

REMark®

Volume 8, Issue 6 • June 1987
P/N 885-2089 Issue 89

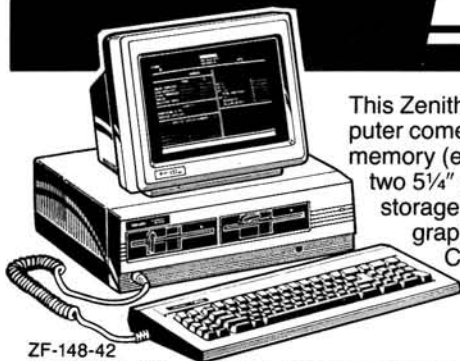
Official magazine for users of **HEATH ZENITH** computer equipment.

A Drawing Program For ALL Heath/Zenith
Z-100 And PC Computers.
See Page 31.

\$2.50



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



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

This Zenith Data Systems computer comes with MS-DOS, 256K memory (expandable to 640K), two 5 1/4" disk drives with 720K storage, both mono and color graphics capabilities. Cable also included.

ONLY \$999*

**Color! Zenith ZF-148-42
with C.I.TOH RGB
color monitor** **ONLY \$1359***



HS-248-S
shown with monitor
(not included)

**NEW! Heathkit advanced
IBM AT Compatible
Computer**

**A sensational 8 Meg, AT
compatible, quick, one-
evening kit. Includes**

**MS-DOS, floppy-only con-
troller, one 5 1/4" 1.2MB disk
drive, diagnostics disk, and
GW-BASIC. Requires video board.
Reg. \$2249.00**

Special Introductory Price \$1999*



DISKETTE SPECIALS!

Get a \$25.95 value Keystone
Pocket Everflash® Camera
FREE when you buy 50 Nashua
MD2D High Quality double-sided
double-density
5 1/4" diskettes at the
special price of \$49.95.*



— OR —



Get a Fuji VHS T120-HS High
Standard video tape **FREE**
when you buy 20 Fuji MF2DD
Micro Floppy high density 3 1/2"
diskettes or 20 Fuji AT
MD2HD Micro Floppy
high density diskettes.
**Your Choice
ONLY \$44.95***



*NO ADDITIONAL DISCOUNTS

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

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

TUCSON, AZ
5616 E. Broadway
602-745-0744

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

SAN JOSE (CAMPBELL), CA
2350 S. Bascom Ave.
408-377-8920

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

SAN DIEGO (LA MESA), CA
8363 Center Dr.
619-461-0110

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

POMONA, CA
1555 N. Orange Grove Ave.
714-823-3543

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

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

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

DENVER (WESTMINSTER), CO
8725 Sheridan Blvd.
303-429-2292

JACKSONVILLE, FL
9426 Arlington Expressway
904-725-4554

HALEAH, FL
4705 W. 18th Ave.
305-823-2280

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

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

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

HONOLULU (PEARL CITY), HI
2402 Andover St.
808-487-0029

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

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

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

KANSAS CITY (MISSION), KS
5950 Lamar Ave.
913-362-4486

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

NEW ORLEANS (KENNER), LA
1900 Veterans Memorial Hwy.
504-467-6321

BALTIMORE, MD
1713 E. Joppa Rd.
301-681-4446

ROCKVILLE, MD
5542 Nicholson Lane
301-881-5420

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

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

FARMINGTON HILLS, MI
29433 Orchard Lake Road
313-852-4171

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

ST. JOSEPH, MI
2387 Lake Shore Drive
616-982-3215

MINNEAPOLIS (HOPKINS), MN
101 Shady Oak Rd.
612-938-6371

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

ST. LOUIS (BRIDGETON), MO
3794 McKelvey Rd.
314-291-1850

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

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

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

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

JERICHO, LI
15 Jericho Turnpike
516-334-8181

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

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

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

CINCINNATI (SPRINGDALE), OH
131 West Kemper Rd.
513-671-1115

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

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

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

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

PORTLAND (TIGARD), OR
10115 S.W. Nimbus Ave.
503-241-3776

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

PHILADELPHIA, PA
6318 Roosevelt Blvd.
215-288-0190

PITTSBURGH, PA
3482 Wm. Penn Hwy.
412-824-3554

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

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

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

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

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

SALT LAKE CITY (MIDVALE), UT
58 East 7200 South
801-566-4626

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

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

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

MILWAUKEE (WAUWATOSA), WI
845 N. Mayfair
414-453-1161

Phone orders accepted.
**OFFER GOOD
THROUGH
JULY 15, 1987**

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

Units of Veritechnology Electronics Corporation

Keystone and Everflash are registered trademarks of Keystone Camera Corporation.

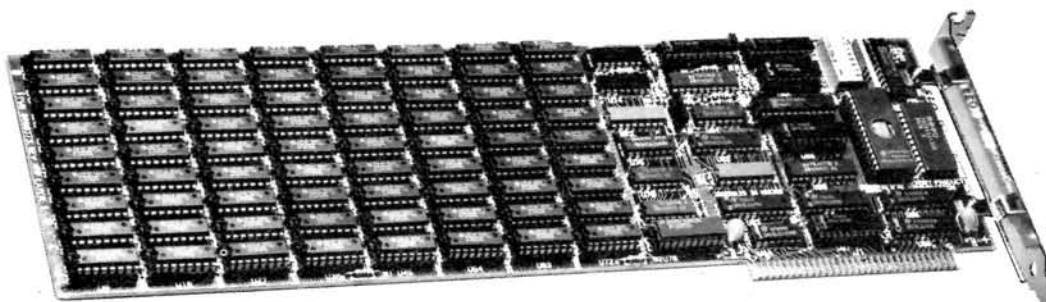
Heath® ZENITH®
Computers & Electronics

HZC-308

Expand your **ZENITH** PC Memory with

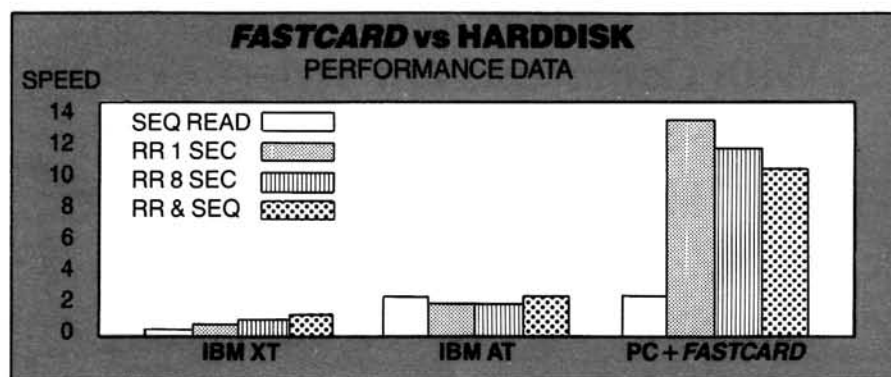
FASTCARD[®]

fully loaded with 2 *MBYTES* for just \$395*



**10 DAY
FREE
TRIAL!**

**FASTCARD speeds disk I/O
by as much as tenfold,
as shown in the following chart.**



This graph shows benchmark test results of a PC equipped with *FASTCARD* versus a PC-XT and PC-AT. These tests were run by a major independent testing laboratory using their standard disk performance benchmark tests.

Specifically designed for compatibility with the 8 MHz Zenith bus, *FASTCARD* provides both extended and expanded memory plus the following standard features:

- Portable between all Zenith models
- Up to 2MB with Split Memory Mapping to
 - Fill memory to 640K
 - Provide Expanded memory over 640K
- Unique Disk Caching
- Ram Disks (up to 8MB)
- Custom Password-Security
- Print Buffering
- Built-in Diagnostics and Automatic Fault Tolerance

*Each *FASTCARD III* comes with 2 MBytes of memory for \$395. *FASTCARD IV*, available with 1 MByte, includes serial/parallel ports and a clock/calendar for \$299. *FASTCARD III* without memory is \$140 and *FASTCARD IV* without memory is \$169.

For additional information, call: **pmi** PERIPHERAL MARKETING, INC. (602) 483-7983
7825 E. EVANS RD., #600, SCOTTSDALE, AZ 85260

Heath/**ZENITH** Users' Group

Managing Editor Jim Buszkiewicz
(616) 982-3837

Software Engineer Pat Swayne
(616) 982-3463

Software Coordinator Nancy Strunk
(616) 982-3838

Production Coordinator Lori Lerch
(616) 982-3794

Secretary Margaret Bacon
(616) 982-3463

HUG Bulletin Board (616) 982-3956

Contributing Editor William Adney

Contributing Editor Joseph Katz

Printer Imperial Printing
St. Joseph, MI

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

* U.S. Funds

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

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

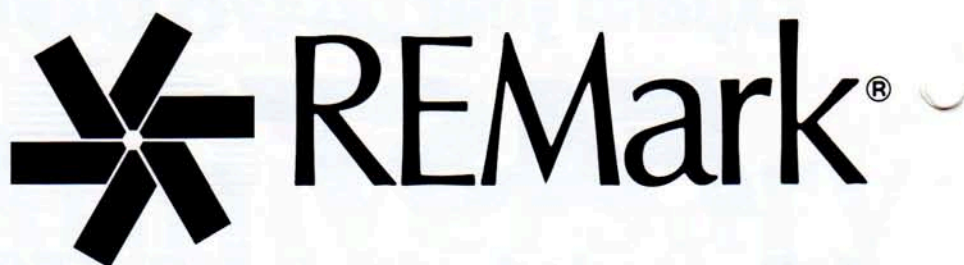
Although it is a policy to check material placed in REMark for accuracy, HUG offers no warranty, either expressed or implied, and is not responsible for any losses due to the use of any material in this magazine.

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

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

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

Copyright (C) 1987, Heath/Zenith Users' Group



Buggin' HUG	7
Introducing HEPCAT: A New Kind Of Pop-Up Calculator	
Pat Swayne	11
What's On The Menu?	
M. D. Zapolski, Sr.	15
Mainstream Computing	
Joseph Katz	21
No Sheet Simple Spreadsheet With Comments On Style	
Theodore J. Stolarz	25
DOODLER-V	
Kevin Lerch	31
Installing And Programming The NEC V20	
Richard L. Ferch	35
Price List	40
HUG New Products	42
ZPC Update #17	
Pat Swayne	45
Performance Tuning Your Software	
Leo R. Hansen	47

Official magazine for users of



computer equipment.

Volume 8, Issue 6 • June 1987

On The Leading Edge

William M. Adney 53

Convert Your Z-100's Keyboard To Dvorak

Dr. Glenn F. Roberts 59

More On Two Terminal Games!!!

Steven W. Vagts 61

How To Convert HDOS MBASIC Programs To Other Microsoft BASICs

Kirk L. Thompson 65

C__Power — Part 4

John P. Lewis 67

Here Comes Help For WordStar

Ami Atir 74

Expanded HUG Discount List

..... 76

COBOL Corner XVIII — Final

H. W. Bauman 77

Official Heath/Zenith Users' Group Conference Registration Form

..... 82

Heath/Zenith Related Products

Jim Buszkiewicz 83

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.
126 Al Davis	79
101 American Cryptronics	64
102 Anapro Corporation	46
127 Bersearch Information Services ...	34
144 Cypher Technologies Corp.	24
104 FBE Research Company	64
105 First Capitol Computer	14
147 H&T Software	30
106 HUG PBBS	6
*** HUGMCP	84
143 Heath Education	9,80
107 Paul F. Herman	34,63
108 Hogware	44
111 KEA Systems	64
148 Laclede Printing Company	51
149 The LaserWriter Connection	12
114 Micronics Technology	69
146 Norcom	24
116 PMI	3
117 Payload Computer Systems	58
132 RAM Technologies	81
119 S&K Technologies	66
121 Scottie Systems	44
122 Secured Computer Systems	57
105 Software Wizardry	52
124 Spectre Technologies Inc.	20
131 Veritechnology Elec. Corp.	2

On The Cover: It's HEERRRE!! What's here, you ask? Well, HEPCAT, of course, the super pop-up calculator! Shown on the EGA displays is HEPCAT popped up over the Microsoft Windows Notepad and over Autocad. For complete details, see Page 11.



"Now I Can Handle More Of You!"

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

MOC

BUGGIN' HUG

Reader Service Is Coming!

Dear Jim:

We're all frantically working on the Reader Card Service program here at HUG, and hopefully, it should be in full operation shortly. Implementing this new service has taken a little longer than expected, and we sincerely apologize for any problems this may have caused your advertisers and readers. Cards have been coming in daily and the reason your readers haven't heard from any vendors yet is entirely our fault. As soon as the software is completed and in place, this card backlog will be eliminated.

Sincerely,

HUG Programming Department

Redirecting Output

Dear HUG:

I recently upgraded to CP/M Plus (Ver. 3.0) for my H-100 and found that the MBASIC routines which manipulated the IOBYTE by PEEK/POKEing to redirect output had to be patched. The same type of redirection of output can be accomplished by altering the "Redirection Vectors" in the System Control Block. CP/M Plus uses these "Vectors" in the same fashion as the IOBYTE except that one can direct output to more than one physical device at a time. The addresses are as follows:

Vector	HEX	DEC
Console Output	0FAC1H	64193
Auxiliary Input	0FAC3H	64195
Auxiliary Output	0FAC5H	64197
List Output	0FAC7H	64199

The physical device is determined by the upper 4 bits of the particular address, thus:

CRT = 128 (080H)
J1 Serial = 64 (040H)
J2 Serial = 32 (020H)
Parallel = 16 (010H)

Examples: To direct listing output to the CRT in MBASIC:

POKE 64199,128

To direct listing output to the parallel port:

POKE 64199,16

It is possible to set these vectors with the sum of the port indices to direct output to two ports at once. Note that directing output simultaneously to ports with different baud rates can occasionally have inconsistent results.

Paul A. Grayce
223 Landing Road
Newport, NJ 08345

JAVELIN .OVL

Dear HUG:

I was delighted to see Pat Swayne's patch for JAVELIN ver 1.1 in the February REMark. I bought a copy recently and have been using it at work for about three weeks. It is an important program and for certain applications superior to LOTUS.

Apparently, there are at least two versions of the OVL (overlay) files distributed with version 1.1. Pat's patch did not work on my copy; but there was enough information in the article so that I could generate one of my own. This is what I used in PATCHER .DAT.

JAVELIN version 1.1 Startup disk
Insert the startup disk

JAV0.OVL
B670,90,90,90,90,90,90
B682,90,90,90,90,90,90
B6BF,90,90,90,90,90,90
B6D1,90,90,90,90,90,90
B724,90,90,90,90,90,90
B742,90,90,90,90,90,90
B7A9,90,90,90,90,90,90
B7C7,90,90,90,90,90,90
B802,90,90,90,90,90,90
B81E,90,90,90,90,90,90
B8B0,90,90,90,90,90,90
B8C2,90,90,90,90,90,90
B8F6,90,90,90,90,90,90
B90A,90,90,90,90,90,90
B94D,90,90,90,90,90,90
1E519,90,90,90,90,90,90
1E538,90,90,90,90,90,90
1E53E,90,90,90,90,90,90

Z

JAVELIN version 1.1 Program disk
Insert the Program disk
(JCONFIG must have been run first)
IBMGGRAPH.DRV

No change same as article

Z

The date and time on my OVL files is 11-24-86 5:00pm. Here is a procedure I would suggest to your readers to tell which patch to use. Load JAV0.OVL into DEBUG, add 100H to one of the four character PATCHER addresses (one of those followed by six 90s), then use the u command — say at

BA0A. The result should be the following instruction sequence:

```
in ax,dx
test ax,0008
jz <somewhere>
```

(Be advised that there is at least one jnz present.)

If that set of instructions does not result, then we have at least 3 different sets of overlays.

The first time I ran my patched program, I got a quick return to DOS and an error message. Apparently, JAVELIN does some sort of directory match on the two overlay files JAV0.OVL and JAV1.OVL. I reset system date and time to match the date and time on JAV1.OVL and then used DEBUG to "fix" JAV0.OVL, i.e.:

```
debug jav0.ovl
-w
```

Setting system time a minute before the desired directory time allows a comfortable margin for loading the file and watching the second hand on a clock.

I hope this information helps others avoid some of the frustrations which I encountered.

Sincerely,

Vernon Reisenleiter
5216 Franconia Road
Alexandria, VA 22310

Unprotect BASIC

Dear HUG:

In the February 1987 issue of REMark, the article on "Unprotect: The Easy (And Safe) Way". I have the following comments for the HUG Engineer's Note: For a much easier way to un-protect GW-BASIC programs.

The following is a simple short way to un-protect BASIC programs written in BASIC and it works with ZBASIC 1.0, GW-BASIC 3.11, BASICA 1.13 that I have and I tried it on the Z-100, IBM PC XT, Tandy 2000 and IBM compatible computers and it works every time. The program is called UNPROT .BAS a 2-byte program created with DEBUG in the following manner:

```
Type: DEBUG <Enter>
-e 100 ff 1a <Enter>
-n unprot.bas <Enter>
-r cx <Enter>
CX 0000
:2 <Enter>
-w <Enter>
Writing 0002 bytes
-q <Enter>
```

The program created above will remove the protected status from BASIC programs,

without over-writing the program already in memory. It will also bring back programs that you used the NEW command on and with ZBASIC if you exit to the operating system and call BASIC back and load UNPROT, the last program you had will still be there for you, this has only worked with ZBASIC.

To use the program you just made, bring up your BASIC and load the protected program and then load UNPROT and the program can be listed saved without the protect flag.

Joseph H. Earl II
CPO Quarters
USS Orion (AS-18)
FPO NY 09513-2570

HD-4040 And The H-8

Dear HUG:

I received a Heath HD-4040 for Christmas and completed the assembly shortly thereafter. I had planned to use the H-8 with it, running under HDOS and HTERM from HUG's P/N 885-1089.

It seems that the H-19 terminal would work correctly, but the H-8 and HTERM would not. The terminal directly to the TNC would work fine, but the H-8 would not print any data back from the TNC, although I could tell by the LEDs on the TNC, that it was receiving data from the H-8 four port serial board (H8-4) with the TNC directly connected to the H-19, I measured the voltages on the DB-25. It seemed that the RTS line on the terminal was about +10 volts, a logic "0" or space with the H8-4 hooked to the TNC, and HTERM in the computer the voltage was -10 volts a logic "1" or mark. Since the TNC requires a +10 volts in order to send data back to the computer, I reasoned that the flaw must reside in the computer or the program. After swapping ports with each other I found them in perfect order.

After a few passes through the source code, I discovered that the modem control port had not been configured, so that a reset would set the port (330Q) to -10 volts without any configuration in the source code. The following code will correct that problem and the TNC now operates in a normal fashion.

After the line that reads:

```
XSPORT EQU XRPORT+5
```

Add the following line:

```
XMPORT EQU XRPORT+4
        XR MODEM CONTROL PORT
```

Then near the end of the menu routine, after the lines that read:

```
MVI A,3
OUT XLPORT
```

Add the following lines:

```
MVI A,3 MODEM CONTROL BYTE
OUT XMPORT SET MODEM RTS LINE +
```

After entering this code to a backup copy of the source code and reassembling, this works fine with the H-8 system here at WOPKN. I hope this letter will be of some use to the huggies out there that are also Hams and have the same setup. I cannot say if this has the same effect on the H-89, but someone out there may have an answer to that question.

Ernest H. Evans, Jr. WOPKN
504 Kingsley Street
Ellsworth, KS 67439

ARAM.COM — Simpler

Dear HUG:

On December 13th, I wrote you a letter about using Ramdrive Utilities on the H-89. In the letter, I outlined the method I used to copy programs and files to the CDR Super RAM 89. Subsequent to sending that letter I learned how to use another program, ARAM.COM, that comes with the Super Ram 89 and greatly simplifies loading programs to the Ram disk.

ARAM.COM is automatically created the same as SRAM.COM when the RAM disk is created with INSRAM.COM. Either ARAM or SRAM can subsequently be used to create the RAM disk/s with the same parameters that were specified when it was last created with INSRAM. The difference between the two is that if the RAM disk/s are created with SRAM, the last step in the process requires the operator to answer "Y" on an initial power up BOOT, and it must be answered "N" to keep what is in Ram disk memory when necessary to Cold Boot without turning power off.

When ARAM is used to create the Ram disk/s, it automatically looks for a flag that indicates whether or not the Ram disk/s are already created. If ARAM detects that the Ram disk/s have been created, it terminates without erasing the Directory. If the Ram disk/s have not been created by the power up BOOT, then ARAM will erase the Directory so the Ram Disk/s can be used.

The detection capability of ARAM makes it possible to use it in a SUBMIT program which automatically copies desired programs and files to the Ram disk with an Autoload command in the Boot routine.

I split the RAM Disk into two drives, A: 248k, and B: 676k. A: for Programs and B:

for Data files. Using Ram disk for logical drive A: permits Warm Booting from the Ram Disk. This makes moving from one program to another or resetting disks almost instantaneous. When Ram disk A: and B: are created by ARAM, (or SRAM) the Boot Drive becomes logical drive C: and any other physical drives become logical drives D: thru n:.

The following is the SUBMIT program that I have loaded into the BOOT routine as "SUBMIT PIPC/A" with SETAUTO.COM. The BOOT floppy drive becomes logical drive C: after ARAM is run. PIPV.COM is CP/M PIP.COM patched to automatically Verify every file transfer.

```
;FileName PIPC/A.SUB
ARAM
C:PIPV A:=C:PIPV.COM
PIPV A:=C:SUBMIT.COM
PIPV A:=C:PIP?????.SUB
PIPV A:=C:D.COM
PIPV A:=C:WS?????.*
PIPV A:=C:KEYWS.*
KEYWS
WS
KEYWS U
;^GDone
```

This SUBMIT program works both on power up BOOT and subsequent Cold Boots. The file execution will continue beyond ARAM only if the RAM disk has not yet been created. If it has not been created, i.e., a power up BOOT, the complete SUBMIT program is run. If ARAM detects that RAM disk is already created, i.e., a Cold Boot without power being turned off, ARAM and the rest of the SUBMIT program are terminated, and the Directory of files previously transferred to RAM disks A: and B: is not disturbed.

While the boot routine actually loads WordStar, it also copies to RAM disk A: the SUBMIT program PIPDBII.SUB:

```
;FileName PIPDBII.SUB
PIPV A:=C:DB?????.*
PIPV A:=C:SB.COM
;^GDONE
```

When called with the command "SUBMIT PIPDBII" this program will copy dBase II and Spellbinder from floppy disk C: to RAM disk A:.

Data Files are copied to RAM disk B: with PIPV.COM as needed. The updated files are copied back to the floppy at the end of each session with either PIPV.COM or with DATSWEEP.COM which can identify those files which have been modified or tagged for copying.

Thanks to ARAM.COM Super Ram 89 is automatic. It is ready to go upon Boot just like a hardware disk. One exception is the requirement to remember to copy

Kitbuilding With Heath Share The Experience

You know the feeling. You've been there. From breaking the seal on your new kit project to finally turning it on, the pride and satisfaction that goes with saying "I built it myself" is extra special. And, you know you have a quality product that will last for years.

Now share this experience with a friend. Check your Heathkit Catalog for the new SK Series Starter Kits. They're the perfect way to introduce a beginner to the pleasures of

kitbuilding. And, as with all of our products, each is backed by our famous service and support. We have a wide variety of kits just waiting to be built, including these new additions featured below.

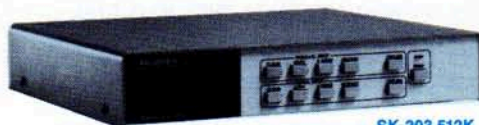
And it's so easy to order. Simply call TOLL-FREE: 1-800-253-0570. Alaska and Michigan residents call: 616-982-3411. Or visit your nearest Heath/Zenith Computers & Electronics Center.



IM-2320
full-function
Handheld Digital
Multimeter kit measures
capacitance, too

Enjoy a wide range of measurement capability with this quality, easy-to-read handheld digital multimeter. The IM-2320 has a single rotary switch for selecting five voltage ranges from 200 mV to 1000 VDC ($\pm 0.5\%$ basic accuracy) and 750 VAC, current ranges from 200 μ A to 10 ADC and from 20 mA to 10 AAC, and six resistance ranges from 200 Ω to 20 M Ω . Even check capacitances in five ranges from 2 nF to 20 μ F with an accuracy of $\pm 2.0\%$ or check a PNP or NPN transistor's h_{FE} .

