d 3pa 15N m 1ah ie 17N 1b uh2 uh2 v 3pa 3pa 3pa"
70 DATA "3pa 15N t i 31 0pa 13N yi u ul 0pa 12N 1aw er"
80 DATA "11N m 3ah 3ie 3n 3pa 1AN uhl 3uhl 0k 18N g 3ehl 3eh en n 3pa 3pa"
90 DATA ""
100 READ A$:IF A$="" THEN 120
110 A$="{"+A$+"}":OPEN "O",1,"HV":PRINT#1,A$:CLOSE#1:GOTO 100
120 DATA 6,"18","1C",4,"1F","1C",2,"27","23",12,"28","24",2,"23","1F"
130 DATA 2,"26","23",2,"24","1F",4,"1C","1F",2,"1E","1B",12,"1D","1A"
140 DATA 6,"1D","17",4,"1F","1A",2,"21","1D",12,"23","1F",2,"21","1D"
150 DATA 2,"1F","1C",2,"1F","1B",4,"1D","1A",2,"26","1D",7,"24","1C",0,"",""
160 OPEN "O",1,"HV":PRINT#1,"{1D F0F}":CLOSE#1
170 READ N,F$,S$:IF N=0 THEN 200
180 A$="{"+F$+"N e "+S$+"N e}"
190 OPEN "O",1,"HV":FOR X=1 TO N:PRINT#1,A$;:NEXT X:PRINT#1,"":CLOSE#1:GOTO 170
200 OUT &H300,0:END
```

Program 1
Lara's Theme

board, with corresponding N values for four octaves.

Producing Sound Effects

Many sound effects consist of producing some basic sound, then altering one or more attributes. For example, to simulate the ringing of a bell, you could output a fricative to make the striking sound, followed by some voiced phoneme for the 'ring' sound, output at successively lower amplitudes to cause the ringing to decay. For example:

```
SPEAK "={FA d a CA a AA a 8A a 6A a 5A
a 4A a 3A a 2A a 1A a A}"
```

Note the attempt to simulate an exponential decay. To get fancier, you might try altering not only the amplitude with time, but the filter and/or inflection frequency slightly, as well. Try improving on the above example and see how closely you can simulate a bell.

Another example is Program 2, which simulates a scene from Star Wars. Here Filter frequency, Amplitude, Rate, and Inflection are the attributes being altered.

A small problem exists if you want to produce a continuous sound effect, such as the steady hiss of a gas leak. The device driver was written with voice synthesis as its primary purpose. When speaking a string of text, you don't want the last phoneme that was output to continue sounding — you want it to terminate. Otherwise, a string ending with the word 'hello', for example, would sound a continuous 'O'. A string ending in a fricative, such as 'test', would terminate, but the final 't' would not be output until the next voiced phoneme or Pause phoneme was output.

Therefore, the device driver automatically outputs a Pause phoneme when it encounters the carriage return at the end of a string.

You could output a long string of 'h' phonemes to give a reasonably long hissing sound, as with the BASIC line:

```
OPEN "O",1,"HV":PRINT#1,"{h h h h h h h
h h h h h h h}":CLOSE#1
```

But even if you loop this line over and over, there will be a short pause inserted on each iteration. So if you want to concentrate on sound effects, and find this to be a problem, here is a patch you can make to the device driver. Be sure to copy the original VOICE.SYS to a backup file, say VOICE.BAK, first.

To make the patch, type the following at the DOS prompt:

```
DEBUG VOICE.SYS <return>
FCS:2211 L3 90 <return>
W <return>
Q <return>
```

Then copy the patched VOICE.SYS to your root directory, and reboot. Then type:

```
SPEAK "hello"
```

You've got a continuous sound now, don't you?! To shut it up, type:

```
SPEAK "{pa}"
```

As you can see, using the patched device driver requires that you make a point to terminate your strings with a Pause phoneme yourself. But now if you want that hissing sound, just type:

```
SPEAK "{h}"
```

Program 3, which requires that you make the above patch to work properly, will produce a wind storm simulation that is convincing enough to send a shiver down your spine. It simply generates the wind sound with an h phoneme, and randomly ramps the filter setting up and down to simulate changes in wind speed. To make the effect more realistic, the amplitude is made to track the 'speed'. (From BASIC, you can stop the sound by writing a Pause

```
10 OPEN "O",1,"HV":PRINT#1,"={D}":CLOSE#1
20 P$="F0F o D0F o B0F o 90F o 70F o 50F 40F o 30F o 20F o 10F o F o pa}"
30 A$=HEX$(INT(12*RND(1)+3)):R$=HEX$(INT(5*RND(1)+10)):
   I$=HEX$(INT(255*RND(1)))
40 FOR X=1 TO 5*RND(1)
50 OPEN "O",1,"HV":PRINT#1,"={"+"D "+I$+"I "+R$+"R "+A$+"A "+P$:CLOSE#1
60 NEXT X
70 FOR T=0 TO 1000*RND(1):NEXT T:GOTO 30
80 OPEN "O",1,"HV":PRINT#1,"{200I}":CLOSE#1
```

Program 2
Phasor War

```
10 OPEN "O",1,"HV"
20 PRINT#1,"={FA h}":CLOSE#1
30 O=N:N=INT(200*RND(1)):S=1:IF N<O THEN S=-1
40 FOR X=O TO N STEP S
50 OPEN "O",1,"HV":A=INT(X/15+2):PRINT#1,"{"+HEX$(A)+"A "+HEX$(X)+"F}":CLOSE#1
60 NEXT X:GOTO 30
```

Program 3
Wind Storm

phoneme directly to the hardware using the BASIC command OUT &H300,0. This assumes that your board is jumpered for address 300H.)

After making the patch, run Program 1 again, and notice that the pauses between the two-note chords have disappeared, giving a more pleasing sound.

Program 2 could be simplified now by outputting the desired phoneme once, and using a FOR-NEXT loop to step through the descending Filter frequencies. This would also make it easy to randomly select the phoneme used to produce the basic sound.

Bypassing The Device Driver

Although the device driver and software utilities included with the HV-2000 were written to provide a simple, yet versatile, interface to the product, there may be applications where you'll want to bypass the device driver, and interface with the hardware directly. In doing so, you should be aware that the SSI-263 has a number of idiosyncrasies that were discovered and compensated for during the writing of the device driver. So if something doesn't work quite as expected, you may need to experiment a little. If you're still game, the following information should be helpful.

Refer to Table 1 for the function of each of the eight addressable registers. Most of the bits shown in the table are self-explanatory, except for the following:

DR1,DR0 — Define the phoneme duration, where 0,0 is the *longest* duration.

P5-P0 — Specify the desired phoneme per Table 2.

The C bit, when set, disables (powers down) the SSI-263 chip. It also selects the inflection mode in the following manner. If bits DR1=DR0=1 when the C bit is taken low, Mode 0 is established. If DR1=1 and DR0=0 when C goes low, Mode 1 is selected. (The other two combinations should not be used.)

T2-T0 — Define the articulation speed.

Reading from any of the eight registers returns, in bit 7, the Ready status of the chip. In polled operation, on outputting a phoneme to the Duration/Phoneme register, bit 7 will read low for the duration of the phoneme. When it returns high, you can output the next phoneme.

Offset From Register		**Bit Position**							
Base Addr	**Name**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
0	Dur/Phoneme	DR1	DR0	P5	P4	P3	P2	P1	P0
1	Inflection	I10	I9	I8	I7	I6	I5	I4	I3
2	Rate/Ext Inflection	R3	R2	R1	R0	I11	I2	I1	I0
3	Ctrl/Speed/Amp	C	S2	S1	S0	A3	A2	A1	A0
4	Filter Frequency	F7	F6	F5	F4	F3	F2	F1	F0
5	Disable								
6	Enable								
7	Reset Interrupt F-F								

Table 1
Register Bit Assignments

If you don't load the device driver, be sure to perform a Write to register 6 to enable the board at the beginning of your program.

Keep in mind that the SSI-263 was designed for speech synthesis, not sound effects. The sound effects that can be simulated using the various sounds and attributes that are available can be useful in some situations, but are definitely limited.

The Contest

Because there are so many potential applications, it's hard to set rules or restrictions for program entries to this contest. Your entry could be a game, an educational program, such as a typing teacher or language tutor, an aid to the handicapped, a talking appointment calendar, or any other useful, entertaining, and/or novel application. It need not be long or complex. It may be written in Assembly, BASIC, or some other common language, and can take advantage of the features of the device driver, or talk to the hardware directly. The HUG staff and I will, as judges, just have to look over (and listen to) all the entries, and fight it out amongst ourselves!

The deadline for submittals will be May 31, 1988. You need not be a member of HUG to enter, and you may submit more than one entry. All entries should be submitted on disk, and will become the property of Heath Users' Group. The following prizes will be awarded:

First Place: $300 HUG Buck
Second Place: $200 HUG Buck
Third Place: $100 HUG Buck

The winners will be announced, and their entries presented, in a subsequent issue of REMARK.

So get started, and good luck!

MNEM	**HEX CODE**	**MNEM**	**HEX CODE**	**MNEM**	**HEX CODE**	**MNEM**	**HEX CODE**
a	08	er	1C	1b	3F	t	28
:a	3A	f	34	1f	22	th	36
a1	09	hf	2C	m	37	thv	35
ae	0C	hfc	2D	n	38	u	16
ae1	0D	hn	2E	ng	39	:u	3C
ah	0E	hv	2A	o	11	u1	17
ah1	0F	hvc	2B	:oh	3B	uh	18
aw	10	i	07	oo	13	:uh	3D
ay	05	ie	06	ou	12	uh1	19
b	24	iu	14	p	27	uh2	1A
d	25	iu1	15	pa	00	uh3	1B
e	01	j	31	r	1D	v	33
e1	02	k	29	r1	1E	w	23
e2	3E	kv	26	r2	1F	y	03
eh	0A	1	20	s	30	yi	04
eh1	0B	11	21	sch	32	z	2F

Table 2
Phoneme Codes

HUG Price List

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!

HEPCAT is here! *HEPCAT* is here! *HEPCAT* is here! 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. Order ***HUG P/N 885-3045-37***.

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!

H8 — H/Z-89/90

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
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)

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
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-[37]	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 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
REMARK VOL 7 ISSUES 72-83	885-4007	N/A	1986	25.00
RF CAD	885-8020-[37]	CPM	UTILITY	30.00

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
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-[37]	CPM	COMMUNICATION	20.00
H/Z-100 (Not PC) Only				
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	885-3006-37	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
GAMES PACKAGE II	885-3044-37	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
KEYMAC	885-3046-37	MSDOS	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
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				
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
HEPCAT	885-3045-37	MSDOS	UTILITY	35.00
HUG BACKGROUND PRINT SPOOLER	885-3029-37	MSDOS	UTILITY	20.00
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
MATT	885-8045-37	MSDOS	MATRIX 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/A	1986	25.00
SCREEN DUMP	885-3043-37	MSDOS	UTILITY	30.00
UTILITIES II	885-3014-37	MSDOS	UTILITY	20.00
PC Compatibles				
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
LASERWRITER CONNECTION	885-8050-37	MSDOS	PRINTER UTILITY	40.00
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***.

ORDERING INFORMATION

For VISA and MasterCard phone orders, telephone the Heath Users' Group directly at (616) 982-3838. Have the part number(s), descriptions, and quantity ready for quick processing. By mail, send your order, plus 10% postage and handling ($1.00 minimum charge, up to a maximum of $5.00) to: Heath Users' Group, P.O. Box 217, Benton Harbor, MI 49022-0217. VISA and MasterCard require minimum $10.00 order. No C.O.D.s accepted.

Questions regarding your subscription? Call Margaret Bacon at (616)982-3463.

Continued from Page 10

Sincerely,

Ray Haythornthwaite
50, Foothills Drive
Nepean, Ontario
CANADA K2H 6K3

Problems Running Programs With The Paradise Graphics Card

Dear HUG:

I have written software using GW-BASIC on my Zenith 158. When the disk is booted up, a menu appears which can be used to run programs. Everything works well except when I attempt to use a Zenith with a Paradise Graphics Card the menu appears, but if I choose to run a program using Screen 1 format, nothing happens.

Can you advise on how to resolve this problem?

Thank you,

Tony Patricelli
1913 Hopi Lane
Mt. Prospect, IL 60056

When CHKDSK Tries To Run, It Corrupts Files On The Partition!

Dear HUG:

I have an antique Z-100 which I refuse to part with because it still does everything I ask, and I prefer its design to anything that has come into general use since. I also appreciate the fact that Heath/Zenith has supported the machine as well as it has. Because I expect to keep my Z-100 for some time, I usually accept any offers Heath makes to update my softwaare, especially system software. Recently, a major software incompatibility has surfaced on my system.

I am running the latest operating system for the Z-100 that I have been offered, MS-DOS 3.1R, OS-63-30. However, to accommodate a few programs, I also have a Z-DOS partition on my hard disk with the appropriate utilities on it. Thus, the Z-DOS partition includes version 1.1 of CHKDSK.

As you probably know, when running a program by default, DOS searches the current directory for the program unless a drive/directory is specified. If it cannot be found in the current directory, DOS version 2 and later will search the PATH for it. Thus, if I run CHKDSK from my Z-DOS partition after having booted up with MS-DOS 3.1, CHKDSK 1.1 will try to run, since it is on the default drive, rather than PATHing to the MS-DOS 3.1 partition.

The problem that arises is that CHKDSK 1.1 attempts to run, rather than reporting "Incorrect DOS version" as later versions do. And when it tries to run, it corrupts most of the files on the partition! CHKDSK will attempt to correct certain "errors" without permission (i.e., no /F switch or equivalent is needed). What happens is that you get a large number of Allocation errors and File Size errors, and the size of the files is truncated. In addition, it appears that files may become cross-linked somehow. It is very difficult to recover from this situation unless you can rebuild the FAT tables and directory on a cluster basis.

Therefore, if you are running both MS-DOS 3.1 and Z-DOS from hard disk partitions where you might transfer from one to the other, I recommend that you rename the CHKDSK on the Z-DOS partition so that it will not run if you boot from MS-DOS. Otherwise, keep your backup current.

Sincerely,

Francis B. Grosz, Jr.
3620 Canal Street
New Orleans, LA 70119

Last Chance

Dear HUG:

Last chance for Floppy Disk Controllers and RAM Boards for the H-89. Due to an increased interest for H-89 products, we will manufacture one more run of these two products *IF* we receive enough orders by March 1, 1988. We would need the orders with a check, money order, VISA or MasterCard number (not to be cashed or charged until the order is shipped). If enough orders are not received to make another run feasible, all checks and money orders would be returned.

Please contact:

C.D.R. Systems, Inc.
7171 Ronson Road
San Diego, CA 92111
(619) 560-1272

Steven Vagts' "Side-Bar: Understanding Your Printer's Escape Codes

Dear HUG:

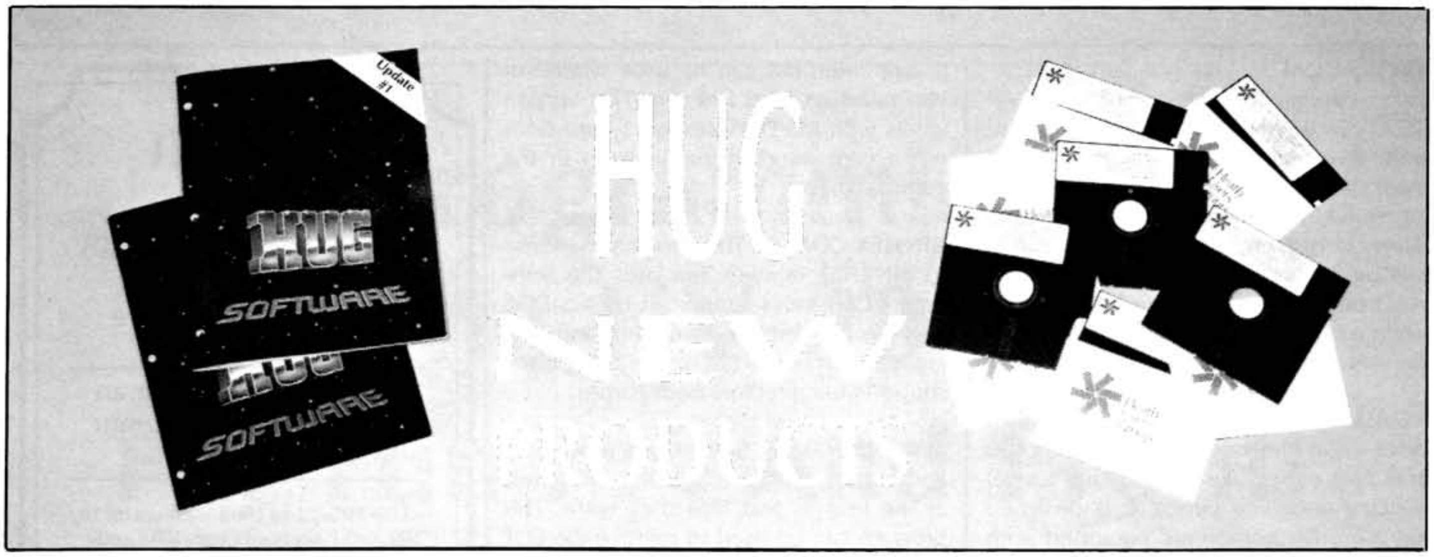
In his informative "side-bar" article "Understanding Your Printer's Escape Codes, Steven Vagts made one little slip, correction of which might make life easier for the less-experienced users. He described use of the ESC key as "holding down the "ESC" key while typing the . . ."

Since ESC is a separate character (ASCII decimal 27), one should press it in a normal, sequential fashion. In particular, ESC is not the same type of a key as SHIFT, CTRL, or ALT, but is more like a letter or a number. The SHIFT, CTRL, or ALT keys have no numeric code of their own, but they modify the numeric code sent for a letter or number key.

I hope this helps.

Al French
201 Central Avenue
Jefferson, LA 70121

EXPLORE
NEW WORLDS
WITH
HUG
GAME
SOFTWARE



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; telephone Heath/Zenith Users' Group directly at (616) 982-3838. Have the part number(s), description, and quantity ready for quick processing. VISA and MasterCard require minimum $10.00 order. By mail, send your order, plus 10% postage/handling ($1.00 minimum, $5.00 maximum) to: Heath/Zenith Users' Group, P.O. Box 217, Benton Harbor, MI 49022-0217. Orders may be placed, by mail only, using your Heath Revolving Charge account. Purchase orders are also accepted by phone or mail. No C.O.D.s accepted.

Questions or problems regarding HUG software or REMark magazine should be directed to HUG at (616) 982-3463.

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 885-6011-37
PS's PC Utilities $20.00

Introduction: This disk contains a number of handy utilities developed by Pat Swayne (HUG Software Engineer) for use with PC-compatible computers. Among the utilities on this disk are a character undelete utility for WordStar, a fast alphabetizing directory program, and a collection of drawing libraries for use with four different CAD/drawing programs.

Requirements: These programs are designed to be used on any PC-compatible computer, such as the H/Z-150 or H/Z-200 series. Most of the programs will work with any version of MS-DOS or PC-DOS, but some require version 2 or above. The programs are all small enough to work in minimum memory configurations.

Program Author: Patrick Swayne, HUG Software Engineer

The PS's PC Utilities disk contains these files.

README	.DOC	NOPRTSC	.COM
INSTRUCT	.DOC	NOPRTSC	.ASM
WSUD	.COM	CLICK	.COM
WSUD	.ASM	CLICK	.ASM
D	.COM	CBEEP	.COM
D	.ASM	CBEEP	.ASM
SSCK	.COM	BIN2HEX	.COM
SSCK	.ASM	BIN2HEX	.ASM
KEYWS	.COM	BINHEX	.COM
KEYWS	.ASM	BINHEX	.ASM
NOBLINK	.COM	FIXEND	.COM
NOBLINK	.ASM	FIXEND	.ASM
BLKCUR	.COM	ACAD	<DIR>
BLKCUR	.ASM	CADD	<DIR>
PROCAP	.COM	PD	<DIR>
PROCAP	.ASM	SKETCH	<DIR>

Here is an explanation of the files:

INSTRUCT.DOC — This file contains the instructions for using the programs and drawing libraries on the disk.

WSUD.COM — This program provides character undeletion for the WordStar word processing program. It is a shell that runs WordStar under its control, so that it can watch for character deletions. The deleted characters are stored in a buffer, and can be undeleted by pressing an Alt key combination. This program works with either version 3 or version 4 of WordStar.

D.COM — This is a new version of Pat Swayne's alphabetizing, columnizing directory program previously released as DIR100.COM and DIR150.COM. This version was designed specifically for use with MS-DOS versions 2 and above, and handles file specifications on the command line in exactly the same way as the DOS DIR command. It also features a much faster screen display than previous versions, configurable colors, and contains the same Shell-Metzner sort routine as the previous versions, making it one of the fastest alphabetizing directory programs around.

SSCK.COM — This program combines Pat Swayne's screen clock program with Jim Buszkiewicz's screen saver program. It provides an on-screen digital clock display in the upper right corner of your screen that works while you run other programs. It also can be set to blank the screen (to save the phosphors in your monitor's CRT) after a user configurable period of non-use. The screen clock and screen saver operations can be disabled independently, so that you can have just a screen clock or a screen saver.

KEYWS.COM — This program is a specially configured version of KEYMAP (HUG p/n 885-6001-37) designed for use with WordStar version 3.3. It provides more functionality with the keypad and function keys than you get with WordStar alone. It is a memory resident program that can be loaded into a batch file when you boot up, and then can be called up when you run WordStar by pressing a special key combination.

NOBLINK.COM — This program provides a non-blinking block cursor that you may find easier to see than the normal blinking underline cursor. It is designed for use with computers equipped with Heath or Zenith CGA- or MDA-compatible outputs, or with computers equipped with EGA cards containing Zenith Data Systems ROM's. Once this program is loaded, you can turn the non-blinking cursor on or off as you need it.

BLKCUR.COM — This program provides a block cursor instead of the normal underline cursor. It is similar to NOBLINK except that it does not affect the blinking of the cursor, and works on non-Zenith equipment. The block cursor will normally remain in effect after you load the program until you re-boot.

PROCAP.COM — This program protects the Caps Lock key from being pressed accidentally. It causes it to work only when another key, such as the left Shift key, is pressed with Caps Lock.

NOPRTSC.COM — This program disables the Shift-PrtSc combination, so that your computer will not lock up if you accidentally press it when no printer is connected.

CLICK.COM — This program provides a key click for Z-171 computers that works just like the key click on Z-150 series computers. Once the program is loaded, you can turn click off or on by pressing Alt-Esc.

CBEEP.COM — This program is the demonstration program presented in the article "An Introduction to TSR's" that appeared in the November 1987 issue of REMark. It provides a concurrent "beep" that does not cause the computer to halt while the beep is sounding.

BIN2HEX.COM — This program converts binary files to the Intel hexadecimal format. It was designed to provide files compatible with the Heath IDS-4801 EPROM programmer, but can be used whenever you need an Intel hex file. This version works with MS-DOS version 1, and does not accept directory path names in the command line.

BINHEX.COM — This program is similar to BIN2HEX (above), but uses the Standard I/O devices supported by MS-DOS versions 2 and above for reading and writing files. It is not as fast as BIN2HEX, but it supports full directory path names.

FIXEND.COM — Some word processing and editing programs put EOF characters at the end of text files they write. This program can be used to remove the EOF characters when the files are to be processed by programs that do not accept EOF characters. Although the COPY command in MS-DOS version 3 or above can be used to strip EOF characters, this program does it much faster, because it only works on the end of the file rather than copying all of it.

WSUD.ASM, D.ASM, etc. — The source code for all of the programs on this disk is included.

ACAD <DIR> — This is a directory containing a library of symbols for use with AutoCAD in drawing schematics containing logic gates (AND gates, OR gates, etc.). The included symbols are the same style you see in professional schematics. A drawing that shows all of the symbols is included, as well as a sample schematic drawn using some of them.

CADD <DIR> — This is a directory containing a library of symbols similar to those described above, but for use with Generic CADD.

PD <DIR> — This is a directory containing a library of symbols for use with Prodesign II or DesignCAD.

SKETCH <DIR> — This is a directory containing a library of symbols for use with AutoSketch.

*

HFS - III

The New Home Finance System from Jay Gold Software

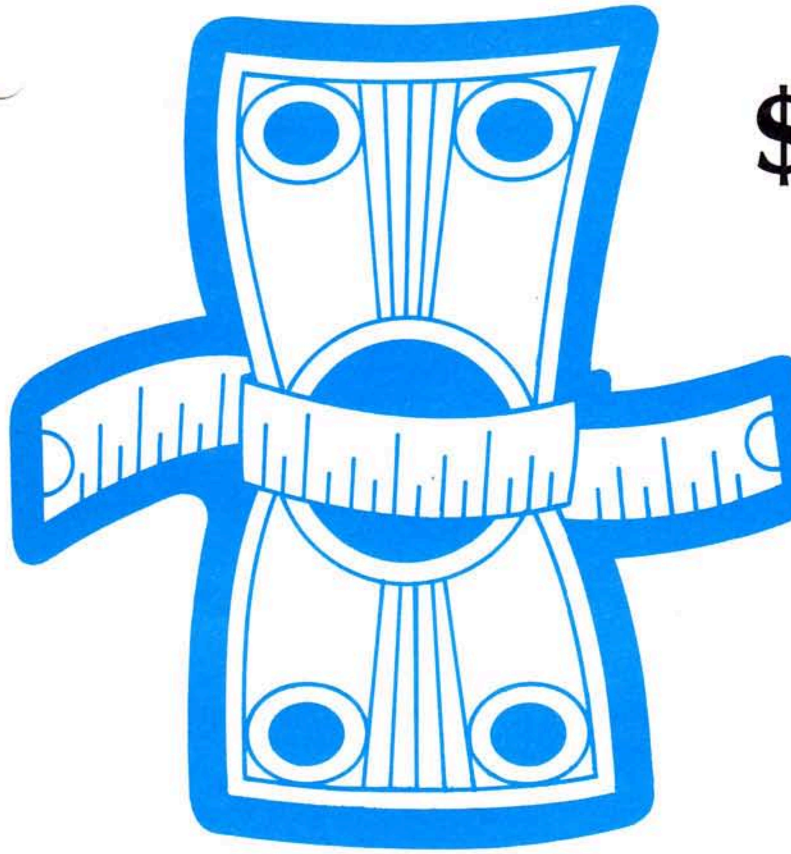
You don't have to be an accountant to keep your accounts in order.

- Use HFS-III to track cash in up to 100 Asset Accounts and 100 Credit Accounts. Number of transactions limited only by disk space.
- **Prints checks** in any of 3 formats that you design using any standard business-sized check. Prints payee address (if desired) for use with window envelopes.
- Asset Accounts include **checking, regular asset accounts** such as savings, IRAs, CDs, or cash, **parent accounts** that keep track of the total balance of their "children" and **reserve accounts** which let you hide funds in your checking account until you need them.
- **Model transactions** allow commonly used transactions to be recalled for later entry with a few keystrokes (number of models limited only by disk space)
- Easy selection of accounts and codes from **pop-up menus** or by entry of name or number.
- Help boxes available at a keystroke.
- **Fast.** 100% assembly language code for the fastest operation possible.
- Up to 100 user defined expense codes and 15 deposit codes. Assign up to 14 expense or deposit codes to each transaction (including separate tax flag).
- Accepts HFS-II data.

Hardware: HZ-100, all Heath/Zenith PCs or any other PC/XT/AT compatible (256K), 2 disk drives, any printer.
Software: MS-DOS 2 or higher.
Price: $99.00 (includes shipping).
MasterCard/Visa accepted, please include phone number.

Order from:

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



$100 (or less) Software For The Z-100

Doris Wagner
19567 Jersey Avenue West
Lakeville, MN 55044

The subject of this article is a brief review of PC software which runs on your Z-100, using only HUG ZPC emulation. The emphasis here is on inexpensive software, either low cost commercial or user supported and/or public domain.

A few words about my background. Some years ago, I had an H-89. The machine and I got along well until a lightning bolt struck. Thirty seconds late to a plug is as good as thirty years. The H-89 was never the same again.

About that time, I contracted to do some software manual writing for a company who thoughtfully provided a PC for my use at home. When they most unthoughtfully went away, the machine stayed. It had one-half of PC-DOS and a word processor. I have kids who refuse to eat or wear software. My interest in affordable and alternative software goes way back, and I have collected a lot of it.

Recently, I acquired a Z-100. Since I already had all that software, the next step was to see what would run on the Z. As Z-100 systems go, it's fairly plain: "standard Z" with 768K, speed up kit, dual floppies, dual eights. Most testing was done using version 2.0 ZPC, but no ZHS board and no patching.

Hard Disk Installation on 8-inch Disks

My first discovery was that a lot of software which is looking for disk swapping or a hard drive can run happily installed on an eight inch disk. You may or may not have to "explain" the installation to the software via MAP commands. Usually, finding all program elements (such as a word processor and its associated dictionary) on drive C: or the default drive is enough. In other cases, the software insists upon being installed from floppy disks in Drive A: to the hard disk. As long as the target drive is C: or above and big enough, the installations worked beautifully.

The Z-100 arrived in the midst of a hectic civic campaign which found me doing a lot of data base work. I was using d-BASE II to organize names, addresses, phone numbers and interests. Most of the data entry was done on my machines. There were, however, 772 additional names and associated data available to us from an XT under Symphony.

A print-out of that list showed that it was set up to make labels --only. It carried a 30 character organization field. Apparently, someone planned to type Junior Jaycees, etc. but quickly settled on JC, etc. A dozen or more JC, etc. A dozen or more of these "name changes" produced a list which didn't sort on organizations worth a darn.

The solution proved to be Reflex. This huge program requires a PC with color graphics or EGA, high memory, hard disk preferred. The program needs 384K and the data base is entirely memory resident. On a dual drive system, things get tight. With the program disk on A: and the data disk on B:, I had to make do without the HELP program on drive B: (it's also a full disk, so forget combination plans).

A graphics only program on the Z-100? Well, why not?

All three disks (main program, help disk and report disk) of Reflex installed on an eight inch disk ran flawlessly under ZPC. Reflex has fine report and translation abilities. My first attempt at translation didn't work, because the Symphony file had "DUPLICATE FIELDS" which Reflex refused to consider. Reflex does have a useful OPTION key. The choices included something called MAIN_DB. A swift cursoring down plus carriage return, and all 772 entries flew across in less time than it takes to tell the story.

Once in Reflex, the data base could be easily altered. With the whole thing in RAM, you can do anything you jolly well please, quite rapidly. If things get hopelessly messed up, you can abandon the whole effort, secure in the knowledge that your original data base waits patiently on disk. When it is correct, you can write the new data base out to a new file while still not altering the old one.

There were six single letter code fields in the d-BASE file. It was a simple matter to add six columns in the Reflex data base, and code them according to the information in that 30 character organization field. The revised data base was then saved to disk. There is one catch: the database must not exceed available memory, and it's amazing how fast even modest efforts grow. There is a newer version of Reflex which does accommodate high RAM machines. Version 1 handles life at 704K just fine.

Going from Reflex back to d-BASE was harder. While the program would write the file to disk in d-BASE format, I had a public domain program which apparently dated from CP/M d-BASE. For whatever reason, my original data base was set up with blank spaces, instead of comma delimiters. I could instruct Reflex exactly which fields, in what order and of what size to write to an ASCII (or SDF, standard delimited file) format, but it took a few tests. An early effort printed to screen in readable double spaced format. It wrote out, and read into dBase, where it proved to contain 1,544 records. Record 1 was good, Record 2 was blank, and so forth and so on. A few tries later, the translated file appended to the old file, sorted nicely, and never gave a bit more trouble.

The final list totalled 2,024, in 150K worth of disk space. There were duplicates, but sorting by phone number made these easy to find. When it was all done, I had a lot more respect for data base programs, Reflex in particular, and the Z-100.

Obviously, if Reflex worked, it made sense that the Reflex Workshop would also work. This is a full disk of application templates, from facilities management to personnel tracking to accounting and financial projections.

I started testing more stuff and results have been most interesting. Sure, I hit some klunkers. But there were many pleasant surprises. Exact version numbers tested and the approximate prices are given in the product list; sometimes a later or perhaps earlier version of a program might work better than the tested one.

More Programs That Work

Another favorite program of mine is Timeslips, which keeps detailed records of both time and expenses for small and medium sized organizations. The nice point about this program is that it lets you print the billing statements as you see fit. Editing may be done even during the selection process; statements may be written up or down discretely, flat fees may be used for time and/or expenses, retainer accounting is carefully done.

Timeslips also runs without any hitches and much more conveniently on an eight inch disk than on floppies. The program has three screen drivers: oddly enough, memory (for "extremely compatible machines") works as does RomBios (for "not very compatible machines") while Snow (for "slightly compatible screens") does not. Timeslips can be used either Foreground (batch entry of time slips) or Background (memory resident, logs your time on the phone to the correct client if you remember to turn it on and off again). The memory part doesn't work on the Z-100, surprise, but the rest is excellent programming, a reasonably priced package doing more than many of its competitors.

Another inexpensive package which runs fine is DacEasy Accounting. This is a total accounting package which again prefers life on an eight inch disk. Some folks find DAC a trifle inelegant; its price (you can get the program, a tutor disk and help for under $100) inclines its users to overlook the frills. It does a fine no-nonsense job of managing accounting work.

Knowing that pfs-Write had needed some patching, I had little hope for pfs-First Choice. A little knowledge could be dangerous; the program runs beautifully.

Two for the Rest of the Family

First Choice is a package which would be a splendid introduction to computers and a useful tool for a spouse who would rather have had new laundry equipment back when you bought your Z. This is especially true if she happens to also run a small business. The documentation is clear enough for almost total beginners (although you might want to revise the key names and locations) and walks you through a hot air balloon business called Flights of Fancy. It has charm without insult.

First Choice is an integrated package with a word processor, an adequate spell checker, a spreadsheet and "File Management", really a small data base program which keeps track of information, organizes lists and provides for the creation of forms for reports. For routine business use, this looks like an excellent package. The local retail does exceed our $100 limit by a few bucks, but not nearly as much as you'd expect for this sort of program.

A graphics program which looked better on the Z-100 than on the PC was NewsMaster. This is affordable desktop publishing for newsletters and the like. There are two disks in the set and if you happen to have PrintMaster, one or more disks, you can use them too. All of that works better on a big eight inch disk.

NewsMaster tries very hard to be easy. Choices are shown in "icons" -- screen graphics to show what they do -- and there is a help key. Still, placing and resizing pictures (you can shrink or expand them, pull so that they get fatter or thinner, crop them, expand the white space, etc.) takes six levels of menus and some dedication.

The program will import ASCII files, so your text can come from your word processor, but you can import only one page at a time. Over 100 printers are supported. While I am used to making my elderly Epson "over achieve," what this program does to it is astonishing.

On my Z, there was one small hitch. The program claimed that it didn't have enough memory, a message which usually translates to mean that the program found more memory than it could handle. Sure enough, MDISK removing 200K or so did the trick. The PC has 640K which is the same amount as ZPC was told to use. It pays to experiment.

Sometimes, I made good guesses. I figured that the Turbo Gameworks package would run fine. It does beautifully, but my next problem was that I couldn't even beat Go-Moku (a child's game, they say) and so did not try the Chess version offered.

User Supported Software and Public Domain

There is a huge world of alternative software beyond the software found in stores. It has many names, user supported being my choice. It's also called shareware, although that term is copyrighted by an early innovator in the field. User supported differs from public domain in that it usually does carry a copyright notice and a request for payment. Public domain is sometimes copyrighted, so that others cannot take the program and attempt to make a profit, but it asks for no reward other than happy users.

I have a long and profitable experience with this software, and I have found many programs to be as good, and sometimes better, than commercially distributed software. It began as an "alternative." Some people wrote some good programs. They tried the traditional distribution methods, but things went wrong. These people didn't care for copy protection. In fact, their credo was that a user should be able to try the software before investing. Registration usually entitles you to updates and support. Thus, the term user supported. If you use it, support it!

The main problem is the immensity of the aggregate collection of available software -- would you believe something on the order of 200 to 300 MEG worth? I was the PC-compatiable librarian for my HUG group a couple of years ago and recently got drafted as resident advisor. The world of user supported software was huge before, and it's bigger now.

My time as librarian suggested some general guidelines. Programs written in C seem most likely to run. Compiled BASIC generally had the biggest problems.

Ironically, the four programs mentioned first do not follow these rules, which may suggest how useful any generalizations are. DacEasy is compiled BASIC; Reflex is ported from a larger machine, perhaps C code; Timeslips is written in Turbo Pascal, First Choice appears to be Pascal, but not Turbo. Thus, it all depends on how the program is written. Screen addressing and keys can be sensibly done. If my limited testing is any indication of anything, it may be a trend toward less machine-dependent code.

The public domain includes literally hundreds of utilities. If you are thinking of writing your own special utility, you might wish to check what's available first. Many utilities will run on all MS-DOS machines, from PC to Z-100 and more. It may be that programmers writing these programs are more careful to write machine independent codes, which in turn might be a reflection of their awareness of the many available machines. In some cases, the code is from CP/M or UNIX.

An excellent example of a fine utility is the ARC program. This compresses your files, after first checking to see which of several internal compression methods will work best. (Making the wrong choice on your own could actually expand the file in some cases.) Later versions allowed you to print the documentation or run the program without first extracting it from its archive. While you see transmission time, it's also useful to be able to compact and store files that you want to keep, but wish to save disk space in storing. ARC is pure MS-DOS; no emulation is necessary.

The last version which I have is ARC512. Beware of a nasty thing once found on the bulletin boards called ARC513 -- it proposed to do nasty things to your hard disk. WHILE THERE ARE MANY FINE PROGRAMS AROUND, A FEW ROTTEN EGGS MAKE IT ADVISABLE TO KNOW THE BOARDS FROM WHOM YOU DOWNLOAD, AND TO ISOLATE FOR TESTING ALL SUCH SOFTWARE!

(If you are interested in being totally safe, you might wish to join PC-Sig. They are a reliable vendor of public domain and user supported software, totalling over 750 at a recent count. PC-Sig software is also distributed by many authorized dealers, such as Newline Software. PC-Sig publishes a magazine which gets bigger with every issue.)

(Another excellent library of user supported software is PC-Blue, a less market oriented group and perhaps harder to find in your area. PC-Blue offers full disks, often of compressed (archived) programs and their documentation. My impression is that this library includes more public domain and MS-DOS (as opposed to PC specific) programs than other groups offer. PC-Blue is up to around 350 disks.)

Not only is almost everything available through various libraries and bulletin boards, bewildered newcomers often find that many of the programs are large and complex. A single program can fill one or more disks and easily rival commercial software. While you can find disks full of various utilities, the emphasis is on the large packages.

The large programs are faster and easier to test as well. What follows is a random sampling of my disks. My collection goes back three years, so often I do not have the latest version in hand. Version numbers are given, and most programs should be still around.

Word, Data and Number Working Programs

A recent entry in word processing software is New York Word. This was originally written for UNIX systems and commercial distribution (target price, $950.) The PC version tested well on the Z, or at least as far as I could demo it briefly. This full-powered program includes macros and 300K worth of documentation; learning could take awhile, even with the help screens.

PC Desk was a surprise. An integrated "desk" program, it includes a word processor, a data base of mailing addresses, a high-powered calculator and a calendar to remind you of whatever (it comes instructed to inform you about April Fools' Day). While some functions appear resident, they are all actually on disk, accessed rapidly and selected by plain number (not function) keys.

Another editor which sort of worked (with some "halts and limping") was Screen. This one is a programmer's editor, so much so that the default extension is .BAS. Perhaps an earlier or later version would work better.

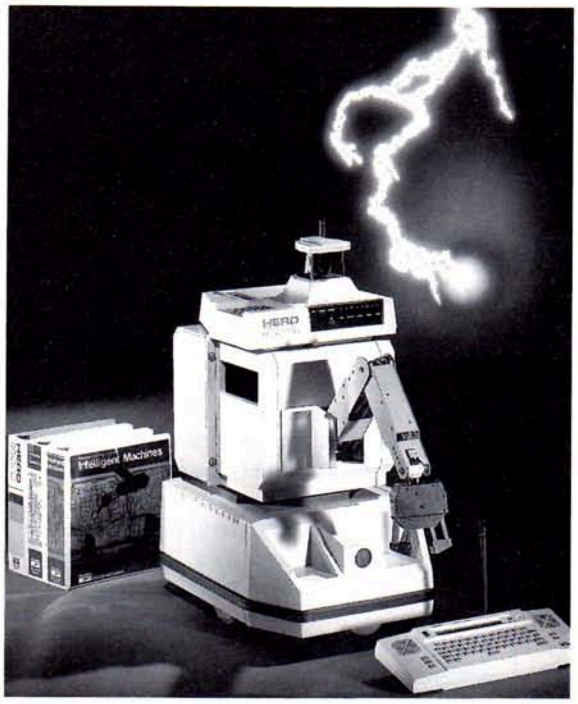
A data base program which ran for me was 3 by 5. This program produces "index card" records which sort with marvelous speed and can be used with some word processors. It looks especially good for applications in which you don't really want to bother with a major data base program, such as modest records of magazines, books, index cards for term papers, whatever.

A home accounting package which appeared and loaded its demo programs just fine is Time and Money. It's a simple yet reasonably complete home finance manager. While it doesn't exactly approve of "shoe box" records, it will accept them, asking only that you number them. (At last, an accounting package which speaks my language.)

MAKE LEARNING AN ADVENTURE

Heathkit/Zenith Educational Systems

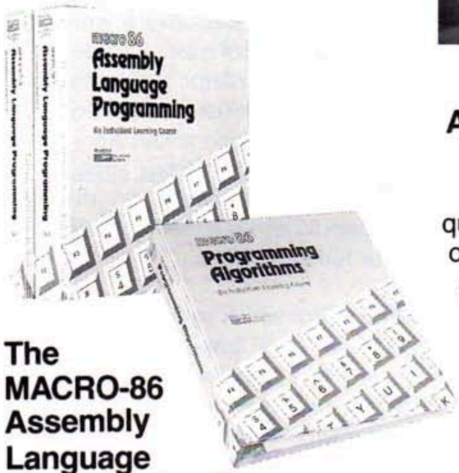
Feature success-oriented courses with a unique blend of technical theory and real-world applications. With our advanced course trainers and the hands-on experience they give you, you'll quickly gain valuable electronics skills. And see exciting electronics concepts come to life before your eyes. Plus 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.



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

I'm still testing spreadsheets. While those I have came with the code, they didn't test well. But there's another one... In user supported software, there are always many alternatives.

More Games...

The field of games is rich and fine, although many are compiled BASIC and stubbornly machine dependent. A helicopter game ran its demo quite nicely, but died when a key was pressed; I'll ask my kids if it's worth patching. Many action games slow down dismally. Text adventure games usually run, if anyone still cares.

A personal favorite does just fine: Hack. This is a superlative version of a dungeon adventure game, originally from UNIX and years in the transportation. There are monster descriptions from all the best in monster literature, more ways to move, more ways to die than even HUG DND offers --and it was the favorite at my house. Hack takes a big machine: 384K minimum, 640 and up preferred, three disks and/or a hard disk. It should run native on a 100, but in true adventure tradition, Hack must be lovingly configured. A PC-configured version ran nicely, something of a surprise since that version comes with NANSI, a fast ANSI driver, and a TERMCAP file, both of which take study. For good measure, if you have a PC configured version, be sure to load ANSISYS with ZPC and set it ON.

(The author of Hack has ported a new dungeon game called Larn, which has just arrived at our house and looks slightly different, equally enchanting. My son is doing the testing.)

Software of All Descriptions

Utilities don't necessarily come in small packages. I collect disk utilities and corrupt disks to test them on. While emulation in this work is not usually a good idea, some are known to work with ZPC. To that list, add the Ultra Utilities. There are three modules: U-File, U-Zap, and U-Format; all quite useful for doing things to disks. Note: in the version which I have and tested, U-Zap and U-Format worked fine. The biggest program is U-File, which has a different opening screen. That display will hang the machine ... BUT you can bypass the opening screen by typing an ESCape while it is loading. Then, U-File comes up fine.

Whatever your fancy, there are programs around. For artificial intelligence, Esie runs perfectly and includes excellent documentation.

Serious writers (or graduate students) might like Peter Norton's book indexing tools, even though they date to 1983. It's semi-automatic (you enter the words, but it does the page numbers and sorting). Code included; rewrite the function key assignments and use on the Z-100 with no emulation needed.

Code for All...

Programmers will find a wealth of public domain code. With minor revision for C code and Pascal, perhaps greater revision for BASIC, this should all run fine. Without emulation.

Related programs, that is worksheets, templates and programs for d-BASE, Lotus and others also abound. Most can be customized to your application. If you have the a Z-100 version of the main program, the worksheets and subprograms usually work with minimal fiddling. The programs which I used for the campaign work were public domain. I had to do a bit of work in order to get what I wanted from them, but for free, I don't think complaining is in order.

There is room in a short article to only scratch the surface of what does work under ZPC on a Z-100, especially from user supported software. My suggestion is that you experiment and share your successes.

Product list:

Prices are local (Mpls-St. Paul) for commercial software or as suggested by the version listed for user supported software. User supported software is indicated by an asterisk (*) in front of program name. If no price is given for user supported, it's genuine public domain. Consult local libraries or bulletin boards for user supported and public domain software.

* 3 by 5, version 1, copyright 1984, Adware, Softshell Corp., P. O.Box 18522, Baltimore, MD 21237. Demonstration program for consulting firm, newer versions available.

* ARC512, copyright 1985-6, System Enhancement Associates, 21 New Street, Wayne, New Jersey 07470, contributions.

* Book Indexing Preparation Programs, copyright 1983, Peter Norton, 2210 Wilshire Blvd., #186, Santa Monica, CA 90403.

Dac Easy, version 1, copyright 1985, Dac Software, Inc., $59; tutor, $19. Revised and C versions available.

* Esie, version 1.1, copyright 1985, Lightwave Consultants, P. O. Box 290539, Tampa, FL., 33617. Suggested contribution, $75. New version available.

First Choice, version 1.0, copyright 1986, Software Publishing Corporation; $119.

* Hack, version 3.6, D. G. Kneller, 2 Panoramic Way, #204, Berkley, CA 94704.

* Larn, version 12, copyright 1986 by Noah Morgan, ported to MS-DOS by D. G. Kneller, 2 Panoramic Way, #240, Berkeley, CA 94704.

NewMaster, version 2, copyright 1986, Kyocera Unison, Inc.; $79.

* New York Word, version 1, copyright, 1984, Mark Adler, 138 - 23 Hoover Ave., Jamaica, NY 11435. Suggested contribution, $35. Newer version available.

* PC-Blue, New York Amateur Computer Club, PC-Blue Users Group, P.O. Box 3442, Church St. Station, New York, NY 10008. Membership in NYACC, $15 year; disks, $7.

* PC-Sig, 1030D East Duane Avenue, Sunnyvale, CA 94086. Membership, $20; catalog, 12.95; disks, $6 plus shipping.

* PC Desk, version 3.2, copyright 1985, Software Studios, 8516 Sugarbush, Annandale, VA 22003, $25; advertised also at $50.

Reflex, version 1.0, copyright Analytica Corporation, 1984-5. Distributed by Borland International, $99; Workshop, $44. Newer version available.

* Screen, version 3.0, copyright 1983, 1985, Basic Business Software, P. O. Box 26311, Las Vegas, Nevada, 89126, suggested: $35.

* Time and Money, version 1, copyright, 1984, Middletown Programming Works, P. O. Box 4099, Middletown, New Jersey, 10940. Suggested contribution, $50.

Timeslips, version 2.24, copyright North Edge Software Company, 1985-86, $79 (hard disk version, $149.) Newer and hard disk versions available.

Turbo Gameworks, copyright 1985, Borland International, $44.

* Ultra Utilities, version 4.0, FreeSoft Company, P.O. Box 27608, St. Louis, MO 63146. Suggested contribution, $40.

Copyright, 1987. Doris A. Wagner *

SPEED MODS - **Assembled and Tested** - $34.95
- H/Z 150/160 Software Select 6.67 or 4.77 MHz
  Hardware Reset included. Satisfaction Guaranteed
- H/Z 89 Software Select 2 or 4 MHz. No Trace Cuts
  Z80A, CP/M and HDOS Software included.

H/Z 89 20 Meg Hard Disk **only** $495!
Boots from Hard Disk. Supports CDR and Magnolia.
All Software and Hardware except Hard Disk $175

H/Z 150/160 Hardware: Smart Watch $ 39
20 Meg ST-225 $269 (drive only $239) 30 Meg $299
MT PAL 640/704k RAM $19.95 Super PAL 1.2 Meg $49

MS-DOS and CP/M Software:
Perfect Funds $29 MT Accoutant $19 Paycheck $39
Perfect Money $19 Perfect Printer $19 T Pascal $65

Micronics Technology **(904)-897-4257**
449 Barbados Way, Niceville, FL 32578
Checks, VISA, MC. Shipping $2, Hard Disk $10
Call or Write for our FREE complete Catalog

Z150/160 HARDWARE ENHANCEMENTS

TURBOPLUS V2.0

Highest Performance From Your 150/160
7.4 or 8.0 MHz Operation (SI=2.8 or 3.1)
Software or Hardware Speed Select at Any Time
Reset From Keyboard at Either Speed
100% Software Compatibility
Correct DOS Clock
Simple, Reversible, Installation
Complete With PC Board, Software, V-20, and High Speed Chips

TurboPlus V2.0 WORKS *Where Others Fail!*

Price $125 ppd.

VMM150 VIDEO CARD ELIMINATOR

Use *ANY* Video Card in 150 (even GB-1)
REMOVE the Heath/Zenith Card
Free One Standard Size Slot
Use Less Power, Run Cooler
Simple Plug-In Installation
Includes PC Board, Decoder and New Back Panel

Price $45 ppd.

TERMS: Check or Money order, UPS COD $1.90 extra.
Dante Bencivengo, P.O. Box 234, Wyandotte, MI 48192
(313)484-4866

ABSOLUTE 15 DAY MONEY BACK GUARANTEE

FBE Products

For the H/Z-150, 160 Series

MegaRAM-150 — Modification kit allows memory board to be filled with 256K RAM chips (1.2 MByte). No soldering. Supplied with RAM disk software. ***$49.95***

ZP640 PLUS — Replacement PAL for standard memory board allows up to 2 banks of 256K and 2 or 3 banks of RAM chips to be installed for 640K or 704K maximum memory. ***$24.95***

COM3 — Replacement PAL allows installation of three serial ports (one an internal modem). Supplied with printer driver software for 3rd port. ***$39.95***

FBE Smartwatch

Calendar/Clock using Dallas Semiconductor's DS1216E SmartWatch module. Works with H/Z-110/120, 138/148, 150/158. Package includes SmartWatch with our software and documentation. Spacer kit ($2) required for Z-100. ***$44.95***

For the H/Z-100 Series

ZMF100a — Modification package allows installation of 256K RAM chips in older Z-100 without soldering. Works only with old-style motherboard. ***$65***

ZRAM-205 — Kit allows 256K RAM chips to be put on Z-205 memory board to make 256K memory plus 768K RAM disk. Requires soldering. PAL ($8) required for new motherboard. ***$49***

For the H/Z-89, 90 Series

SPOOLDISK 89 — 128K byte electronic disk and printer interface/spooler card. ***$195***

H89PIP — Dual port parallel interface card. Use as printer interface. Driver software included. ***$50*** Cable ***$24***

SLOT4 — Extender card adds 4th I/O expansion slot to right side bus. ***$47.50***

FBE

FBE Research Company, Inc.
P.O. Box 68234, Seattle, WA 98168
(206) 246-9815, M-F 9-5

UPS/APO/FPO Shipping Included.
VISA or MasterCard Accepted.

Computers, Music And MIDI

Part 2

T. E. Thompson
61 Sampee Road
Ottawa, Ontario
CANADA K1V 9T9

PC Compatible Owners Take Note: *Even though the original computer used for the completion of this article was a Heath H-89, the Asynchronous Communications Element (8250 ACE) is the same device used for the 'COM' serial ports in all PC compatibles. Further, a good portion of the support software is written in a higher level language, making it easily transportable. The Z-80 assembly language code could easily be hand translated to 8088/96 code. -ed.*

Last month, we talked about some details of MIDI and its history and usefulness. This month, I will discuss available music hardware and an interface for the H-89 computer. There are a wide variety of hardware interfaces and software available commercially for IBM PCs and compatibles and Macintosh, Commodore 64, Amiga, Atari ST and other types of personal computers. Some computers, such as the Atari STs, have a MIDI interface built in and for others both simple and complex interfaces are available. There are, however, no available facilities for the H-89.

This month, we will cover the details of making a MIDI port for the H-89. Why the H-89? Well, there's some life in the old girl yet. I bought my '89 in May 1981 and it has had some very heavy use since then. I have built several add-on devices and I have made all the normal modifications to power connectors and motherboard. When I became involved in computer music, I could not justify the expenditure required for a new computer just to handle a MIDI port, so I decided to try my luck with just one more interface for the venerable '89. All in all, it worked well and was a lot of fun, too.

This article will offer some results of my experimentation. I found very quickly that there are some problems with the '89 in this application, particularly when dealing with the system 2mS clock. The clock handler in the boot ROM has an overhead approaching 200 microseconds for every clock tick. This is because of measures taken to support the H8 architecture while using a boot ROM that was compatible between the H8 and the H-89. Normally, this overhead is not a problem, but when the MIDI data bytes are arriving at a rate of one every 300 microseconds, it is an insurmountable difficulty. There just is not enough time left to do any significant processing between arriving data bytes. This problem will be addressed a little later, but it is the main limitation on the ability of the H-89 to record real-time MIDI inputs from a keyboard.

Now that the bad news is out of the way, here is what can be done on an H-89. I can play 6 voices at once on several synthesizers from pre-entered data files. I can record voice parameters, save them to disk and reload them to the synthesizer using a patch librarian program. I can experiment with real time sound modification, random new sound generation and other interesting MIDI tricks. Last, but not least, I can release some of those creative instincts that are lurking in me.

All these capabilities are available with the interface that is described in the paragraphs to follow. The software that I will cover next month is far from being a saleable product, but there is a lot of functionality included in a form that should be fairly easy to modify and expand. This month, I have included a simple MBASIC diagnostic program that you can use to check the interface and to do some limited MIDI functions.

Music Equipment

I promised last month that I would offer some advice on the purchase of a keyboard for computer music studio use, so I'll do that before I go on to the interface details. I use a Sequential Circuits SixTrak synthesizer for my work. This machine has been bypassed by the very latest in chrome plated high-tech wonders, but there are still very few machines that can match its flexibility at any reasonable price. The fact that I use this particular unit has an effect on the software as one must make some use of manufacturer system exclusive MIDI capabilities, particularly in building a patch librarian. I have attempted to indicate where such machine specific codes have been used in the software. The features of the SixTrak that make it especially suitable are listed below:

1. Multi-timbral — It can play up to 6 completely different sounds at once. Few synthesizers can do so. Ask the salesman to demonstrate how your prospective purchase could play a flute line with a piano accompaniment. This 6 voice limit appears in the software packages.

2. Built-in sequencer — It can record and play back on its own without an attached computer.

3. MIDI ports — Note that many inexpensive new keyboards do not have MIDI port connectors. Make sure that your purchase has.

4. Real-time parameter control — The voice parameters can be modified in real time very easily using the defined MIDI controller commands. This is not a common capability.

5. Manuals — Full documentation of the capabilities of the MIDI implementation are included in the manual. Some manufacturers do not supply adequate data.

6. Price — The SixTrak is available on the used market for around $400, an excellent price for the capabilities.

The big names in keyboards today are Yamaha, Roland, Korg, Sequential Circuits and Casio. Almost any piano store will carry at least one line, but the best place to see the full selection is in a rock instrument shop among the drums and guitars. Such stores love to see a healthy credit card for a change and are glad to help.

MIDI H/W And S/W Available

So far, I have concentrated on keyboard instruments. There are far more types of MIDI equipment than one might, at first glance, expect. MIDI is used to interconnect the following instrument types today:

- synthesizers
- drum machines (sound and rhythm pattern generators)
- drum pads (rubber, artificial drum head surfaces)
- sequencers and MIDI recorders
- microcomputers
- mechanical pianos with retro-fit MIDI OUT packages
- electric guitars
- audio mixer consoles
- special effects and sound processors
- light controllers
- unusual and experimental input control devices
- wind instruments

Consider the possibilities of using a single electric guitar to control 6 different synthesizers that are generating the sound of a full orchestra, as well as the lights and drums! There are even devices that will evaluate an audio input signal (such as a flute or singing voice) and generate the appropriate MIDI note-on and note-off commands. The orchestra plays along with you.

For various machines and at various levels, the following types of software are available:

1. Sequencer software — For recording, editing and playback of MIDI command sequences.

2. Music notation software — For recording MIDI command sequences and producing a printed page sheet music output.

3. Patch editor and librarian software — For modifying and storing the parameters necessary to reproduce a specific instrument sound.

4. Music training software — For beginning and advanced training in music theory and practice. Remember that with a MIDI keyboard, the computer can play a sequence of notes, ask for the student to repeat the sequence and check the student's work for accuracy. There are MIDI devices that will translate a sung note into the appropriate MIDI commands that a voice training program could interpret.

This software is not difficult to construct, in theory, and the routines in next month's installment will illustrate many of the techniques required.

One of the best sources for the latest market information and lots of detailed data on MIDI and its application is Keyboard Magazine (1), a magazine for professional keyboard musicians. Keyboard can often be found on newsstands or in music stores.

H-89 H/W

The hardware for the H-89 MIDI interface follows the recommendations of the MIDI 1.0 Specification and is adapted to the H-89 system bus. The MIDI spec does not define the details of the serial output device, but is restricted to the drivers and receivers used for the interface itself. The circuit is shown in Figure 1, and the section enclosed in the dotted line comes directly from the MIDI specification. The remainder of the circuit is the same as the serial ports on the normal 3 serial port H-88-3 board. I had other I/O devices in my H-89 at the time with an address decoder, but the design shown here is for a standalone interface built on an H-89 prototype board and operating as the third serial port at address 320/327 octal. This address series is decoded and presented on the right side interface bus by the motherboard address decoder chip. Remember that there must be no serial chip (the 8250) in the center socket on the H-88-3 when the MIDI interface is installed.

It would have been nice to use one of the existing serial channels, but the basic problem with this approach is in the required baud rate for the MIDI port. The speed of 31.25 kilobaud (+/- 1%) cannot be generated from the serial clock rate available in the H-89 (1.8432 MHz). The divisor can only be programmed to generate this baud rate from a round number, such as 4.0MHz or 2.0MHz. So a separate oscillator for generating the required frequency is included on the circuit board.

Notice also, that the interface signals are not RS232 voltage levels, but rather a 5 mA current loop. The output is driven directly by a TTL driver gate and the input is

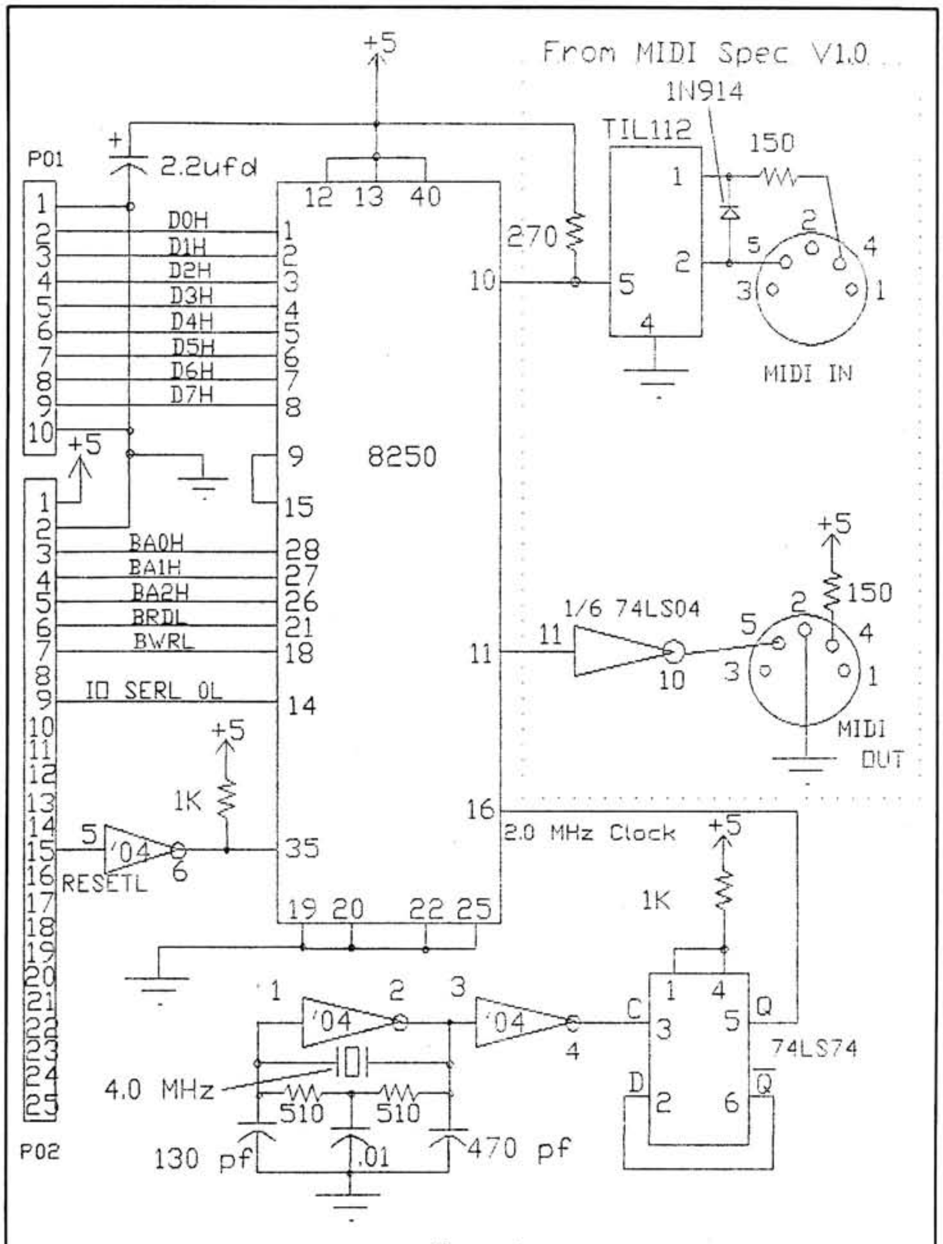


Figure 1
H-89 MIDI Interface

isolated by a high speed optical coupler. Be careful about substituting for the specified opto-coupler. Many devices are just not fast enough to operate properly at these speeds. The rise and fall times should be less than 2 microseconds.

The connectors used were the specified DIN 5 pin (180 degree) female panel mount receptacles mounted in unused spaces on the rear panel of the computer. Only three of the five pins are used; two for the twisted signal pair and one for the cable shield. Notice especially that the ground pin on the MIDI IN connector is not connected. This is to avoid any possibility of a ground loop between equipments.

A small piece of test equipment that I found very useful is shown in Figure 2. It is a female DIN connector with a single LED attached to the MIDI signal leads. It will give a visual indication of output from a MIDI cable attached to a MIDI OUT port on any instrument. A similar device built using a DIN plug would also be useful for plugging directly into the output port. Just be sure to get the LED polarity correct.

In conjunction with the test lamp, I used several short utility programs to generate known signals and to read simple MIDI inputs. The most useful, shown in Listing 1, is a Microsoft BASIC program that will send MIDI data bytes as entered by their decimal values or will receive individual MIDI input bytes. BASIC cannot keep up with a continuous stream of MIDI input bytes! It will also operate in a loop-back mode by sending MIDI data bytes to itself via a cable connecting the MIDI OUT and MIDI IN connectors. The program can be easily changed to allow for any other utility functions you may desire. How about a short BASIC program to play random notes?

I have talked about the musical equipment available for MIDI use and given a diagram for a MIDI interface for the H-89. An alternative implementation would be to cut the track on the H-88-3 board that supplies the clock to the 8250 chip, and to substitute a 2MHz independently derived clock. You would then be able to pick off the serial in and out signals from the UART itself to drive a set of MIDI I/O drivers and receivers. This circuitry could be placed on a small daughter board as it would need room for only two 14-pin ICs, one opto-isolator, a crystal and a few resistors and capacitors. A little ingenuity is all that's required. Good luck.

Next month, I'll present the software to go with the interface. There are 3 programs in C and assembler to make use of the interface and really make it sing.

1. Keyboard Magazine Subscription Department, 20085 Stevens Creek, Cupertino, CA 95014

Listing 1