JUST \$69.95



SK-203 512K
Parallel & Serial Printer Buffer

Stop wasting time waiting for your printer. The SK-203 accepts your document with the speed of your computer! Revise a new document while the buffer prints the last one. Super high speed parallel (to 100,000 cps) AND serial (to 38,400 baud) inputs and outputs. Data is passed unchanged, important for plotters. Special priority print function interrupts the document being printed to print a more important one. Prints multiple copies. 64K memory standard, upgrade with low cost 256K DRAMS. Order SK-203. A terrific value at

\$199.95



GD-3610
Convenience
Light Kit
with Optional
GDA-3511-1
Dual Floodlight
Accessory

The GD-3610 Convenience Light Control brings you the latest in infrared technology to improve the quality of your life and protect your property. The GD-3610 offers automatic lighting to greet guests, light your way at night, even triggers on heat from your car. Also provides the most effective deterrent to crime...light, to ward off an intruder BEFORE he breaks a window or causes other damage. New 8-sensor array offers more sensitivity, better range (easily 60ft) and immunity to false triggering. Auto off during daylight hours. Adjustable delay turns off lights after 1 to 15 minutes. Uses existing wiring and can be turned on from wall switch inside.

ONLY \$99.95

Accessory floodlights GDA-3511-1 only \$29.95

FOR MORE INFORMATION AND ADDITIONAL PRODUCTS, SEE OUR LATEST HEATHKIT CATALOG.

KB-100R1

Heathkit®
Heath
Company

changed or new files back to a floppy before turning off the power. This could also be automated by expanding the SUBMIT program, but for the present I prefer having manual control over which files are copied to a floppy.

I have found the speed of Super Ram 89 makes it well worth the cost. Elimination of the time waiting for drive access to take place has also made the 4mhz modification previously installed worthwhile. WordStar commands: Find and Replace, Move Block, etc. and dBase commands: Find, Index, Sort, Replace, etc. really fly.

Sincerely,

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



**EXPLORE
NEW WORLDS
WITH
HUG
GAME
SOFTWARE**

TIMARQ

Dear HUG:

My reason for writing is to hopefully share the enclosed program with HUG readers. TIMARQ is a program that prints the date and time in reverse video on the 25th line. The date and time will run continuously accommodating leap years. Simultaneously a marquee message continuously repeats across the center of the screen. The

program prompts and remarks are self explanatory and were written in MBASIC-CP/M on an H-89 computer.

I think the readers of REMark magazine would appreciate this short and unique program.

Yours truly,

Dale Schueman
c/o Moraine Valley Community College
10900 88th Avenue
Palos Hills, IL 60465

```

10 GOSUB 530
20 PRINT "COMPLIMENTS OF DALE SCHUEMAN ...12-26-85..."
30 PRINT "...TIMARQ... RECORDS DATE, TIME & MESSAGE ON SCREEN":PRINT
40 GOTO 550
50 PRINT "EXAMPLE ...6..."
60 INPUT "ENTER MONTH..... ";MO$:MO=VAL(MO$):PRINT
70 PRINT "EXAMPLE ...3..."
80 INPUT "ENTER DAY..... ";DA$:DA=VAL(DA$):PRINT
90 PRINT "EXAMPLE ...85..."
100 INPUT "ENTER YEAR..... ";YR$:YR=VAL(YR$):PRINT
110 PRINT "EXAMPLE ...9... OR ...13..."
120 INPUT "ENTER HOUR..... ";H$:H=VAL(H$):PRINT
130 PRINT "EXAMPLE ...5..."
140 INPUT "ENTER MINUTE..... ";M$:M=VAL(M$):PRINT
150 PRINT "TO BEGIN TIMING PRESS ..RETURN.."
160 PRINT "TO END TIMING PRESS ..E.."
170 INPUT ;K:S=0:GOSUB 530
180 PRINT CHR$(27);CHR$(106):' REMEMBER CURSOR POSITION
190 '
200 ' LINE 240 IS THE TIMING LOOP -- INCREASE X,Y,W TO SLOW TIME
210 '
220 FOR X=1 TO 4:FOR Y=1 TO 3:FOR W=1 TO 7:NEXT W:NEXT Y:NEXT X:S=S+1
230 PRINT CHR$(27);CHR$(120);CHR$(49):' ENABLE 25TH LINE
240 PRINT CHR$(27);CHR$(112):' ENTER REVERSE VIDEO MODE
250 IF S=60 THEN M=M+1:S=0
260 IF M=60 THEN M=0:H=H+1:IF H=24 THEN H=0:DA=DA+1
270 IF MO=1 AND DA=32 THEN MO=2:DA=1
280 IF YR=88 OR YR=92 OR YR=96 THEN 300:' TEST FOR LEAP YEAR
290 IF MO=2 AND DA=29 THEN MO=3:DA=1:GOTO 310
300 IF MO=2 AND DA=30 THEN MO=3:DA=1
310 IF MO=3 AND DA=32 THEN MO=4:DA=1
320 IF MO=4 AND DA=31 THEN MO=5:DA=1
330 IF MO=5 AND DA=32 THEN MO=6:DA=1
340 IF MO=6 AND DA=31 THEN MO=7:DA=1
350 IF MO=7 AND DA=32 THEN MO=8:DA=1
360 IF MO=8 AND DA=32 THEN MO=9:DA=1
270 IF MO=9 AND DA=31 THEN MO=10:DA=1
280 IF MO=10 AND DA=32 THEN MO=11:DA=1
390 IF MO=11 AND DA=31 THEN MO=12:DA=1
400 IF MO=12 AND DA=32 THEN MO=1:DA=1:YR=YR+1
410 E$=INKEY$:IF E$="E" THEN 470
420 PRINT CHR$(27);CHR$(89);CHR$(56);CHR$(32):' POSITION CURSOR START OF LINE
430 PRINT MO;"-";DA;"-";YR;" "H";"M";"S";
" TO END PRESS ..E.."
440 PRINT CHR$(27);CHR$(113):' EXIT REVERSE VIDEO MODE
450 GOTO 600
460 '
470 PRINT CHR$(27);CHR$(113):' EXIT REVERSE VIDEO MODE
480 PRINT CHR$(27);CHR$(107):' SET CURSOR TO PREVIOUS POSITION
490 PRINT CHR$(27)+"y1":' DISABLE 25TH LINE
500 GOSUB 530:PRINT " NOW LOOK WHAT YOU DID !!!"
510 END
520 '
530 PRINT CHR$(27)+"E":RETURN:' CLEARS SCREEN
540 '
550 PRINT "ENTER MESSAGE FOR THREE LINES WITH SPACES AFTER EACH LINE"
560 INPUT "PRESS ..RETURN.. AFTER EACH LINE";Z$:PRINT
570 C$=CHR$(27):PRINT C$"E"C$Y,1LINE"1+1:LINE INPUT" ";A$:I=I+1:
IF A$<>" " THEN B$=B$+A$:GOTO 570
580 PRINT:GOTO 50
590 '
600 PRINT C$"x5"C$Y,="LEFT$(B$,20):A$=INKEY$:IF A$=" " THEN PRINT C$+"y5":
END ELSE B$=MID$(B$,2)+LEFT$(B$,1):FOR U=1 TO 90:NEXT:GOTO 220

```



Introducing HEPCAT

A New Kind Of Pop-Up Calculator

Pat Swayne
HUG Software Engineer



One of the most useful programs you can get for a personal computer is the memory-resident pop-up utility, such as Perks or Side Kick. I use Perks on both my Z-100 and PC-compatible computers, and of its functions, the one I use the most is the calculator. But there is one thing that I always wished that Perks, Side Kick, or one of the other pop-up utilities could do that they can't. Since I write programs, my computer spends periods of time doing assemblies or compilations, and during these periods, it is not much use to me. The Perks and Side Kick manuals warn you not to pop them up during disk activity (which can be heavy during an assembly) and even if you do, your assembly or compilation stops while the utility is popped up. So while the computer is engaged in some on-going activity that takes time, there is no advantage to having a pop-up utility.

Enter HEPCAT — The Concurrent Pop-Up

To alleviate this situation, I wrote HEPCAT. HEPCAT is an acronym for HUG Engineer's and Programmer's Calculation Tool. It is a pop-up calculator with more features than most other pop-up calculators. More than that, however, it is the only pop-up calculator that I know of that allows the background task to continue functioning while it is popped up. That means that you can pop it up while your computer is grinding away at a compilation, or while it is re-

calculating a huge Lotus spreadsheet, or whatever, and perform calculations without causing the background activity to stop. HEPCAT only steals processor time while it is actually calculating or updating the screen, not during the entire time it is popped up. It is supplied in versions for both PC-compatible and H/Z-100 (not PC) computers.

HEPCAT Is Compatible

HEPCAT is designed to be compatible with more programs than just about any other pop-up utility. The PC version can pop up during any CGA or EGA video mode, or the monochrome (Z-329) text mode. If you pop up Perks or Side Kick while your PC is in a CGA graphics mode, your computer is forced into a text video mode, and the screen outside the utility window is filled with annoying miscellaneous characters, half of which are usually blinking. When you pop up HEPCAT during a CGA graphics mode, the screen outside the window is undisturbed, and fully restored when you exit from HEPCAT. And HEPCAT is the only pop-up utility that I know of that can pop up on a system with EGA video while one of the EGA graphic modes is active.

HEPCAT is not only compatible with more background programs, but it is also compatible with Perks and Side Kick. You can pop up HEPCAT while Perks or Side Kick are popped up. As an example, you could be

working in the Perks Notepad and pop up HEPCAT if you needed to do any calculations. HEPCAT can be loaded into memory BEFORE Perks or Side Kick, thus preserving the need of those programs to be the last resident utility loaded.

HEPCAT is compatible with most programs that "steal" the keyboard interrupt, and can therefore be popped up over such programs as QuickBASIC or Microsoft Windows. Neither Side Kick nor the PC version of Perks can pop up over these programs.

HEPCAT Is Powerful

HEPCAT is not just a simple four-function calculator. It is actually two calculators in one. It is a full-featured scientific floating point calculator, and a programmer's binary calculator that supports any number base you are likely to need. The floating point calculator features four display modes: floating point, fixed point with 2 to 8 places after the decimal, scientific notation, and engineering notation (scientific notation with the exponent forced to a multiple of 3). The floating point calculator provides 8 significant digits of accuracy with a 2 digit exponent, and its range is 1.0 E-65 to 9.9999999 E+62. It has several built-in functions, including trig functions (in degrees or radians), square root, powers, natural and base 10 logarithms, and even rectangular to polar and polar to rectangular coordinate conversion. It also

THE LASERWRITER CONNECTION

by Joseph Katz
connects an
Apple LaserWriter
printer
to your
IBM-compatible
computer and
lets you profit
from the best of both worlds.



THE LASERWRITER CONNECTION links any Apple LASERWRITER—including the LASERWRITER PLUS—to any IBM-compatible computer. Now anyone can do it, fast.

It lets you use a LASERWRITER reliably on a serial port of any PC, XT, AT, or PS/2 computer, with any version of MS-DOS, on either COM1 or COM2.

Because Apple's LASERWRITER manual and software are for its own MACINTOSH computer, **THE LASERWRITER CONNECTION** takes over where they leave off. They don't do what you need.

You need instructions and software for your IBM-compatible computer and MS-DOS. **THE LASERWRITER CONNECTION** is what you need.

The illustrated manual (produced on a LASERWRITER and a Zenith Z-248) supplements the MACINTOSH manual that comes with the printer. It tells you what you need to know.

For example, you need the right cable for your computer. This manual tells how to buy or build it, where and how to connect it.

You also have to set the LASER-

WRITER switch for proper operation with your computer and your software: use this manual's diagrams as your guide.

Just follow the simple, step-by-step instructions for quick-and-easy installation of the printer on your computer.

The software is a suite of easy-to-use programs for new and advanced users.

LASERONE is the traffic director between your computer and LASERWRITER. Run it once to install communications protocol and handshaking, and you can even use the MS-DOS PRINT.COM as your LASERWRITER spooler.

DELASER lets you use a parallel printer also.

MACHEATH translates MACINTOSH text (ASCII) files to MS-DOS text (ASCII) files.

PSPRINT does simple (no features like italics or boldface) POSTSCRIPT formatting of text (ASCII) and WORDSTAR files. Sends them to a file for further editing or directly to the LASERWRITER for fast printing.

STAT reports important information about your LASERWRITER, including

ROM version, fonts (permanent and downloaded), and communications parameters.

HANDSHAKE selects software (XON/XOFF) or hardware (DTR/RTS) handshaking on LASERWRITERS that support both.

THE LASERWRITER CONNECTION

is published by HUG.
Only \$40.

Part Number: 885-8050-37.

Order from Heath Company, Hilltop Road, St. Joseph, MI 49085 (Att: Parts Department). Telephone 616/982-3571. You must specify you are ordering Part Number 885-8050-37. Visa and MasterCard accepted.

Software and manual are for Heath, Zenith, IBM, and all other true compatible computers.

Copyright © 1987 by Joseph Katz. All Rights Reserved. Logos are trademarks of Apple Computer and International Business Machines, used here for identification only.

does conversions between several American and Metric measurement units.

The binary calculator is a 32-bit calculator that works in the following number bases: binary, tetral (base 4), octal, split octal, decimal, and hexadecimal. It supports the 4 basic calculations (add, subtract, multiply, and divide), and also modulo (finding the remainder of a division), shift left or right, and the logical functions AND, OR, and XOR. You can easily convert a number from one base to another, and transfer numbers from the binary calculator to the floating point calculator and vice versa.

HEPCAT has 10 memory cells into which you can store either floating point or binary numbers, and a unique memory map feature that lets you see which cells are empty, which hold binary numbers, and which hold floating point numbers. If you recall a binary memory cell while the floating point calculator is active, the number is automatically converted to floating point, and vice versa.

The HEPCAT calculators use infix notation. That means to calculate $100 - 99.99$, you enter $100 - 99.99 =$.

HEPCAT Is Easy

With all of its power, HEPCAT is designed to be easier to use than other pop-up calculators. The commands are intuitive, and easy to remember. For example, to change radices or floating point display modes, you just hold down the shift key and use the up or down arrow keys (on a Z-100, the separate arrow keys can be used without the shift key). The number on the screen is automatically converted as you go through the bases and modes. When you are in the binary mode, the left and right arrow keys can be used to shift a number left or right. And in the fixed point display mode, the left and right arrow keys are used to move the decimal point.

The backspace key is the only key you need to remember for clearing the display. While you are entering a number, the backspace key can be used to back up one digit at a time. When you have completed a calculation, the backspace key can be used to clear the calculator. HEPCAT is intelligent enough to know what to do with the backspace key in each situation, and does not need separate clear and master clear keys.

The many functions in the floating point calculator all use easy-to-remember single character commands along with the shift key for certain functions. For example, you use S to calculate the sine of a number, and Shift-S to calculate the arc sine of a number. L is used to calculate the logarithm (base 10) of a number, and Shift-L is used to get the anti-logarithm (10^X). Similarly, N and Shift-N are used for the natural logarithm and natural anti-logarithm (e^X). I think that you will find the HEPCAT command structure easier to remember than just about any other pop-up calculator.

HEPCAT is designed to let you do as much as possible from the keypad. On a Z-100, the enter key is used as a plus key, and you can use shift-plus (shift-enter on a Z-100) for multiplication, and shift-minus for division. Since the plus key can serve as the equal key (once again, HEPCAT is intelligent enough to know what to do when), you can perform any basic function from the keypad, without having to remember which arrow key is for multiply, which one is for divide, and which one is equal, as you must do with Perks (Z-100 version). On PC-type computers, the Prt Sc key (which is also a * key) can also be used for multiplication, and on both computer types, the \ key can be used for division in the floating point mode (\ is used for Mod in the binary mode). Of course, you can also use the main keyboard +, -, *, and / keys.

HEPCAT only uses two function keys. F1 is used to exit HEPCAT without clearing it from the screen (ESCape is the normal exit), and F2 is used to release the keyboard to the background program while HEPCAT is still active (a useful feature for jotting down information from the HEPCAT display).

HEPCAT Is Tiny

With all of its power, HEPCAT only uses about 16k of memory in your computer or on your disk. That means that you can use it on bare-bones systems on which other pop-ups are out of the question. Since the HEPCAT command structure is so easy to learn, there was no need to display prompts on the screen, so the HEPCAT window is only two lines by 34 characters, and is normally located near the top right corner of your screen (you can move it over a small range), out of the way of most of your work.

HEPCAT Is Informative

HEPCAT does not waste its window space with information you don't need, and instead provides information that the others don't. It shows you which angular measurement mode is active (degrees or radians), which number base or display mode it is in, which accumulator is active (X or Y), and the current mathematical function. When you use one of the built-in immediate functions, it shows that on the screen, too. For example, it shows SIN if you take the sine of a number, or SQRT if you use the square root function.

And That Ain't All

Have I said enough to get you interested in HEPCAT? Well, I almost forgot to mention that it also includes an ASCII table, another important aid for programmers. To learn how you can get your own copy of HEPCAT, see the New Products page in this issue of REMark.



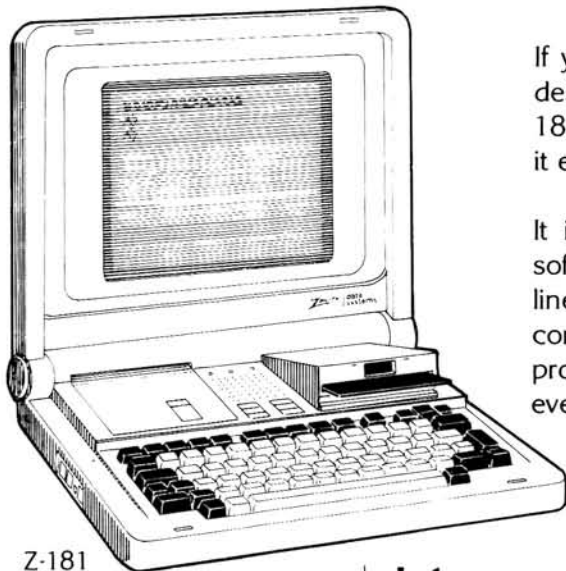
**Are you reading
a borrowed copy of REMark?
Subscribe now!**

MOVING?



**Please let us know 8 weeks in
advance so you won't miss a
single issue of REMark!**

Portable Computing from Zenith and FCC



Z-181

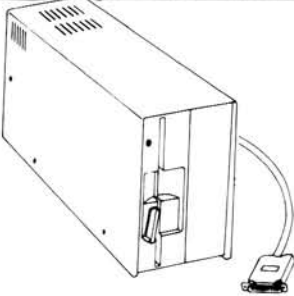
ZENITH data
systems

AUTHORIZED SALES AND SERVICE

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

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

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



**FCC External
5" Drive**

\$250

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



**Diconyx
Printer**

\$395

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



**Ruggedized
Carrying
Case**

\$295

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

**First
Capitol
Computer**

1106 First Capitol Dr.
St. Charles, MO 63301

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

1-314-724-2336

Technical support and order status

Before you buy

**Call Us First.
And Last.**

What's On The Menu?

M. D. Zapolski, Sr.

226 North West Avenue
Bridgeton, NJ 08302

Introduction

Do you have several BASIC program disks lying around? Wonder what's on them? When it comes time to play a game or run a program, do you have to figure out what's where, then load BASIC, and enter the program name to run the program. Wouldn't it be nice if there was a program that would run when BASIC was loaded, locate all the .BAS files, display them on a menu where you simply moved the cursor to the proper program name, and then pressed the enter key to run it? If that kind of operation sounds interesting, then read on as I take you through the development and operation of MENU.BAS.

Besides describing MENU's operation, I will discuss the following BASIC concepts and topics:

1. The SHELL command
2. Program structure
3. Specialized subroutines for cursor, color, and message control.
4. Use of the 25th line.
5. Functions and Error trapping

MENU is written in GW-BASIC to take advantage of the SHELL command, which is not found in ZBASIC. I have a ZBASIC version of MENU, but it is not as automated as the program described in this article. The ZBASIC version uses a random access file to store the file names, where in GW-BASIC, the SHELL command can extract

this information directly from the disk directory. For those of you that would like a copy of the ZBASIC program, leave me a note on the HUGPBBS (St. Joseph, MI) and I will upload it as ZMENU.ARC.

The Concept

I have seen several programs that displayed a menu of files, yet none automatically read the disk directory, or ran the program by positioning the cursor adjacent to the program's name. But this is what I wanted the program to do. So with this need, and my "rusty" knowledge of BASIC, I started to plan MENU's operation. Figure 1 illustrates the functional flow of the program. Invoking MENU at the start of BASIC was no problem. Just enter the command line `BASICA MENU`. BASIC will load, and then run the program `MENU.BAS`. To further automate this process, a batch file may be used to enter this command line.

Program Development

The next task was to display the menu itself. What should it look like? How large should it be? These and other questions needed to be answered. Since a full file name uses 12 characters and the CRT displays 80 characters, you could have 6 names per menu row. And, the names should be separated by spaces for readability. I opted to use 2 spaces. Now each name on the menu occupies 14 spaces allowing room for 5.71 columns. This

meant the menu could have 5 file names and leave 79 free pixels for the menu's borders ($0.71 \times 14 \times 8$). A similar analysis can be performed to find the maximum number of rows. However, in this case, I did not use all 24 rows of the CRT screen. I left some room for message displays. Consequently, the maximum # of rows was set at 10. Although it could have more rows, I found it difficult to consider more than 50 games, or programs on a single disk, or in a subdirectory. With MENU's display size set, we can fix its graphic appearance. For that I chose a simple boxed border wide enough to allow a title and messages to be displayed inside the border. As the H/Z-100 characters are 9 pixels high, the border should be at least 11 pixels wide.

The key to MENU's operation is the SHELL command. This "gem" lets the user execute a DOS command while in BASIC. Essentially, MENU will obtain a directory listing of all .BAS files on the current disk, store them in a sequential file, filter out of that file only the file names, and then store them in an array for future use. All these actions are "transparent" to the user, and take about 3 seconds.

Let's look at this process in more detail. Figure 2 shows a small program using SHELL. Line 30 causes the `DIR *.BAS` command to be executed within BASIC, and the output of DIR is redirected to the file `MENU.DAT`. Did you notice this file was

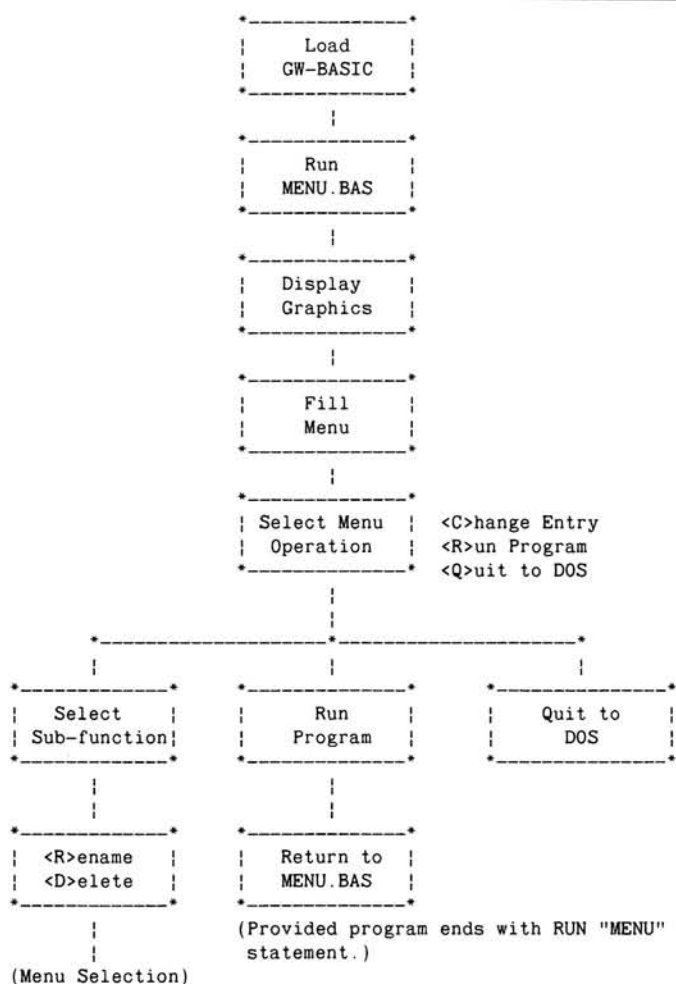


Figure 1
Program Flow

not OPENed first? To save memory, the program relies on the fact that output redirection can force the creation of a file even if it never existed. Line 40 OPENs the file MENU.DAT for input to the BASIC program, and INPUTs (reads) each record as F\$. Line 50 prints the record # and the record. The result of this operation is displayed below the program. As you can see, not every record has information that will be needed by MENU. We only want the file name. In the actual MENU program, these records will be filtered such that only the proper data remains.