```
5 CLEAR 1500
7 DEFINT D
47 DEV=208    'address of MIDI interface
48 REM *****SET UP THE MIDI PORT*****
50 OUT DEV+1,0
55 OUT DEV+4,16
60 OUT DEV+3,128
65 OUT DEV,4
70 OUT DEV+1,0
75 OUT DEV+3,3
80 X=INP(DEV)
85 OUT DEV+4,11
87 REM**********ASK FOR FUNCTION*******
90 INPUT "IN=1 OUT=2 LOOP=3";A
92 IF A=3 THEN 300
95 IF A=1 THEN 200
96 REM*******DO MIDI OUTPUT********
100 FOR I=1 TO 3
105 INPUT D(I)
110 NEXT I
120 FOR I=1 TO 3
125 PRINT D(I),CHR$(D(I))
130 OUT DEV,D(I)
140 NEXT I
150 GOTO 100
200 REM********* INPUT LOOP ********
210 X=INP(DEV)
220 PRINT X
230 GOTO 210
```

```
300 REM **********LOOP BACK**********
310 REM     GET A BYTE
320 FOR I=0 TO 255
325 D(1)=I
330 OUT DEV,D(1)    'SEND IT OUT
340 D(2)=INP(DEV)    'READ IT BACK
350 PRINT D(1),D(2)
355 IF D(1)<>D(2)THEN PRINT CHR$(7):STOP
360 NEXT I
370 END
```

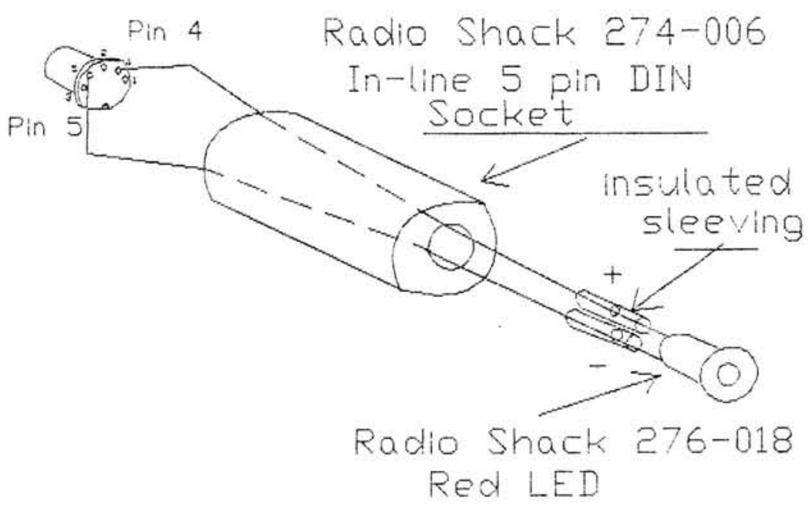


Figure 2
MIDI Tester

Microsoft Word And Page Composition

David M. H. Butler
1619 East Columbia Street
Seattle, WA 98122
Copyright 1987 David M. H. Butler

Programs often perform tasks that far surpass their everyday use. Printing banners with 1-2-3 is a recent example I noticed in PC Magazine. Desktop composition is a hot new software arena, and Microsoft *Word* has built-in power that gives it considerable page composition performance. While *Word* is not a true page layout tool, such as Aldus Corporation's *PageMaker* or Xerox's *Ventura Publisher*, its printer control capabilities and its style sheets make it a well-groomed document processor. I prefer to avoid the phrase Desktop Publishing. None of the systems encompass the aspects of printing and afterpress distribution. Aspects that I consider an integral part of publishing. That doesn't subtract one bit of respect I have for their page composing abilities.

A full desktop system requires a number of extras, like an image scanner and laser printer. These are items many users have to do without. Still, those who wish to produce multiple column documents and use their printer to its fullest, can do so with Microsoft *Word*.

To illustrate the point, I have taken a company newsletter that is usually typeset, and formatted it using *Word* (Version 3.1). A few names of people and products have been changed by request. Printing the newsletter with a Quadram Laser printer provided the near typeset quality results. I will use aspects of the newsletter to discuss *Word*, which I have used since buying my Zenith 150-PC two years ago.

Features

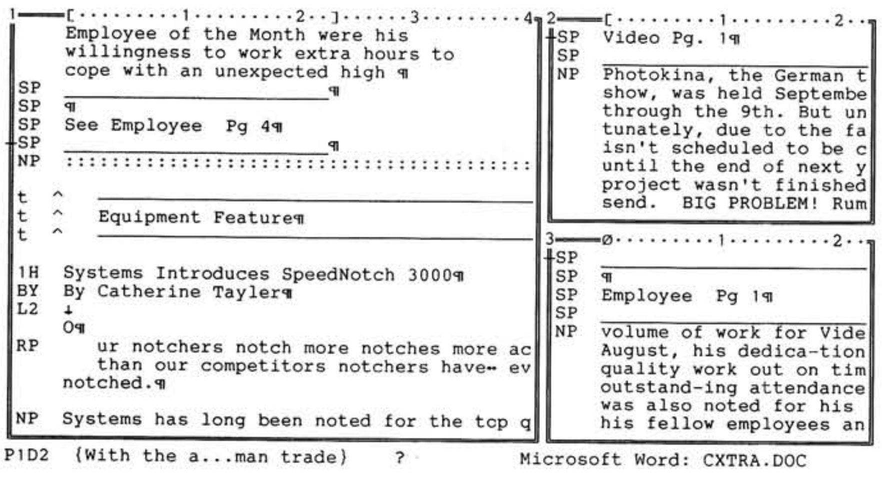
Like all word processors, *Word*'s main task is entering and editing text. You can also edit any ASCII file or, using *Word*'s conversion utilities, work with *WordStar* or DCA compatible files. Unlike other word processors, *Word* handles text formatting and printer control with a finesse few others can match.

Its display, while not true WYSIWYG, allows you to view italics, bold and super/subscripts on the screen. Its "printer display" option shows you exactly how many characters will fit on each line when you are using proportionally spaced fonts. *Word* does not import drawings. However, if the printer supports the IBM extended character set, *Word* can display and work with them. This allows you to create boxes and shaded or solid lines. Unfortunately, this is where any true graphics ability ends. Blank areas have to be left in the document and conventionally produced art pasted in later.

Word offers exceptional mouse support. The mouse is a valuable tool when your main task is formatting. The mouse was one of my first reasons for falling in love with *Word*. Text selection is effortless, and a single click of the mouse opens a window or turns on the ruler. The keyboard commands are nearly as simple, and quickly picked up for those without the mouse.

Word allows up to 8 windows for displaying parts of the same document or different documents at the same time. A multiple window configuration gives you a clipboard area to hold portions of files temporarily. To Move text from one window to another, simply select the text and copy it using the mouse or the scrap buffer. Figure 1 shows *Word* divided into three windows. The narrow column width of the newsletter meant I could still view all my main text in the larger left-hand window and keep track of temporarily cut sections in the smaller right-hand windows.

Word can define pages in multiple columns, although only a single column appears on the screen. The columns are of equal width, but custom layouts are possible with a little creative use of standard features. The first page of the newsletter (Figure 2) shows an example of this. The first two paragraphs flow across the center column.



```
1──[·········1·········2··]·····3·········4┐2──[·········1·········2··┐
     Employee of the Month were his          +SP  Video Pg. 1¶
     willingness to work extra hours to       SP
     cope with an unexpected high ¶           NP  Photokina, the German t
SP   ___________________________¶                 show, was held Septembe
SP   ¶                                            through the 9th. But un
SP   See Employee  Pg 4¶                          tunately, due to the fa
+SP  ___________________________¶                 isn't scheduled to be c
NP   ::::::::::::::::::::::::::::::::::::::       until the end of next y
                                                  project wasn't finished
t  ^   _________________________________         send.  BIG PROBLEM! Rum
t  ^   Equipment Feature¶
t  ^   _________________________________     3──Ø·········1·········2··┐
                                             +SP  ______________________
1H   Systems Introduces SpeedNotch 3000¶      SP  ¶
BY   By Catherine Tayler¶                     SP  Employee  Pg 1¶
L2   ↓                                        SP  ______________________
     O¶                                       NP  volume of work for Vide
RP      ur notchers notch more notches more ac    August, his dedica-tion
        than our competitors notchers have- ev    quality work out on tim
     notched.¶                                    outstand-ing attendance
                                                  was also noted for his
NP   Systems has long been noted for the tcp q    his fellow employees an
P1D2  {With the a...man trade}    ?           Microsoft Word: CXTRA.DOC
```

Figure 1. Word screen divided into three windows. Notice how the drop-cap appears on a separate line from the paragraph it will appear with. Word takes care of merging the two paragraphs during printing. The two-letter codes running down the left side of the window are Style sheet tags for that paragraph.

Figure 1

Word screen divided into three windows. Notice how the drop-cap appears on a separate line from the paragraph it will appear with. Word takes care of merging the two paragraphs during printing. The two-letter codes running down the left side of the window are Style sheet tags for that paragraph.

For formatting, *Word* offers built-in hyphenation, justification and variable leading. Kerning is not supported. For editing, *Word* includes an integrated spelling checker, an on-line thesaurus and an outline processor.

An Undo command allows you to change your mind about your last action. The search and replace command searches for tabs, spaces and paragraph marks, along with the standard characters. You can select a column within a table and move it, sort it, or perform standard math functions on it.

Perhaps *Word*'s greatest feature is the style sheet. Virtually the entire document's look can be described in it; each character, each paragraph type, each page. A style is identified with a one- or two-letter tag. Assign the style to the appropriate element of the document by highlighting the word or paragraph and pressing the Alt-key, plus the tag. For example, Alt-1H might be the tag for a first level heading. Major formatting changes are made by editing the style sheet or attaching a different style sheet; the change is made instantly.

Printer Control

Word's printer support is impressive and includes most of the current laser printers. Available type styles and sizes depend on the printer you are using. Specific printer driver (PRD) files contain the control sequences for the printer's fonts and formatting capabilities. The PRD file also contains the character width tables of proportionally spaced fonts. *Word* performs all its formatting of page and line lengths based on the width tables. It is these tables that allow *Word* to know that it can fit more lower case i's on a line than upper case M's.

The Apple LaserWriter PRD file permits *Word* to communicate with any device that is Adobe Postscript compatible. That includes many laser printers and a number of typesetting devices, as well. The value in this is that you can proof a document for both content and format, using a Postscript printer, and when it's just right, send the same output to a Postscript typesetter and know you will get the same results in higher resolution. The Quadram laser printer is on the verge of releasing an upgrade to Postscript compatibility, which will greatly enhance its future in desktop composition.

Word supplies a utility and examples to modify print drivers or to create your own. I used the utility to add extra fonts and width tables to the Quadlaser PRD file. Quadram supplies a font editor that I used to modify the character spacing of several fonts. The changes were added to the print driver.

The printout of all 4 pages took less than 1 minute. I have only included 2 pages for space considerations. By modifying the style sheet slightly, *Word* produced an acceptable, but low resolution, printout on an Epson FX-85.

Page Layout

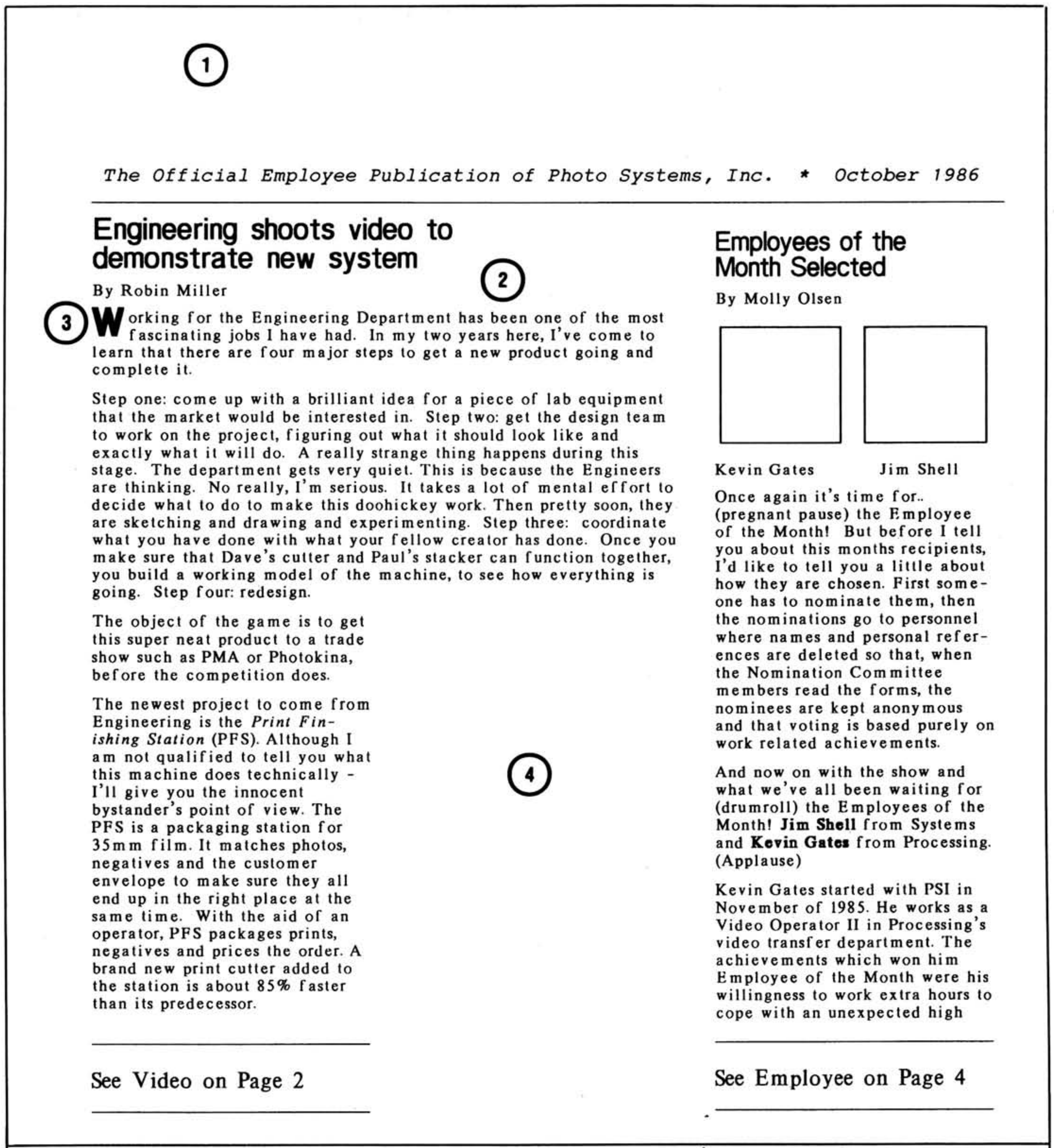
All layouts should start with a quick thumbnail sketch, to provide a rough guess as to what the pages will look like. Even in the computer age it doesn't hurt to use a pencil. Jot down which articles will appear on which page and draw in boxes to show where columns of text or photos will appear. Decide how much space you'll need for the banner and margins. It's a small amount of work that pays back big time savings.

The Format Division command describes the overall page layout. The division describes margins, layout, and page numbering. Using your thumbnail sketch as a guide, fill in the pertinent fields; generally in the margin and layout menus. This will be your basic "page description".

Page widths are variable from one inch up to the width of your printer or 22 inches (whichever is smaller). Page length can also be set from one inch up to 22 inches. *Word* supports up to 6 columns on an 8-1/2 inch page. A gutter can be defined for documents that will be printed double-sided; allowing an after-printing bindery edge. By bindery edge, I simply mean an area where, for example, staples or a three-hole punch would go.

For the newsletter, I defined a standard division to have .5 inch margins and three columns with a .3-inch space between columns. Page numbering was turned off and added manually, so that I could get multiple divisions on a page. Let me explain.

Word uses its columns in a threaded fashion. That is, it fills each column to its limit, then moves the text to the next column. That is fine for a novel or a textbook, but in a newsletter you quite often want an article to occupy the top or bottom half of



①

*The Official Employee Publication of Photo Systems, Inc. * October 1986*

Engineering shoots video to demonstrate new system ②

By Robin Miller

③ **W**orking for the Engineering Department has been one of the most fascinating jobs I have had. In my two years here, I've come to learn that there are four major steps to get a new product going and complete it.

Step one: come up with a brilliant idea for a piece of lab equipment that the market would be interested in. Step two: get the design team to work on the project, figuring out what it should look like and exactly what it will do. A really strange thing happens during this stage. The department gets very quiet. This is because the Engineers are thinking. No really, I'm serious. It takes a lot of mental effort to decide what to do to make this doohickey work. Then pretty soon, they are sketching and drawing and experimenting. Step three: coordinate what you have done with what your fellow creator has done. Once you make sure that Dave's cutter and Paul's stacker can function together, you build a working model of the machine, to see how everything is going. Step four: redesign.

The object of the game is to get this super neat product to a trade show such as PMA or Photokina, before the competition does.

The newest project to come from Engineering is the *Print Finishing Station* (PFS). Although I am not qualified to tell you what this machine does technically - I'll give you the innocent bystander's point of view. The PFS is a packaging station for 35mm film. It matches photos, negatives and the customer envelope to make sure they all end up in the right place at the same time. With the aid of an operator, PFS packages prints, negatives and prices the order. A brand new print cutter added to the station is about 85% faster than its predecessor.

See Video on Page 2

④

Employees of the Month Selected

By Molly Olsen

Kevin Gates Jim Shell

Once again it's time for.. (pregnant pause) the Employee of the Month! But before I tell you about this months recipients, I'd like to tell you a little about how they are chosen. First someone has to nominate them, then the nominations go to personnel where names and personal references are deleted so that, when the Nomination Committee members read the forms, the nominees are kept anonymous and that voting is based purely on work related achievements.

And now on with the show and what we've all been waiting for (drumroll) the Employees of the Month! **Jim Shell** from Systems and **Kevin Gates** from Processing. (Applause)

Kevin Gates started with PSI in November of 1985. He works as a Video Operator II in Processing's video transfer department. The achievements which won him Employee of the Month were his willingness to work extra hours to cope with an unexpected high

See Employee on Page 4

Figure 2
Elements of the Sample Newsletter.

1. The laser printer does not have any fonts over 24 point. The name "Shutter Bug" would have to be produced using transfer lettering or commercial typesetting. The lines and boxes were created using a font that contains the IBM extended character set.
2. This page is formatted as a 3 column page. The first two paragraphs have a negative right indent causing them to extend across the center column. Two column breaks at the end of the first column forces the text that follows over to the third column.
3. The drop-cap was produced using *Word*'s "side-by-side" paragraph format. Type style was limited to 24 point.
4. *Word* cannot import drawings. The image to appear here must be dropped in later.
5. The italic text, as well as the bold text on page one, was defined in the style sheet. Applying the style is a simple

Equipment Feature

Systems Introduces SpeedNotch 3000

By Catherine Tayler

Our notchers notch more notches more accurately than our competitors notchers have ever notched.

Systems has long been noted for the top quality of its film notchers. Ninety-five percent of all large photofinishers use our notchers, so when we build a new notcher the expectations are naturally high. 5 The *SpeedNotch 3000* is faster and more accurate than any notcher manufactured. With the compactly written software the notcher can work at the remarkable rate of 28,000 frames per hour without losing accuracy and with an error rate of a mere 0.2 percent. The average speed of other notchers is 14,000 notches per hour with an average error rate of 2.83 percent.

Notchers perform an important function for large photofinishing labs. They take a batch of film and examine each frame on that batch. The notcher scans each frame looking for the leading and trailing edges. When the edges of the frame have been determined, the notcher cuts out a half moon shape from the edge of the film located in the middle of the frame. When the film goes to the printer, this notch gives the printer a reference point to perfectly align the negative frame when it prints. By notching accurately a large photofinishing lab can save time and money by reducing the quantity of mis-printed negatives.

6

The SpeedNotch 3000 is the only notcher in the world that notches both sides of a negative strip. By having dual edge notching capacity, photofinishers can program the notcher to notch the edge of a negative which corresponds to the program of the printer. Every forth frame, the SpeedNotch 3000 puts an additional notch on the opposite edge of the film. At the packaging station, this notch informs the automatic film cutter where to cut the negative for packaging.

SpeedNotch 3000 is helpful when working with improperly or poorly exposed films. The extra-sensitive eye can differentiate between a blank frame and a "thin" frame. That means that the SpeedNotch 3000 recognizes a negative with fireworks exploding at night and notches it. The printer will know that an image exists on the negative and prints it accordingly. A print, that otherwise would not have been made, is printed and sold.

Our new notcher is the latest in a line of world class notchers. We are showing it in Chicago this spring and production units will be ready in June.

Video (Pg. 1)

Photokina, the German trade show, was held September 3rd through the 9th. But unfortunately, due to the fact the PFS isn't scheduled to be completed until the end of next year, the project wasn't finished enough to send. BIG PROBLEM! Rumors have it that the competition will have a working model of a roll station at the show. How do we get the market to wait another 12 to 15 months before buying a 35mm packaging station when we can't show them what ours does? The folks in Marketing must have been watching MTV, I'm not sure, but they came up with a brilliant idea. Everyone agreed that showing a video of what the PFS is supposed to do should help stave off the competition.

Putting the video together was an interesting experience. The Technical Publications department along with a professional video team, spent August 12th making the station "work". Various members of the Engineering staff spent several days scurrying around putting PFS together. This was a prototype however and wasn't exactly working; not without help anyway.

PFS was hooked up to several computers. One fed the envelopes to the operator, another worked the print handler, not to mention the print cutter & stacker, negative cutter & stacker, wallet folder and a joystick which caused the wallet feeder to work. With several Engineers pushing buttons in sync, the print station was ready for it's starring role.

In the six minutes that the video runs, first we see frustration of a processing worker using a slow packaging station, and then the advantages of the new Print Finishing Station. The video eliminates the bloopers when PFS didn't quite function properly. All we see is a smooth running station -- well semi-smooth.

matter of selecting the text and then pressing the Alt-key and the appropriate two letter code.

6. Each half of this and subsequent pages is considered one page by *Word*. This allowed the two column format above a three column format and allowed columns to break more conveniently.

the page. To have two articles in a newsletter appear in three columns, one article above the other, as page 2 of the example does, requires a bit of preparation. Describe the page (a division command) as being only half its true length, ie., 5.5 inches instead of 11. Modify the print driver to use linefeeds, instead of a formfeed, to position the next page in the printer (this is not as hard as it might sound). The second article can then be printed on the same physical piece of paper. This is an advanced feature that can be worked around. If you are not up to modifying your PRD file, just skip it for now and remember that it's possible.

Most measurements can be set using inches, centimeters, points or lines (there are generally 6 lines to the inch). Paragraphs can be indented or, using a negative number, extended, much like a margin release. This margin release allows you to produce a format calling for columns of uneven widths. In the sample newsletter, a negative right indent causes the first two paragraphs to extend across the center column. Two division marks at the end of the first column forces the text that follows over to the third column. Thus leaving a space for a drawing and not overtyping the extended text. The second page uses a similar scheme to form a figure window.

Paragraph formats are applied through menu selections or by using an Alt-two-letter sequence described in the style sheet. An optional "style bar" along the left edge of the screen shows each paragraph's style code. A ruler line on the top of the screen changes to show each paragraph's tabs and margins.

To format a page, position the text and attach character, paragraph and division styles. Repaginating the document determines where page and column breaks occur. Each column break and its column number appears next to the style bar on the left side of the screen. To force a column break, insert a properly formatted division mark. The division can be formatted to break to the next column or page. If the text overruns the area you wish it to fill, select it and move it to the clipboard window. Be sure to allow room for a "continued on" line at the end of the page. Drop down to the page where you want your story to continue, type in the continued heading and move the text from the clipboard back into the main document.

A drop-cap is a standard element in newsletters and brochures. *Word* can handle it by defining what *Word* calls "side-by-side" paragraphs. The left paragraph contains only the initial letter ("W", in the example). This one-letter paragraph is formatted to have a right margin indent that leaves only a .3 inch line length. The rest of the paragraph is formatted with a .3 to .4 inch left indent. To force the drop-cap into its slot, highlight the character and press <Shift-Return>. Adjust the line spacing of the drop-cap paragraph to align the character with the following line. I should add that on screen, the drop-cap will be above the paragraph that it drops into. *Word* takes care of merging the two paragraphs during printing. To finish off the drop-cap pocket, add a hidden paragraph mark near the end of the second line of type. Highlight the following space and format the paragraph with a 0" left indent.

There's a lot to learn about typesetting and page composition. And, in time, that knowledge will be in the hands of many computer users. Much of the knowledge I've acquired from preparing manuals and brochures has proved vital in electronic layout.

Using What You Already Have

Most programs do much more than their original, intended application. Often, it is not as elegant as a program intended for that specific use, but provides low-budget access to the capabilities of the computer system. Microsoft *Word* has proved itself a powerful formatting tool with formatting power for desktop composition. ✱

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***.

Use your **Z181/183**
as a portable VT220/240 workstation with:

- TRUE DOUBLE HIGH/DOUBLE WIDE
- "TRUE" 132 COLUMN MODE (128 actual)
- TRUE SMOOTH SCROLLING
- KERMIT & XMODEM FILE TRANSFERS
- VMS SERVICES FOR MS-DOS SUPPORT

ZSTEMpc™VT220 Emulator	**$150**
ZSTEMpc™VT240 Emulator	**$295**

including 4014 and ReGIS graphics.

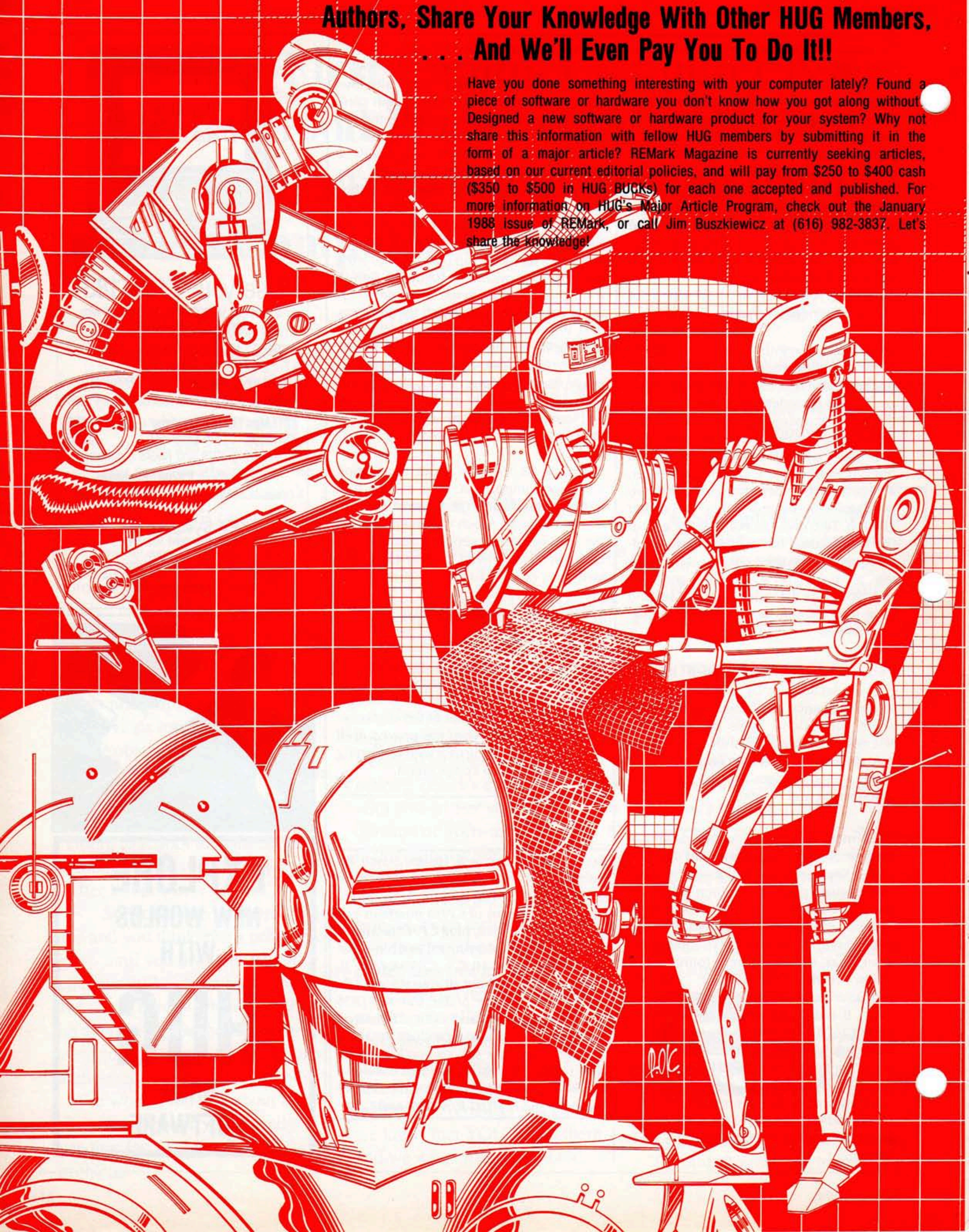
also available **PS200™ Keyboard**
(see our PowerStation advertisement)

KEA Systems Ltd.
#412 - 2150 West Broadway
Vancouver, B.C. Canada V6K 4L9
Support (604) 732-7411
TELEX 04-352848 VCR **FAX** (604) 732-0715
Order Toll Free (800) 663-8702
30 day money back guarantee AMEX/MC/VISA

EXPLORE
NEW WORLDS
WITH
HUG
GAME
SOFTWARE

Authors, Share Your Knowledge With Other HUG Members, . . . And We'll Even Pay You To Do It!!

Have you done something interesting with your computer lately? Found a piece of software or hardware you don't know how you got along without? Designed a new software or hardware product for your system? Why not share this information with fellow HUG members by submitting it in the form of a major article? REMark Magazine is currently seeking articles, based on our current editorial policies, and will pay from $250 to $400 cash ($350 to $500 in HUG BUCKs) for each one accepted and published. For more information on HUG's Major Article Program, check out the January 1988 issue of REMark, or call Jim Buszkiewicz at (616) 982-3837. Let's share the knowledge!



Structured Screen Control

Lynwood H. Wilson
2160 James Canyon
Boulder, CO 80302

Why

Time was, and not long ago, that the users were grateful for whatever they got and never missed things, like ease of use and friendliness. Programmers had less to worry about in those days, but the industry has matured (a bit), and now we must concern ourselves with the user interface, as well as the function of the program. In fact, a program which is difficult to use and confusing to the user is less functional than another which produces the same results, but communicates them better. We, as programmers, must begin to think of our work as part of a communication process, as well as a computing process.

A lot of us, even if we are pretty good programmers, tend to use our CRTs just like teletypes. We don't do anything more than write text to the bottom of the screen, and let it scroll off the top. A fuller use of the display can contribute a lot to the communication with the user. User-friendliness is more than just menus and windows, but they can certainly help. The important thing is to present the user with a simple and easily understandable picture of what is happening, what has happened, and what is likely to happen next. This is a lot easier to do if you take control of the screen and create displays that use more of the capability of the machine. Don't you find that a well organized screen display helps you too, particularly in the case of a program you don't use often?

We read and talk a lot about structure, and how the structure of the program should reflect its operation. By now, there is probably little question that structured programs are easier to write, easier to debug, and easier to verify. But there is another area where such considerations have been largely overlooked, and that is the structure of the user interface. The structure of the program helps communicate to the programmer how the program works, and that is certainly important. But the structure of the user interface communicates to the user how the program works and that is at least as important, since he probably starts off knowing less about it than the programmer. What the user sees on the screen is all he has (unless he is the one out of a hundred who reads the manual first) to tell him what is happening in the program or even what the program does. We must give him structure in what he sees to help him follow the program. There must be continuity, as well as information. The user will never know about the logic of your program unless it is wrong, and perhaps not even then. He won't ever appreciate that little tricky bit that saves so much code. All he will know about is what he sees on the screen, and on that basis, he will judge your work.

What

In order to be able to communicate well and accurately with the user, we need to improve the presentation we make on the screen. We need to be able to do more than just write on the bottom line. We need a set of functions to control the display, clear the screen, move the cursor around, and such. And more, we need functions that are easy to use, that work in a reasonable and intuitively sound fashion. The better we fulfill all these requirements, the easier these functions will be to use, the more they will be used, and the easier the code and the programs will be to understand. We have to look out for ourselves, as well as the user.

I would like to show you several ways of doing these things, varying in complexity and effectiveness, so you can choose the one that best fits your needs.

These routines were written in Microsoft C.

**The Simple Approach —
Screen Control With C Print Functions**

It is actually possible to do a lot of useful display stuff by writing to the bottom of the screen. To clear the screen, for instance, you merely write 25 new lines. You don't even have to admit what you are doing. Just hide this in a header file somewhere.