Program Structure

Over the past four years, I have read several articles on the merits of various program structuring techniques. From these articles, and some trial and error testing, the organization of all my BASIC programs has evolved to the structure shown in Figure 3. Line 1 contains information that may seem too basic to consider. But, you may be surprised at the number of programs I've obtained from various Bulletin Boards that

don't have it. To find the size of your program, you need only type the statement: ? xxxx-FRE(0), where xxxx is the # of bytes shown free when your version of BASIC is loaded. Did you know that ? is expanded by BASIC as PRINT ? I can't recall any BASIC manual ever describing that item; but, is it handy!! The next 3 lines are quite important. This is the area where program assumptions, limitations, and the operating environment are summarized. Have you ever written a program, then needed to change it 3 months later only to find you forgot some key point? Well, lines 2-4 are the place for that kind of information.

Lines 5 and 90 are the boundaries of a special area divided into two ranges: lines 5-29 and 30-90. The first range is used for program initialization statements:

1. Declare variable types
2. Dimension arrays
3. Assign initial values to variables
4. Define functions

The end of this range is always identified by a GOTO 100 statement. The second range

is a merge area. In it will be placed a group of frequently used subroutines for cursor control and message display. Since I use the same routines for each BASIC program, I have a separate file for them that is MERGE'd into this area. In this way, program development is made much easier as the purpose and routine # of each routine are quite familiar. Moreover, a past REMark article (August 1984, pg 14) cited the value of "placing frequently used routines near the beginning of the program . . . to save execution time. Each time a program statement, such as GOTO, GOSUB, etc., sends the MBASIC interpreter out of [the] line number sequence, the interpreter has to start back at the first line of the program and examine each line number in sequence until it finds the one for which it is searching. Therefore, if the line is near the beginning of the program . . . it will be found sooner."

MENU.BAS

The main program (Listing 1) starts at line 100. After clearing the screen and loading MENU.DAT with the .BAS file names, the program draws the boxes that make up the borders using LINE statements. Did you notice that the border is exactly 11 pixels wide? This dimension allows sufficient room for the menu's title and messages.

Lines 120-130 input the .BAS files from the current disk, stores them into array F\$, and sets the value for NF (number of files). Line 210 then places these names at the proper menu position. The key to this process is subroutine 65. It computes the actual row and column number (AR & AC) for each file name primarily as a function of the counter I. As a side-note, one of the most gratifying parts of programming is the development of algorithms, such as routine 65. I believe it is the heart of the programming process. To understand how the routine works, look at Figure 4 as I describe the processes involved.

Assume that the menu will have a maximum of 4 columns and 6 rows. Thus, MC=4 and MR=6. We know a file name occupies 12 characters, and they will be set 2 spaces apart. Consequently, W=12 and SP=2. In our analysis, we will assume 10 BASIC files will be listed on the menu. Figure 4A shows new row numbers start at even multiples of MC. It would then seem that the actual row (AR) number of a file name was some function of the counter I and MC.

To help discover what expressions will produce the desired result, I often prepare a table like Figure 4B. I first list what is known

(I & MC), and then look for repetitive sequences that can be expressed in mathematical relationships. The file number (I) will increment from 1 to 10, if it is divided (integer-wise) by MC, the result will be as shown in the 3rd column. This result is only off by 1 from what we want. Just add 1 you say? Well, then Fn4 would be in row 2 instead of row 1, and Fn8 would be in row 3, etc. Using combinations of BASIC operators, I came to develop the $\text{NOT}(\text{I MOD MC})=0$ term. As you can see, it is perfectly acceptable to mix logical, arithmetic, and relational operators in a single term. It would take quite some time to explain exactly how the right combination of operators were put together to arrive at a workable formula. I don't think it would be sufficiently instructive to go into that detail. In as much as this formula works, there are most likely other methods that also work.

Knowing the actual row number is half the battle. In a similar fashion, the formula for AC (actual column) was developed. For this calculation, I observed that AC would be a multiple of AR, and be spaced evenly at widths equal to the sum of W and SP. Further, AC would have to be offset from the left margin by a fixed amount (STC — starting column number). Using these facts, and lots of paper and pencil, the formula for AC was produced (Figure 4C). Before returning to the main program, AR is adjusted for a constant offset (STR — starting row number).

Cursor Control

With the file names now displayed on the menu, the file name counter (I) is set to 1, the cursor box color (BC) is set to cyan (3), and a box is drawn around the first file name. Subroutine 61 draws this box which serves to identify the file the user wants to work with. To be useful, the box should move on a single keystroke. In doing so, the routine would have to produce pixel values from AR and AC to properly position the box. Figure 5 shows the cursor box and some of its important characteristics. One thing I found interesting was the pixel reference for the LOCATE statement. On Figure 5 you'll see a # mark. This is the apparent reference point (pixel-wise) for characters positioned by LOCATE. I noticed this when searching for the correct formulas for X1 and Y1 used in routine 61's LINE statement. If the LOCATE reference were in the upper-left corner of the character matrix (8 pixels x 9 pixels), the X1 and Y1 formulas would have AC and AR as factors, instead of (AC-1) and (AR-1).

After printing the menu's main selection prompt, line 230 waits for a key to be

pressed. If the key is an arrow key, the program is diverted to the routine at line 90. This routine moves the cursor box over the physical limits of the menu. First, it blanks the existing box. Then, it determines which arrow key was pressed, and changes the value of I accordingly. Finally, a box is drawn at the new location and the routine returns to wait for the next keystroke. Although the existing program only works with the arrow keys, it can be easily

allows the user to delete or rename a file, or return to the main menu options.

Selecting "D" invokes the Delete routine. A special, boxed "DELETE FILES" message is shown on line 22 to remind the user of the option in effect. On line 25, the user is prompted for a file name. Once entered, the user is given a chance to change his mind before the file is actually deleted. If the user answers "yes" to this question, the

```
30 SHELL "dir *.bas >menu.dat":I=0
40 OPEN "I",1,"MENU.DAT":WHILE NOT EOF(1):INPUT #1,F$
50 I=I+1:PRINT I;" ";F$
60 WEND

1
2 Volume in c is FDISK-1
3 Directory of C:\bas
4
5 TEST      BAS      129   12-01-86   2:07p
6 TEST1     BAS      159   12-02-86   4:09p
7 MUSICA    BAS     15072  10-20-83   12:00p
8 LOAN      BAS      4634  10-03-86   8:56a
9 4 File(s) 5361664 bytes free
```

Figure 2
Directory Extraction Method

modified to accept the 2, 4, 6, 8 keys for cursor direction.

Line 240 takes an action depending upon which key the user presses. If it is one of the "authorized" keys (C, R, Q), program control is transferred to the proper line. Otherwise, an "Oooops" message is displayed. This item will be discussed later in greater detail. Pressing "Q" causes BASIC to return to the DOS prompt. An "R" will RUN the program highlighted by the cursor box.

If the chosen program includes a RUN "MENU statement at its end, another program may be selected from the menu. Pressing "C" changes the lower menu message to the one listed in line 310, and

file is deleted, and the new menu is shown. Instead of returning to the main menu options, the user is asked if he wants to delete another file. The Rename routine, starting at line 700, operates similar to the Delete routine. From this point on, the program is relatively straightforward, except for two areas: message handling & error trapping.

Message Handling

As I mentioned earlier, all my BASIC programs have a section where messages and key inputs are handled. Generally, I display all questions, cautions, and error messages on the 25th line. This gives me the freedom to use the rest of the screen for graphics without concern over how a message

```
1 REM [ date ]           [ Program name ]       [ # of bytes ]
2 REM
3 REM                   [ Program Notes ]
4 REM
5
6                   [ Program Initialization ]
7                   [ Last line contains GOTO 100 ]
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100 REM Program Start
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2
```

might interfere with them. A foundation for the message display routine is my message centering function [FNCT(M\$)]. Based on the length of the message (M\$), this function returns the proper starting column # to center the message on any line:

```
FNCT(M$)=40-INT((2+LEN(M$))/2)
```

The Print Message routine starts on line 40. When called, it first deletes any previous message on the 25th line (PRINT D25\$), then saves the cursor's position. Next, the 25th line is enabled, and the message is LOCATE'd and PRINTed. For this example, the message is in normal video. But by inserting a SCREEN,1 statement, a reverse video message can be displayed.

The Key Input routine starts at line 70. It begins similar to routine 40 and prints M\$ on line 25. Occasionally, M\$ is set to a null string (i.e., M\$=""). This is done to capture a keystroke for processing (R\$=INPUT\$(1) takes the first key pressed). This information is modified by the "return upper case" function:

```
FNCP(R$)=CHR$(ASC(R$) AND &HDF)
```

This function saves programming time (and space), since you don't have to worry whether or not the user has the CAPS LOCK key pressed. All lower case letters will be changed to upper case. But a word of caution is needed here. If you use this function for characters other than letters, you can easily change them to something you may not want.

The "Oooops" routine begins at line 80. Its name is derived from its use. For example, look at line 240. Once the program has responded to any cursor movement requests, it checks to see if one of the "authorized" keys was selected. If the user inadvertently touched the wrong key, routine 80 would be invoked. The "Oooops" message is displayed for an interval, governed by routine 30, and then the user is given another chance. In effect, an error trap was activated.

Error Trapping

One of the more difficult aspects of programming is to anticipate user errors, and control their result. Moreover, the user should be given a chance to correct the problem with little, or no, effect on the program's operation. All that sounds good, but what does it mean? When line 650 executes, a disk file will be deleted. What happens if the user enters the wrong file name, or it doesn't exist on the disk? Normally, an error message would be displayed at the current cursor position, and the program would stop. This would be

		Column			
		1	2	3	4
Row ->	1	Fn1	Fn2	Fn3	Fn4
	2	Fn5	Fn6	Fn7	Fn8
	3	Fn9	Fn10		

(A)

I	MC	I\MC	I MOD MC	NOT (I MOD MC)=0	AR
1	4	0	1	-1	1
2	4	0	2	-1	1
3	4	0	3	-1	1
4	4	1	0	0	1
5	4	1	1	-1	2
6	4	1	2	-1	2
7	4	1	3	-1	2
8	4	2	0	0	2
9	4	2	1	-1	3
10	4	2	2	-1	3

```
AR=I\MC-(NOT(I MOD MC)=0)
```

True = -1, False = 0

(B)

```
AC=STC+(I-MC*(AR-1))*W-W
```

Where W=W+SP

(C)

Figure 4
Actual Row and Column Number Calculations

unacceptable for general use. Instead, the programmer should anticipate this error, trap it, and control how (and where) the error message is displayed. The user should then be given the opportunity to correct the mistake, and the program should resume operation as if the error never occurred. For example, line 20 tells the program ON ERROR GOTO 800. This statement "sets" the trap. Once at line 800, the program will check the error number (ERR), and which line produced it (ERL). In this case, a "file not found" error (#53), would have occurred at line 650. Consequently,

line 860 would display the error message on the 25th line. When it "timed-out", the program would RESUME execution at line 220. In this way, the effect of the error has been controlled, and the program continues without having to re-RUN, or reboot.

How do you know what errors to look for? Perhaps experience in using and writing programs provides the best insight. But if you're still unsure, review the available error messages provided by BASIC, and look at your program for operations that

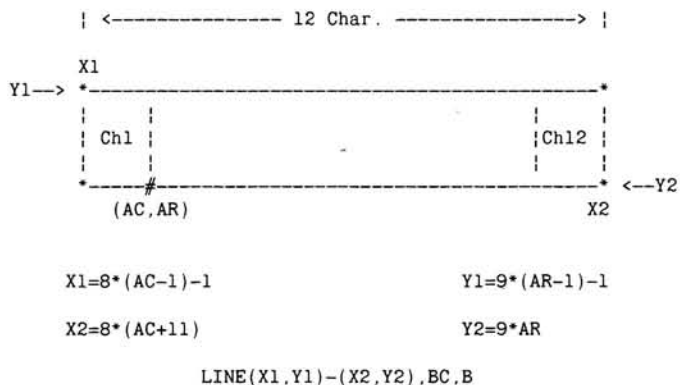


Figure 5
Cursor Box

could generate those errors. Then, add the appropriate IF...THEN statements to service the errors that are likely to occur. Follow these items with RESUME statements to allow the program to continue execu-

tion at the proper line.

Summary

MENU is a program that should allow the user quite a bit of flexibility in RUNning

GW-BASIC programs. Its main features are its cursor selectability, and use of the SHELL command. I hope that many HUGgers will find the program, and the explanation of its development, useful. *

```

1 REM 12/03/86                GW-BASIC Menu Selection Program          2,820 bytes
2 REM
5 DEFINT A-Z:MR=10:MC=5:W=14:STC=6:STR=5:SP=2:DIM F$(MC*MR):E$=CHR$(27):
  E25$=E$+"x1":D25$=E$+"y1"
20 DEF FNCT(M$)=40-INT((2+LEN(M$))/2):DEF FNCP$(M$)=CHR$(ASC(M$) AND &HDF):
  :ON ERROR GOTO 800:GOTO 100
30 FOR S=1 TO 3000:NEXT:PRINT D25$:SCREEN ,0:LOCATE R,C,1:RETURN
40 PRINT D25$:R=CSRLIN:C=POS(I):PRINT E25$:LOCATE 25, FNCT(M$),1:PRINT M$:
  RETURN
56 LINE(250,183)-(390,203),7,B:LINE(260,187)-(380,198),7,B:PAINT(251,196),3,7:
  RETURN
57 LINE(112,162)-(520,170),0,BF:RETURN
59 LINE(250,183)-(390,203),0,BF:RETURN
61 LINE(8*(AC-1)-1,9*(AR-1)-1)-(8*(AC+11),9*AR),BC,B:RETURN
65 AR=I\MC-(NOT(I MOD MC)=0):AC=STC+(I-MC*(AR-1))*W-W:AR=AR+STR:RETURN
70 PRINT D25$:COLOR 7,0:R=CSRLIN:C=POS(I):PRINT E25$:LOCATE 25, FNCT(M$),0:
  PRINT M$:R$=INPUT$(1):PRINT D25$:R$=FNCP$(R$):RETURN
80 COLOR 4,7:PRINT D25$:M$="Ooops... try again please":GOSUB 40:GOSUB 30:
  M$="" :COLOR 7,0:RETURN
90 REM Cursor Movement Routine
91 BC=0:GOSUB 61
92 IF ASC(R$)=28 THEN I=I+1:IF I>NF THEN I=1
93 IF ASC(R$)=29 THEN I=I-1:IF I<1 THEN I=NF
94 IF ASC(R$)=31 THEN IF I+MC<=NF THEN I=I+MC
95 IF ASC(R$)=30 THEN IF I-MC>=1 THEN I=I-MC
96 BC=3:GOSUB 65:GOSUB 61:RETURN
100 REM Main Program
105 KEY OFF:CLS:SHELL "DIR *.BAS >MENU.DAT":I=0:OPEN "I",1,"MENU.DAT"
110 LINE(22,17)-(600,171),7,B:LINE(32,27)-(590,161),7,B:PAINT(23,23),6,7:
  LOCATE 3,33:COLOR 3,0:PRINT "File Selection":COLOR 7,0
120 WHILE NOT EOF(1):INPUT #1,F$
125 IF MID$(F$,10,3)="BAS" THEN I=I+1:F$(I)=LEFT$(F$,8)+".BAS"
130 WEND:NF=I
210 FOR I=1 TO NF:GOSUB 65:LOCATE AR,AC:PRINT F$(I):NEXT I=1:BC=3:GOSUB 65:
  GOSUB 61:BEEP
220 LOCATE 19,15:COLOR 3,0:PRINT "Press.. <C>hange name, <R>un Program,
  <Q>uit to DOS":COLOR 7,0
230 GOSUB 70:IF ASC(R$)<32 AND ASC(R$)>27 THEN GOSUB 90:GOTO 230
240 IF R$="C" THEN GOSUB 57:GOTO 300 ELSE IF R$="R" THEN CLS:RUN F$(I) ELSE
  IF R$<>"Q" THEN GOSUB 80:GOTO 230
250 REM Quit Program
260 COLOR 7,0:LOCATE 1,1,1:CLS:SYSTEM
300 REM Change Filename Data Routine
310 BEEP:LOCATE 19,15:PRINT "Press.. <D>elete, <R>ename, <M>ain Functions"
320 GOSUB 70:IF R$="D" GOTO 600 ELSE IF R$="R" GOTO 700 ELSE IF R$="M" THEN
  GOSUB 57:BEEP:GOTO 220 ELSE GOSUB 80:GOTO 320
330 GOTO 230
600 REM Delete Filename(s) Routine
610 GOSUB 56:LOCATE 22,35:PRINT "DELETE FILES"
620 M$="Move Cursor to File to be DELETED, then press 'D'...":GOSUB 40
630 R$=INPUT$(1):R$=FNCP$(R$):GOSUB 90:IF ASC(R$)<>95 AND R$<>"D" GOTO 630
640 M$="Are you sure (Y/N)...":GOSUB 70:IF R$<>"Y" GOTO 770
650 KILL F$(I):FOR K=I TO NF-1:F$(I)=F$(K+1):NEXT:NF=NF-1
655 LINE(34,29)-(588,159),0,BF:FOR I=1 TO NF:GOSUB 65:LOCATE AR,AC:
  PRINT F$(I):NEXT
660 M$="Delete another file (Y/N)...":GOSUB 70:IF R$="Y" GOTO 620 ELSE 770
700 REM Rename Menu File(s) Routine
710 GOSUB 56:LOCATE 22,35:PRINT "RENAME FILES"
720 M$="Move Cursor to File to be RENAMED, then press 'R'...":GOSUB 40
730 R$=INPUT$(1):R$=FNCP$(R$):GOSUB 90:IF R$<>"R" GOTO 730
740 M$="Enter New Filename...":GOSUB 40:INPUT "":F$:M$="Are you sure (Y/N)...":
  GOSUB 70:IF R$<>"Y" GOTO 770
750 NAME F$(I) AS F$:F$(I)=F$:GOSUB 65:LOCATE AR,AC:PRINT SPACES(12):
  LOCATE AR,AC:PRINT F$
760 M$="Rename another file (Y/N)...":GOSUB 70:IF R$="Y" GOTO 720
770 GOSUB 59:M$="":BEEP:GOTO 220
800 REM Error Trapping
810 PRINT ERR;" ";ERL
860 IF ERL=650 AND (ERR=53 OR ERR=71) THEN M$="ERROR >> File Not Found":
  GOSUB 40:GOSUB 30:M$="":GOSUB 59:RESUME 220

```




**EXPLORE
NEW WORLDS
WITH
HUG
GAME
SOFTWARE**

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

LONG & LOUD!

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



	January	February	March	April	May
REVENUES					
Widget Sales (units)	3454	2345	3215	4322	2345
Price Each	1.28	1.28	1.28	1.28	1.28
Widget Sales (\$)	4421.12	3001.60	4115.20	5532.16	2996.40
Gadget Sales (units)	4221	3432	3434	4547	2345
Price each	2.34	2.34	2.34	2.34	2.34
Gadget Sales (\$)	9877.14	8077.68	8035.34	10686.78	5487.30
TOTAL REVENUES	14354.26	11079.28	12150.54	16218.94	7983.70

**SHOUT YOUR
MESSAGE IN
A BANNER!**

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

\$34.95

**Special Offer: one MS/DOS and
one CP/M version for only...
\$49.95**

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

PRESTO!™ Multi-Function Software Supercharger for CP/M NEW for Heath/Zenith

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

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

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

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

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

Rembrandt

Complete Business Graphics Toolkit™

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

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

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

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

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

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

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

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

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

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



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

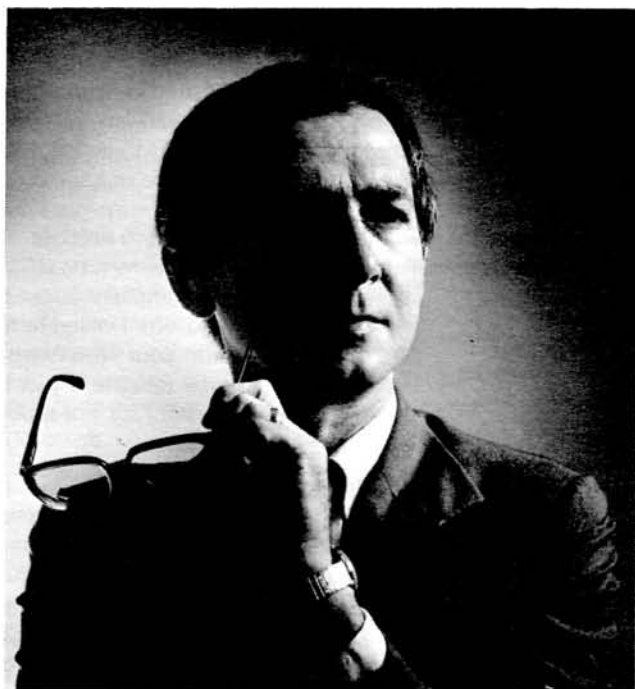
We accept VISA, MASTERCARD and AMERICAN EXPRESS

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

800-628-2828 (Extension 918)

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

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



Mainstream Computing

Joseph Katz

103 South Edisto Avenue
Columbia, SC 29205

Copyright (C) 1987 by Joseph Katz. All Rights Reserved.

I can't imagine anyone having more fun with a printer than I'm having with our Apple *LaserWriter*. I've reached the point now of giving up even the pretense of calling what I'm doing "work." It's fun.

Some of what I'm doing is exploring the insides of the *PostScript* interpreter that distinguishes the *LaserWriter* from other laser printers. *PostScript*, you'll recall from the early installments of this saga, is the PDL (Page Description Language) from Adobe Systems that resides in the *LaserWriter* ROM (Read-Only Memory) and gives it powerful graphics capabilities. I won't participate in any Laser Wars, and I certainly don't intend to start any, but you really have no idea of what current laser printers can do if all you know are printers like the Hewlett-Packard *LaserJet*. Those are good laser printers for general-purpose office work — far better for such things than the *LaserWriter* — but they simply are not *PostScript* printers. As I've said previously, the *LaserWriter*'s superior graphics capabilities make it a very good typesetting machine as well. If that's the direction you want to follow in laser printing, I really don't think you can do better than the *LaserWriter* right now or, probably, in the foreseeable future.

Nope, you who know me so well know I'm not shilling for Apple. In fact, I can't get the right time of day from them: I have found Apple downright uncooperative. The *La-*

serWriter is beautiful, though. That's why I'm spending so much time figuring things out on my own.

Because my Puritan heritage impels me to put what I know to practical use, I've been talking with HUG about publishing a small package now tentatively entitled "*The LaserWriter Connection*". Swell title, right? It's a collection of utilities and a manual. Together they're intended to get and keep the *LaserWriter* and *LaserWriter Plus* running on IBM-compatible computers. The big idea is to save you the blood, sweat, and tears I've had to go through to make the *LaserWriter* sing *Sweet Sue*. I want to make it possible for you to walk into a store, buy a *LaserWriter* for your IBM-compatible, plug the printer into the computer, and start using it. I'm full of such revolutionary ideas. If I get filthy rich in the process, I promise you I'll enjoy every penny.

In the meantime I'm trying to keep up, not too successfully I suspect, with the growing stream of software that supports *PostScript* or the *LaserWriter*: think of the two as going together. There still isn't much, but there will be soon. You know that IBM has recently announced its desktop publishing system that will use its own *PostScript* printer, Microsoft's *Windows*, and Aldus' *PageMaker PC*, don't you? (I'll be talking about *PageMaker PC* soon.) We, therefore, have what might be described as a bandwagon about to leave town. Expect that

stream of *PostScript* software to become a flood right soon. And remember you heard it here first.

WordStar Professional Release 4

It's the new *WordStar* we've been awaiting but, alas, it doesn't support *PostScript* or the *LaserWriter* . . . yet. The lack is more a nuisance than anything else for me. I have my Epson *FX-185* and the *LaserWriter* both connected to a swell new printer buffer that handles both parallel and serial devices. I have everything set up so that the output of each program I use goes to the right device for it. In addition, the new *WordStar* supports two printers. I have one of them set up so that this *WordStar* "prints" to a text file, *ASCII.WS*. One of the programs in *The LaserWriter Connection* — *PsPRINT* — will bang out text files on the *LaserWriter* all day long, so when I want speed instead of italics and boldface, I simply *PsPrint ASCII.WS* on the *LaserWriter*. But I'm getting ahead of myself.

WordStar Professional Release 4 arrived too recently for me to do more than give you a general overview, a few general impressions, and some recommendations that boil down to this: If you ever liked *WordStar*, you really ought to get the new *WordStar*. It's still *WordStar*, which appears to disappoint reviewers who talk about this new version in ways that imply they really want it to be some other program. It's not.

It's *WordStar*, which is why I'm interested in it in the first place. But it's evidently a well-evolved *WordStar*, which is why I retain my interest in it.