```
#define CLEARSCREEN printf("\n\n\n\n\n
\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n
\n\n")
```

Then you can write CLEARSCREEN; in your programs, and lo, the screen will be cleared. Yes, I admit it, I've done it this way. I've done whole screens this way. First, clear it. It isn't pretty, even with a fast terminal. The old stuff visibly scrolls off the top as the blank lines push up. Then start writing each line of the new screen, with appropriate spaces, tabs, and new lines to space the words on the screen as you desire. The new screen begins to appear as it scrolls in from the bottom. It's a little quicker to begin writing the new screen after just enough of the old is gone to ensure that it will all have scrolled off by the time the new screen is complete. After you have written 25 new lines, you have a new screen. Takes a lot of tinkering to get all the spacings right, and nothing you do can make it fast, but it works. And it is portable. This kind of primitive screen and cursor control will run on anything that can compile the code. It will even run under other operating systems. Slowly.

A Better Approach — The ANSI Functions

A better alternative is the ANSI.SYS terminal driver, which is a feature of MS-DOS 2.0 and later versions. It provides a software emulation of the ANSI standard terminal, including a useful set of commands to move the cursor, clear the screen or a line, and such as that.

In order to use the ANSI commands on most MS-DOS machines, you must write a config.sys file that contains "DEVICE =ANSI.SYS", or add it to your existing config.sys file. Also, in the root directory of the boot disk, you must have the file ANSI.SYS. Then you just print to the screen some odd sequences of chars beginning with ESC. For example, to clear the screen and move the cursor home, you need merely to write "ESC[2j" to the screen. (ESC here stands for the ASCII escape character, hex 1B, not the three characters "ESC".)

There are several problems with this approach. It is slow, although faster than scrolling everything up from the bottom. You can see the cursor flashing around the screen. On many machines, you must construct a config.sys file to load the ANSI stuff, and that is another thing to confuse the user of your programs. And several of the interesting capabilities of the machine go to waste. ANSI.SYS can be extremely useful, of course, if the disadvantages do not bother your application. See your MS-DOS manual for the commands and details.

A Much Better Approach — Using The ROM BIOS Calls

The IBM computers and their clones have a wide range of services available in their BIOS, which is in ROM on the main board. These intermediaries between the

```
----------------------------------------------------------------
|*                                                            *|
|1,1                                                      1,80|
|                                                              |
|                                                              |
|                                                              |
|                                                              |
|                             *                                |
|                           12,40                              |
|                                                              |
|                                                              |
|                                                              |
|                                                              |
|                                                              |
|25,1                                                    25,80 |
|*                                                            *|
----------------------------------------------------------------
```

Figure 1

Locations on the screen are described in rectangular coordinates. The first number is the line number, starting from the top, and the second number is the column number, starting from the left side. Home is the upper left corner of the screen, variously characterized as 0,0 and 1,1. ANSI calls home 1,1 and MS-DOS calls it 0,0. If you want to send the cursor to a particular location, you must identify the location differently depending on who you are talking to. Figure 1 shows some screen locations as ANSI defines them. To get the MS-DOS definitions, subtract 1 from each of the numbers.

hardware and software are easy to use, easy for your programs to use for screen and cursor control, disk access, fetching keyboard input, and all sorts of input and output. I am going to show you how to use a few of them for simple screen and cursor control. If you want to know more about how they work, you should consult Peter Norton's "Programmer's Guide to the IBM PC", which I recommend very highly, as well as the technical manuals for your computer and operating system. The BIOS interrupts are very nearly as easy as ANSI.SYS to use, and offer better and more detailed control over the screen at a much higher speed. Plus, the added benefit of simplifying and improving the readability of the program.

Microsoft C has a library function called int86(intno, inregs, outregs) which causes the CPU to execute the software interrupt specified by "intno". Before performing the interrupt, int86() copies the contents of "inregs" into the registers of the CPU. Upon returning from the interrupt, the function copies the new register values from the CPU to "outregs". "Inregs" and "outregs" are unions of type REGS, which is declared in the include file "dos.h". Here is the relevant portion of dos.h.

```
/* dos.h
*
* Defines the structs and unions used to
handle the input and output
* registers for the DOS interface
routines defined in the V2.0 to V3.0
* compatability package. It also
includes macros to access the segment
* and offset values of MS C ;;far;;
pointers, so that they may be used by
* these routines.
*
*/
/* word registers */
struct WORDREGS {
  unsigned int ax;
  unsigned int bx;
  unsigned int cx;
  unsigned int dx;
  unsigned int si;
  unsigned int di;
  unsigned int cflag;
  };
/* byte registers */
struct BYTEREGS {
  unsigned char al, ah;
  unsigned char bl, bh;
  unsigned char cl, ch;
  unsigned char dl, dh;
  };
/* general purpose registers union -
overlays the corresponding word and
* byte registers.
*/
union REGS {
  struct WORDREGS x;
  struct BYTEREGS h;
  };
```

As to the interrupts themselves, there are many of varied function, but the only one we are going to use is number 16. (All numbers are decimal unless specified otherwise.) Interrupt 16 is video-display services. Under this interrupt are a number of different services, which are selected by the value in the AH register of the CPU when the interrupt is called. This value is called the service number. Data is passed to and from the service by values placed in other CPU registers. This is done with a union of type "REGS" as described above. Microsoft declared it as a union so it could be used with either byte or word size data. I declared it as a structure in the accompanying code, since all my data is bytes, and it was simpler. In any case, what happens is that you first fill the various elements of the structure "regs" with the values to be moved to the CPU registers. As you can see from its declaration, the structure has elements corresponding to each of the registers. You need not specify values for each of the elements of the structure, only the ones that are significant in the operation you are performing. You will, however, always specify the value of AH, since it selects the service. Notice that you use only one structure, regs, to pass data to the bios video service and get data back from it.

When you call the library function "int86(16, regs, regs)", it first loads the CPU registers with the data in the structure "regs" and then executes software interrupt 16. Interrupt 16 looks at the number in the AH register and calls the service routine that corresponds to that number. The data in the other registers is passed along to the service routine in the CPU registers. When the interrupt service routine is finished, it returns control to int86(), which in turn returns control to the calling program after copying the contents of the CPU registers into "regs". In some cases, data is passed back in this fashion from the service routine. Most of these routines have nothing of interest to return, so you won't usually pay attention to the values returned in "regs".

If you want to know more about what is really happening here, registers, etc., get a good book on assembly language and the architecture of the 8088 chip. Anyway, in time honored cookbook fashion, here is what we are going to do, and here is what is going to happen.

The Primitives — Design Choices

Careful choice of names for your functions, as well as care given to the exact operations they perform is important. You want to be able to look at your code a year from now and see exactly what the screen functions do from their names without needing to go to the functions themselves.

I chose the functions to be included in this list of primitives carefully, since I plan to use them from now on. I found that there were more factors involved in this choice than I expected. First, they must be good tools. They must each do a useful and easily defined job. Second, they must be simple. These are primitives, after all. Third, they must be complete enough to do what you need. And speed, of course, is one of the major reasons we are doing all this. I placed the emphasis on choosing functions and names for them that were obvious when used in real programs. As important as increasing my ability to control the cursor was improving the readability of my code. You should, of course, change the names and even the functions themselves if they do not seem as transparent to you as they do to me.

The Primitives — Implementation

The first function is called cur_goto(row,col) and it moves the cursor to a specified location. It is called with two integers, the first representing the new row and the second the new column. The row, the vertical position, may take values in the range 0 - 24, and the column, the horizontal position, may take values in the range 0 - 79. Remember, the upper left corner of the screen is 0,0. Since this is one of the basic routines available from MS-DOS, it is pretty simple. The two we load into the AH register (through the structure "regs") invokes the proper service routine which moves the cursor to the position we specified in registers DH (row) and DL (column). The BH register specifies the page, and is loaded with 0. We will stay on page 0 (which is the default) throughout this article. Then the call to int86 occurs. The 16 specifies interrupt 16 and the two pointers to "regs" specify that structure as the source of the values to be loaded into the CPU registers and the recipient of the values found in the registers upon return from the service routine. The variables in the structure "regs" that we didn't set are left with whatever garbage they might happen to contain, and the structure returns with whatever the service routine happened to leave in the registers. We won't pay attention to "regs" after the service routine returns, except to notice that we specified a place for the data from the registers to be returned, even though we have no use for it. We could use two different structures for sending and receiving the register values, but this way is simpler and saves memory, considering that we have no further use for the values we put in "reg" once the interrupt has been called.

The second function, get_pos(rowptr, colptr), gets the current cursor position without changing anything. It is called with two pointers to integers, which integers are loaded with the row and column of the cursor position when the function returns. Here, we specify routine 3 and page 0, and upon return the DH and DL registers contain the row and column of the current cursor position. These are transferred to "regs" and then to the locations pointed to by "rowptr" and "colptr".

The third of our primitives, clr_screen(), clears the screen and leaves the cursor in the upper left corner. In this case, the service we invoke is number 6, which scrolls a designated window upward. We select the service as usual by loading 6 into the AH register. The AL register gets the number of lines to scroll. (This is the distance to scroll the designated window, not the size of the window.) This is 0 because scrolling no lines clears the window. CH and CL get the row and column of the upper left corner of the window to be scrolled, while DH and DL get the row and column of the lower right corner. BH in this routine gets the display attribute for the cleared lines. 7 gives normal black and white. You might want to change it for a color monitor. The last part of this function consists of an invocation of number 2, the cursor move service, with row and column both set to 0 to move the cursor home.

The fourth function, clr_eol(), clears the line the cursor is on from the cursor to the right end. It does so by first getting the current cursor position and then scrolling a window one line high to the right of the cursor zero lines.

The fifth through eighth functions, cur_right(n), cur_left(n), cur_up(n), and cur_down(n) simply move the cursor n spaces in the indicated direction. These work by first getting the current cursor position, adding or subtracting n to or from row or column as appropriate, and moving the cursor to the new position. This is simplified by the gratifying regularity of the MS-DOS functions. The function that gets the current cursor position, returns the row and column in the same registers that the function that moves the cursor expects to find them in. Therefore, in a function like these four, we need to modify only one register and may leave the other unchanged.

The ninth and last function is a time delay called wait_sec(n). It is here just to slow down the test sequence, so you can watch it run. It is neither accurate nor portable.

I have included a short test program which will test all of these functions and demonstrate their operation on your computer. It is always possible that some combination of hardware and software could produce a problem, but these should work on any MS-DOS machine. The test program also serves as an example of how to use the functions.

About The Author

Lynwood Wilson *is a programmer, teacher, and writer specializing in small computers. In addition to his freelance work, he is a founder and VP of Mandarin Systems, an expert system house in North Carolina. Mr. Wilson lives in the mountains west of Boulder, Colorado.*

Listing

```
/* scr_tst.c
** Tests the screen & cursor control routines from screen.c
*/


main()
{
    int  row, col;

    clr_screen();
    printf(;;Here we are at home.;;);
    wait_sec(1);
    cur_right(30);
    printf(;;Right 30;;);
    wait_sec(1);
    cur_down(15);
    printf(;;Down 15;;);
    wait_sec(1);
    cur_up(10);
    printf(;;Up 10;;);
    wait_sec(1);
    cur_left(20);
    printf(;;Left 20;;);
    wait_sec(1);
    cur_goto(5,70);
    printf(;;Goto 5,70;;);
    wait_sec(1);
    cur_goto(5,0);
    printf(;;Goto 5,0;;);
    clr_eol();
    printf(;;Clear eol;;);
    wait_sec(1);
    get_pos(&row, &col);
    printf(;;Position was %d,%d;;, row, col);
}


/* end scr_test.c */



/* screen.c
** cur_goto(row, col)
** get_pos(rowptr, colptr)
** clr_screen()
** clr_eol()
** cur_right(n)
** cur_left(n)
** cur_up(n)
** cur_down(n)
** wait_sec(n)
** Interrupt 16 calls the ROM BIOS services. The desired service is
** specified by the number in the AH register. Data is sent to &
** received from the service routine in the other registers. Numbers
** are written to & read from the CPU registers through the structure
** ;;regs;;. See the Microsoft C compiler books re int86().
*/

#include <dos.h>
```

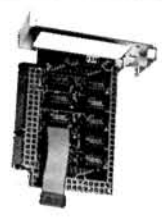
EXPLORE
NEW WORLDS
WITH
HUG
GAME
SOFTWARE

```
/* cur_goto(row, col)
** Takes in 2 integers, the first corresponding to a row number & the
** second to a column, & moves the cursor to the position they represent.
** The valid range is 0 - 24 for the row & 0 - 79 for the column.
** Home (the upper left corner) is 0,0.
*/

cur_goto(row, col)

int row, col;

{
  struct  REGS  regs;

  regs.h.ah = 2;        /* 2 is the routine to set cursor position */
  regs.h.dh = row;      /* these registers hold the new position */
  regs.h.dl = col;
  regs.h.bh = 0;        /* this is the page number */
  int86(16, &regs, &regs);
}


/* get_pos(rowptr, colptr)
** Called with two pointers to integer variables. These variables
** are assigned the column & row numbers of the current cursor position.
** Returns nothing.
*/

get_pos(rowptr, colptr)

int  *rowptr, *colptr;

{
  struct REGS regs;

regs.h.ah = 3;        /* 3 is read cursor position */
regs.h.bh = 0;        /* page number */
int86(16, &regs, &regs);

*rowptr = regs.h.dh;   /* these registers return with the */
*colptr = regs.h.dl;    /* cursor location */
}


/* clr_screen()
** Called with no parameters. The screen is cleared and the cursor
** is left at the upper left corner.
** Returns nothing.
*/

clr_screen()

{
  struct  REGS  regs;

  regs.h.ah = 6;        /* clear screen by scrolling (service 6) */
  regs.h.al = 0;        /* 0 lines. The window to be scrolled is */
  regs.h.ch = 0;        /* from 0,0 to 24,79 which is the whole */
  regs.h.cl = 0;        /* screen. */
  regs.h.dh = 24;
  regs.h.dl = 79;
  regs.h.bh = 7;        /* display attribute for blank lines */
  int86(16, &regs, &regs);

  regs.h.ah = 2;        /* cursor to 0,0 see cur_goto above */
  regs.h.dh = 0;
  regs.h.dl = 0;
  regs.h.bh = 0;
  int86(16, &regs, &regs);
}


/* clr_eol()
** Called without parameters. Clears the line the cursor is on from
** the cursor position (which is cleared) to the right end. The cursor
** does not move.
*/

clr_eol()

{
  struct  REGS  regs;

  regs.h.ah = 3;        /* get cursor position, dh = row, dl = col */
  regs.h.bh = 0;
  int86(16, &regs, &regs);

  regs.h.ah = 6;        /* scroll window */
  regs.h.al = 0;        /* to scroll 0 lines clears the window */
  regs.h.ch = regs.h.dh;  /* upper left corner of window gets */
  regs.h.cl = regs.h.dl;   /* current cursor position */
  regs.h.dl = 79;       /* lower right corner is same row, col 79 */
  regs.h.bh = 7;        /* display attribute for blank line */
  int86(16, &regs, &regs);
}


/* cur_right(n)
** Called with one integer parameter.
** Moves the cursor right n spaces.
** Returns nothing.
*/

cur_right(n)

int  n;              /* n is the number of spaces to move */

{
  struct REGS regs;
  regs.h.ah = 3;        /* get old cursor position */
  regs.h.bh = 0;
  int86(16, &regs, &regs);

  regs.h.ah = 2;        /* move cursor to new position */
  regs.h.dl += n;       /* first add n to column number */
  regs.h.bh = 0;
  int86(16, &regs, &regs);
}
```

```
/* cur_left(n)
** Called with one integer parameter.
** Moves the cursor left n spaces.
** Returns nothing.
*/

cur_left(n)

int  n;              /* n is the number of spaces to move */

{
  struct  REGS  regs;
  regs.h.ah = 3;       /* get old cursor position */
  regs.h.bh = 0;
  int86(16, &regs, &regs);

  regs.h.ah = 2;       /* move cursor to new position */
  regs.h.dl -= n;       /* first subtract n from column number */
  regs.h.bh = 0;
  int86(16, &regs, &regs);
}


/* cur_up(n)
** Called with one integer parameter.
** Moves the cursor up n spaces.
** Returns nothing.
*/

cur_up(n)

int  n;              /* n is the number of spaces to move */

{
  struct  REGS  regs;
  regs.h.ah = 3;       /* get old cursor position */
  regs.h.bh = 0;
  int86(16, &regs, &regs);

  regs.h.ah = 2;       /* move cursor to new position */
  regs.h.dh -= n;      /* first subtract n from row number */
  regs.h.bh = 0;
  int86(16, &regs, &regs);
}


/* cur_down(n)
** Called with one integer parameter.
** Moves the cursor down n spaces.
** Returns nothing.
*/

cur_down(n)

int  n;              /* n is the number of spaces to move */

{
  struct REGS regs;
  regs.h.ah = 3;       /* get old cursor position */
  regs.h.bh = 0;
  int86(16, &regs, &regs);

  regs.h.ah = 2;       /* move cursor to new position */
  regs.h.dh += n;      /* first add n to row number */
  regs.h.bh = 0;
  int86(16, &regs, &regs);
}


/* wait_sec(n)
** call with one integer parameter
** pauses for n seconds
** returns nothing
** 4.77 MHz. CPU
** not accurate
*/

wait_sec(n)

int n;

{
  long i;

  for(; n > 0; n--)          /* what can I say? */
   for(i = 27250; i > 0; i--)
     ;
}