In fact, if you haven't used *NewWord 3* and hadn't read my description of it here a few months ago, you might think MicroPro has been working on this version of *WordStar* for some time. Maybe they have. As I said a few months ago, though, it was *NewStar* who kept the spark going in its *NewWord*. When MicroPro, under the leadership of its new president Leon Williams, finally figured out that its neglect had run a gold mine into the ground (*En garde*, John Walker!), it bought *NewStar* and *NewWord*. If you did the same on my recommendation, I have good news for you at the end. Stay tuned.

The way I heard it from MicroPro, that happened after MicroPro was well along in developing its own revision of *WordStar* and what they got from *NewWord 3* was nifty features. Who really cares except the eventual historian of word processing? This new *WordStar* is installed the same nice way *NewWord 3* was installed: You copy the working programs to your system, then run a simple installation program. If you're satisfied with the defaults that result — and you ought to be — you can start word processing. You ought to go from opening the package to opening your first document in under five minutes. If you're a whacker — my coinage to describe an old-line *WordStar* addict who is, by definition, too far gone to be called a "hacker" — you'll want to go on to another program that will allow you to screw up the defaults to your heart's content. You can change almost anything with this program, and you can make the changes so easily you can wile away the hours in pleasant fiddling, instead of doing any real work. Fortunately, this same program allows you to reinstate the defaults whenever you'd like.

Two immediately-obvious differences from *NewWord 3* are the inclusion of Writing Consultants' *Word Finder* thesaurus program and the substitution of MicroPro's *CorrectStar* spelling checker for the creaky old *The Word Plus* that *NewWord 3* used. You won't find me arguing against either of these. I've been using *Word Finder* for a long time and I think highly of it. As for *CorrectStar*, it was written by Ed Greenberg and I taught with his beautiful mother at Winthrop Junior High School in Brooklyn. I don't know any other spelling checker of one can say anything like that. Other writers may focus on relative trivia such as number of words, accuracy of their spell-

ing, and algorithms. Of course, *CorrectStar* rates high by such standards too (although I wish Ed would teach it plurals). But no other spelling checker has the one ineluctable distinction that makes *CorrectStar* unique. Leon Williams should give Ed an immediate raise. Thank goodness, though, that *WordStar* retains the *NewWord*'s absolutely marvelous interface to the spelling checker. *WordStar Release 4* now has the only spelling checker I've seen that really suits the way a professional writer works.

The one dominant impression I've formed so far about *WordStar Professional Release 4* is of its speed. This *WordStar* is not merely fast: It's FAST! It loads fast and moves fast. It is faster both ways than *NewWord 3* and much, much faster in loading than that speed demon *XyWrite III*. I don't know for sure, but I think *WordStar* now is one of the fastest word processing programs around. I can't believe I said that. I never thought I would say anything like that about *WordStar*. But so far I think I'm right.

Just so you don't think I'm making this up, I should tell you a few things that go to explaining some obvious sources of this speed. First, *WordStar* has been completely rewritten to make it smaller, tighter, and more efficient. One result is that the old WSOVLY1.OVR has gone. Now there are only WS.EXE and WSMMSG.SOV involved in editing. (The printer drivers are segregated into another overlay, WSPRINT.OVR, which is called only during printing. Intelligent.) Second, you can install this *WordStar* so it resides entirely in RAM while you edit. What you give up by doing so is the so-called "help" that explains the so-called "menus." Experienced *WordStar* or *NewWord* users won't miss them at all and will admire the speed earned by their sacrifice. Third — at least on Heath and Zenith computers — you can kill most screen delays with the WCHANGE program I mentioned earlier. The first thing I did, in fact, was whack them all down to 0: no delay whatsoever. (Remember what I said about whackers?) The only thing I haven't figured out how to do yet is kill, or at least speed up, the initial *WordStar* banner. When I do, this thing will fly. MicroPro will hate me, but I'll have the fastest word processing program on my block.

Here's some advice. MicroPro currently has an almost incredibly generous upgrade policy. If you are or ever were a *WordStar* user, I suggest you take advantage of it. You send \$89 and "proof of purchase" of a "registered" *WordStar* to this place: MicroPro, Order Update Department, P.O. Box 7978, San Rafael, CA 94901. In turn they

send you — by Airborne Express, no less — *WordStar Professional Release 4*.

Nope, you're not going to catch me on this.

To the fellow out there who plans on writing to ask if I *really* mean any version of *WordStar* — even a CP/M version 1.4 or one of those old Osborne or KayPro cripplines can be upgraded this way: Yes. I knew you would ask, so I called to find out. Yes. The other guy, who owns *NewWord*, was starting to write me a complaint about being left out in the cold. No. I called to find out. You can upgrade your *NewWord* to *WordStar Professional Release 4* just the same way: \$89 and proof it's not a bootleg copy.

What's proof? Great. I forgot to ask. But when a friend asked me that a few weeks ago, I took his *WordStar* disks to his Xerox machine and copied them. He sent those copies with his check and got his new *WordStar* package, so I guess Xerox copies of your distribution disks are proof. I don't think a note from your mother is proof, unless of course you're Ed Greenberg. Nor do I think it would be a really swift idea to try upgrading a bootleg copy of *WordStar* or *NewWord*. There probably are less costly ways to meet MicroPro's lawyers. My hunch, though, is that MicroPro would be pretty reasonable about anything pretty reasonable you provide.

You probably should take this opportunity to get on the *WordStar* upgrade path, even if you also use another word processing program. As you know, I do: I use several, each for its own special benefits. As you also know, I think a *WordStar* pretty nearly essential. I also think, as the result of what I've heard in various quarters, that *WordStar* really is moving now, and fast. It might be smart to get on the upgrade path right now — even if you're not currently a *WordStar* user.

The company operators now answer its phones by saying, "Hello. The new MicroPro." At first, I ran into a few snafus that made me wonder if I was talking to the same old MicroPro with a new way to answer the phones. The snafus seem to have been unsnafued quickly though, and there's enough evidence to suggest that we really do have a new MicroPro here. That generous upgrade policy certainly suggests so. Leon Williams certainly has to be smart enough to know that there are only a few second chances and no third chance at all. Of course, I'll stay in touch. My hunch is it's going to be a lot of fun and I'm going to see some great products.

Version 2 of *WindowDOS*

I can't imagine how I missed version 1 of *WindowDOS*.

Among the more modest of my fantasies has been a software gadget to format floppy disks on the fly, while I am using an application program. Remember the "Plan Ahead" sign with the last few letters crammed together because its designer didn't? I'm that way. To be fair about me — which is one of my major principles — I'm really good in unstructured situations, like those that involve exploring computer stuff. I, therefore, spend much time on *CompuServe* and various special-interest remote bulletin board systems from which I download software and pick up electronic mail. You can't tell what will be there when you log on. Sometimes, albeit rarely, there's nothing at all of interest. Other times there can be bunches of things, many of them big. All right: I could plan ahead by having a neat stack of blank formatted floppy disks on a shelf near my computer, but I don't. So what happens is that I download files and capture messages to my hard disks, figuring that some day I will plan ahead to a time when I will sort through the accumulated mess and arrange it on floppies. I haven't yet found time to schedule "some day." Indeed I do tend to fill hard disks that way. That's one reason why I keep adding hard disks and computers. An on-the-fly formatting program would make life easier and save me money.

It's here, among the multitude of useful features in *WindowDOS*, a TSR (Terminate and Stay Resident) disk utility. Don't yawn. I started to yawn at the thought of YADU (Yet Another Disk Utility). But *WindowDOS* is different. It is not the stuff that yawns are made of. It is, in fact, most useful. Load *WindowDOS* to make it RAM resident and it sits in the background while you run an application program in the foreground. Then, when you need what *WindowDOS* does, press Control-Insert (the CTRL and INS keys at the same time) and *WindowDOS* leaps into the foreground with a menu of its major functions.

Of course my favorite of those is the one that formats a disk from within an application program. Press "F" and a panel overlays the main menu so you can select the drive to format, either A or B. For safety's sake, *WindowDOS* won't format a hard disk drive. An equally-nice touch is that on an AT-compatible like the '241 or '248, *WindowDOS* lets you format either a standard 360KB double density disk or a 1.2MB high capacity disk in Drive A. Since *WindowDOS* works in the foreground, your

application program hibernates until you return to it by pressing the ESC key.

WindowDOS does more than format disks. The main menu is at the bottom of a more-or-less standard disk manager's directory display and gives you the following choices: Copy, Dir, Erase, Format, Global, List, Mkdir, Rename, Sort, Tree, and View. The Copy and Erase functions work on one file at a time or, if you mark files by pressing the + key while the cursor is on them, in batches. The Global function takes an entire drive as its domain, allowing you to copy, erase (the term in *WindowDOS* is "purge"), or find any group of files named according to some pattern: "JUNK.TXT," "*.C," and "K????.DOC," for example. It's a marvelous aid for uncluttering a hard disk fast.

View is the equivalent of the DOS "Type" command, letting you read the contents of an ASCII file. List is a "dump" function: it shows you a file in hexadecimal and ASCII representations. Mkdir, Sort, and Tree work on directories: The first lets you make a subdirectory, as you would with the DOS "MD" command; the second sorts *WindowDOS*'s directory list according to file name, extension, creation date, or size; and the third displays the directory tree (with all subdirectories branching out from their roots) of an entire drive, so you can figure out where you are and what you want to do.

Incredibly, those functions offer the merest taste of *WindowDOS*. Press the F10 key and you get a list of twelve more functions, including some I consider almost priceless. Press CTRL-E, for example, and you see the DOS environment with such things as the list of device drivers and TSRs, including the size of each and the vectors they use. (You'll also see some miscellaneous information about the machine, including its ROM date. Because Heath/Zenith computers and, presumably, other compatibles have some of this information in non-standard places, you'll see question marks instead of the ROM date. It's a trivial matter in this kind of program on these kinds of machines.) Press F6 to hide or unhide a file or subdirectory. That feature is useful when you need minimal security on your computer. Press F6 to protect or unprotect a file against erasure or writing. That feature is useful when you have something precious you don't want hit by an accident. Press F8 to set up an Epson/IBM-compatible parallel printer for such things as bold-face or condensed mode — the kinds of things for which you normally need a separate setup utility. (*WindowDOS* is intel-

ligent. The default printer port is LPT1, but you can switch among LPT1-LPT3 if you have the ports installed. *WindowDOS* checks.) Press F9 to see the DOS time and date, which you can change. Then — and this feature I think I love — you can press CTRL-C on AT-compatible computers like the H-241 and H-248 and set the system clock to the DOS time and date. That way you don't have to reboot the computer and use the CMOS setup program whenever the system clock needs resetting.

Here's a *WindowDOS* feature I really like. I always keep the door of my university office open when I have office hours. It's an invitation to students and colleagues to drop in. Of course, they often drop in while I'm working on something private or personal at my computer there. Then it's been a nuisance to close the document and clear the screen. With *WindowDOS* now I press CTRL-INS to activate the program and then press F7. That function password-protects the computer, substituting a pattern and request for the password in place of the document I was doing. When the visitor leaves, I enter the password and resume working. Oh sure, it's not ultimate security, but it's a heck of a lot more dignified than throwing my body across the screen.

WindowDOS is like a Swiss army knife for IBM-compatible computers: It has most functions you might want, and all you should conceivably need, on your daily forays through the digital terrain. Like the Swiss army knife, *WindowDOS* packs those functions into a remarkably small space: around 50KB of disk space and about the same amount of RAM when resident. You can run it in non-resident mode too, as a conventional transient program like standard disk managers, but I don't know many occasions on which you would want to. *WindowDOS* doesn't use much RAM in its TSR mode, it hasn't had a conflict with anything but *XyWrite III* (which doesn't like TSRs) so far, it has an EGA mode, and it can be removed from RAM with one of its command-line switches. (Be careful, though, to make sure you haven't loaded another TSR after *WindowDOS* or else you'll leave a hole in RAM when you unload *WindowDOS*.) At \$49.95 *WindowDOS* is priced most fairly. I've been using it for a month now, and it's a charmer. I can't imagine why I hadn't come across version 1: I'd have been using *WindowDOS* long before now.

A Handy Household Hint

While I was exploring those bulletin boards I came across EMOFF.COM and EMON

.COM. They're intended to speed up a system that has an EMM (Expanded Memory Manager) board when you're running version 1 of Lotus' 1-2-3, which doesn't take advantage of that memory.

I don't use 1-2-3 much: Microsoft's *Multiplan* suits my needs for a spreadsheet, is faster, and integrates nicely with other Microsoft software I like right now. But I figured that the system should run faster if I turned off access to my BOCARAM EMM

board when it's not being used, so I downloaded the two public domain programs.

Sure enough, things go faster when the BOCARAM is not being used and I run EMOFF. When I want to use the board, I run EMON to turn on access to it again. I've uploaded the two programs to *CompuServe's* HUG SIG (GO PCS48). If you have any kind of EMM board, you might want to download these programs and see if they do anything to make your life better.

Products Mentioned

WindowDOS Version 2	\$49.95
Upgrade from Version 1	\$25.00
WindowDOS Associates Box 300488 Arlington, TX 76010 (817) 467-4103	
WordStar Professional Release 4 Upgrade from various vers.	\$89.95
MicroPro International Corporation 33 San Pablo Avenue San Rafael, CA 94903 (800) 227-5609	



Bold, Stylish Text for H-19 and H-89!

T-Prom provides an entirely new look for your CRT display. All of the text characters are enhanced for greater style, clarity and boldness. Now you have a choice between "stick" letters and clear, stylish screen text.

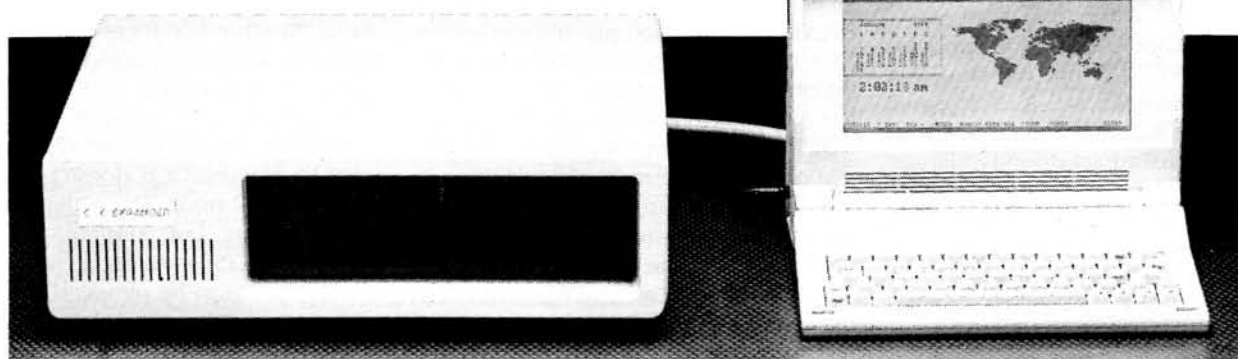
T-Prom is a plug-in replacement for the original character generator ROM, and is priced at only \$19.95 including shipping, documentation, and installation instructions. (Another new product, **GT-Prom** offers the combined benefits of T-Prom and G-Prom for only \$24.95. Write for information on G-Prom.)

Send your order to: **NORCOM**

**9630 Hayes
Overland Park, KS 66212**



EXPAND YOUR PORTABLE COMPUTER Enjoy Laptop Portability PLUS Desktop Power With Your Z-171



THE CTC EXPANDER™ lets you use hard disks, LANs, expanded memory, and most other IBM-compatible add-on cards with your laptop computer. Now you can have portability AND power, with no compromises.

Flexible design fits virtually any combination of add-ons*

- 4 full-size and 3 half-size slots
- 135-watt UL-listed power supply
- Mounts up to 4 half-height disk or tape drives

Designed for high-speed processing

- Compatible with 0 wait-state processors

Reliable, trouble-free construction

- Easily connected and disconnected for unimpaired portability
- FCC Class B approved for both home and business use
- Made in the U.S.A.
- 90-day factory warranty

Dealer Inquiries Welcome

**CYPHER
TECHNOLOGY
CORPORATION**

14003 Ventura Boulevard Sherman Oaks, CA 91423
(818) 905-8161 FAX: (818) 905-0211

AVAILABLE NOW FROM YOUR HEATH/ZENITH DEALER

* Call for list of compatible add-ons.

No Sheet Simple Spreadsheet With Comments On Style

Theodore J. Stolarz

200 Yost Avenue
Park Ridge, IL 60068

Background

I think I got one of the first H-8 kits sold by the Heath St. Joseph Customer Center in 1977. With the additions and improvements included as they became available, the system is still working well. A few months back, after seeing the ZF-158-24 at HUG-CON85, I replaced my H-8 with the new machine. I also got the ZVM-135 color monitor. Since I never used CP/M with the H-8, it was still a 56K machine with HDOS. Needless to say, the difference in performance is hard to describe.

I am a very active programmer and I have been reading language specification manuals since the days of UNIVAC I and the IBM 650. The GW-BASIC Ver. 2 is the most powerful programming language (in terms of its capabilities) that I have ever seen. With the companion compiler I can do everything I want to do. I will, of course, still use FORTRAN for certain computationally intensive programs, especially when I get the arithmetic chip installed.

No "file transfer" program I found seemed to be able to allow me to do a direct transfer from my single-sided hard-sectored disks to the new disks. I, therefore, decided to take this opportunity to "clean up" my programs. This involves "keying in" some eight years of software development. After some 220 hours at the keyboard, I am seeing some "light at the end of the tunnel." Most of the programs I have transferred are my own BASIC and FORTRAN routines. Some are programs written by others which I had adapted to the old system. This experience led me to make some observations about programming style which I will discuss and illustrate with one of my programs. But first a comment.

As new advances in hardware technology come along, we will have to rewrite our programs to take advantage of these advances. It is not sufficient to take a program that "worked" on an H-8 and change it enough for it to "work" on an H/Z-150. Some of the commercial programs I have bought for my new system seem to use this approach. The temptation is to not care what the

source code of a program is like since we do not see it once we start using the program. The original code IS the program. If it is written so that it can be understood and easily modified, the program will not only "fit" on some future machine, but hopefully, will take full advantage of its improved structure. Remember, the H-8 was the "state of the art" for us only seven years ago. Perhaps the less said about the H-9 and H-10 the better. What will we be using in 1993? I hope it will not be the exact same programs we used for the H-8.

Programming Style

While still teaching at the university [I am a retired Prof], my office was on the same floor as our Department of Computer Science. As a frequent computer user and statistical consultant on many research grants, I spent much time with the computer science faculty. I have to admit that I heard the cliches "top-down" and "structured programming" until I became ill. One young faculty member said I must never use a GOTO statement. To my knowledge he had never written a useful program. I even bought a book on PASCAL. Then I went home and continued coding statistical analysis programs. I came to the conclusion that much of this was a lot of [expletive deleted]! But it is time some of us got rid of habits that have outlived their usefulness.

In my H-8, by the time I loaded MBASIC, I had 25K of memory left for programs and also the variable "heap" [arrays, etc.]. So I packed coding like we all did. The last program I wrote for it in decent style took 911 lines of coding. With a 60K work space and the ability to link programs with the compiler, the need to pack is over.

I now follow some conventions and I propose that, without making a fetish out of "structured programming", we adopt some common sense practices. It is true that most program transfers from system to system will be by disk exchange or by telephone lines. But we often have to modify programs and some programs will still be "keyed in". The comments I make here are the result of modify-

ing over a hundred programs from B.H. BASIC and MBASIC to my current GW-BASIC Ver. 2. For instance:

1. For the most, part the old adage of "one statement per line" makes sense. I would not want to carry this too far. For instance, to advance printing a few lines, the statement: 50 PRINT : PRINT : PRINT is by no means confusing.
2. Comments should usually appear on separate lines with the REM coding used. The apostrophe does not separate the comments well and at the end of lines it is confusing. Also, comments at the end of executable statements do not "open up" the coding and should be used with care.
3. All GOTO and GOSUB statements should carry REM comments either before their use or on the same line to tell where the program logic is leading. For instance:

```
1130 GOTO 750 : REM BACK TO MENU,  
or  
1240 GOSUB 6520 : REM PRINT MATRIX
```

are clear to the reader.

4. In subroutines, the entrance point should be the first executable statement NOT a comment statement. Someone may wish to eliminate the REM statement. Some programs I have transferred say that the REM statements can be eliminated. If you try it you are in for a lot of work. The subroutines should be identified as to what they do, however.
5. Most important of all, all FOR-TO loops should be indented to show the level of nesting and should be written one statement per line. This is to remind us that these statements will be executed more than once or, in some cases, not at all. Let me explain why this is so important.

One of the frustrating features of FOR-TO loops is that different versions of BASIC treat them differently. This may not be apparent unless you do a lot of programming. If, for example, the initial conditions of the loop are met before it is entered, many versions of BASIC will execute the loop once. In these versions, all FOR-TO loops are executed at least once. In most compiled BASICs and in GW-BASIC Ver. 2 the loop is NOT executed. The latter is, by far, preferred. This can cause problems that are hard to debug, especially if you have to search for the loop and try to visualize its level of "nesting." Try this on your version of BASIC:

```
10 FOR I=3 TO 1  
20 PRINT "THE LOOP HAS BEEN EXECUTED"  
30 NEXT I  
40 END
```

By logic, nothing should print. If it is coded in a form like:

```
10 FOR I=J3 TO N4 STEP X0:PRINT "X":NEXT:END
```

and "hidden away", you get one of those programs where you say, "I never got it to run on my system." The programmer may have "taken advantage" of the fact that it will execute at least once. This can be very difficult to find.

Also, "nested" loops can be difficult to visualize unless they are indented properly. Lines 510-560 in NSSS clearly read two matrices by rows.

6. Make frequent use of subroutines and be sure they are at the end of your program. If the subroutine calls are well explained, it is possible to read through the program without having to skip around.

"Human Engineering" The Program

One old habit that dies hard is the notion of programs having to rely heavily on separate documentation for their use. It is, of course, true that an undocumented program is quite useless. In my transfer of files I ran into a few programs that originally appeared in magazine articles. I have lost the copies of the original articles; I had to discard the programs. It is true that years ago many of us had drawers full of card decks and files of separate program listings. In order for us to prepare program input on cards, the printed documentation was essential.

With today's fast terminals with rapid screen displays, a program can present diagrams, menus, instructions, etc. to the user very rapidly. Some of the displays can be made optional. Unless a program produces results that need some explanation, the program should lead the user through its steps without him or her needing to learn "codes" and/or "conventions." Most of the statistics programs I have written have substantial external documentation, but this is because all input data looks alike in this case. It is critical that the user select the right program. Once this choice is made, the programs are very easy to use.

In a properly "human engineered" program it is possible for a person to learn to use the program without external documentation of any kind. Note that I said POSSIBLE, not "easy." External documentation can facilitate learning more rapidly. I do not think most word processing or spreadsheet programs meet this criterion.

Now I am, of course, aware that if a program is so structured, it is easily "pirated." Having to rely on lengthy manuals which are difficult to duplicate helps sell software and also minimizes the work to adapt programs to different systems. No software company can stay in existence if everyone duplicates his buddy's disks. This has nothing to do with improving computer software, however. I am talking from an "ideal" standpoint. I warned you that I am an academic "egghead".

I will illustrate my point using as an example some "Spreadsheet" programs. I wanted a program to keep track of income and expenditures so I could prepare taxes at year's end and budget for expensive computer equipment. I found these programs represented "over kill." Also, the concept of a large sheet of paper used to "spread" expenses, etc. is foreign to me. I did not know what a spreadsheet was. I tried to explain it to a friend once. He said he kept track of his expenses and income like I did - in a notebook. One page is used for each category and a page for totals at the end. If you want to add a category, you insert another page and update the total list. This is logical and makes sense. It will not do for a professional accountant. I found I needed to keep notes when using these programs.

This brings me to one important point. The computer is limited, to some extent, by its structure. It has some unique features. To the extent possible, we should try to make it adjust to the way we think. To use it effectively, however, we must adjust to it to some extent. Unless you are a businessperson very familiar with categories of expenditures, etc., the idea of using a computer screen to look at a large hypothetical sheet of paper like a "window" is not appealing to most people.

It is not surprising at all to see computer companies trying to sell their products by telling people that they will not have to change their thinking. After all, before the automobile was perfected, several attempts were made to construct mechanical horses. It took years before people came to accept cars that did not look like

"horseless carriages". Implementing an accountant's spreadsheet on a computer is building a mechanical horse.

The "No Sheet Simple Spreadsheet"

In writing this program, I attempted to adopt an approach that was logical and as simple as possible. The goal was to construct a way for people to keep track of money matters such as expenses, income, budgets, etc. in a way that makes it unnecessary to memorize any codes or symbols. The hope was that it would be so easy people would not hesitate to turn on their computer to enter a small amount.

This program is limited, but can be expanded. The approach is what I am trying to illustrate. It was originally written for the H-8 with MBASIC. In its present form it is for the H/Z-150 computers using MS-DOS and GW-BASIC Ver. 2. It will work on the compiler. Since I paid for a color monitor, there is lots of color used. A lot of this can be eliminated if need be.

Here is the program description; the actual operating instructions are included in the program:

- NSSS allows the user to create 15 categories for use in keeping track of expenses or income. It uses disk files so many different areas can be maintained.
- Select an area such as "Household Expenses" and a file name such as HHEX86.DAT. Decide on some categories (15 maximum) such as "Taxes," etc.
- You can enter up to 50 entries in each category with a short description of each entry. All totals are updated automatically.
- Printed output is available on request. All output is automatic and can be punched to fit into a notebook.

Nothing more need be said. The program does the rest. Some people who have used this program found it very easy and uncomplicated. This approach can be expanded and the program extended. If you key it into your computer, you will find the coding easy to follow.

Some Final Notes

The concept of the stored program computer is quite new from an historical perspective. The technical hardware development is far ahead of our software approaches. It makes no sense, for instance, for a computer user with over a half million bytes of memory to be limited to a 60K workspace in BASIC. Once a programmer writes programs of more than a dozen pages in length, 50 or 60 pages pose no problem if he uses decent style and can mentally handle them. The cost of computer memory was once very high; it is now trivial.

We should take a new look at the programs we use everyday such as our word processing programs from a "human engineering" standpoint. I should be able to tell my word processing program in simple English how I want my pages to look and it should remember that until I change it. Right now I have a program that keeps consulting disk overlays while I edit text. I have a huge memory; load them into it. Part of this, I suppose, is economics. One project I plan down the line is to write my own program; that is, when I get a compiler or interpreter that will use all available memory in one program.

Some decades back, when I served as a college dean, I supported transfer of our transcript files to a computer format. Some people wanted to use the computer printouts as records, making

changes on them in pen and ink. Others suggested that we print up a set of transcripts to keep in filing cabinets. We burned the original records and had to retire a few people. Old habits die hard. We want to insure that those of us who think we are at the leading edge of computer thought do not become relics before our time.

Listing

```
10 REM Program NSSS - For MS-DOS -
20 REM
30 REM No Sheet Simple Spreadsheet - By Theodore J. Stolarz
40 REM Version 1.1 - 27-Nov-85
50 REM (C) 1984 T. Stolarz
60 REM
70 REM For ZF-158-24 [IBM PC] with MS-DOS (GW-BASIC)
80 REM INITIALIZE VARIABLES, ETC.
90 ND=15 : YM=50
100 DEFDBL G: DEFDBL T
110 DIM A(15,50),B(15),A$(15,50),M$(15),T(15)
120 D$="###.###.###" : GT$="#.###.###.###"
130 REM
140 REM IDENTIFY AND CHECK FOR FILE
150 CLS : KEY OFF
160 COLOR 10,0
170 PRINT TAB(10);"NO SHEET SIMPLE SPREADSHEET"
180 PRINT TAB(10);"Ver. 1.1 (C)1984 T. Stolarz"
190 PRINT
200 PRINT TAB(5);"Is there a previously created file"
210 COLOR 26,0
220 PRINT TAB(5);"[1=Yes;0=No] ";
230 INPUT YN
240 COLOR 10,0
250 PRINT TAB(5);"Enter Year ";
260 INPUT Y$
270 REM CHECK FOR VALID YEAR ENTRY
280 IF LEN(Y$)=2 THEN Y$="19"+Y$
290 IF LEN(Y$)<>4 THEN 250
300 REM ENTER DATE
310 PRINT TAB(5);"Enter Month - Jan, Feb, Mar, Apr, May, Jun, Jul, ";
320 PRINT "Aug, Sep, Oct, Nov, or Dec"
330 INPUT " " : MO$
340 PRINT TAB(5);"Enter Day of Month [2 digits] ";
350 INPUT DA$
360 REM CREATE DATE FOR PRINTER LISTING
370 DT$=DA$+"-"+MO$+"-"+RIGHT$(Y$,2)
380 IF YN=0 THEN 610
390 REM
400 REM READ STORED FILE
410 PRINT
    "Enter file name in MS-DOS format - e.g. B:HHEX83.DAT "
420 INPUT F$
430 OPEN "I",#1,F$
440 INPUT#1,T$
450 INPUT#1,NCAT
460   FOR I=1 TO NCAT
470     INPUT#1, B(I)
480     INPUT#1, M$(I)
490   NEXT I
500 REM
510   FOR I=1 TO NCAT
520     FOR J=1 TO B(I)
530       INPUT#1, A(I,J)
540       INPUT#1, A$(I,J)
550     NEXT J
560   NEXT I
570 CLOSE #1
580 GOTO 630 : REM PRINT MENU
590 REM
600 REM OBTAIN NEW FILE PARAMETERS
610 CLS : GOSUB 3960
620 REM PRINT MENU AND OBTAIN CHOICE
630 GOSUB 3110
640 REM
650 REM BRANCH TO CHOICE
660 IF S=0 THEN GOSUB 4190
670 COLOR 14,0
```

```

680 ON S GOTO 720, 1020, 1460, 2090, 2770
690 REM
700 REM -- 1 -- ADD TO CURRENT FIGURES
710 REM DISPLAY CATEGORIES
720 GOSUB 3580
730 REM SET FLAG TO WRITE NEW FILE AT END
740 FG=1
750 REM PRINT TO 25TH LINE
760 X$=M$(C)
770 GOSUB 3300 : COLOR 14,0
780 REM DISPLAY STORED FIGURES
790 GOSUB 3800
800 PRINT
810 IF B(C)<50 THEN 850
820 PRINT "CATEGORY IS FULL - Create another if possible."
830 PRINT "ENTER 'C' and [CR] to continue - "; : INPUT C$
840 GOTO 630 : REM PRINT MENU
850 PRINT TAB(5),
      "ENTER New Amount [0 if no entry] and press RETURN ";
860 B(C)=B(C)+1
870 BB=B(C)
880 INPUT A(C,BB)
890 IF A(C,BB)=0 THEN B(C)=B(C)-1
900 IF A(C,BB)=0 THEN 630 : REM RETURN TO MENU
910 PRINT "ENTER DESCRIPTION [14 Spaces Max] ";
920 INPUT A$(C,BB)
930 IF LEN(A$(C,BB))<15 THEN 950
940 PRINT "More than 14 spaces - Please abbreviate -" :
      GOTO 910
950 PRINT "Another Entry [1=Yes;0=No] ";
960 INPUT E
970 REM IF NO GO BACK TO MENU - IF YES GET VALUE
980 IF E=0 THEN 630 ELSE 760
990 REM
1000 REM -- 2 -- CORRECT ERROR IN FIGURES
1010 REM DISPLAY CATEGORIES
1020 GOSUB 3580
1030 REM SET FLAG TO WRITE NEW FILE AT END
1040 FG=1
1050 REM PRINT TO 25TH LINE
1060 X$=M$(C)
1070 GOSUB 3300 : COLOR 12,0
1080 PRINT TAB(5);"CORRECTION ROUTINE"
1090 PRINT
1100 PRINT "Is category correct [1=Yes;0=No] ";
1110 INPUT C2
1120 REM IF CORRECT GO FORWARD IF NOT GO BACK
1130 IF C2=0 THEN 630 : REM RETURN TO MENU
1140 REM DISPLAY STORED FIGURES
1150 GOSUB 3800
1160 PRINT
1170 NP=0
1180 PRINT "NOTE: TO DELETE A VALUE REPLACE WITH ZERO"
1190 PRINT "      TO CHANGE A DESCRIPTION REPLACE VALUE"
1200 PRINT
1210 PRINT
      "ENTER Line Number of Incorrect Amount and Press RETURN ";
1220 INPUT ER
1230 PRINT
1240 PRINT "Present Entry is ";
1250 PRINT USING D$; A(C,ER);
1260 PRINT "      "; A$(C,ER)
1270 PRINT "ENTER correct figure ";
1280 INPUT A(C,ER)
1290 IF A(C,ER)=0 THEN 1350
1300 PRINT "ENTER correct description [14 Spaces Max]"
1310 INPUT A$(C,ER)
1320 IF LEN(A$(C,ER))<15 THEN 1350
1330 PRINT "More than 14 spaces - Please abbreviate":
      GOTO 1300
1340 REM IF DELETE, MOVE UP VALUES IN ARRAY
1350 IF A(C,ER)<>0 THEN 630 : REM RETURN TO MENU
1360 IF ER=50 THEN 630
1370 FOR K=ER TO B(C)
1380 A(C,K)=A(C,K+1) : A$(C,K)=A$(C,K+1)
1390 NEXT K
1400 B(C)=B(C)-1
1410 REM GO BACK TO MENU

1420 GOTO 630
1430 REM
1440 REM -- 3 -- DISPLAY CURRENT VALUES
1450 REM GET TOTALS
1460 GOSUB 3420
1470 CLS
1480 REM PRINT 25TH LINE
1490 X$=T$
1500 GOSUB 3300 : COLOR 14,0
1510 PRINT TAB(5);"SELECT ACTION OR DISPLAY"
1520 PRINT
1530 PRINT TAB(10);"1 - Display totals"
1540 PRINT TAB(10);"2 - Display Category"
1550 PRINT TAB(10);"3 - Return to FUNCTIONS AVAILABLE"
1560 PRINT
1570 COLOR 20,0
1580 PRINT TAB(5);"Choice [Number] ";
1590 INPUT C
1600 COLOR 12,0
1610 REM SELECT **1** OR **2**
1620 IF C=1 THEN 1680
1630 IF C=2 THEN 1880
1640 REM RETURN TO MENU IF 3 OR OTHER
1650 GOTO 630
1660 REM
1670 REM **1** DISPLAY TOTALS
1680 CLS
1690 REM PRINT TO 25TH LINE
1700 X$="TOTALS"
1710 GOSUB 3300 : COLOR 14,0
1720 PRINT TAB(5);"Category Totals"
1730 PRINT
1740 FOR I=1 TO NCAT
1750 PRINT M$(I);TAB(35);
1760 PRINT USING GT$; T(I)
1770 NEXT I
1780 PRINT
1790 PRINT "Grand Total is $ ";
1800 PRINT USING GT$;GT
1810 PRINT
1820 PRINT "Type 'C' and press RETURN to continue ";
1830 INPUT C$
1840 REM RETURN TO "SELECT ACTION"
1850 GOTO 1470
1860 REM
1870 REM **2** DISPLAY CATEGORY
1880 CLS
1890 REM PRINT CATEGORIES
1900 GOSUB 3580
1910 IF C=0 THEN 630 : REM RETURN TO MENU
1920 CLS
1930 REM PRINT TO 25TH LINE
1940 X$=M$(C)
1950 GOSUB 3300 : COLOR 14,0
1960 REM DISPLAY STORED FIGURES
1970 GOSUB 3800
1980 PRINT
1990 PRINT TAB(10);M$(C);" TOTAL - ";
2000 PRINT USING GT$; T(C)
2010 PRINT
2020 PRINT TAB(5);"ENTER '1' to display another category"
2030 PRINT TAB(5);"ENTER '0' to return to 'DISPLAY VALUES'"
2040 INPUT C
2050 REM ANOTHER CATEGORY OR "SELECT ACTION"
2060 IF C=1 THEN 1880 ELSE 1460
2070 REM
2080 REM -- 4 -- PRINT DATA ON LINE PRINTER
2090 NP=0
2100 CLS : PRINT : PRINT
2110 PRINT TAB(20);"***** PRINTING - PLEASE WAIT *****"
2120 REM OPEN OUTPUT FILE
2130 REM CALCULATE TOTALS
2140 GOSUB 3420
2150 REM SEE IF OUTPUT NEEDS EXTRA PAGE
2160 MAX=B(1)
2170 FOR I=1 TO NCAT
2180 IF B(I)>MAX THEN MAX=B(I)
2190 NEXT I

```

```

2200 REM PRINT 3 OR 6 CATEGORIES PER PAGE
2210 L2=1
2220 LPRINT "System ZF-158 Date: ";DATE$; Time: ";
    TIMES : LPRINT
2230 NP=NP+1
2240 LPRINT TAB(10);T$;TAB(55);"Page ";NP
2250 LPRINT
2260 LPRINT
2270 LPRINT TAB(22);"Categories as of ";DT$
2280 LPRINT
2290 LPRINT TAB(5);M$(L2);TAB(30);M$(L2+1);TAB(55);M$(L2+2)
2300 LPRINT
2310 TB=5
2320   FOR I=1 TO MAX
2330     FOR J=L2 TO L2+2
2340       IF J>NCAT THEN 2410
2350       SK=SK+A(J,I)
2360       LPRINT TAB(TB);
2370       TB=TB+25
2380       IF A(J,I)=0 THEN LPRINT " ";
2390       IF A(J,I)<>0 THEN LPRINT USING D$; A(J,I);
2400       IF A(J,I)<>0 THEN LPRINT " "; A$(J,I);
2410     NEXT J
2420   REM STOP IF ALL ZERO
2430   IF SK=0 THEN 2470 ELSE SK=0
2440   LPRINT
2450   TB=5
2460   NEXT I
2470 L2=L2+3
2480 IF L2>NCAT THEN 2560
2490 IF L2=7 OR L2=13 OR L2=19 THEN 2520
2500 IF MAX<25 THEN 2280
2510 REM START NEW PAGE
2520 LPRINT CHR$(12)
2530 GOTO 2230
2540 REM
2550 REM START NEW PAGE
2560 LPRINT CHR$(12)
2570 NP=NP+1
2580 LPRINT TAB(10);T$;TAB(55);"Page ";NP
2590 LPRINT
2600 LPRINT
2610 LPRINT TAB(22);"TOTALS as of ";DT$
2620 LPRINT
2630   FOR I=1 TO NCAT
2640     LPRINT TAB(6);M$(I);TAB(35);
2650     LPRINT USING GT$;T(I)
2660   NEXT I
2670 LPRINT
2680 LPRINT TAB(10);"GRAND TOTAL";TAB(33);"$ ";
2690 LPRINT USING GT$;GT
2700 LPRINT CHR$(12)
2710 GOTO 630 : REM RETURN TO MENU
2720 REM
2730 REM -- 5 -- RETURN TO MSDOS
2740 REM PUT FILE TO DISK, CLEAR SCREEN, RETURN TO BASICA
2750 REM WRITE FILE TO DISK
2760 REM DO NOT WRITE FILE TO DISK IF NO CHANGES
2770 IF FG=0 THEN 3030
2780 CLS
2790 COLOR 12,0
2800 PRINT TAB(5);"FILE NAME is ";F$ : PRINT
2810 COLOR 28,0
2820 PRINT TAB(5);"Is FILE NAME correct [1=Yes;0=No] ";
2830 INPUT F
2840 COLOR 12,0
2850 IF F=1 THEN 2880
2860 PRINT : PRINT TAB(5);
    "ENTER desired FILE NAME [e.g. B:FNAME.DAT]";
2870 INPUT F$
2880 OPEN "0",#1,F$
2890 PRINT#1,T$
2900 PRINT#1,NCAT
2910   FOR I=1 TO NCAT
2920     PRINT#1,B(I)
2930     PRINT#1,M$(I)
2940   NEXT I
2950 REM

```

```

2960   FOR I=1 TO NCAT
2970     FOR J=1 TO B(I)
2980       PRINT#1,A(I,J)
2990       PRINT#1,A$(I,J)
3000     NEXT J
3010   NEXT I
3020 CLOSE #1
3030 COLOR 7,0 : CLS : KEY ON
3040 PRINT : PRINT : PRINT
3050 END
3060 REM
3070 REM      *** SUBROUTINES ***
3080 REM
3090 REM PRINT MENU
3100 REM PRINT TO 25TH LINE
3110 X$=T$
3120 GOSUB 3300
3130 COLOR 9,0
3140 PRINT : PRINT TAB(5);"FUNCTIONS AVAILABLE"
3150 PRINT
3160 PRINT TAB(10);"1 - Add to current figures"
3170 PRINT TAB(10);"2 - Correct of delete figures"
3180 PRINT TAB(10);"3 - Display current values on screen"
3190 PRINT TAB(10);"4 - Print out data on Line Printer"
3200 PRINT TAB(10);"5 - Exit to MS-DOS or BASICA"
3210 PRINT TAB(10);"0 - Increase number or change categories"
3220 PRINT
3230 COLOR 27,0
3240 PRINT TAB(5);"INPUT Selection [Number] ";
3250 INPUT S
3260 COLOR 11,0
3270 IF S<6 THEN RETURN ELSE GOTO 3130
3280 REM
3290 REM Create 25th Line - Set color
3300 CLS
3310 COLOR 12,1
3320 X$=" "+X$+" "
3330 WI=LEN(X$)
3340 AL=(80-WI)/2
3350 LOCATE 25,AL,1
3360 PRINT X$;
3370 COLOR 7,0
3380 LOCATE 1,1,1
3390 RETURN
3400 REM
3410 REM - CALCULATE TOTALS
3420   FOR I=1 TO NCAT
3430     T(I)=0
3440   NEXT I
3450 GT=0
3460   FOR I=1 TO NCAT
3470     FOR J=1 TO B(I)
3480       T(I)=T(I)+A(I,J)
3490     NEXT J
3500   NEXT I
3510 REM
3520   FOR I=1 TO NCAT
3530     GT=GT+T(I)
3540   NEXT I
3550 RETURN
3560 REM
3570 REM PRINT CATEGORIES
3580 X$="Category List"
3590 GOSUB 3300 : REM PRINT IN 25TH LINE
3600 COLOR 10,0
3610 PRINT : PRINT TAB(5);
    "SELECT FROM THE FOLLOWING CATEGORIES"
3620 PRINT
3630   FOR I=1 TO NCAT
3640     PRINT TAB(10);I;"- ";M$(I)
3650   NEXT I
3660 PRINT
3670 IF NCAT>9 THEN TB=12 ELSE TB=11
3680 PRINT TAB(TB);"0 - Return to 'FUNCTIONS AVAILABLE'"
3690 PRINT
3700 COLOR 26,0
3710 PRINT TAB(5);"Choice ";
3720 INPUT C

```



```

3730 COLOR 10,0
3740 REM RETURN TO MENU
3750 IF C=0 THEN GOTO 630
3760 REM CHECK FOR VALID ENTRY
3770 IF C>NCAT OR C<0 THEN GOTO 3630 ELSE RETURN
3780 REM
3790 REM -- DISPLAY STORED FIGURES
3800 IC=0
3810 IF B(C)=0 THEN 3930
3820 FOR I=1 TO B(C)
3830 IC=IC+1
3840 PRINT I;TAB(5);
3850 PRINT USING D$; A(C,I);
3860 PRINT TAB(30);A$(C,I)
3870 REM HALT SCREEN OUTPUT AT 15
3880 IF IC<15 THEN 3920
3890 PRINT "ENTER 'C' and [CR] to continue ";
3900 IC=0
3910 INPUT C$
3920 NEXT I
3930 RETURN
3940 REM
3950 REM -- INITIALIZE VARIABLES --
3960 PRINT
3970 COLOR 14,0
3980 FG=1
3990 PRINT
" Enter title for the new record - 30 spaces max. -"
4000 INPUT T$
4010 IF LEN(T$)<31 THEN 4030
4020 PRINT
"More than 30 spaces - Please abbreviate " : GOTO 3990
4030 PRINT
" Enter file name in MS-DOS format - e.g. B:HHX83.DAT -"
4040 INPUT F$ : PRINT
4050 PRINT
" Enter the number of categories [";ND;"Max] -":
INPUT NCAT
4060 IF NCAT<ND+1 THEN 4080
4070 PRINT "More than";ND;"categories - re-enter " :
GOTO 4050
4080 PRINT " Enter CATEGORY TITLES [20 spaces max.] -"
4090 FOR I=1 TO NCAT
4100 PRINT "Cat. No. ";I; : INPUT M$(I)
4110 IF LEN(M$(I))<21 THEN 4130
4120 PRINT "More than 20 spaces - Please abbreviate -" :
GOTO 4100
4130 NEXT I
4140 RETURN
4150 PRINT TAB(5);
" '0' to add to number of categories." : PRINT
4160 REM
4170 REM ** 0 ** ADD TO OR CHANGE CATEGORIES
4180 REM PRINT TO 25TH LINE
4190 X$="Add to or Change Categories"
4200 GOSUB 3300 : COLOR 10,0
4210 FG=1
4220 PRINT TAB(5);"ENTER '1' to change a category title;"
4230 PRINT TAB(5);
" '0' to add to number of categories." : PRINT
4240 INPUT CA
4250 IF CA=1 THEN 4400
4260 IF NCAT=ND THEN 4330
4270 PRINT "Enter new CATEGORY TITLE [20 Spaces Max.] -"
4280 INPUT CA$
4290 IF LEN(CA$)<21 THEN 4310
4300 PRINT "More than 20 spaces - Please abbreviate -" :
GOTO 4260
4310 NCAT=NCAT+1
4320 IF NCAT <ND+1 THEN 4370
4330 PRINT "More than";ND;
"categories. Enter 'C' and [CR] to continue";
4340 INPUT C$
4350 REM RETURN TO MENU
4360 GOTO 630
4370 M$(NCAT)=CA$
4380 GOTO 630 : REM RETURN TO MENU
4390 REM PRINT TO 25TH LINE

```

```

4400 X$="Change category TITLE" : GOSUB 3300 : COLOR 10,0
4410 PRINT "CURRENT CATEGORIES ARE:" : PRINT
4420 FOR I=1 TO NCAT
4430 PRINT TAB(10);I;"-";M$(I)
4440 NEXT I
4450 PRINT
4460 PRINT "ENTER number of category to be changed -";
4470 INPUT I
4480 REM CLEAR SCREEN WITH SUBROUTINE 3250
4490 GOSUB 3300 : COLOR 10,0
4500 PRINT "Present title is -";M$(I) : PRINT
4510 PRINT "ENTER new title - " : INPUT CA$
4520 IF LEN(CA$)<21 THEN 4540
4530 PRINT "More than 20 spaces - Please abbreviate - " :
GOTO 4500
4540 M$(I)= CA$
4550 GOTO 630 : REM RETURN TO MENU
4560 REM
4570 REM **** END LISTING OF NSSS ****

```

*

MOVER - Moves file entry(s) from one directory to another. Sub dir to root, Root to sub dir, sub dir to sub dir. Uses wildcard's (*,?).

DIRT - Directory Totals, Similar to Dos DIR but lists (all) files plus actual file sizes in both bytes and clusters, all attribute flags and actual file size totals.

Requires MSDOS Ver 2 or 3 - Runs on H/Z100 or any

PC compatible. Media = 5 1/4" disk 40tk DS
Both for \$19.95 - Terms, Check or Money Order
H&T Software Associates

Pa. Residents P.O. Box 17417
Add 6% Sales Tax Pgh Pa.15235 (412)372-6527

MSDOS tm Microsoft Inc. Z100 tm zenith



Please let us know 8 weeks in advance so you won't miss a single issue of REMark!

DOODLER-V

Kevin Lerch

Technical Consultant
Heath Company

Hey, I've got a question for you. Is it me or does everyone love to doodle on an envelope or piece of scrap paper when they're on the phone? It like draws me in when I see a pencil just sitting there, beckoning to be used, and oh that blank white paper, what a sin, it must be violated with graphite! Sorry, I got carried away.

Well anyway, here I go again, I descend into the depths of my musty basement and kick on the stereo, sit down in front of my computer and hack away. But today, I am actually "doodling" on my computer, and the best way to doodle on your computer is by using the new "DOODLER-V" software package from Paul F. Herman, Software Graphics Tools.

I spoke to the people at P.F.H., and they sent me the whole nine yards, so I could tell everyone about this new release of an old friend. Some of you might remember an article that ran in REMark, June 1985 on the version of Doodler that was released for the Z-100. Well, the new version, "DOODLER-V", that I received is a horse of a different color.

The first version of DOODLER was released in the spring of '83. This version was written in ZBASIC interpreter. After selling just a few copies of DOODLER, Paul made the necessary changes to the source code and compiled the program with the ZBASIC compiler. The resulting DOODLER.EXE

program was much faster and more practical to use. A font editor (CREATE.EXE) was added to the system to allow the user to create or modify their own character fonts.

Well, low and behold, in 1985 Paul introduced DOODLER-PC for the Z-150 (or any PC compatible). DOODLER-PC actually represented a completely new program written in the "C" language.

Paul, foreseeing the release of MS-DOS Ver 2.0 with the support for path names and the like, realized that another version of DOODLER was needed. The initial con-

cept for DOODLER-V was to write a program which would take full advantage of all the new MS-DOS 2.xx features, and would be capable of running on either the Z-100 or PC compatibles (without the need for two separate programs). What started out as a simple upgrade, turned into a major rewrite of the whole program.