/* end screen.c */
```

*

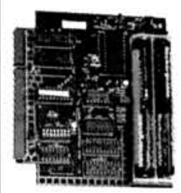
HEATH/ZENITH
88, 89, 90 PERIPHERALS

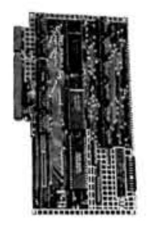


16K RAM EXPANSION CARD

Only $65.00
Shipping & Handling $5.00

REAL TIME CLOCK
Price $130.00 with Batteries
Shipping & Handling $5.00
$114.00 without Batteries





2 PORT SERIAL
3 PORT PARALLEL
I / O CARD

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

PRICES ARE LESS SHIPPING & TAX IF RESIDENT OF CALIFORNIA.
MAIL ORDER: 12011 ACLARE STREET, CERRITOS, CA 90701
(213) 924-6741
TECHNICAL INFO / HELP:
8575 KNOTT AVENUE, SUITE D
BUENA PARK, CA 90620
(714) 952-3930

Secured Computer Systems

TERMS & SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE
VISA & MASTER CARD GLADLY ACCEPTED

New Ways To Protect And Use A Hard Disk

Ed Demaree
P.O. Box 23944
Tigard, OR 97223

Minimizing Head Crash Damage Using DOS Partitions

Most hard disk users have no need for more than 1 DOS partition, and therefore advise against using more than 1 partition of the 4 allowed by MS-DOS. I disagree with this, since carefully planned partitioning can improve your hard disk access speed slightly as well as giving additional protection from some potential head crashes.

Like death and taxes, a head crash is inevitable if you use a hard disk. Sooner or later your A.C. power will fail for half a second. Even if you use a U.P.S., someone can pick up the (running) computer to dust under it, and drop a corner. Despite all your precautions, sooner or later your computer will be in the path of a falling telephone book, or you may never know why it happened. Regardless of the cause, the damage from a minor head crash or power glitch can often be restricted to an area of the hard disk that contains only a few expendable files that are backed up and don't change.

First, a quick look at hard disks in general. A hard disk can be pictured as a stack of constantly spinning rigid disks with spaces between them. Each disk is divided into hundreds of concentric tracks containing a number of sectors (usually 17 sectors of 512 bytes each). Two heads are used per disk, one above and one below each disk. The heads are moved together in fixed steps from track to track, so that they can access the same track on each disk surface simultaneously. (The vertical "stack" of tracks that can be read without moving the heads is called a cylinder.) The number of rigid disks (or platters) used (usually 2 to 4) and the number of tracks on each platter vary from manufacturer to manufacturer, and with the capacity of the disk.

A typical 20 Meg hard disk might have 4 surfaces on 2 disks with 17 sectors per track for 34,816 bytes per cylinder. If it has 610 cylinders, the total capacity would be 21,237,760 bytes less DOS overhead for directories, tables, etc.

Data is stored as a magnetic pattern on the disk surface. Older disks and some newer disks are coated with ferric oxide. Other newer disks are plated or sputtered with a thin metallic film. The technology for coating disks has been proven. The plating or sputtering techniques are relatively new and unproven. On the other hand, a slightly higher density is possible with the metallic surface, which is also harder so that the heads can supposedly bounce off, leaving your data unscathed in a minor head crash. (A head crash occurs if the head touches the spinning disk.)

In order to tightly pack the data onto the disk, the heads are kept almost touching the disk. They "fly" on a cushion of air about 30 millionths of an inch from the disk surface. (That's about 1% of the thickness of a human hair.) Almost all hard disks are sealed to avoid damage from dust or other contaminants. Unfortunately, a mechanical shock to the running computer can cause the head to vibrate slightly. If the head barely touches the disk, it can plow a furrow in the magnetic coating of the disk. This can degrade or destroy parts of your data while leaving the head relatively undamaged. A minor crash may degrade your data slightly, and pass unnoticed by the user. Minor crashes such as this, greatly outnumber the dreaded major head crash that damages a head, or leaves the entire hard disk unusable.

Any motor vibration or disturbance to the air movement inside the sealed disk drive can have a similar effect. Unfortunately, power line glitches or short power outages can cause problems. It may even be possible for a minor head crash to occur when the computer is turned on or off. Some newer disks have a feature that "parks" the heads at an unused cylinder next to the disk hub whenever the computer is powered down to eliminate this problem.

Short power outages present another hazard to your data. Your computer can usually handle a sudden power outage without trouble. However, if the power is restored before the computer completely shuts down, your hard disk can write garbage onto whatever cylinder the heads happen to be located over. That's another reason for waiting at least 30 seconds after turning your computer off before turning it back on again.

The Zenith versions of MS-DOS provide help for safely moving the computer with the **SHIP** command. This command moves the heads to a cylinder next to the hub of the hard disk that isn't used for data. I recommend using this command any time you power down your computer, not just when you intend to move it.

Head crashes caused by accidental mechanical shock to the running computer or by power outages are not helped by the SHIP command, since they can happen at any time, without warning. Some newer software will automatically move the heads to a "park" area, usually next to the hub of the platters, when the hard drive has not been accessed for a long time (10 seconds or so). Since most computers spend 50% to 99% of their running time doing things that do not require frequent hard disk accesses, this is a sensible precaution that greatly improves the odds of surviving potential head crashes unscathed. (There is a slight penalty in hard disk speed, since the heads must move further on the first seek after inactivity to find the next file needed, but the few milliseconds involved are worth it in my opinion.)

If your hard disk is an older model, without the "park" feature, you can achieve almost the same thing easily using the much neglected partition feature of MS-DOS (2.11 or later). This involves setting up a 1 cylinder partition, copying a few simple files on it, and setting up your batch files to access this track last. That will leave your hard disk heads over the expendable cylinder whenever you are "between tasks". This is not as efficient as the "park after timed inactivity" feature of the newer systems, but by prolonging the life of your existing hard disk, it should be well worth one cylinder (about 34K) of disk space.

A typical 20 Meg hard disk for a Heath 158 consists of 610 cylinders. Cylinder #0 contains the partition table, bad sector table, and other critical information. The heads should never be left parked above this cylinder since even a minor head crash here can be fatal to the entire disk. While any other cylinder can theoretically be used for the "head parking zone", I recommend using the highest numbered cylinder (typically #609). There are two reasons for this. First, it is closest to the hub of the platter and therefore more resistant to mechanical shock. Secondly, DOS requires contiguous cylinders in partition assignments. If a cylinder in the middle is used, then it forces you to divide your disk into three partitions. By using a cylinder at one end, after it becomes unusable, the disk can be re-partitioned to make the next cylinder out (#608) the new parking zone, and the old parking zone can be easily left unallocated. By keeping your usable sectors contiguous, you also get maximum flexibility in future disk use.

Here's how to do it, with some things that the DOS manual doesn't tell you. If you're already using a hard disk, copy all of your files to floppies before you start! This will make it much easier to set up the newly partitioned hard disk. Revising a single partition to multiple partitions will effectively wipe out everything on your disk, as both FATs and directories are scrambled.

If you're revising the partitions on an existing disk, I recommend running **PREP** if you have used your hard disk for any length of time. It seems to give a more thorough test of sector integrity than DETECT. If you don't use PREP, you should at least use **DETECT**. The time to locate and isolate any bad sectors is before you reload everything onto the hard disk.

If you're starting with a brand new hard disk, supplied by Heath, **PREP** is probably not needed. Follow the instructions that come with your disk. Two words of caution; first, the instructions that came with my hard disk didn't say which side was up (and it does make a difference). Second, be very careful not to drop the hard disk or the computer containing it, especially after power has been applied (presumably moving the heads away from their relatively protected shipment position).

After running PREP (or with a new hard disk), you must boot from a floppy. Load a back-up copy of your DOS distribution disk #1 into drive A and try to boot. Unlike the IBM PC, the Heath 150 series computers don't first try to boot from a floppy, but instead boot as set by an internal switch. If that switch was set to boot from the hard drive, you'll get an error message, but don't worry; its simple to override this feature from the keyboard. Instead of pressing CTRL ALT DEL, press **CTRL ALT INS**. This places you in the ROM monitor (which is handy for a number of undocumented things that are extraneous here). Then enter **BF0** to **B**oot from **F**loppy drive **0**(A:). (Note that DOS distribution disk #2 is not bootable. Also, the diagnostic disks boot an incomplete older DOS which should be used for diagnostics only.)

PART (located your back-up copy of DOS distribution disk #2) should be run to set up the partitions used. When first setting partitions, partition #1 defaults to include the entire disk, so you must start by reducing the size of partition #1 to make room for the others. Don't forget to set the disk to boot from the desired DOS partition (probably #1). Then exit to the ROM monitor, place a copy of DOS distribution disk #1 in drive A, and boot by entering BF0.

If you use any programs (such as Word Perfect) that have the capability of doing periodic automatic back-ups, careful planning at this point can give a disk access speed improvement. I set up a separate partition large enough to hold all of Word Perfect's permanent files (dictionary, thesaurus, etc.) and the largest possible automatic back-up files. This is about 950K for my use of Word Perfect. This guarantees that DOS won't slow down my automatic back-ups by spreading them all over a large fragmented area of the hard disk. Additionally, by placing all of the word processing files close together (and using floppies for permanent storage of the text files), I get the fastest possible hard disk operation. As an added bonus, if the hard disk should have a minor crash during word processing, all I should lose is the current text. The spreadsheet and other files are guaranteed to be in different cylinders and should survive unscathed.

The following is an example partition assignment:

Partition type	Start Cylinder	End Cylinder	Size Kilobytes	Percentage of the disk	Use partition
--------------	--------------	--------------	--------------	--------------	--------------
DOS (16)	0	580	19745	95.2%	Main DOS
DOS	581	608	952	4.6%	Word Perf.
DOS	609	609	34	0.2%	Parking

DSKSETUP (located on DOS disk #1) should be run next. Set the flag to MANUAL (if it isn't already set to MANUAL), and you should remove Format Protection from all partitions so that they can be formatted. Save this information to both memory and the disk.

ASGNPART (also on the first DOS disk) is the next program to run. Unlike the other programs, it is not menu driven. It must be run once (with the exact syntax shown) for each partition to be assigned a drive designation. For example, to assign hard drive 0 partition 2 as drive D: you must enter **ASGNPART 0:2 D:**. Partition 1 seems to default to drive C:. After assigning the partitions, enter **ASGNPART 0:** to display the assignments for drive 0. Unfortunately, these assignments seem to be only temporary, as will be explained later.

Finally, run **FORMAT X:** where X is each assigned drive to be formatted. Don't forget to place DOS on the boot drive (usually C:) with FORMAT C:/S. Contrary to Richard Mueller's otherwise excellent article on DOS 3.2 (May '87 REM), FORMAT does *not* display head and cylinder numbers when reformatting a partitioned hard disk, but only when formatting floppies. By the way, does anyone know why DOS 3.2's Format calls floppy disk tracks "Cylinders"?

A new version of DOS wouldn't be complete without a nasty surprise. In DOS 3.2, mine came after all this the first time I rebooted the computer. The computer booted and started to run until I tried to access drives D: and E: They were no longer assigned, but the partitions were still there. After repeating the entire process several times, in the hope that I had missed something (I hadn't), I discovered that I could get around the problem by placing ASGNPART 0:2 D: and ASGNPART 0:3 E: in the AUTOEXEC.BAT file (after setting up the PATH, so DOS knows where to find ASGNPART. Fortunately C:\BIN is located on the default drive).

You are now ready to transfer your files to the hard disk. A little planning at this point can make the hard disk easier to use and minimize your disk access time.

I recommend the following to maximize the speed (and efficiency) of the hard disk: Load files in the following order: (COMMAND.COM and the hidden DOS boot files should already have been loaded by Format) CONFIG.SYS, AUTOEXEC.BAT, and other files used only during boot operation; such as a clock reader, and TSR programs executed by AUTOEXEC.BAT.

Files ending with .COM, .EXE, or .BAT can be placed either in the root directory, or in the \BIN subdirectory along with some transient DOS commands as described below. (Booting should be slightly faster with all of the boot files you use located contiguously in the root directory, but you'll probably never notice the difference if binary files are located in the \BIN subdirectory, and they won't clutter up your root directory that way.)

Next you should use the MD (Make Directory) command to create a subdirectory (I use C:\BIN, so the syntax is MD C:\BIN). This file will contain .EXE and .COM files that you don't want to clutter up the root directory with, and selected DOS 3.2 files (from ANSI.SYS to ZSPOOL.COM). This subdirectory should be specified in the AUTOEXEC.BAT file by PATH = C:\BIN. This procedure allows DOS to find its transient commands when needed without cluttering up your root directory with them.

There are a number of files on your DOS distribution disk that you will probably not want on your hard disk. PREP.EXE will destroy all of your data if it is run. FORMAT.COM may be an acceptable risk since DOS 3.2 allows the hard disk partitions to be selectively format protected. Some files are simply useless. I can't imagine anyone using all 25 foreign language keyboards. Other files such as TREE.COM may simply lie dormant if you use the Norton Utilities, PC Tools, or similar programs instead. I therefore suggest copying DOS to another pair of floppies, then deleting files you don't want on your hard disk from the copies, (COMMAND.COM should be deleted also, since it will be loaded in the root directory from your by FORMAT). Then add any other .COM and .EXE files you want in the \BIN subdirectory. This kills two birds with one stone. First, it gives a permanent back up of these unchanging files, and saves repeating the selection process in the event of a major head crash. Second, it makes loading the files contiguously into the \BIN subdirectory possible with a single COPY *.* command.

While .BAT files could go in the \BIN subdirectory, I suggest keeping them in the root directory. If you use batch files as much as I do, you'll soon get tired of searching for them in the \BIN subdirectory. My root directory contains only CONFIG.SYS and batch files. Everything else is in subdirectories.

Careful planning of the subdirectories you will use and their contents can speed up your disk slightly by keeping related files that don't change contiguous with each other and their subdirectory files. Each subdirectory contains its own directory file on the disk. If you can keep the files in each subdirectory contiguous with this directory file and each other, the heads will need less movement to read them. This requires loading the unchanging files for each subdirectory immediately after using the MD command to create the subdirectory. Files that change will be relocated by DOS to wherever DOS finds space on the disk.

This procedure reduces disk search time at bootup, and places your unchanging files together in a block that should be contiguous, with related files close together. This reduces the disk search time for these files. It also reduces the disk area available for file fragmentation, and therefore reduces the access time lost due to fragmentation.

To save time, I also suggest keeping your Root Directory files backed up on a single disk. (COMMAND.COM should not be backed up to this disk, since it will be loaded onto the hard along with the hidden DOS boot files when the hard disk is formatted.) This disk should be kept current, with the files arranged like this:

CONFIG.SYS	(recommend 10-20 Files, 15-20 Buffers, ANSI.SYS)
AUTOEXEC.BAT	(specify PATH=C:/BIN, Asgnpart, TSRs used, etc.)

Batch files and any other unchanging files that you want in your root directory such as:

ME.BAT	(Places menu on screen, parks hard disk.)
WP.BAT	(Accesses Word Perfect in its own partition.)
SP.BAT	(Accesses spreadsheet in its own subdirectory.)
JUNK	(Reserves 2K space for future batch file.)

Here again loading (or reconstructing) the entire directory with contiguous files becomes a simple matter of a copy *.* command.

Having set up cylinder #609 as drive E: for a "parking zone", a small surprise awaits. The 34K cylinder contains space for only 4 files of 4K each. The rest of the cylinder is occupied with DOS overhead for the partition. This has not been a problem for me, since I have only 2 small files in the parking partition, but keep this limit in mind.

The first file is MEN.BAT, which prints a menu of available root directory batch files on the screen (handy if others use the computer). This batch file is then accessed by the command E:MEN.BAT which is the last line of AUTOEXEC.BAT, and all other root directory batch files. E:MEN.BAT must be the last line, since if it isn't, the hard disk heads must go somewhere else to read the remaining lines, which won't leave the heads over the parking zone.

The other file is P:BAT which puts the message "Hard Disk Heads Parked" on the screen using ECHO. This allows the heads to be manually parked from DOS with an E:P at any time.

Other head parking techniques are possible, and limited only by your ingenuity. By keeping the files you use backed up, and keeping the heads over the parking zone whenever practical, much of the possible head crash damage can be limited to the parking zone. May all of your head crashes be minor.

EXPLORE
NEW WORLDS
WITH
HUG
GAME
SOFTWARE

Let us know
8 weeks in advance
so you won't miss a single issue
of REMark

ENABLE

Part 3

George Elwood
1670 N. Laddie Court
Beavercreek, OH 45432

A tutorial Spreadsheet An Introduction

As I said in the first part of this series, ENABLE has a lot of capability and to cover it in one or two articles would not provide the level of understanding necessary. This series will provide a basic understanding of each of the three main modules before moving on to a higher level. This is the method of instruction that the company I work for provides and it has received very high marks from students. For some of you, this article will be very basic, but hang in there because the intermediate and advanced levels of this series will show the complete capabilities of ENABLE. The last installment in this series provided a basic understanding of the ENABLE word processor. This article will cover the spreadsheet and also provide some basics of spreadsheet operations. Although this series is based on the Z-100 version of ENABLE, the PC version works the same way. I will continued to point out the differences in the two versions as I go along. This article covers ENABLE version 2.0, which is the Z-100 version.

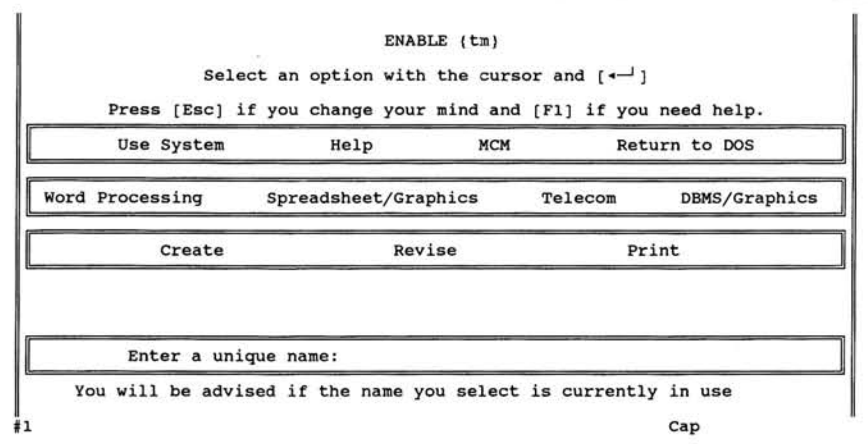
ENABLE's spreadsheet can be used for any of the typical spreadsheet functions. Spreadsheets can be used to display data that needs to be manipulated. All spreadsheets can handle both numeric and textual data with varying degrees of complexity. Spreadsheets are better used to display and calculate numeric data, which use formulas and interrelationships between the cells, than a database.

ENABLE's spreadsheet is LOTUS-like in many ways, not as good at some things and much better in other areas. I will not get into graphics in this article, but plan on covering it in detail in the intermediate and advanced sections. If you know LOTUS or one of the other spreadsheet programs on the market, you should have no problems with ENABLE's spreadsheet.

Entry into the spreadsheet is done from the Main Menu like all of the modules in ENABLE. You highlight the selection with the cursor keys or by typing the first letter of the selection. To enter the spreadsheet, type (U)se System, (S)preadsheet/Graphics. The next choice is (C)reate, (R)evise or (P)rint. Since this is the first entry, select (C)reate and enter a name of your choice. If you enter the name of a file that is already in existence, ENABLE will beep and indicate that the file exists and to either use the file or to type in another name. Note that ENABLE has different extensions on files so you could have a file called "TEST" as a word processor file, a spreadsheet file and a database file. The ENABLE spreadsheet extensions are as follows:

.SSF	Spreadsheet files
.SSP	Spreadsheet print file
.SSM	Spreadsheet macro
SS.MNU	Spreadsheet menu
X.SM	Named spreadsheet menu

After you have entered the above keystrokes, you are placed in the spreadsheet. The screen display is similar to the LOTUS screen. The top line is identified with letters and the left side is identified with numbers. This is the border area. The intersection of a vertical column and horizontal row is called a cell, the basic unit in a spreadsheet. As you move around the spreadsheet, the cell you are in is high-



```
                              ENABLE (tm)

              Select an option with the cursor and [←┘]

      Press [Esc] if you change your mind and [F1] if you need help.

      Use System          Help           MCM          Return to DOS

 Word Processing      Spreadsheet/Graphics       Telecom       DBMS/Graphics

         Create                   Revise                  Print



        Enter a unique name:

    You will be advised if the name you select is currently in use
#1                                                        Cap
```

Figure 1
ENABLE's Opening Menu To Create

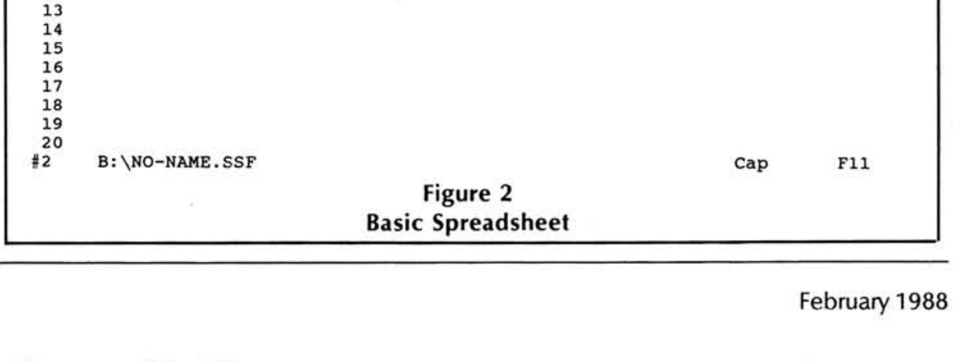
lighted. The address is displayed in the upper left hand corner of the screen and consists of a letter(s) and number. In the upper right corner of the screen is the mode indicator of the spreadsheet. This can display the following with the meaning as indicated: BUSY — Action in progress; CMD — Macro in progress; COMM — Command in progress; EDIT — Indicates modification of cell; LABEL — Creating a label; POINT — Pointing to a range or formula; READY — Ready for next command; VALUE — Entering a number or formula. This part of the display is called the Information Display Area. Like all of ENABLE's modules, the bottom line is the status line. From left to right the following things are displayed. First is the window number, then the file name including drive and directory. Word processing attributes will be displayed next, if selected. The last item displayed is the highest cell in use. This entry is helpful in determining if an extraneous entry has been made outside of the normal work area.

Moving around the spreadsheet is done with the cursor keys for small moves. Moving a page down or up can be made using the SHIFT/3 (PgDn for the PC) or SHIFT/9 (PgUp for the PC). Moving a screen right or left is done by using the TAB for both Z-100 and PC or SHIFT/HELP (SHIFT/TAB for the PC). Another way to move if you know the cell address you wish to move to, is to press the F2 key. ENABLE will respond with "ENTER ADDRESS TO GO TO:". Enter a letter number and <RETURN> and the highlighted cell will be the selected one. Pressing the HOME key will return you to cell A1. F2 -> or F2 <- will move you to the last active cell in the row selected. These are the same keystrokes that are used in the word processor to move to the end or beginning of a line. SHIFT/1 (END HOME for the PC) will take you to the bottom right corner of the spreadsheet.

All spreadsheets take three types of data: value (numbers), formulas and labels (all other entries). Values are any numbers, both positive and negative can be used. The "-" sign must be entered to indicate a negative number. Note that as soon as you enter a number, the mode indicator displays VALUE. As you enter the number, it will be displayed below the cell address in the upper left-hand corner of the screen. Until you press <RETURN>, ENTER or a cursor key, the value can be edited by using BACKSPACE. Using the cursor keys will cause the entry to be placed in the spreadsheet and the cursor to move in the selected arrow direction to the next cell. Once this key has been pressed, the value is displayed to the right of the cell address and in the spreadsheet. This value can be changed by typing in a new value and pressing return or by pressing F4, which is the edit key. By pressing F4, the value appears below the address and editing can be accomplished using the cursor keys or the same keys as in the word processor.

Entering formulas is accomplished in the same manner as entering values with the following exceptions. A formula must be entered with one of the following characters: + - . ($ @. A simple formula might require the values in cells A1, A2 and A3 be added together with the result placed in cell A4. To accomplish this, enter values in cells A1, A2, and A3. Move to cell A4 and type in +A1+A2+A3 <RETURN>. The sum will appear in cell A4. All spreadsheets use the same numeric operator for their formulas. The basic operators are: + (plus) for addition, - (minus) for subtraction, * (asterisk) for multiplication, / (slash) for division. These operators are placed between the cell addresses, but the + is required as the opening symbol for most simple operations.

Entering labels is done in the same manner as values or formulas. When you enter a letter, the mode indicator reflects "LABEL". To enter a number in a cell as a label, you must first type a space and then a number. Failure to do this will result in an error if you include text with the entry. Word processing attributes can be used with labels by using the same keystrokes as in the word processor, i.e., F0 B for bold (ALT/b for the PC). You can use bold, underline, and italics in the spreadsheet and they will print as such. Label alignment in a cell can be accomplished by using the (>), (<), or (x) for right, left or center alignment. The backslash (X) permits repeating entries. By typing "X=", the cell



```
F9:  Above is a repeating entry                                        READY
        A        B         C         D         E        F        G        H
 1     23                           1495
 2     34                       2.826086
 3     65
 4    122
 5
 6
 7         This is a label
 8                                                   #########
 9                     Left                          Above is a repeating entry
10                      Center
11                         Right
12
13
14
15
16
17
18
19
20
#2    B:\NO-NAME.SSF                                           Cap       F11
```

Figure 2
Basic Spreadsheet

will be filled with "=". By copying this cell across the screen, a break can be created.

Now that a basic spreadsheet has been built, it is time to save it. ENABLE uses two methods to gain access to the top menu. Like LOTUS, the slash will bring up the menu. But in keeping with the rest of ENABLE, the F10 key will also bring up this menu. Select (S)ave and ENABLE will prompt for the default setting or to change the setting. If you select (C)hange, you can save the work in other formats, parts of the spreadsheet, or provide a new name for the file. ENABLE will permit you to save the spreadsheet in LOTUS 123 V1.1A and V2.0, SuperCalc3, DIF, or ASCII format. Once the spreadsheet has been saved, select F10 again and quit, note that ENABLE reminds you to save your work.

Now that you are back at ENABLE's Main Menu, we will return to the spreadsheet. Again press (U)se System, (S)preadsheet/Graphics, (R)evise. ENABLE will prompt you for the name of the file. Like the word processor, you can type a "?" and <RETURN> and ENABLE will display all spreadsheet files on the data disks as established in the PROFILE. Using the cursor keys, highlight your choice and type <RETURN> and the file will load. The basic spreadsheet border will be displayed. Note the cell counter in the lower right-hand corner counts up as the spreadsheet is loaded. The data is loaded column by column from left to right. A large spreadsheet will take some time to load. The PC version of ENABLE does not do this while loading as it remains at the Main Menu.

If you load another type file, such as LOTUS or SuperCalc3, you must type in the extension or ENABLE will not find it. The "?" will only search for files with the ENABLE extension *.SSF. If you load one of the other formats that ENABLE can read, you will be prompted for the type. Again, you can use the first letter (number) or highlight the choice using the cursor keys. ENABLE uses the same convention for all selection choices, so it is easy to remember.

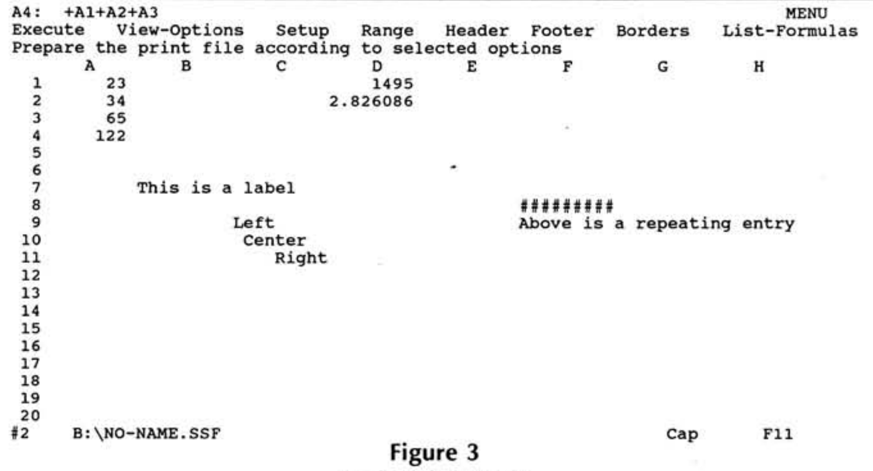
I have found some LOTUS worksheets that will not convert completely. Talking with The Software Group (TSG), they indicated that the problem is with LOTUS. LOTUS will sometimes put extra nulls at the end of the file and ENABLE will not pick this up. ENABLE should convert all LOTUS macros and run them, except for LOTUS print routines. ENABLE does have problems with very large LOTUS files with complex formulas. Again, not all of the spreadsheets will be converted. While this conversion is taking place, ENABLE will display "Converting LOTUS" on the Status Line. ENABLE will convert Symphony files if you select LOTUS V2.0. During the conversion, ENABLE will ignore LOTUS functions not supported.

Printing what has been created is done in the same manner as the word processor with a few added features that are unique to the spreadsheet. First, display the top menu by typing either a "/" or the F10 key. Select (P)rint and the next print menu will appear. As you move the highlighted area across the menu, expanded explanations will appear below the choice. The first option on the list is (E)xecute. With this selection, you can print to a file or to the printer. The print file is used if you print from the opening menu. (S)etup will display the basic printer screens that were explained in the word processor section. This permits you to change printers, select pages, select print sizes, etc. (R)ange permits you to select the range of the spreadsheet you want printed. The default is (A)ll or you can select the range by typing in the "letter/number . . . letter/number" of the upper left corner and lower right corner, i.e., "B5..D10." (H)eader will permit you to select a header that will be printed at the top of every page. You are prompted to enter the one line entry if this is selected. (F)ooter is like the header selection. (B)orders will permit you to select either top, side or both borders. These borders will be ones that you select from the spreadsheet and not the borders displayed on the sides of the screen. The last option is (L)ist-Formulas. This option will print the formulas in the spreadsheet in one long column with the cell address on the left-hand side. ENABLE will print files larger than one sheet of paper left-to-right and then down to the next set of rows based on the size of the spreadsheet.

The one main difference between ENABLE and LOTUS is the extra step required to get to the operation menu. The first "/" or F10 gets you to the basic ENABLE Top Line menu, the <RETURN> at the (W)orksheet choice gets you to the spreadsheet command menu. Note that the additional menu choices are displayed when your highlighted area is over the Worksheet choice. This work area is basically the same as LOTUS with enhancements. These spreadsheet commands provide the capability to copy, insert, delete, change cell widths, plus many more functions.

The first choice is GLOBAL. The global option provides the following commands: Format, Alignment, Width, Protection, Recalculation, Dimension, Transpose. Format and width do not affect cells that have been previously set using format or width under one of the other options. This is convenient when you discover a need to change globally most cells after a few have been set.

Global Format provides a means to select a single attribute for all cells in the spreadsheet. These attributes are: fixed, providing for a fixed decimal point; Integer, for displaying only whole numbers (the default); $, for displaying numbers with a $ and commas every three digits; Commas, displaying numbers with commas every three digits; Percent, will display values as percentages; Date, will display dates in any of seven ways (DD-MMMM-YY, DD-MMM, MMM-YY, YY/MM/DD, YYDDD, YYYY/MM/DD, and MM/DD/YYYY); Time, displays time in



```
A4:   +A1+A2+A3                                                        MENU
Execute   View-Options   Setup   Range   Header  Footer  Borders   List-Formulas
Prepare the print file according to selected options
       A         B         C         D         E         F         G         H
  1     23                          1495
  2     34                      2.826086
  3     65
  4    122
  5
  6
  7        This is a label
  8                                                  #########
  9                  Left                            Above is a repeating entry
 10                   Center
 11                      Right
 12
 13
 14
 15
 16
 17
 18
 19
 20
#2     B:\NO-NAME.SSF                                        Cap       F11
```

Figure 3
Basic Print Menu

The other cats get to sing along!

That's because HEPCAT runs **with** your other programs, not ***over*** them. *HEPCAT (HUG Engineer's and Programmer's CAlculation Tool) is a powerful pop-up calculator for all Heath/Zenith MS-DOS and Z-DOS based computers. Unlike other pop-up calculators, HEPCAT does not stop the currently running program while it is popped up. That means that you can do calculations while your computer is busy with something else. For example:*

- *While Lotus (tm) is loading a huge spreadsheet, you can check your kid's math homework.*
- *While Dbase (tm) is sorting a large database, you can add up some grocery prices.*
- *While your computer is busy compiling one program, you can work on number base conversions needed for another program.*

HEPCAT is safe to pop-up during just about any running program — even during disk activity. And HEPCAT has other features the other guys can't touch.

HEPCAT gets along with everyone . . .

HEPCAT supports more video configurations than any other pop-up, and always pops up in the current video mode, rather than forcing the screen into a text mode as other pop-ups do. It also works properly with more programs than any other pop-up. You can pop up HEPCAT over Microsoft Windows (tm) and many other programs that other pop-ups can't work with, and even over some other pop-ups.

HEPCAT works harder . . .

*HEPCAT provides a multi-function floating point calculator and a programmer's binary calculator that work together to do more than the basic four (+, −, *, /). The floating point calculator includes the following built-in functions: powers, pi, factorial, square root, sine, arc sine, cosine, arc cosine, tangent, arc tangent, log (natural and base 10), e^X and 10^X. It also includes the following conversions: degrees-radians, radians-degrees, Celsius-Fahrenheit, Fahrenheit-Celsius, centimeters-inches, inches-centimeters, meters-feet, feet-meters, kilometers-miles, miles-kilometers, grams-ounces, ounces-grams, kilograms-pounds, pounds-kilograms, milliliters-fluid ounces, fluid ounces-milliliters, liters-quarts, quarts-liters. The binary calculator works in these number bases: binary, tetral (base 4), octal, split octal, decimal, and hexadecimal; and it supports these operations: MOD, AND, OR, XOR, SHL, SHR.*

The HEPCAT floating point calculator supports 8 significant digits and can display numbers four ways: floating point, fixed point, scientific notation, and engineering notation. Numbers are handled internally in BCD format to eliminate binary round off errors in addition and subtraction.

HEPCAT eats less . . .

HEPCAT uses less than 16k of memory — less than any other pop-up calculator that we know of. It also uses less than 16k of disk space, so you don't have to worry about where to put it on a small system. The HEPCAT window uses less screen space, too. It shows you more real information than other pop-up calculator displays, but it doesn't waste space by showing you a keypad layout. You already know what your keypad looks like! HEPCAT is easier to learn, too, with commands that make sense.

If you are tired of pop-ups that can only sing solo, give HEPCAT a try. HEPCAT is available from HUG as part no. 885-3045-37 for $35.00. It works on any Z-100 PC, Z-200 PC, or Z-100 (not PC) system and any version of MS-DOS or Z-DOS.

one of four ways (HM:MM:SS, HH:MM:SS, elapsed DDD HM:MM:SS and elasped DDD HH:MM:SS) (HM is 24 hour clock, like the military uses); Scientific, for displaying numbers in exponential format; General, is the default format; and +/-, will display either plus (+) or minus (-), instead of numeric values for a pictorial display.

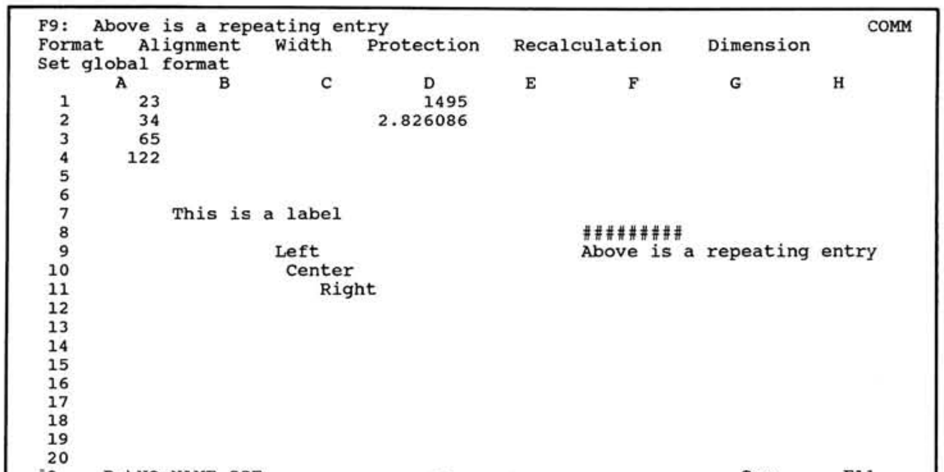
Global Alignment permits you to align labels left, center or right in the cell. This is the same as using the "<", "x", or ">" with the values individually.

Global Width permits you to change the size of the cell for all cells in the worksheet. The default value is 9, but can be changed to values between 1 and 72. If you have changed the size with the Worksheet Width, the global value change will not affect that column.

Protection protects specified ranges in the worksheet that cannot be changed. The global protection permits you to temporarily turn off this protection to make a change and then turn it back on.

Global Recalculation permits you to specify how you wish the calculation to be performed. You can select Automatic (default), Manual (you have to use the F5 key to recalculate), Iteration is a means to calculate automatically around circular references. This is a formula that required the answer of a formula after the first formula to complete the answer. Iteration prompts you for the number of recalculations up to 50. An example is an income tax spreadsheet I wrote. The federal form provided an input to the state form, which in turn provides an input to the federal form. Using iteration prevents numerical errors because of circle references in the spreadsheet.

The one unique aspect of ENABLE is the capability to size the spreadsheet. ENABLE is not as large as LOTUS 1-2-3. Because of the integration, the size of the spreadsheet had to be controlled. The maximum number of cells permitted in ENABLE is 65,000. The Global Dimension permits you to select the shape of the spreadsheet. The six options are: 255 × 255, 511 × 127, 1023 × 63, 2047 × 31, 4065 × 15 or 127 × 511. In the year that I have been using ENABLE, I have not had a problem with the size of the spreadsheet. The few problems that I have heard about were resolved by converting to a database, which they should have been in, in the first place.



```
F9:  Above is a repeating entry                                              COMM
Format   Alignment   Width   Protection   Recalculation   Dimension
Set global format
        A        B        C        D        E        F        G        H
  1     23                      1495
  2     34                  2.826086
  3     65
  4    122
  5
  6
  7         This is a label
  8                                            #########
  9                  Left                      Above is a repeating entry
 10                   Center
 11                      Right
 12
 13
 14
 15
 16
 17
 18
 19
 20
#2     B:\NO-NAME.SSF                                          Cap       F11
```

Figure 4
Global Menu Line

Global Transpose permits you to switch one or more rows or columns. This command will move cell entries from rows to columns or from columns to rows. This is helpful if you need to rotate the spreadsheet 90 degrees.

The next options after Global is Range. Range provides many of the same features as global, but only for selected cells. The options available with Range are: Format, Alignment, Protect, Unprotect, Erase, Data-Fill, Name, Sort. Format and Alignment work the same way as the Global version, but you are prompted for the range of cells.

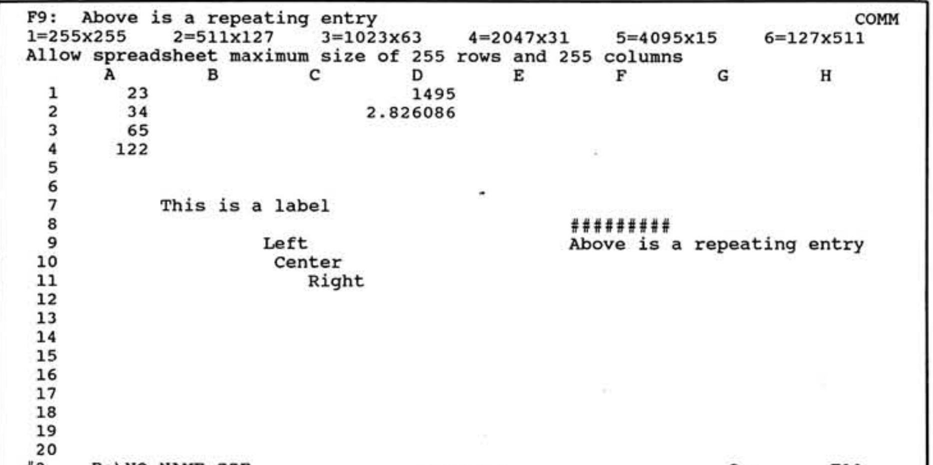
Range Protect and Unprotect provides a means to protect ranges from accidental change. Once selected, you cannot enter the protected cells for any reason, unless it is unprotected first. This can be used to protect labels or a starting value for continued calculations. I used this feature to protect a dollar amount in a spreadsheet designed for non-computer users. They were instructed on basic operations, but could not change the protected going in value.

Range Erase allows you to erase a specified range in the spreadsheet.

Range Data-Fill permits you to enter a value and increment or decrement it by a specified value.

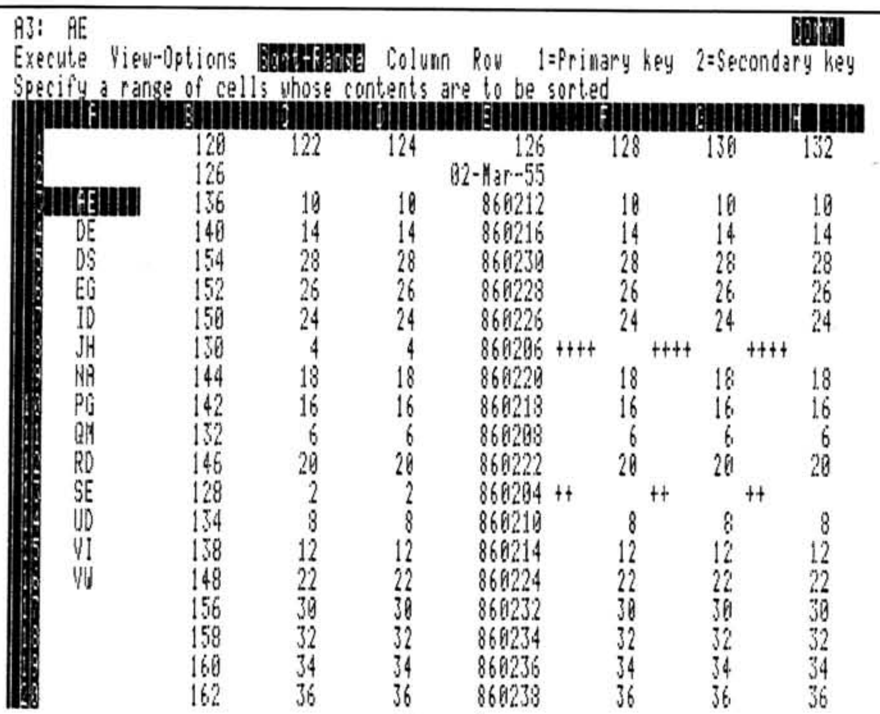
Version 2.0 ENABLE added the sort capability. When you select sort, you identify the range for the sort, the primary and secondary fields, and ascending or descending. Note: Make sure that you select all cells attached to the fields being sorted or the data will get scrambled. You must select Execute when you are done identifying the data for the sort.

Other commands available are single line entries. Insert is the first option and allows you to insert columns or rows into the spreadsheet. The inserted column will be



```
F9:  Above is a repeating entry                                              COMM
1=255x255    2=511x127    3=1023x63    4=2047x31    5=4095x15    6=127x511
Allow spreadsheet maximum size of 255 rows and 255 columns
        A        B        C        D        E        F        G        H
  1     23                      1495
  2     34                  2.826086
  3     65
  4    122
  5
  6
  7         This is a label
  8                                            #########
  9                  Left                      Above is a repeating entry
 10                   Center
 11                      Right
 12
 13
 14
 15
 16
 17
 18
 19
 20
#2     B:\NO-NAME.SSF                                          Cap       F11
```

Figure 5
Global Dimension Line