I spoke earlier about the article that ran in REMark, June 1985. Below is a comparison of new features added to DOODLER-V, which were not included in the original DOODLER.

One of the strongest features of the new DOODLER-V is the ability to run on either

Feature	Original DOODLER	DOODLER-V
Packaging	Paper report cover	Cloth covered slip case binder
Manual	28 pages, 8-1/2" x 11"	140+ pages, 5-1/2" x 8"
Operating System	Z-DOS or MS-DOS	MS-DOS 2.xx or higher
MS-DOS Pathnames	No support	Fully supported
Print Drivers	12 printers supported	Guaranteed support for all printers
Print to File	Not supported	Yes
Sideways Printing	Not supported	Yes
Magnify Mode	No	Yes
"Undo" Capability	No	Yes
Active Frames	1 at a time	Up to 9 active frames
Texturing	Not supported	Yes
Rubber-Banding	DOODLER-PC only	Yes
90° Rotation	DOODLER-PC only	Yes
Avg. PIC File Size	53K bytes	5 to 10K (compressed format)
(Full Screen)		
Auto Playback	Not possible	Yes

the Z-100 or the PC compatibles. Being a portable system, any drawings done on the Z-100 will run fine on the PC compatibles.

To begin, the manual is packaged in an attractive hard cover binder, with a hard shell case. Inside the manual, it is broken into six sections.

1. Table of Contents
2. Tutorial
3. Getting Started
4. Reference Guide
5. Appendices
6. Index

The tutorial section is a great place to start. This portion of the manual is a basic step-by-step procedure of how to draw a simple picture of a house. It familiarizes the user with a lot of the basic commands that are available. But this section is in no way a complete overview of all commands DOODLER-V has available.

My first experience drawing with DOODLER-V seemed to be going a little slow. You see, DOODLER-V is a menu driven program, just about everything you want to do is menu driven. After the slow start and having completed the tutorial, I found things went much faster. It seems that the tutorial was very well written, since upon completion I could run quickly through the basic menus with no problems.

Some of the things covered in the tutorial were as follows:

1. **Changing Cursor Step Rate** — The cursor step rate can be changed from a rate of 1 pixel to 9 pixels per keystroke. To do this, you use the F2 key to get to the cursor menu and continue hitting F2 until the desired rate is displayed.
2. **Drawing With The Cursor** — There are two modes that are explained to the user in the tutorial. The MOVE mode and the DRAW mode. If you just want to move the cursor, push the arrow key that corresponds to the direction. But if you want to "SKETCH" or draw a line as you move the cursor, you can do this in the DRAW mode.

In the DRAW mode, you can also choose the color you want to draw in by selecting another menu. You choose the COLOR menu and DOODLER displays the available colors to choose from.

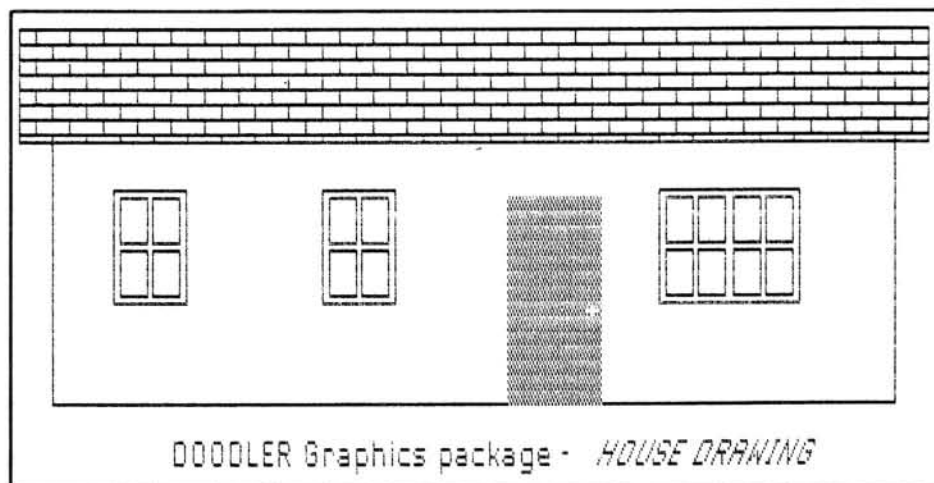
3. **Drawing Point-To-Point** — When drawing point-to-point, you establish a start and end point and DOODLER

draws the line in-between. The nice feature about this command is that once you establish a starting point, and move the cursor from that point, it's like you are "DRAGGING" the line around with the cursor. In the manual they call this "RUBBER BAND", but I always called it "dragging". With a command like this, you can position the line where you want it, and see it in place to see if it is going to look like you want it.

4. **Drawing Forms** — When the word "forms" is used here it means geometric shapes. DOODLER allows you to draw squares without drawing four lines; this saves a lot of time. All you have to do is go to the appropriate menu and tell DOODLER you want to draw a square. When you return to the main menu where the cursor was, one corner of your square is there. As you move the cursor around the screen, you actually drag the square (or rectangle) around with the cursor. This also holds true for a circle. Just go to the menu, tell DOODLER you want to draw a circle, and when you return to the main menu, you can drag a circle around with the cursor. When you get the shape the correct size, you hit one key and it draws it in.
5. **Filling Areas With Color** — This command is like a paint command. You go into an area of your drawing that has a border (like the door on the house drawing) and tell DOODLER you want to color that area. You must first choose a color to paint with. Then, you must tell DOODLER what the border color is so the paint command knows when to stop painting. **CAUTION:** This command can be fatal to your drawing. You see, if the border is not absolutely solid, if there is just one pixel that isn't the same color as the border, the paint leaks out and paints your whole drawing. I did

not experience this, but I thought I should mention it. One more thing before I leave this section, DOODLER also allows you to "TINT" a color to a different shade; like the door on the house picture. When I first drew the door it was white, but then later in the tutorial they have you tint it to a grey color.

6. **The Frame Command** — This command is a useful one. What you are allowed to do is put a box around any portion of your drawing. Whatever is inside that box is treated like a whole entity. That is to say, you can copy, erase, move, save to disk, change size, print, rotate or reverse a frame. Once again on the house drawing, I drew one window on the left. Then I used the frame command to first copy it to two other locations. Then, I reflected it for the double window.
7. **Texturing A Frame** — This is cool, what I used this for is to draw the shingles on the house. Boy, it would have been a real bum trip to have to draw all the lines one at a time. How to do this is you go to an unused portion of the screen and draw the shingle pattern (three shingles). Then, you create a frame around them. I feel I must explain one thing here, DOODLER allows the user multiple frames at one time (nine at one time). Now back to the shingles, then you make a "frame" around the entire roof. All you have to do now is select the color of the pattern, select the pattern frame, and select the frame you wish to put the pattern in.
8. **Text In Your Drawing** — This is a real tuff one. Check this command out. To put text in your drawing you ... TYPE! Hard isn't it. You see, DOODLER uses function keys for menu movement, so there is no need for a text mode. You just put the cursor where you want text and type





away. You can also choose a text font, thus changing the style of text. There are 7 different text fonts included with the base DOODLER-V package. You can also make your own character fonts with the Create command. I was also sent an optional font library disk which contained an additional 44 fonts. For a small price (there's a price list at the end of this article), you receive a folder with a complete graphics representation of all the character fonts included in the package.

9. **The Playback Command** — One thing that sets DOODLER apart from most other graphics programs is its ability to store and play back the commands which were used to create a drawing. Each time you hit a key on the keyboard (or move the mouse), the keystroke is added to the "playback string", which is stored in memory. At any time, you may instruct DOODLER to "playback" your drawing from the beginning. This feature also has another use, called the "UNDO" command. What this allows the user to do is if you make a mistake when drawing, you simply issue a command to play it back. DOODLER clears the screen and starts to redraw your picture — just as you drew it — but very quickly. Before you get to the point in the playback where your original error occurred, you place DOODLER in the single-step mode. In single-step mode, one playback keystroke is executed each time you hit a key on the keyboard. All the user does now is keep single-stepping until you reach the point just previous to your original mistake. Then, abort the playback and continue editing your corrected picture.

Well, that is a little overview of the tutorial section of the manual. When doing the tutorial I used the keypad, but I did receive

the DOODLER mouse pack. There are some things that are to be considered when choosing which method to use when drawing. If you use the mouse and you intend to use the "playback" command, caution must be used. DOODLER sets aside 25K of memory for the use of playback string storage. If the user is inputting with the mouse, the 25K of memory will be filled up much faster than inputting from the keypad. If you make a mistake and you try to use the "undo" command, and the playback string has exceeded 25K, you won't be able to correct the mistake using that command. Just a small warning to you mouse users. But on the other hand, drawing with the mouse is about 100 times faster than with the keypad, it's your choice.

The Reference Guide section of the manual has a detailed description of all the commands available to the user. I wish I had the space to go through all the commands, but this section would be 10 pages

long. So I'll go through the most important ones that I didn't cover in the tutorial explanation.

First, there is the PEN command, DOODLER allows the user to choose the style of pen you are drawing with. Not only does it give you the choice of style, but you can also vary the size of each pen. The six pen styles that are available are diamond, square, vertical, horizontal, backslant diagonal, and italic diagonal.

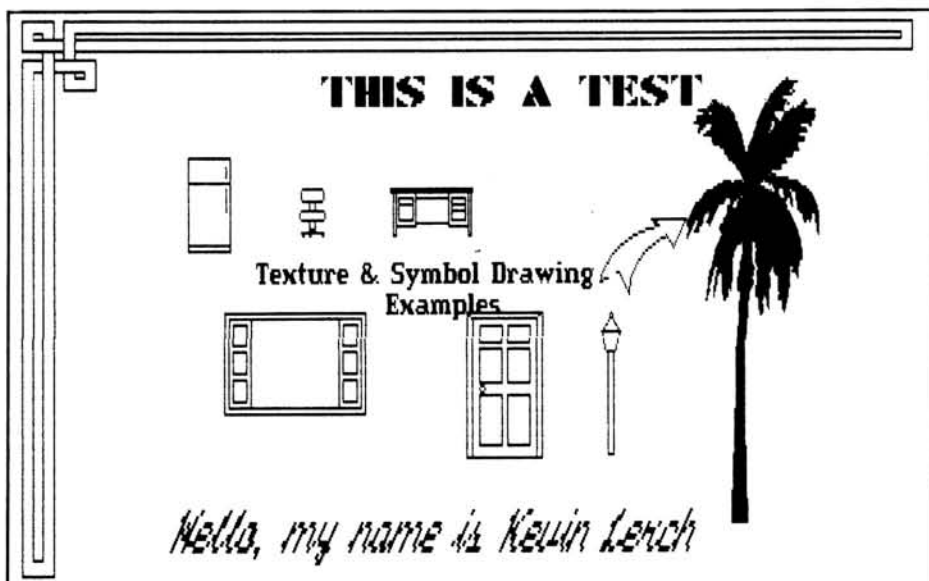
When going through the tutorial, you use a command to draw a box and a circle. You are not limited to drawing just boxes and circles, you may also draw ovals and arcs.

To draw an oval, the center must be selected, then just move the cursor and the vertical and horizontal extent is determined by the final cursor position. One of the best features of DOODLER-V is the rubber-banding or drag effect when drawing an object. You can see the object before the final key is struck and the object is put in place.

To draw an arc, select one end point, then move the cursor to the other end point. The final cursor position when drawing an arc is a third point anywhere on the arc. The center of curvature for the arc is the point which is equidistant from all three points. During the ARC operation, a rubber-band arc will be displayed on the screen. Without this rubber-banding effect drawing an arc would be difficult.

Well, that's all I am going to cover about the Reference section of the manual, I'll leave a little to your imagination.

I received one more thing from the people at Paul F. Herman, and that was a "TEXTURE & SYMBOL" Library. This is a disk with a ton



of little drawings and textures. I thought I would give you a little example of the drawings included. There are a total of 279 pictures in this library, so I can't show you too many, but here are a few.

Some of the other drawings supplied include architecture objects, architecture textures, arrows, borders, repetitive designs, electronics symbols, furniture, grids, hatching, logic symbols, and cross sectional hatching.

Well, this kind of wraps up what I have time and room for, and as a summary I would like to say, GOOD JOB! The software is a good, useful program for anyone who would like to use their computer for a sketchpad. But there are a lot of other uses for DOODLER-V than just doodling. Myself, I sometimes make posters and such things. The different text fonts and the ability to change the scale of the text is very useful. With a good printer, this package is like a drawing machine. Well thanks, Paul F. Herman for the opportunity to play with your software, and I'll talk to the rest of you HUG people later.

Doodler Price List

DOODLER-V Graphics Package	\$ 99.00
For Zenith Z-100 or PC-Compatibles	
MOUSE PACK Mouse Driver Package	\$ 29.95
For Zenith Z-100 or PC-Compatibles	
Font Library Disk for DOODLER	\$ 29.95
44 Ready-to-Use Character Fonts	
Texture & Symbol Library Disk	\$ 29.95
Hundreds of Small Useful Pictures	

Special Packages Available



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

We couldn't have said it better ourselves...

- * Your update policy and newsletter are the best I've seen.
M. Frederick, Omaha, NE
- * Beautiful and elegant in simplicity!
S. Greenberg, Douglaston, NY
- * The software which I have purchased from Paul F. Herman Inc., and the support, has been excellent.
J.M. Hall, San Diego, CA
- * Dollar for dollar, Paul F. Herman software consistently proves to be a best buy. The support is far beyond what might be expected for the price. It reminds one of a users' group attitude.
D.C. Johanson, Jamison, PA
- * Paul Herman's "DOODLER Graphics Package" provides me with every capability I need to develop and print graphics of the highest quality.
S. Revere, Trenton, NJ
- * Paul Herman seems to be filling the gaps in Z-100 graphics that Heath/Zenith didn't.
J. White, Columbus, OH
- * A quality product for the Z-100.
E. Sevin, Bethesda, MD
- * I don't know how I would do the work I do without DOODLER. I recommend it to my customers.
T.W. Flynn, Apollo Beach, FL
- * DOODLER-V is the best Z-100 graphics program I have purchased - the only one I use.
R.W. Hilton, Las Vegas, NV
- * The best graphics package for the Z-100. Simple as that!
J. Fiegel, Gretna, LA
- * Quality software is still being developed for the Z-100 (poor orphan) and P.F. Herman is a leader.
R.A. Tilden Jr., Somerville, MA
- * I've never found such friendly support in both technical and business questions like from P.F.H.
J. Poellmann, W. Germany



Software Graphics Tools
3620 Amazon Drive
New Port Richey, FL 33553

EVERYTHING YOU NEED... \$279⁰⁰

Now it's easy to program the Heath-Zenith HERO-1* Robot with an Apple* II. HERO* Macros for the S-C Software 6800 Cross Assembler program in Heath's Robot Interpreter Language with easily remembered mnemonics.

For example, the line: 1130 > MVWRIM GRIP, OPEN, 60, FAST instructs the HERO* to open his gripper 60 units at fast speed. Motor position is expressed in base-10.

The HERO* Macros come with 30 pages of documentation.

- Transfer to HERO* with ROBI... an affordable interface for the robotics experimenter... is simple.
- ROBI is a complete package. No additional hardware required for Apple* or HERO*.
- ROBI installs quickly in an Apple* II, II+, or IIe. Once installed, no hardware changes are needed. Within minutes, you will be programming HERO*.
- With ROBI and the Cross Assembler, the programmer uses Apple*'s memory to write the program, and HERO*'s memory to run the program.
- Not "copy protected," archival copies may be made as needed.
- ROBI offers expansion potential.

VISA and MasterCard accepted

BERSEARCH
Information Services
26160 Edelweiss Circle
Evergreen, Colorado 80439

The Cross Assembler with HERO* Macros sells for \$100.00; the ROBI Interface sells for \$199.00. Both as a package — \$279.00.
To order, or for more information, call (303) 670-6137.

APPLE* is a trademark of Apple Computer. HERO* is a trademark of Heath Electronics.

Installing And Programming The NEC V20

Richard L. Ferch

1267 Marygrove Circle
Ottawa, ON K2C 2E1

In recent months, there has been considerable interest in the NEC V20 CPU chip. For example, a recent pair of articles in REMark discussed its speed improvements (April 1986, p.63), and its additional features (May 1986, p.45) relative to the 8088. However, neither explained how to use these new features in an actual program. This article is an attempt to provide such an explanation. It begins with some general background information on the 8088, V20 and related CPU chips, follows with detailed explanations of the new machine instructions the V20 offers, and ends with a discussion of my experience with the hardware aspects of installing a V20 (which wasn't quite as simple as I had expected, thanks to other modifications I had made previously).

Background

To begin with, it will be useful to describe briefly the differences between members of the Intel 8086 family. The 8086 itself is a true 16-bit CPU chip, which can transfer data to/from memory and I/O devices in 16-bit words. To reduce the pin count, the first 16 address lines (of 20) are multiplexed (shared) with data lines. The 8088, which is used in the H/Z-100, H/Z-100 PCs and "clone" PCs, is identical to the 8086 from the software point of view (except for timing). It is also very similar electrically, except that only 8 of the address pins are multiplexed with data. This allows the

8088 to interface readily with 8-bit memory and peripherals (and makes the H/Z-100's dual processor design feasible). As a result, though, the 8088 has to transfer 16-bit data words in two separate consecutive 8-bit bytes. Therefore, even at the same clock speed, an 8088 will be considerably slower than an 8086. Unfortunately, you cannot speed up an 8088-based system by substituting an 8086, without major redesign of the rest of the system.

The next members of the 8086 family are the 80186 and 80188. Electrically, these are completely incompatible with their predecessors, since many support functions were formerly done by auxiliary chips and are included on one chip. To the programmer, they are "upward-compatible", meaning that they execute all of the 8086/8088 instructions, plus several "enhanced" instructions. There are also a number of speed improvements: The calculation of effective addresses, which takes 5 to 12 clock cycles on the 8086/8088, takes only 2 cycles; iterative instructions (string moves and multi-bit shifts/rotates) are a lot faster; and multiply and divide speeds are much improved.

The H/Z-200 or "AT"-type computers use the 80286 CPU. This chip, which only comes in a 16-bit version, includes all of the software-related 80186 improvements, plus further hardware changes and additional instructions intended to sup-

port multi-user virtual-memory operating systems. However, MS-DOS does not make use of these "protected mode" instructions. Instead, it treats the 80286 as if it were an 80186. For example, recent versions of MASM have a ".286C" pseudo-operation to support 80286 instructions, and all of these new instructions are available on the 80186/80188 as well.

Now for the NEC V20: The V20, or uPD70108, is intended as a direct plug-in replacement for the 8088. (There is also a V30, or uPD70116, which replaces the 8086, and more advanced members of the "V" series may be able to replace other Intel CPUs.) Electrically, the V20 is the same as the 8088, except that it draws less power, and hence runs cooler. To the programmer, however, it looks like an 80188 with 8080 emulation added, and it also has several other new "unique" instructions. That is, when compared to the original 8088, the V20 does effective address calculations in only 2 clock cycles; string, shift/rotate, multiply and divide instructions run faster; the "enhanced" 80186/80286 instructions can be executed; there are several brand new instructions for packed BCD arithmetic and bit manipulation; and the V20 can be switched to 8080 mode to run 8-bit 8080 code.

Much of the initial interest in the V20 was based on its 8080 emulation capability. This is of particular interest to H/Z-100 PC

Table 1
"Enhanced" V20 Instructions (Intel Mnemonics)

BOUND	reg16,mem32	= 62 (mod reg16 r/m) [disp-low] [disp-high]
ENTER	aabb,cc	= C8 bb aa cc
IMUL	reg16,aabb	= 69 (11 reg16 reg16) bb aa
IMUL	reg16,cc	= 6B (11 reg16 reg16) cc
IMUL	reg16,mem16/reg16,aabb	= 69 (mod reg16 r/m) [disp-low] [disp-high] bb aa
IMUL	reg16,mem16/reg16,cc	= 6B (mod reg16 r/m) [disp-low] [disp-high] cc
INS	string,DX	= 6C/6D (for byte/word "string" type)
INSB		= 6C
INSW		= 6D
LEAVE		= C9
OUTS	DX,string	= 6E/6F (for byte/word "string" type)
OUTSB		= 6E
OUTSW		= 6F
PUSH	aabb	= 68 bb aa
PUSH	cc	= 6A cc
PUSHA		= 60
POPA		= 61
RCL	mem8/reg8,cc	= C0 (mod 010 r/m) [disp-low] [disp-high] cc
RCL	mem16/reg16,cc	= C1 (mod 010 r/m) [disp-low] [disp-high] cc
RCR	mem8/reg8,cc	= C0 (mod 011 r/m) [disp-low] [disp-high] cc
RCR	mem16/reg16,cc	= C1 (mod 011 r/m) [disp-low] [disp-high] cc
ROL	mem8/reg8,cc	= C0 (mod 000 r/m) [disp-low] [disp-high] cc
ROL	mem16/reg16,cc	= C1 (mod 000 r/m) [disp-low] [disp-high] cc
ROR	mem8/reg8,cc	= C0 (mod 001 r/m) [disp-low] [disp-high] cc
ROR	mem16/reg16,cc	= C1 (mod 001 r/m) [disp-low] [disp-high] cc
SAR	mem8/reg8,cc	= C0 (mod 111 r/m) [disp-low] [disp-high] cc
SAR	mem16/reg16,cc	= C1 (mod 111 r/m) [disp-low] [disp-high] cc
SHL	mem8/reg8,cc	= C0 (mod 100 r/m) [disp-low] [disp-high] cc
SHL	mem16/reg16,cc	= C1 (mod 100 r/m) [disp-low] [disp-high] cc
SHR	mem8/reg8,cc	= C0 (mod 101 r/m) [disp-low] [disp-high] cc
SHR	mem16/reg16,cc	= C1 (mod 101 r/m) [disp-low] [disp-high] cc

Note: The (mod reg r/m) byte and "disp" bytes are explained in Table 2.

(and "clone") owners, since they don't have the 8085 processor that H/Z-100 owners do. Commercial software is available for the H/Z-100 PC computers to run CP/M using the V20, giving them similar capabilities to the H/Z-100's CP/M-85. The V20 appears to be somewhat slower than the 8085 at the same clock speed. On the other hand, it has the advantage that improvements to the V20's clock speed will also speed up its 8080 emulation, whereas the 8085 on the H/Z-100 always runs at 5 MHz, regardless of the 8088's clock speed. The one simple benchmark 8080 program I tried took 103 seconds on the 8085 at 5 MHz, 123 seconds on the V20 at 5 MHz, and 82 seconds on the V20 at 7.5 MHz. The exact speed ratio probably depends on the instruction mix.

The speed improvements of some individual 8088 instructions on the V20 have been described elsewhere. In my experience, using typical higher-level language code, the V20 is about 5 to 10 percent faster overall than the 8088 at the same clock speed, although certain specialized applications may be improved slightly more. This speed-up is almost all due to the faster effective address calculations, since the other improvements affect only a small proportion of the typical instruction mix

(even when doing number-crunching jobs, the CPU spends much of its time just moving data around and performing simple logical operations).

Without software changes, the other capabilities offered by the V20 (apart from this slight speed improvement) will go to waste. If your compiler (or assembler) has an 80186 or 80286 switch, you can take advantage of the "enhanced" instructions simply by telling the compiler you have an 80186 or 80286. Without such a switch, the only way to use these instructions is to hand-code them in Assembly language (or to write macros to do the encoding). The "unique" V20 instructions, including those related to 8080 emulation, always have to be hand-coded.

The following information is based on the NEC specification sheets, IBM documentation for MASM, and experiments with my V20. You will need experience with 8086 Assembly language to understand the explanations below.

"Enhanced" Instructions

The actual hexadecimal machine codes for these instructions are given in Table 1. The NEC mnemonics for many of these instructions differ from the Intel 80186/80286 mnemonics. I have used the Intel mne-

monics, since they are more widely known, and are supported by recent versions of MASM. If your assembler has the ".286C" or equivalent pseudo-operation, you will be able to use all addressing modes with these instructions. Otherwise, you will not be able to use relocatable (i.e. assembler- or linker-resolved) addresses without resorting to undesirable techniques. Only addressing modes based on registers and fixed offsets (e.g., 4[BX + SI]) can be readily hand-encoded.

ENTER (NEC: PREPARE): This instruction is used to prepare the stack frame for subroutines in higher-level languages. The first argument, "aabb", is a word containing the number of bytes of local storage to reserve. The second argument, "cc", is a byte. In the most common case, when "cc" is zero, "ENTER aabb,0" is equivalent to:

```
PUSH BP
MOV BP,SP
SUB SP,aabb
```

The "aabb" bytes of local storage are addressed using negative offsets from BP, while the subroutine's arguments are addressed using positive offsets from BP. It is also possible to specify a non-zero value for "cc", in which case "cc" words of previous frame pointers are also saved:

```
PUSH BP
MOV FP,SP
REPT cc-1
SUB BP,2
PUSH BP
ENDM
MOV BP,FP
PUSH BP
SUB SP,aabb
```

(FP is an inaccessible hardware register.) Regardless of which form of ENTER is used, LEAVE should be used before every RET in the subroutine.

LEAVE (NEC: DISPOSE): This releases the frame set up by ENTER, and is equivalent to:

```
MOV SP,BP
POP BP
```

It is normally followed immediately by a RET.

PUSH immediate: An immediate word "aabb", or byte "cc" sign-extended to a word, is pushed onto the stack:

```
MOV FP,aabb
PUSH FP
```

OR:

```
MOV FP,cc
PUSH FP
```

PUSHA (NEC: PUSH R): Registers AX, CX, DX, BX, the original SP, BP, SI and DI are pushed onto the stack:

MOV FP,SP
 PUSH AX
 PUSH CX
 PUSH DX
 PUSH BX
 PUSH FP
 PUSH BP
 PUSH SI
 PUSH DI

POPA (NEC: POP R): Registers DI, SI, BP, SP (discarded), BX, DX, CX and AX are popped from the stack:

POP DI
 POP SI
 POP BP
 POP FP
 POP BX
 POP DX
 POP CX
 POP AX

IMUL immediate (NEC: MUL immediate): The first operand is always a 16-bit destination register, and the last operand is an immediate word "aabb" or byte "cc". If three operands are specified, the signed word addressed by the second operand ("mod", "r/m" and "disp" — for a summary of the "mod", "reg" and "r/m" bits, see Table 2) is multiplied by the immediate quantity "aabb", or "cc" sign-extended to 16 bits, and the lower 16 bits of the result are placed in the first operand register. If the result is longer than 16 bits, the carry and overflow flags are set; AF, PF, SF and ZF are undefined. If only two operands are specified, the source and destination are the same register.

ROL immediate, ROR immediate, RCL immediate (NEC: ROLC immediate), RCR immediate (NEC: RORC immediate), SAL/SHL immediate (NEC: SHL immediate), SHR immediate, SAR immediate (NEC: SHRA immediate): All of these instructions are exactly like their counterparts which use CL for the shift count, except that the shift count "cc" is an immediate byte quantity. Note that, like the 8088 but unlike the 80286, the V20 allows shift/rotation counts greater than 31. (Of course, large shift counts don't do anything small shifts can't do — they just take longer.)

INS/INSB/INSW (NEC: INM): This instruction works the same way as STOS, except that the data comes from the I/O port addressed by DX, instead of from AL or AX. The destination string is addressed by ES:[DI], and DI is modified after every transfer, depending on the data type of the string and on DF. If this instruction is preceded by a repeat prefix, the input device must be fast enough to supply a new data value every 8 clock cycles; if INSW is used, the port must transfer 16 bits at a time. These

conditions are unlikely to be met in most systems.

OUTS/OUTSB/OUTSW (NEC: OUTM): This instruction works like LODS, except the data is sent to the I/O port addressed by DX. The source string is addressed by DS:[SI]. If a repeat prefix is used, the output device must be fast enough to accept a new data item every 8 cycles; if OUTSW is used, the port must transfer 16 bits at a time. These conditions are unlikely to be met in most systems.

BOUND (NEC: CHKIND): The second operand must be a doubleword memory location (mod=11 is not allowed). If the signed value of the first operand is either less than the first word or greater than the second word, an INT 5 interrupt occurs. If your system uses INT 5 for the Print Screen interrupt (as Z-DOS/MS-DOS/PC-DOS do), you will be unable to use this instruction (unless your desired response to an out-of-range value happens to be a print screen operation!).

"Unique" Instructions

The following NEC-only instructions are not supported by widely-available software. Therefore, they will have to be hand-coded, and relocatable addresses cannot be used. They can be divided into three groups: Packed BCD instructions, bit manipulation instructions, and processor control (including 8080 emulation). NEC mnemonics are used. The actual hexadecimal machine codes are given in Table 3.

ADD4S: This adds the packed BCD string at DS:[SI] to the packed BCD string at ES:[DI], and stores the result at ES:[DI]. The length of the strings (1-254) is specified by CL. Carry and zero flags are affected, but they will only be as expected if CL is even. If

Table 2
Explanation of Effective Address Calculations

r/m bits	mod = 00		mod = 01		mod = 10		mod = 11	
							(byte)	(word)
000	[BX+SI]	[BX+SI+disp8]	[BX+SI+disp16]				AL	AX
001	[BX+DI]	[BX+DI+disp8]	[BX+DI+disp16]				CL	CX
010	[BP+SI]	[BP+SI+disp8]	[BP+SI+disp16]				DL	DX
011	[BP+DI]	[BP+DI+disp8]	[BP+DI+disp16]				BL	BX
100	[SI]	[SI+disp8]	[SI+disp16]				AH	SP
101	[DI]	[DI+disp8]	[DI+disp16]				CH	BP
110	[disp16]	[BP+disp8]	[BP+disp16]				DH	SI
111	[BX]	[BX+disp8]	[BX+disp16]				BH	DI

Notes:

1. "reg" bits are the same as "r/m" bits (for the case mod = 11).
2. "disp8" is sign-extended to 16 bits for effective address calc.
3. "disp" byte(s) follow(s) (mod reg r/m) byte, lower byte first.
4. The byte/word choice depends on the preceding byte.

CL is odd, the upper 4 bits of the highest byte may also be affected by this instruction, and the flag values will depend on their contents. The AF, OF, PF and SF flags are undefined after this operation.

SUB4S: The same as ADD4S, except subtracts instead of adding.

CMP4S: The same as SUB4S, except that the result of the subtraction is not stored, so ES:[DI] is unchanged. Only the zero and carry flags are affected; if CL is odd, the flag values will not be as expected, but will depend on the contents of the upper 4 bits of the highest byte.

ROL4: The single byte addressed by "mod", "r/m" and "disp" is rotated left by 4 bits through the lower 4 bits of AL. The flag bits and the upper 4 bits of AL are unaffected.

ROR4: The same as ROL4, except the rotation is to the right.

INS: (Warning: note the conflict in mnemonics between this instruction and the "enhanced" Intel input string instruction.) The lower 4 bits of the second operand, which may be either an 8-bit register or an immediate byte, are used as a bit count. The specified bits are moved from the low end of AX to the memory location specified by ES:[DI] at the bit offset specified by the lower 4 bits of the first operand, which must be an 8-bit register. The first operand register, and DI if necessary, are updated to point to the next bit field in memory.

EXT: This instruction is the inverse of INS. The bit field at DS:[SI], bit offset in the lower 4 bits of the first operand, length in the lower 4 bits of the second operand, is transferred to the low end of AX. The first operand register, and possibly SI, is/are updated to point to the next bit field.

Table 3
"Unique" Instructions (NEC Mnemonics)

ADD4S		= 0F 20		
BRKEM	cc	= 0F FF cc		
CLR1	mem8/reg8,CL	= 0F 12 (mod 000 r/m)	[disp-low]	[disp-high]
CLR1	mem16/reg16,CL	= 0F 13 (mod 000 r/m)	[disp-low]	[disp-high]
CLR1	mem8/reg8,cc	= 0F 1A (mod 000 r/m)	[disp-low]	[disp-high] cc
CLR1	mem16/reg16,cc	= 0F 1B (mod 000 r/m)	[disp-low]	[disp-high] cc
CMP4S		= 0F 26		
EXT	reg8a,reg8b	= 0F 33 (11 reg8b reg8a)		
EXT	reg8a,bb	= 0F 3B (11 000 reg8a) bb		
FPO2	fpop	= 66 (11 yyy zzz) 0R		
		67 (11 yyy zzz)		
FPO2	fpop,mem	= 66 (mod yyy r/m) [disp-low]	[disp-high]	0R
		67 (mod yyy r/m) [disp-low]	[disp-high]	
INS	reg8a,reg8b	= 0F 31 (11 reg8b reg8a)		
INS	reg8a,bb	= 0F 39 (11 000 reg8a) bb		
NOT1	mem8/reg8,CL	= 0F 16 (mod 000 r/m)	[disp-low]	[disp-high]
NOT1	mem16/reg16,CL	= 0F 17 (mod 000 r/m)	[disp-low]	[disp-high]
NOT1	mem8/reg8,cc	= 0F 1E (mod 000 r/m)	[disp-low]	[disp-high] cc
NOT1	mem16/reg16,cc	= 0F 1F (mod 000 r/m)	[disp-low]	[disp-high] cc
REPC		= 65		
REPNC		= 64		
ROL4	mem8/reg8	= 0F 28 (mod 000 r/m)	[disp-low]	[disp-high]
ROR4	mem8/reg8	= 0F 2A (mod 000 r/m)	[disp-low]	[disp-high]
SET1	mem8/reg8,CL	= 0F 14 (mod 000 r/m)	[disp-low]	[disp-high]
SET1	mem16/reg16,CL	= 0F 15 (mod 000 r/m)	[disp-low]	[disp-high]
SET1	mem8/reg8,cc	= 0F 1C (mod 000 r/m)	[disp-low]	[disp-high] cc
SET1	mem16/reg16,cc	= 0F 1D (mod 000 r/m)	[disp-low]	[disp-high] cc
SUB4S		= 0F 22		
TEST1	mem8/reg8,CL	= 0F 10 (mod 000 r/m)	[disp-low]	[disp-high]
TEST1	mem16/reg16,CL	= 0F 11 (mod 000 r/m)	[disp-low]	[disp-high]
TEST1	mem8/reg8,cc	= 0F 18 (mod 000 r/m)	[disp-low]	[disp-high] cc
TEST1	mem16/reg16,cc	= 0F 19 (mod 000 r/m)	[disp-low]	[disp-high] cc
CALLN	cc	= ED ED cc (WHILE IN 8080 MODE)		
RETEM		= ED FD (WHILE IN 8080 MODE)		

Note: The (mod reg r/m) byte and "disp" bytes are explained in Table 2.

TEST1: The second operand is either CL or an immediate byte. The first operand may be either a byte or a word, either in memory or a register. The bit specified by the bit offset in the lower 3 (byte) or 4 (word) bits of the second operand, at the address specified by the first operand, is tested. If the bit is zero, ZF is set to 1, and if the bit is 1, ZF is reset to 0. The carry and overflow flags are cleared, and AF, PF and SF are undefined.

CLR1: The bit specified by the bit offset in the second operand, at the address specified by the first operand, is cleared; the flags are unaffected.

SET1: Like CLR1, except the specified bit is set to one.

NOT1: Like CLR1 or SET1, except that the specified bit is inverted.

REPC/REPNC: These repeat prefixes are similar to REPZ/REP NZ, except that CF is used instead of ZF. That is, the following string operation is repeated until CF is cleared (REPC) or set (REPNC), or until CX becomes zero.

FPO2: This instruction is similar to ESC. Its existence leads me to speculate that the

NEC replacement for the Intel 8087 may have additional new instructions (above and beyond the 8087 instruction set).

BRKEM: This instruction is used to enter 8080 emulation mode. It works the same way as "INT cc", except that the CPU is placed in 8080 mode (by clearing the Mode Flag, which is bit 15 of the Flag Word). Only the 8080 instruction set is supported (neither the Z-80 nor the 8085 enhancements are implemented). While in 8080 mode, instruction addresses are calculated relative to CS (set by the interrupt vector), and data addresses are relative to DS (normally set to the same value as CS by the calling program, immediately before executing the BRKEM). The following 8088 registers are used as 8080 registers: AL as A, CH as B, CL as C, DH as D, DL as E, BH as H, BL as L, BP as SP, and IP as PC. The SP, SI, DI, AH and segment registers are unaffected. While the CPU is in 8080 mode, external interrupts are handled in "native" 8088 mode as usual, but IRET causes a return to 8080 mode. The RETEM instruction is used to return to "native" mode.

RETEM (8080 MODE ONLY): This instruction plays the same role after a BRKEM that

IRET does after an INT. It causes a return to "native" mode, at the instruction immediately following the BRKEM.

CALLN (8080 MODE ONLY): While in 8080 mode, this instruction can be used in exactly the same way that "INT cc" would be used in "native" mode. The interrupt routine it invokes will be in "native" 8088 mode, and must not include BRKEM. The IRET at the end of the interrupt routine will cause a return to 8080 mode at the instruction following the CALLN.

Hardware Notes

The V20 is supposed to be a plug-in replacement for the 8088, and many users have had success with a simple direct substitution. However, there are some possible pitfalls, especially when sped-up older systems are involved, as my experience shows.

I have an old H-100 (85-2653-1 motherboard), to which I have added the CDR ZS100 7.5 MHz speed-up and the FBE Research ZMF100 768k memory modification. This system was working fine at 7.5 MHz with the original ICs (except for the new 256k RAM chips, which are rated at 150 ns). When I substituted an 8 MHz-rated V20 (uPD70108D-8) for the 8088, however, I started experiencing intermittent system crashes. These went away at 5 MHz, or at either speed when the 8088 was re-installed.

My first reaction was to try upgrading some of the support chips, as suggested in various letters and articles in REMark. Far from fixing the problem, however, the new chips made it much worse. The more high-speed chips there were in the system, the shorter the interval between crashes. Evidently, there is a timing glitch on the old board when fast parts with rapid switching times are used.

My next step was to replace the original support chips and the 8088 CPU, while I studied the wiring changes suggested for installing the HA-108 upgrade kit on older H/Z-100s (REMark, July 1985, p.22). The memory modification and 256k RAM modification were not relevant, since the ZMF-100 did the same job and I knew it worked at 7.5 MHz. (Actually, the ZMF100 upsets the H/Z-100's memory map options, but none of my operating systems or other software uses this feature anyway.) The wait state modification was likewise not needed, and the IC replacement had made things worse, while I preferred the ZS100 speed-up to the crystal replacement (if something goes wrong, I can try it again at



If you have an old H/Z-100 motherboard and are planning to experiment with clock

For the price of the V20, a few IC sockets, the ZMF100 and ZS100 kits and 27 256k RAM chips, my old H-100 is now a 768k machine which runs about as fast as the 8 MHz H/Z-108, and it also has the “enhanced” and “unique” V20 instruction set improvements. If only I hadn’t bought those unneeded fast support ICs!

Appendix

Ready Logic Modification

- U205 – 14-pin IC socket, pins 12 and 13 bent up, plugged into old socket
- connect pin 13 to new U206 socket pin 1 (bent up)
- connect pin 12 to new U206 socket pin 13 (bent up)
- plug U205 into new socket

- U206 - 14-pin IC socket, pins 1 and 13 bent up, plugged into old socket
- connect pin 1 to U205 socket pin 13, U220 socket pin 5 (both bent up)
- connect pin 13 to U205 socket pin 12, U236 socket pin 4 (both bent up)
- plug U206 into new socket

- U220 - 14-pin IC socket, pins 5 and 6 bent up, plugged into old socket
- connect pin 5 to U206 socket pin 1 (bent up)
- connect pin 6 to U236 socket pin 3 (bent up)
- plug U220 into new socket

- U236 – 18-pin IC socket, pins 3, 4 and 15 bent up, plugged into old socket
- connect pin 4 to U206 socket pin 13 (bent up)
- connect pin 3 to U220 socket pin 6 (bent up)
- connect pin 15 to keyboard (GND) end of R123
- if you are using a ZS100 or similar speed-up module, plug it into

this prepared socket; otherwise,
plug U236 into new socket

Refresh Clock Modification

First, check that there are no jumpers connected to U130 pins 8, 9 and 10 on the back of the motherboard; if there are, this change has already been made.

- U130 - 14-pin IC socket, pins 8, 9 and 10 bent up, plugged into old socket
- connect pin 10 to keyboard end of R104
- connect pin 8 to feedthrough beside (connected to) U152 pin 1
- connect pin 9 to feedthrough 1/8" out from inner end of U243, just left of centerline of U243 (connected to U225 pin 3)
- plug U130 into new socket

- U168 - 14-pin IC socket, pin 11 bent up (N.C.), plugged into old socket
- plug U168 into new socket

Swap Logic Modification

If you have a ZMF100 kit installed, you will have to replace the small piggyback board at U173 to make room for the U156 socket:

- U173 - 20-pin IC socket, pins 1 and 2 bent up, plugged into old socket
- connect pins 1 and 2 together, and to the keyboard (GND) end of C168
- connect the P1 pin nearest the keyboard on the ZMF100 large

- piggyback board to the feedthrough between U156 pin 7 and C169 (formerly connected to U173 pin 1)
- connect the other P1 pin to the feedthrough nearest U173 pin 4 (formerly connected to U173 pin 2)
- plug U173 into new socket and set ZMF100 small board aside

- U155 - 14-pin IC socket, pin 8 bent up,
plugged into old socket
- connect pin 8 to U156 pin 13
(bent up)
- plug U155 into new socket

- U156 - 14-pin IC socket, pins 11, 12 and 13 bent up, plugged into old socket
- connect pin 13 to U155 pin 8 (bent up)
- connect pin 11 to U171 pin 10 (bent up)
- connect pin 12 to feedthrough next to (connected to) U171 pin 1
- plug U156 into new socket

- U171 - 14-pin IC socket, pin 10 bent up, plugged into old socket
- connect pin 10 to U156 pin 11 (bent up)
- plug U171 into new socket



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.

HUG Price List

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

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

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

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
H8 — H/Z-89/90				
ACCOUNTING SYSTEM	885-8047-37	CPM	BUSINESS	20.00
ACTION GAMES	885-1220-[37]	CPM	GAME	20.00
ADVENTURE	885-1010	HDOS	GAME	10.00
ASCRTY	885-1238-[37]	CPM	AMATEUR RADIO	20.00
AUTOFILE (Z80 ONLY)	885-1110	HDOS	DBMS	30.00
BHASIC 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
XVET ROBOT X-ASSEMBLER	885-1229-[37]	CPM	UTILITY	20.00
Z80 ASSEMBLER	885-1078-[37]	HDOS	UTILITY	25.00
Z80 DEBUGGING TOOL (ALDT)	885-1116	HDOS	UTILITY	20.00

H8 — H/Z-89/90 — H/Z-100 (Not PC)

ADVENTURE	885-1222-[37]	CPM	GAME	10.00
BASIC-E	885-1215-[37]	CPM	LANGUAGE	20.00
CASSINO GAMES	885-1227-[37]	CPM	GAME	20.00
CHEAPCALC	885-1233-[37]	CPM	SPREADSHEET	20.00
CHECKOFF	885-8011-[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
DDCUMAT & DOCLIST	885-8019-[37]	CPM	TEXT PROCESSOR	20.00
DUNGEONS & DRAGONS	885-1209-[37]	CPM	GAMES	20.00
FAST ACTION GAMES	885-1228-[37]	CPM	GAME	20.00
FAST EDDY & BIG EDDY	885-8018-[37]	CPM	TEXT PROCESSOR	20.00
FUN DISK I	885-1236-[37]	CPM	GAMES	20.00
FUN DISK II	885-1248-[37]	CPM	GAMES	35.00
GAMES DISK	885-1206-[37]	CPM	GAMES	20.00
GRADE	885-8036-[37]	CPM	GRADE BOOK	20.00
HRUN	885-1223-[37]	CPM	HDOS EMULATOR	40.00
HUG BINDER	885-0004	N/A	REMARK BINDER	5.75
HUG FILE MANAGER & UTILITIES	885-1246-[37]	CPM	UTILITY	20.00
HUG SOFTWARE CATALOG UPDATE #1	885-4501	VARIOUS	PRODUCTS 1983 THRU 1985	9.75
KEYMAP CPM-80	885-1230-[37]	CPM	UTILITY	20.00
MBASIC PAYROLL	885-1218-[37]	CPM	BUSINESS	60.00
MICRONET CONNECTION	885-1224-[37]	CPM	COMMUNICATION	16.00
NAVPROGSEVEN	885-1219-[37]	CPM	FLIGHT UTILITY	20.00
REMARK VOL 3 ISSUES 24-35	885-4003	N/A	1982	20.00
REMARK VOL 4 ISSUES 36-47	885-4004	N/A	1983	20.00
REMARK VOL 5 ISSUES 48-59	885-4005	N/A	1984	25.00
REMARK VOL 6 ISSUES 60-71	885-4006	N/A	1985	25.00

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

H/Z-100 (Not PC) Only

ACCOUNTING SYSTEM	885-8048-37	MSDOS	BUSINESS	20.00
CALC	885-8043-37	MSDOS	UTILITY	20.00
CARDCAT	885-3021-37	MSDOS	BUSINESS	20.00
CHEAPCALC	885-3005-37	MSDOS	SPREADSHEET	20.00
CHECKBOOK MANAGER	885-3013-37	MSDOS	BUSINESS	20.00
CP/EMULATOR	885-3007-37	MSDOS	CPM EMULATOR	20.00
DBZ	885-8034-37	MSDOS	DBMS	25.00
ETCHDUMP	885-3005-37	MSDOS	UTILITY	20.00
EZPLOT	885-3023-37	MSDOS	PRINTER PLOTTING UTILITY	20.00
FAST EDDY	885-8025-37	CPM	TEXT PROCESSOR	20.00
FAST EDDY	885-8029-37	MSDOS	TEXT PROCESSOR	20.00
GAMES CONTEST PACKAGE	885-3017-37	MSDOS	GAMES	25.00
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
HUGPBB5	885-5006-37	CPM86	COMMUNICATION	40.00
HUGPBB5 SOURCE LISTING	885-5007-37	CPM86	COMMUNICATION	60.00
ICT 8080 TO 8088 TRANSLATOR & HFM	885-5008-37	CPM86	UTILITY	20.00
KEYMAP	885-3010-37	MSDOS	UTILITY	20.00
KEYMAP	885-5001-37	CPM86	UTILITY	20.00
KEYMAP CPM-85	885-1245-37	CPM	UTILITY	20.00
MAPLE	885-8023-37	CPM	COMMUNICATION	35.00
MATHFLASH	885-8030-37	MSDOS	EDUCATION	20.00
MATT	885-8045-37	MSDOS	MATRIX UTILITY	20.00
ORBITS	885-8041-37	MSDOS	EDUCATION	25.00
POKER PARTY	885-8042-37	MSDOS	ENTERTAINMENT	20.00
SCICALC	885-8028-37	MSDOS	UTILITY	20.00
SKYVIEWS	885-3015-37	MSDOS	ASTRONOMY UTILITY	20.00
SMALL-C COMPILER	885-3026-37	MSDOS	LANGUAGE	30.00
SPELL5	885-3035-37	MSDOS	SPELLING CHECKER	20.00
SPREADSHEET CONTEST PACKAGE	885-3017-37	MSDOS	VARIOUS SPREADSHEETS	25.00
TERM86 & DSKED	885-5004-37	CPM86	COMMUNICATION & UTILITIES	20.00
TREE-ID	885-3036-37	MSDOS	TREE IDENTIFIER	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
HUG BACKGROUND PRINT SPOOLER	885-3029-37	MSDOS	UTILITY	20.00
HUG BINDER	885-0004	N/A	REMARK BINDER	5.75
HUG EDITOR	885-3012-37	MSDOS	TEXT PROCESSOR	20.00
HUG MENU SYSTEM	885-3020-37	MSDOS	UTILITY	20.00
HUG SOFTWARE CATALOG UPDATE #1	885-4501	VARIOUS	PROD 1983 THRU 1985	9.75
HUGMCP	885-3033-37	MSDOS	COMMUNICATION	40.00
HUGPBB5 SOURCE LISTING	885-3028-37	MSDOS	COMMUNICATION	60.00
HUGPBB5	885-3027-37	MSDOS	COMMUNICATION	40.00
ICT 8080 TO 8088 TRANSLATOR	885-3024-37	MSDOS	UTILITY	20.00
MISCELLANEOUS UTILITIES	885-3025-37	MSDOS	UTILITIES	20.00
REMARK VOL 5 ISSUES 48-59	885-4005	N/A	1984	25.00
REMARK VOL 6 ISSUES 60-71	885-4006	N/A	1985	25.00
REMARK VOL 7 ISSUES 72-83	885-4007	N/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
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
TSPELL	885-8044-37	MSDOS	SPELLING CHECKER	20.00

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

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

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

HEPCAT is 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. Watch for Pat's write-up in this issue of **REMark**.