```
A3:  AE                                                              [illegible]
Execute  View-Options  Sort-Range  Column  Row   1=Primary key  2=Secondary key
Specify a range of cells whose contents are to be sorted
         A         B         C         D         E         F         G         H
                  120       122       124       126       128       130       132
                  126                          02-Mar-55
        AE        136        10        10    860212        10        10        10
        DE        140        14        14    860216        14        14        14
        DS        154        28        28    860230        28        28        28
        EG        152        26        26    860228        26        26        26
        ID        150        24        24    860226        24        24        24
        JH        130         4         4    860206 ++++      ++++      ++++
        NA        144        18        18    860220        18        18        18
        PG        142        16        16    860218        16        16        16
        QM        132         6         6    860208         6         6         6
        RD        146        20        20    860222        20        20        20
        SE        128         2         2    860204 ++        ++        ++
        UD        134         8         8    860210         8         8         8
        VI        138        12        12    860214        12        12        12
        VW        148        22        22    860224        22        22        22
                  156        30        30    860232        30        30        30
                  158        32        32    860234        32        32        32
                  160        34        34    860236        34        34        34
                  162        36        36    860238        36        36        36
```

Figure 6
Sort With View-Option Window

to the left or above the cursor. Note: You must copy formulas into these new rows or columns, if necessary.

Delete will remove columns or rows from the spreadsheet. The selected row/column will be under the cursor.

The Width command permits you to change the size of the column from 1 to 72. The default value is 9. You can change only one column at a time with this command. If the value is larger than the size of the cell, "*******" will fill the cell completely. By changing the width, the value can be displayed.

Copy is a valuable command in any spreadsheet. With this command, you can copy formulas, values, or labels to any other part of the spreadsheet. When selected, you are prompted for the range to COPY FROM and the range to COPY TO. Note: You cannot include the COPY FROM range within the COPY TO range. When you copy formulas, the address of cells moves relative to the copy, i.e., if you copy the formula "c1+c2" from cell c3 to d3, the formula will read "d1+d2". If you want the reference cells to remain the same, you must identify these cells by using a dollar sign, i.e., c1, will always point to cell c1 no matter where it is copied. If the cell reflects "c$1", the column will change, but the row will not.

The MOVE command permits you to move parts of a spreadsheet to another part of the worksheet. MOVE differs from COPY in that the contents of the cells moved are deleted from those cells. During a COPY, the contents remain and duplicate entries are made. During MOVE, you establish the range of cells to move and the starting location at where you want it. Formulas will be updated to reflect the moved location.

TITLES freeze cells above or to the left of the cursor. To enact this command, place the cursor below and/or to the right of the area that you want to freeze. This command can be used to lock titles so that you can see top and/or side labels. Once the area is locked, cursor movements are not possible in the area. Both Horizontal and Vertical titles can be locked.

The LIST command displays the status of the contents of the spreadsheet.

The HILITE command provides you with visual presentation of related cells. When you select HILITE, ENABLE prompts for a cell. Once selected, all related cells are highlighted. On the status line, the total number of related cells will be given. This command is helpful when you plan on updating a spreadsheet. This command will highlight the interrelated cells and "holes" will be displayed if you miss something.

Worksheet ERASE will delete ALL cells in the spreadsheet. This will give you a completely blank worksheet. The file is not deleted from the disk, only the contents of the spreadsheet. ENABLE will ask you if you are sure before deleting the contents, so you have one chance to quit the command with (N)o, the default value.

The last command for this article is VCOPY. This command differs from COPY in that it copies the results of a cell to another cell. You select a range of cells and indicate the TO location. The results are then moved to that location.

Now that you have developed a spreadsheet, it is time to quit. Save the work you have been doing. Press "/" or F10 and (S)ave. Accept the default value by pressing <RETURN>. ENABLE will prompt with the name of the file you are working with, press <RETURN> and ENABLE will prompt with file found, do you wish to (C)ancel or (R)eplace, press (R)eplace and the revised file will be written over the top of the file on the disk.

If during the save, you wish to change the format to LOTUS, SuperCalc, etc., select (C)hange Options. This permits you to change the format, save only value and labels without the formulas, or all, and the range.

This completes the introduction to ENABLE's spreadsheet. Next month, an introduction to ENABLE's database management system (dbms) will be presented. This is one of the good features of ENABLE, especially if you have been using dBase. See you next month.

EXPLORE
NEW WORLDS
WITH
HUG
GAME
SOFTWARE

Enable 2.0 Revisited

Earl R. Zimmerman, Jr.
169 Spinning Road
Dayton, OH 45431

In part 2 of my series of articles on Enable 2.0 for the Z-100, I'll discuss the graphics and menu generator portions of this integrated package. In addition, I'll discuss how you can program a Logitech C7 mouse for Enable. (Part 1 appeared in the August 1987 edition of REMark). As I hinted in the last article, the graphics in Enable are top-notch. Enable created graphics will greatly enhance any presentation you have to make or any report you have to write. This article will present some sample graphs so you can judge for yourself how good Enables' graphics are. Being an Air Force officer, and having had some experience with the government contract Z-248 version of Enable, my fictitious graphs will demonstrate how it can be used to prepare briefings or information packages for my superiors, (with the exception of the open-high-low-close graph).

Types Of Graphs

As George Elwood indicates in his article, you can create two-or-three dimensional vertical bar charts, exploded or standard pie charts, and line, XY, or open-high-low-close graphs. In addition, the vertical bar charts can be stacked.

Graphs can be created from the spreadsheet or the data base and can either be displayed on the screen, printed in various densities, plotted, or copied into a word processing document.

Creating Graphs

Before you can begin creating graphs, you must create the data. The source for all my graphs are spreadsheets. Using the procedure I described in my first article, I called up the spreadsheet Top Line Menu and selected GRAPH and told Enable to CREATE a graph. I then assigned each graph a name of up to 8 letters. A task menu then appeared prompting me to either select OPTIONS, DISPLAY, PRINT, or L=PLOT. Selecting OPTIONS allowed me to begin creating my graph by identifying the data groups, type of graph, and global features.

Generating Graphs

Open-High-Low-Close Graphs

The first graph I will present is the open-high-low-close graph. This graph is used to present a graphic display of stock quotations. Data groups in the spreadsheet represent the daily open prices, high stock prices, low stock prices, and closing stock prices. Low stock prices and high stock prices are such that corresponding high and low prices are connected by a vertical line. High and low points are marked by a dash "-", closed by an "X", and opened as a "<". The following is the spreadsheet (Table 1) the graph was created from.

Table 1
XYZ Stock Quotes

	HIGH	LOW	CLOSE	OPEN
MON	7.00	5.75	5.75	6.00
TUE	6.00	5.50	6.00	5.75
WED	6.50	5.75	6.25	6.00
THUR	8.00	6.25	8.00	6.25
FRI	7.50	7.25	7.25	7.50

The following is the graph I created from this spreadsheet. (Figure 1)

You probably have noticed the different fonts on this graph. Enable has nine different fonts you can select from. There is the default font, and two Italic, Roman, Block, and Script fonts. All fonts will be displayed in the sample graphs.

Vertical Bar Graphs

As a financial manager for a weapons acquisition program, I am responsible for reporting how much of the program's money is obligated monthly against what was originally forecasted. If the original forecast is later changed, a revised forecast is necessary. The best way to display this information is in a vertical bar graph. The following is the spreadsheet (Table 2) and the graph created from it (Figure 2).

Notice that the graph is three-dimensional. This is one feature that sets Enable one step above other Z-100 graphic packages. I have also added a horizontal grid to help viewers determine the actual amounts without difficulty. Grids can be displayed vertically, horizontally, or both horizontally and vertically. In addition, I instructed Enable to ignore some data points in the X axis. It prints every other month with a setting of two. I also set the Y-axis setting to two and what the highest and lowest values should be. Designs depicted in the legends are default values. Enable will print any one of the eight designs you

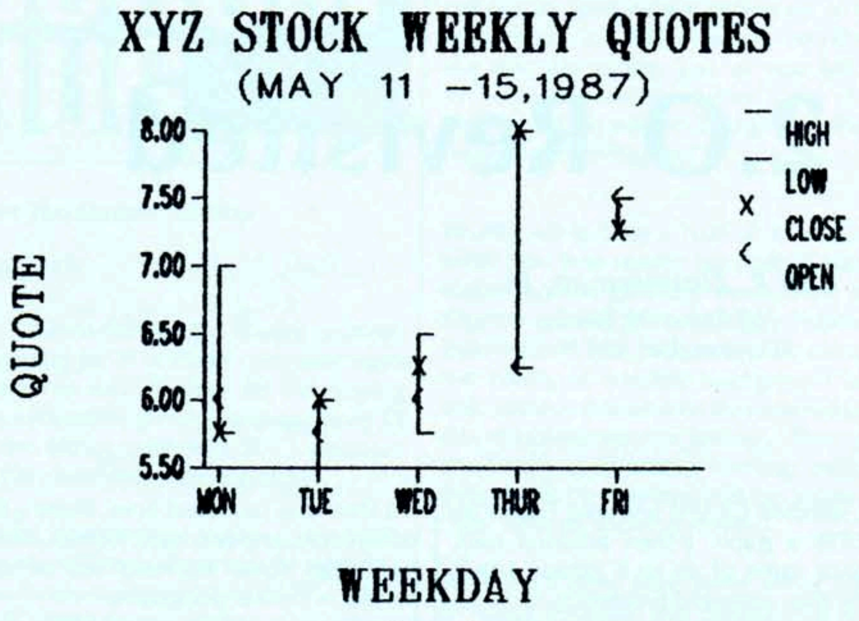


Figure 1
Open-High-Low-Close Graph

Table 2
FY 87 Obligations (Millions)

	FORECAST	ACTUAL	REVISED
OCT	1.00	0.90	
NOV	1.50	1.20	
DEC	2.00	1.63	
JAN	4.39	3.79	
FEB	5.88	5.00	
MAR	6.50	7.00	
APR	8.00		8.50
MAY	9.50		10.00
JUN	10.20		10.50
JUL	10.50		11.00
AUG	11.00		11.50
SEP	14.00		14.90

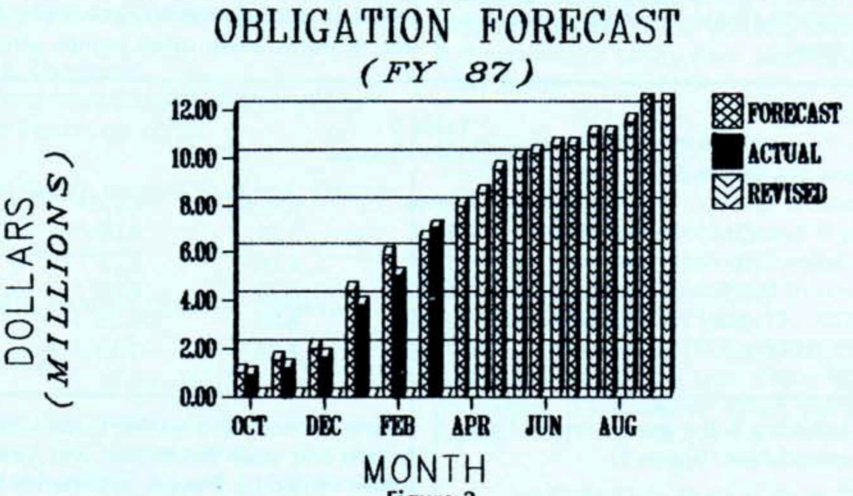


Figure 2
Vertical Bar Graph

have specified.

Pie Chart

Some weapon acquisition programs are funded by more than one branch of the Armed Services. The percentage contributed by each branch can be displayed using a pie chart. In Enable, you can explode an entire pie chart or selected sections if you wish to emphasize them. As with a vertical bar graph, you can specify what shades you want to have or what color you want the graph to be plotted in. Table 3 is the spreadsheet that the pie chart (Figure 3) was created from.

Table 3
% Of Dollars Applied On Program

Airforce	75
Navy	13
Army	12

While I like the explode feature, there is still room for improvement. You should have the capability to create three dimensional exploded pies.

Stacked Vertical Bars

Often it is necessary to compare one group of figures against another and explain how that figure was derived. For instance, a commander may want detailed information on officer manning. He wants to know how many officers are assigned versus authorized, and what rank the officers are. A three-dimensional stacked bar graph will do the trick. Table 4 is the spreadsheet the stacked vertical bar graph (Figure 4) was derived from. All eight shades are displayed in the graph.

Table 4
# Authorized vs Assigned By Rank

	AUTH	ASSIGNED
GEN	1	1
COL	15	14
LC	20	18
MAJ	35	34
CAPT	60	61
1LT	80	77
2LT	90	85
TOTAL	301	290

XY Line Graph

In addition to tracking obligations against a forecast, I must track expenditures in

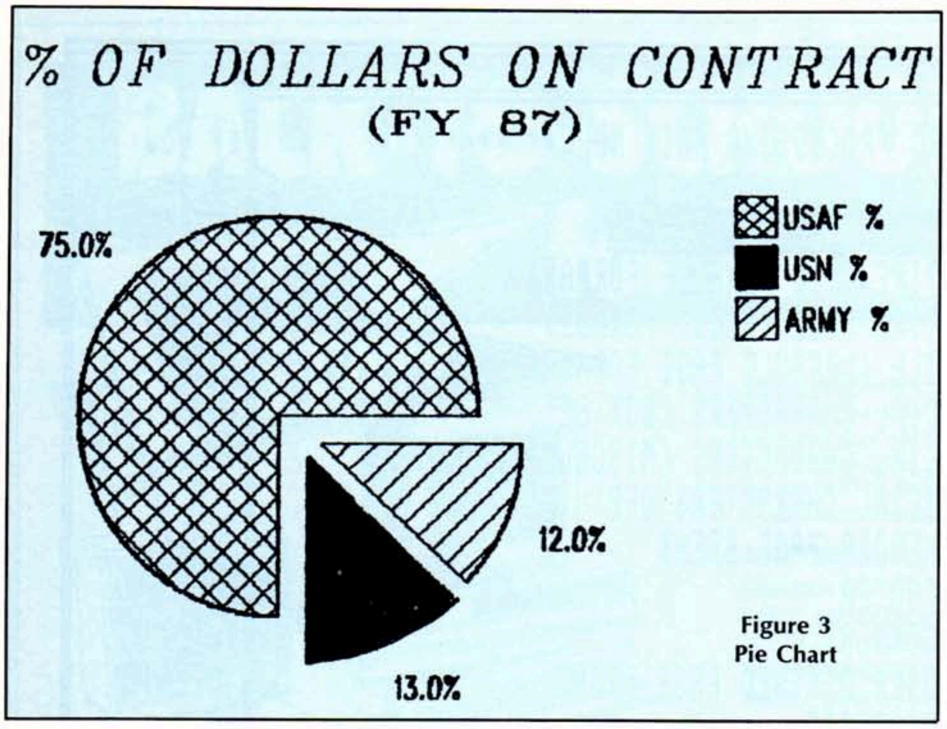


Figure 3
Pie Chart

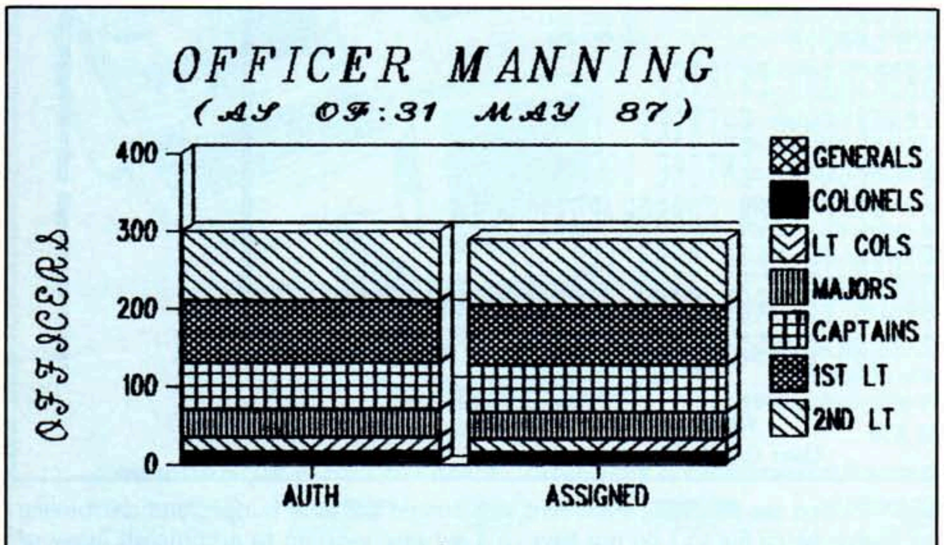


Figure 4
Vertical Stacked Bar Graph

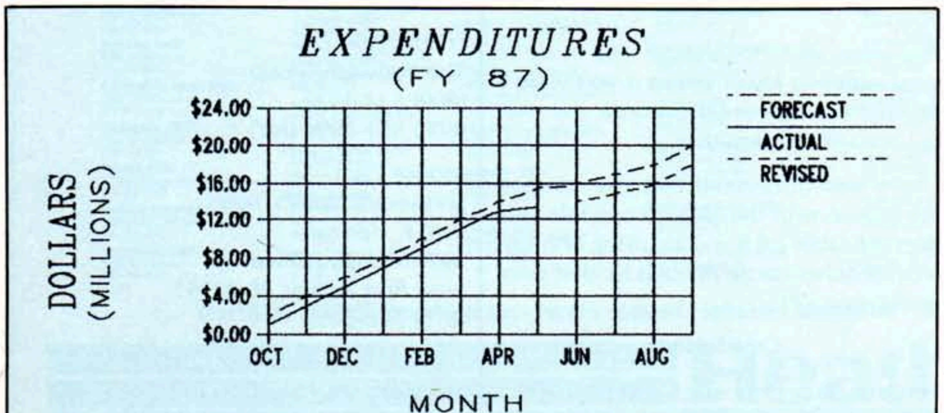


Figure 5
XY Line Graph

the same manner. This can be done with a line graph. The spreadsheet is not shown, as it is similar to the spreadsheet used for the vertical bar graph. Figure 5 is a depiction of a line graph.

You will notice that the grid is both horizontal and vertical. The legend has default shading, and the Y axis is set to dollar format. Other Y axis options include: fixed, integer, comma, percent, date, time, and general. Data points can be user selected, keyboard selected, or special characters can be selected.

Other Graphics Features

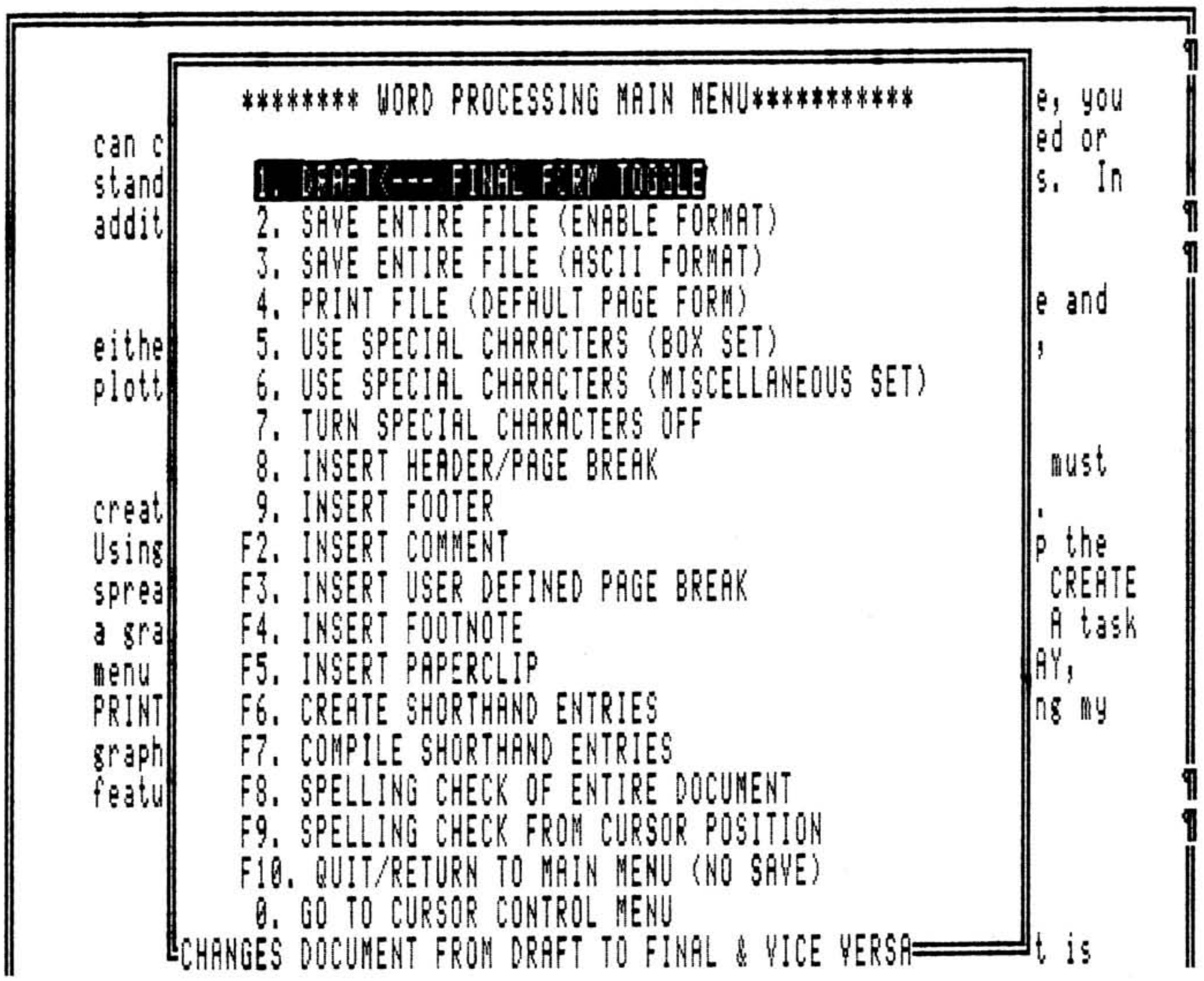
Another feature of the graphics package is the ability to print full and half page graphs in single to quad density (if your printer supports it). Graphs can also be displayed or plotted in color.

Using the Master Control Module (MCM), your graphs can be saved as separate word processing files or copied into word processing documents. Using the MCM windows feature, you can shrink your graphs before incorporating them into your document.

Menu Generator Feature

One advanced feature of Enable is the ability to create your own menus, in addition to those already supplied by Enable. Enable uses expert commands which is a quicker method of accomplishing some tasks on the menus. The trouble with these expert commands is that you have to remember the right combination of keystrokes. One way around this is to incorporate the fast commands into a user created menu.

Figure 6 is what a user created menu looks like inside a word processing document. This is the menu I use when I am using the word processing package. Simply depress the Shift/F10 key and select the option you wish. Before doing this, check that the top line menu is not displayed or the menu will not appear. You have the capability to enter help messages for each option, as well as information messages. These messages can appear on the line below your option, at the bottom of the menu window, or in a separate window you create. Information messages appear when the pointer rests on the option, while help messages appear when the F1 key is depressed.



```
        ******** WORD PROCESSING MAIN MENU***********
                                                         e, you
can c                                                    ed or
stand   1. DRAFT<--- FINAL FORM TOGGLE                   s.  In
addit   2. SAVE ENTIRE FILE (ENABLE FORMAT)
        3. SAVE ENTIRE FILE (ASCII FORMAT)
        4. PRINT FILE (DEFAULT PAGE FORM)                e and
eithe   5. USE SPECIAL CHARACTERS (BOX SET)              ,
plott   6. USE SPECIAL CHARACTERS (MISCELLANEOUS SET)
        7. TURN SPECIAL CHARACTERS OFF
        8. INSERT HEADER/PAGE BREAK                        must
creat   9. INSERT FOOTER                                 .
Using  F2. INSERT COMMENT                                p the
sprea  F3. INSERT USER DEFINED PAGE BREAK                 CREATE
a gra  F4. INSERT FOOTNOTE                                A task
menu   F5. INSERT PAPERCLIP                              AY,
PRINT  F6. CREATE SHORTHAND ENTRIES                      ng my
graph  F7. COMPILE SHORTHAND ENTRIES
featu  F8. SPELLING CHECK OF ENTIRE DOCUMENT
       F9. SPELLING CHECK FROM CURSOR POSITION
      F10. QUIT/RETURN TO MAIN MENU (NO SAVE)
        0. GO TO CURSOR CONTROL MENU
  CHANGES DOCUMENT FROM DRAFT TO FINAL & VICE VERSA      t is
```

Figure 6
User Created Menu

Macros or expert commands can be included within the menu file or they can be separate files. Creating the macros within the menu will save disk space. You also can create sub-menus which you can access from the default menu. For instance, in figure 6, if I select "0", a sub menu that controls cursor movement will appear on the screen.

Using A Mouse With Enable 2.0

To further increase my productivity I use Enable with a Logitech C7 mouse and Paul F. Hermans' Mouse Driver Package. The Logitech C7 Mouse is a three button mouse. I programmed the left button as the Escape key, the middle button as the Return key, and the right button as the F10 key. Using these three keys, I have found I can easily call up and select from the top line menus and the sidebar menus. I included the following statement in my Enable batch file so I do not have to type it in at every session:

```
PCMouse
XS50 YS50 LB27 MB141 RB160
```

You may wish to change the X or Y sensitivities to lower values if you wish. I found the more I use the mouse, the lower I can set the sensitivity.

I have also discovered that you can use the mouse with the IBM compatible version of Enable 2.0 if you are using ZPC 2.0, provided the mouse driver is loaded prior to entering ZPC.

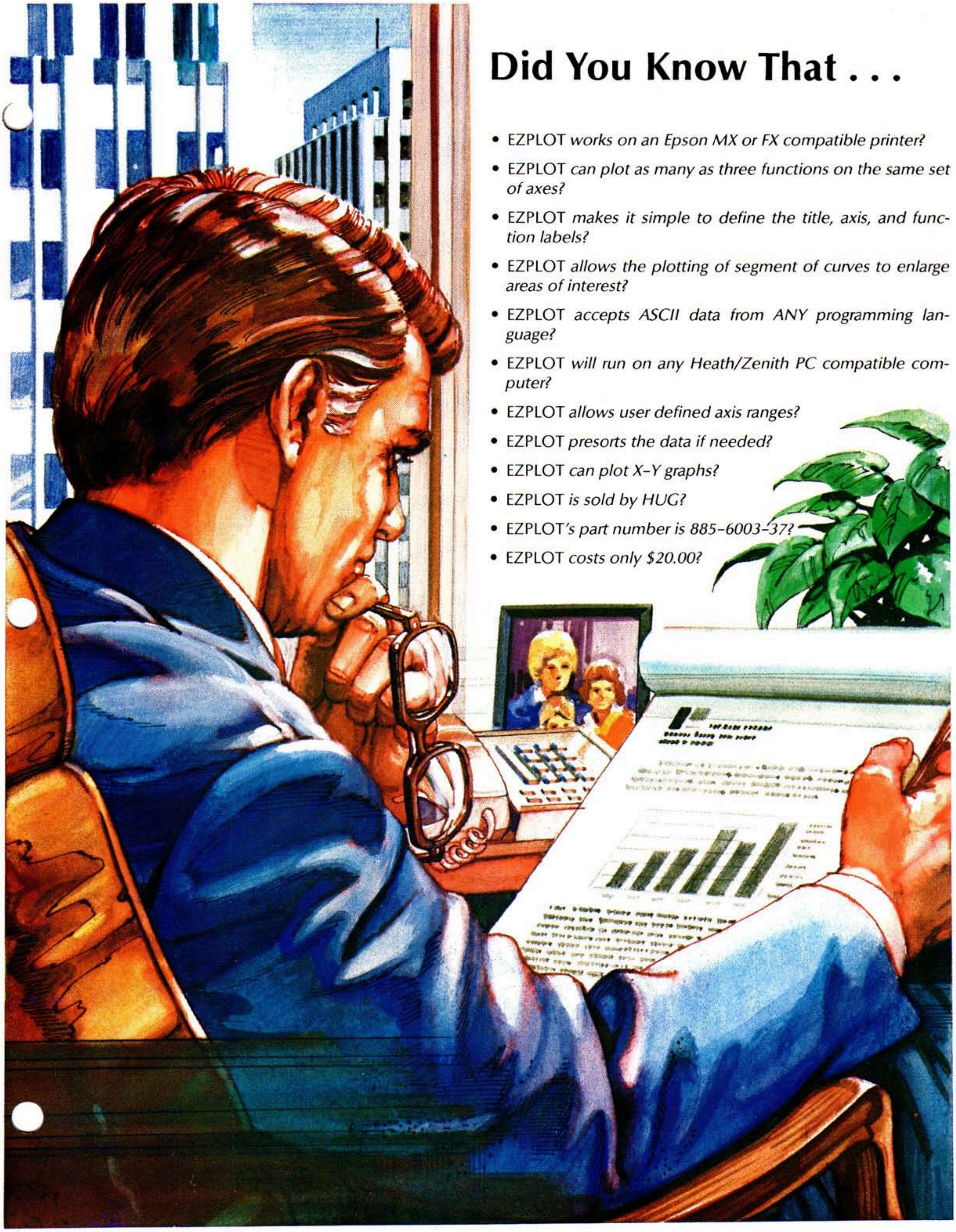
Conclusion

This concludes my article on Enable 2.0 for the Z-100. I hope I have given you a good idea of some of the capabilities of this outstanding program. If you have a limited software budget, and are looking for one program to accomplish many of your needs, I highly recommend this program.

Enable 2.0
The Software Group
(800) 634-3470
(800) 551-1004 (NY)

Mouse Pack
Software Graphics Tools
Paul F. Herman
3620 Amazon Drive
New Port Richey, FL 33553
(813) 376-5457

Logitech Mouse
Logitech, Inc.
805 Veterans Boulevard
Redwood City, CA 94063
(415) 365-9852



Did You Know That . . .

- EZPLOT *works on an Epson MX or FX compatible printer?*
- EZPLOT *can plot as many as three functions on the same set of axes?*
- EZPLOT *makes it simple to define the title, axis, and function labels?*
- EZPLOT *allows the plotting of segment of curves to enlarge areas of interest?*
- EZPLOT *accepts ASCII data from ANY programming language?*
- EZPLOT *will run on any Heath/Zenith PC compatible computer?*
- EZPLOT *allows user defined axis ranges?*
- EZPLOT *presorts the data if needed?*
- EZPLOT *can plot X–Y graphs?*
- EZPLOT *is sold by HUG?*
- EZPLOT*'s part number is 885–6003–37?*
- EZPLOT *costs only $20.00?*

NEW

Eco-C88 Rel. 4.0 Modeling C Compiler

The Eco-C88 C compiler now supports four different memory models, allowing you to have up to 1meg of code and 1meg of data. You also get:

- All data types and operators, prototyping, enum, void, structure passing and assignment
- Over 200 standard library functions; most are ANSI and System V compatible
- Integrated text editor with on-line function help, split windows, edit-compile-link capability
- cc program with mini-make to simplify program development
- Anti-cascading error checking; no more avalanche of false error messages
- User-selectable semantic checking
- Simple installation procedure
- New manual with over 100 program examples

C-more Source Code Debugger

When used with the Eco-C88 Rel. 4.0 C compiler, C-more allows you to:

- set simple, conditional, or counting breakpoints
- examine variables, including indexing
- single-step a program
- evaluate simple or complex expressions
- examine CPU registers
- output values in printf( )-type formats
- easy to read user's manual

plus other features that enable you to find even the most elusive of bugs. C-more requires the Eco-C88 Rel. 4.0 (or later) C compiler. Price is $49.95.

The price for the Eco-C88 modeling compiler is $99.95. For a limited time, we will include C-more, our source code debugger, free of charge when you order the Eco-C88 C compiler. We also have a complete line of other C products and programming aids. For details, call or write:

1-800-952-0472 (orders)
1-317-255-6476 (info)

MasterCard VISA AMERICAN EXPRESS

Ecosoft Inc.
6413 N. College Ave.
Indianapolis, IN 46220

FREE DEBUGGER WITH ORDER

ECOSOFT

Heath/Zenith Users' Group

P.O. Box 217
Benton Harbor, MI 49022-0217

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

POSTMASTER: If undeliverable, please do not return.

P/N 885-2097