HUG NEW PRODUCTS



10 - Very Good
9 - Good
8 - Average

TABLE C
Product Rating

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

- 7 - Hardware limitations (memory, disk storage, etc.)
- 6 - Requires special programming technique
- 5 - Requires additional or special hardware
- 4 - Requires a printer
- 3 - Uses the Special Function Keys (f1,f2,f3,etc.)
- 2 - Program runs in *Real Time**
- 1 - Single-keystroke input
- 0 - Uses the H19 (H/Z-89) escape codes (graphics, reverse video)

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

ORDERING INFORMATION

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

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

NOTES

The [-37] means the product is available in hard-sector or soft-sector. Remember, when ordering the soft-sectored format, you must include the "-37" after the part number (e.g., 885-1223-37).

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

HUG P/N 885-3045-37
HEPCAT \$35.00

Introduction: HEPCAT is an acronym for HUG Engineer's and Programmer's CALCulation Tool. HEPCAT is a memory resident pop-up calculator with two important differences from other programs of this type. 1) When HEPCAT is popped up, the background program continues to function. This means that you can calculate while your computer is grinding away at a long compilation, or calculating a huge spreadsheet, etc., without halting the background process. 2) The PC version of HEPCAT is compatible with all CGA and EGA video modes, and can be popped up over programs such as Microsoft Windows, which other pop-ups cannot do. HEPCAT also offers more features than other pop-up calculators.

Requirements: To use HEPCAT, you need any Heath/Zenith PC-compatible (H/Z-100 PC series, H/Z-200 series, etc.), or H/Z-100 (not PC) computer, or an expanded ET-100 trainer, and any version of MS-DOS or Z-DOS. HEPCAT only uses about 16k of memory, so you should be able to run it in a minimum 128k system.

Specifications

HEPCAT is actually two calculators in one — a scientific floating point calculator and a programmer's binary calculator. Both calculators use standard infix notation. The floating point calculator features 8 significant digits and a two digit exponent, with a range from 1.0 E-65 to 9.9999999 E+62. It has four display modes: fixed point (with 2 to 8 places to the right of the decimal, 16 places total), standard floating point, scientific notation, and engineering notation

(a form of scientific notation where the exponent is forced to a multiple of 3).

The binary calculator is a 32-bit calculator that works in the following number bases: binary, tetral (base 4), octal, split octal, decimal, and hexadecimal.

Converting a number from one radix to another, or from the binary calculator to the floating point calculator and vice versa is simply a matter of pressing an up or down arrow key.

HEPCAT is at least as accurate as a good BASIC interpreter in the transcendental functions, and it is absolutely accurate within the range of 8 significant digits in addition, subtraction, multiplication, and division, because it uses BCD math, which does not introduce round-off errors. Try PRINT 100-99.99 in BASIC to see an example of a round off error.

The floating point calculator in HEPCAT can perform the following operations: add, subtract, multiply, divide, powers (X^Y), rectangular to polar conversion, polar to rectangular conversion. It also can perform the following transcendental functions: pi (returns 3.1415926), factorial, square root, sine, cosine, tangent, arcsine, arccosine, arctangent, log (base 10), anti-log (10^X), log (natural), and anti-log (natural, e^X). The trig functions can be done using angles in radians or degrees. The HEPCAT calculator can also perform the following American/metric and other conversions: degrees to radians, radians to degrees, Fahrenheit to Celsius, Celsius to Fahrenheit, centimeters to inches, inches to centimeters, meters to feet, feet to meters, grams to ounces, ounces to grams, kilograms to pounds, pounds to kilograms, milliliters to fluid ounces, fluid ounces to milliliters, liters to quarts, and quarts to liters.

The binary calculator in HEPCAT can perform the following operations: add, subtract, multiply, divide, modulo (find the remainder of a division), shift left, shift right, AND, OR, and XOR.

HEPCAT contains an ASCII table, which is always available while you are in the binary mode.

When HEPCAT is "popped up", it opens a small (34 column by two line) window, normally near the top right corner of your screen. The window shows you the numbers you enter, your answer, and other essential information about the calculation in progress. The HEPCAT commands are designed to be easier to remember than those of other pop-up calculators, and the basic four calculations (add, subtract, multiply, divide) can be done entirely at the keypad.

HEPCAT comes with source code to the floating point and binary math packages, which are separate modules. These packages can be used in your own Assembly

language programs that require math capabilities. The documentation is supplied in printed form.

The HEPCAT disk contains the following files:

README	.DOC	INSTALL	.BAT
HEPCAT	.POM	HEPCAT	.ZOM
HEPSET	.COM	SCRNCLK	.POM
SCRNCLK	.ZOM	CLK	.COM
BCD	.ACM	TRAN	.ACM
BINMATH	.ACM	SCRNCLK	.PSM
SCRNCLK	.ZSM	CLK	.ASM

Here is an explanation of some of the files:

INSTALL.BAT — This is a batch file that makes it easy for you to install the version of HEPCAT that is correct for your system onto your disk.

HEPCAT.POM, HEPCAT.ZOM — These are the PC and H/Z-100 versions of HEPCAT.

HEPSET.COM — This program allows you to configure certain aspects of HEPCAT,

port. It takes over the computer's printer interrupt, installs a software protocol handler, and sets the proper parameters for communication between IBM-compatibles and the *LaserWriter*. These printers require *LaserOne* to operate reliably. With *LaserOne* installed (once each computing session, before you print the first time), you need not use the DOS MODE.COM. You, therefore, can run *LaserOne* from AUTO-EXEC.BAT and set up the system for a naive user. An important benefit of *LaserOne* is that it makes the DOS PRINT.COM an excellent printer spooler and *PostScript* dump program for the *LaserWriter*. *LaserOne* has been tested with an extensive body of application software, including Microsoft *Windows*, and has been found completely compatible with everything.

STAT determines the current status of important programmable features of the *LaserWriter*. It makes a printout record including: fonts (both permanent and temporary) in the printer; version of the *PostScript* ROM; total number of pages printed; and the communications settings for each channel. STAT tells you if your *LaserWriter* supports both hardware and software handshaking (some do), or only software handshaking (some don't).

If your printer supports both hardware and software handshaking, HANDSHAKE lets you set the *LaserWriter* for your choice. You may change handshaking as appropriate, to the capacity of the *LaserWriter* EEROM.

such as the display colors and the initial display mode or radix.

SCRNCLK.POM, SCRNCCLK.ZOM — These are improved versions of the screen clock program that has been listed in REMark magazine for PC and H/Z-100 computers. These versions are fully compatible with HEPCAT and with nearly all other programs. They provide an on-screen time display in the upper right corner of your screen that is always there (if you want it to be) while you run other programs.

CLK.COM — This is a control program for the screen clock programs.

BCD.ACM, TRAN.ACM, BINMATH.ACM — These are the floating point and binary math packages used in HEPCAT. These packages are provided for use in your Assembly language programs that do mathematical calculations.

Comments: none

TABLE C Rating: (2, 3, 10)

DELASER.COM "unhooks" *LaserOne* from the computer's printer interrupt in case you want to use a parallel printer instead of the *LaserWriter* in the computing session.

MACHEATH.COM translates ASCII files from the Apple Macintosh to the format expected by MS-DOS programs.

PSPRINT.COM does a simple *PostScript* encoding of an ASCII file so the *LaserWriter* can print it. Although PSPRINT can handle *WordStar* files, it does not reproduce print-formatting features, such as underlining or boldfacing. Its aim is to let you use the *LaserWriter* as a simple line printer for proofing and quick printing of doc files. PSPRINT can print directly to the *LaserWriter* or to any other device, such as a modem. It also can output to a disk file you may edit before printing.

All those programs are easy to use. *LaserOne* has no options. The other programs may be used in either a command mode or a prompt mode. There is extensive error checking and almost no possibility of a fatal error.

In addition to instructions for each program, *The LaserWriter Connection* instruction manual gives detailed instructions on connecting the *LaserWriter* to the IBM-compatible computer. It includes an illustrated explanation of the *LaserWriter* switch settings and wiring diagrams for the required cables.

You also may use *LaserOne* as the handler for software handshaking with other serial laser printers, such as the Hewlett-Packard *LaserJet*.



HUG P/N 885-8050-37 LaserWriter Connection \$40.00

Introduction: *The LaserWriter Connection* is software and a printed, illustrated instruction manual for using the Apple *LaserWriter* and *LaserWriter Plus* printers with IBM PC-, XT-, AT-, and PS/2-compatible computers. These printers are among the least expensive *PostScript* devices available, which makes them superb for typesetting, desktop publishing, and graphics. Apple, however, supports them almost exclusively for use with its own Macintosh computers and has no good information on using them with IBM-compatibles. This is the very same package Joseph Katz (columnist for REMark and Sextant) developed for his own use.

Requirements: Software in *The LaserWriter Connection* provides the interface and basic support for both *LaserWriters* on either COM1 or COM2 of a true IBM-Compatible computer with any version of MS-DOS. Of course it supports all Heath and Zenith Z-100 PC and Z-200 PC computers, and Zenith's versions of MS-DOS for them.

Author: Joseph Katz

Program Contents: *LaserOne*, the heart of *The LaserWriter Connection*, controls traffic between your computer and the *LaserWriter*. You use one of two programs provided: LAS1.COM is used with the COM1 port; LAS2.COM is used with the COM2



IBM-PC IN An H / Z-100

**Basic Board W / HUG Software \$149.00
HUG ZPC V2 & UPGRADE**

Both Included

An \$80.00 Value

This EXCELLENT COMBINATION is used by many universities, colleges and the U.S. Naval Academy.

Options Available

- No Solder H / Z-100 Mod Kit \$5.
- COM 1 \$44. • COM 2 \$39. • Clock \$44.
- 8Mhz V20's \$11.
- 8Mhz Interrupt Kits, \$12.

H / Z-100 Requires Modifications and 768K of RAM

CHECKS AND M.O. IN US FUNDS, VISA AND MC ACCEPTED
US ORDERS ADD \$4.00 S & H, APO & FPO ADD \$7.00 S & H
CA RESIDENTS ADD SALES TAX

IBM-PC is a registered trademark of IBM Corp.
ZPC is a product of the Heath Users Group

OTHER MFR'S PRODUCTS at LOWEST PRICES

•HARD DISK DRIVES

Fujitsu, Seagate, Miniscribe, Rodime and others
5.25", 3.5", naked or packaged
e.g. 20mb \$269., 30mb & 299, SCSI PC \$149.
ST-225-N & CDR-11B for Z/100 \$799
Call for current pricing

•FANTASTIC FLOPPIES

5.25", DS/DD includes tyvek sleeve, label
write protect and reinforced hub ring
Thousands in use without a single bug!
Life time guarantee. 20 for \$10.00

•6 OUTLET POWER STRIP

15Amp, Surge and Noise suppression,
SL Waber model no. PM6-NS \$15.00

All prices are plus S & H

Call or write for further information

Scottie Systems

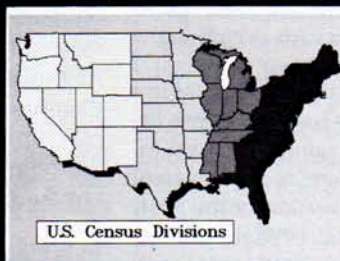
2667 Cropley Ave. #123
San Jose, CA 95132
(408) 259-6226

For the Z100

WITH

ShowOff

**YOUR GRAPHICS
WILL BE OUTSTANDING**



High Resolution Advantages

- Professional Appearance
- Special Effects
- Less Distortion

Improve your presentations and reports with ShowOff, the hi-res graphics editor for the Z100. ShowOff helps you draw, paint and include text to create outstanding graphics.

640 x 480 resolution
92 fill colors • 92 patterns
25 text styles

Easy to use, ShowOff will also display and enhance MacPaint pictures, Lotus graphs, CAD drawings, and other graphics.

**ShowOff \$79 ShowOff with Logitech Mouse \$174 demo disk \$3
ShowOff with Epson Laser Printer \$CALL**

ShowOff versions are available for Logitech, Microsoft, and PC Mouse; HIPAD and Summagraphics digitizers.

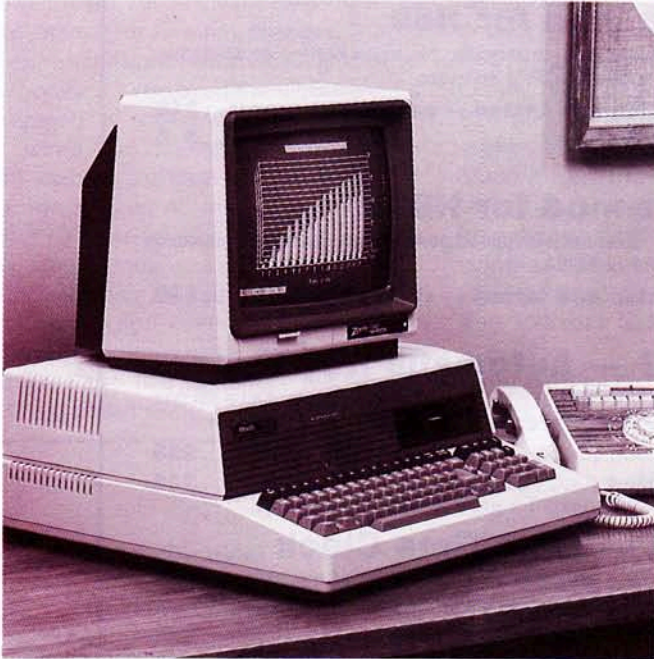
Order direct • VISA • MASTERCARD • check • CALL TODAY

min requirements: Z100 • COLOR RAM • 384K RAM • MS-DOS 2.0



HOGWARE COMPANY
470 BELLEVIEW
ST LOUIS, MO 63119
(314) 962-7833

ZPC Update #17



Pat Swayne
HUG Software Engineer

This is the seventeenth in a series of articles in support of ZPC, a program that allows you to run IBM PC software in H/Z-100 (dual processor) computers. ZPC is available from HUG as part no. 885-3037-37. An upgrade disk for ZPC is also available as part no. 885-3042-37.

In this ZPC Update, I will present a corrected version of the dBASE III + version 1.1 (unprotected) that first appeared in the December '86 REMark. I will also present patches for QuickBASIC version 2.1, Microsoft Word version 3.1, Best Menu version 1.20, and a release of Javelin version 1.1 not covered by previous patches. I am assuming that by now you know how to add patches to your PATCHER.DAT file, and use PATCHER to make patches, so I will not hash over the instructions again in this Update.

dBASE III + Patch Correction

The patch for dBASE III + that was published in the December '86 REMark was missing a one-byte patch. The program would still run OK on ZHS-modified systems, but not on unmodified systems. Here is the corrected patch, which will allow dBASE III + version 1.1 to run on unmodified Z-100's.

DBASE III + vers. 1.1 (unprotected)
Insert the disk containing DBASE.EXE
DBASE.EXE
2E02,F9,90,90
2E0F,F8,90,90
3099,90,90,90,90,90
30A3,90
34C4,90,90,90,90,90
34CE,90
3505,90
z

I have been reading in the BUSS newsletter (published by Sextant Publishing Co., 716 E Street, S.E., Washington, DC 20003) of complaints about copy protection on the Z-100 version of dBASE III. The solution to this problem is to not buy the Z-100 version, but rather buy the PC version (version 1.1, which is unprotected), and run it with ZPC.

QuickBASIC Version 2.1 Patch

The patch originally provided for QuickBASIC version 2.0 does not work for version 2.1. If you have version 2.1, use the following patch:

QUICKBASIC version 2.1
Insert the disk containing QB.EXE.
QB.EXE
A9C4,90,90,90,90,90,90,90,90,90,90,90,90,90,90,90
A9E3,90,90,90,90,90,90,90,90,90,90,90,90,90,90,90
AA4A,90,90,90,90,90,90,90,90,90,90,90,90,90,90,90
11034,90,90,90,90,90,90,90,90,90,90,90,90,90,90,90

1CAB3,CD,90
26BDC,CD,90
29F51,CD,90
z

If you tried the older patch on version 2.1, be sure that you start over with an unpatched copy to try the new patch. QuickBASIC only works with ZPC version 2.1.3, that is, ZPC version 2 upgraded with the Upgrade disk.

Microsoft Word Version 3.1

As with QuickBASIC, the older Microsoft Word patches do not work with version 3.1. Here is the correct patch for that version.

MICROSOFT WORD version 3.1
Insert the disk containing WORD.COM.
WORD.COM
6E5A,0,0
6E61,B0
z

Best Menu Version 1.20

The Best Menu program can be run under ZPC only with the following patch made, even if you have a Scottie board. It may run without patching with the ZHS circuit as presented in the December '86 REMark (because my video retrace port works a little differently from Scottie's).

BEST MENU version 1.20
 Insert the disk containing OVERLAY.COM
 and MAKEMENU.COM.
 OVERLAY.COM
 3C5D,90
 3DC4,90,90,90,90,90,90,90,90,90
 46AD,90,90,90,90,90
 x
 MAKEMENU.COM
 737E,90,90,90,90,90,90,90,90,90
 8B9C,90,90,90,90,90
 z

The video retrace test code in Best Menu is a little different from what it usually is in programs that must be patched. Here is what the code looks like.

```

2D9C:3EC2 B409      MOV     AH,09
2D9C:3EC4 EC        IN      AL,DX
2D9C:3EC5 D0D8      RCR     AL,1
2D9C:3EC7 72FB      JB      3EC4
2D9C:3EC9 FA        CLI
2D9C:3ECA EC        IN      AL,DX
2D9C:3ECB 20E0      AND     AL,AH
2D9C:3ECD 74FB      JZ      3ECA
  
```

If you get a version of Best Menu other than version 1.20, and the above patch does not work with it, you can use DEBUG to search for code like that shown above.

A New Javelin Patch

If you have Javelin version 1.1 dated 11-24-86, the patches presented previously will not work with it. Here is the patch for the 11-24-86 release.

JAVELIN V. 1.1 (11-24-86) Startup
 Insert the Startup disk.
 JAV0.OVL
 B670,90,90,90,90,90,90
 B682,90,90,90,90,90,90
 B6BF,90,90,90,90,90,90
 B6D1,90,90,90,90,90,90
 B724,90,90,90,90,90,90
 B742,90,90,90,90,90,90
 B7A9,90,90,90,90,90,90
 B7C7,90,90,90,90,90,90
 B802,90,90,90,90,90,90
 B81E,90,90,90,90,90,90
 B8B0,90,90,90,90,90,90
 B8C2,90,90,90,90,90,90
 B8F6,90,90,90,90,90,90
 B90A,90,90,90,90,90,90
 B94D,90,90,90,90,90,90
 1E519,90,90,90,90,90,90,90,90,90,90
 90,90
 1E52A,90
 1E530,90
 1E53E,90,90,90,90,90,90,90,90,90,90
 90,90
 z

As I stated before, if you have tried the previous patches, you should start over with an unpatched copy to try this patch. Be sure to set the DOS date and time to the date and time of your JAV0.OVL file (to the minute) before you make the patch, then correct them to the current date and time afterwards. The JAV0.OVL file must have the same date and time on it after the patch (to the minute) that it had before, or the program will not work. *

ANAPRO CORPORATION

805/688-0826

213 Teri Sue Lane Buellton, CA 93427

We Support the H89!

4MHz mod for H89

Easy to install plug-in module. No trace cutting or soldering required. Includes CP/M software.

Assembled and tested — specify disk format . . \$45
HDOS software \$ 5

6MHz mod for H89

Provides maximum high speed performance presently available for the H89 or H89A.

Assembled and tested — specify disk format . . \$59

REP3 — Automatic Key Repeat

Modernize your H89 or H19. Hold any key down for half a second and the key begins repeating. Simple plug-in installation.

Kit \$35
Assembled \$45

TIM2 — Real Time Clock

Installs in the left hand expansion slots of the H89. Includes battery backup. Requires soldering 4 wires to the CPU board.

Kit \$65
Assembled and tested \$75
Software on disk — specify format \$10

DATESTAMPER

Product of Plu*Perfect Systems. Provides automatic time and date stamping for CP/M 2.2 files.

For CP/M — specify disk format \$45

ZCPR3 for H89 and H8

We are licensed by ECHELON to distribute the Z-System. Includes ZCPR3 system, ZRDOS +, supporting utilities, ZCPR3 manual by R Conn, ZRDOS manual and additional instructions. Comes installed with a bootable disk ready to run.

Z-System — specify format and hardware \$98

EMULATE

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

Actrix	Eagle II	Morrow MD	Sanyo 1100
AMPRO	Epson QX-10	NCR DecMate 5	Superbrain Jr.
Beehive Tpr	Fujitsu CP/M86	NEC PC-8001A	Televideo
CDR	IBM CP/M86	Osborne 1	TRS80 CP/M
Cromemco	IMS 5000	Otrona	Visual 1050
DEC VT180	Kaypro II	PMC MicroMate	Xerox 820
DEC Rainbow	Magnolia	Royal/Adler	Zorba

For H37 with Heath CP/M \$59
For C.D.R. BIOS 2.91 \$49
Check for Magnolia version.

CALL OR WRITE FOR CATALOG

PRICES SUBJECT TO CHANGE

Terms: Check or Money Order — Visa, M/C — C.O.D.
 Add \$3 per order for shipping and handling
 California residents add 6% tax

Performance Tuning Your Software

Leo R. Hansen

6771 E. 4th Street
Tucson, AZ 85710

Background

The modern programming languages ADA, Pascal, and Modula-2 offer various ways to write programs. All three of these languages have included capabilities to make the programs "self-documenting". This means that just by reading the code you can determine what is happening, how is it happening, and generally, why is it happening in the program without a separate description document. This is a desirable feature for a language, since most of us love to document programs so well that we save it until last, under the theory that the best is saved until last. Some of these self-documenting features have impacts on the speed of execution of the program and are greatly affected by the way programs are actually executed by the processor. In this article, we will look at the way the 8088 executes programs and how we can take advantage of the hardware in performance tuning our software.

A general purpose microprocessor performs three basic operations in executing a program. These operations are: 1) fetch, 2) decode and 3) execute. The fetch operation "fetches" the instruction or data from memory and puts it in special high speed locations that the central processing unit (CPU) uses for the operations that it performs. Next, the CPU "decodes" or figures out what is supposed to be done with the information that it fetches. Then, after it has decided what to do, it "executes" it. These operations are repeated in sequence until the program runs to completion.

There are ways to improve performance with this type of operation. One way is to have the CPU execute at the highest possible rate. This is one way that the "Cray" machines achieve high performance, they execute instructions in the 10 nano-second (one hundredth of one millionth of a second) range. This speed places tremendous difficulties on the design of the machine and is one reason that they cost \$5 million or more. Another way is to pick up more information on each fetch, so you don't have to perform a fetch as often. This also is a technique used by the "Cray" machines and the Control Data Corp. main frame computers, which were also designed to Mr. Cray. They use a 10 byte word which is 60 bits long. Another way is to use separate fetch and execute operations which anticipate the program flow and try to pre-fetch the instructions for execution. This is what the designers of the 8088 did.

The 8088 used in the Z-100 family has added a little more hardware to the CPU and has modified the normal operational sequence. A separate unit (buffer interface unit — BIU) has been added to perform the fetch which does not execute the instruction. The fetched information is put into a special buffer that is used to feed the hardware that executes the instructions. In the case of the 8088, this buffer is 4 bytes long. The BIU tries to keep this buffer full at all times so that the execution of the instructions is not delayed because instructions are not available. It tries to keep the "pipeline" full of information for the execution unit, hence, this type of design is referred to as a "pipelined" architecture.

The BIU tries to keep the pipeline full by loading the sequential memory locations as soon as space is available. This is fine until a branch or jump in the instruction flow is encountered, then the information that is in the pipeline is not the right information. The pipeline is then flushed and the right information is loaded and executed. Thus, programs with a lot of calls or executed branches run slower than those with fewer calls.

There are ways to take advantage of this "pipelined" design when we code our programs. One way is to keep the number of fetches at a minimum. The question is where and how can we do this? The primary way to do this is to use the smallest possible variable to represent the data we are using. Use single precision math variables, if that is all you need. Another way is to try to write "fall through" code as much as possible. What is "fall through" code? It is code that has no branches or jumps, it just "falls" straight through executing one instruction after another. Now, it obviously is impossible to write pure "fall through" code in any reasonable program, but the number of branches or jumps can be reduced by being aware of what is happening in the program.

Desirable Code Characteristics

The ideal programming language would be your natural native tongue. However, we are not quite there, so what would be a near ideal programming language? Again, one with commands to the compiler (these commands are the programming language) that are close to our mother tongue. The code by itself, with no comments, would present a clear picture of what is being performed and on what variables. Again, we are not quite there,

but we are getting close in some areas. For example, the following Pascal program is almost self documenting:

```
Program Benchmark;
var
  x: integer;

begin
  write('input the value to be counted down to zero');
  read(x);
  repeat
    x:= x - 1;
  until x < 1;
  write('done');
end.
```

The program name obviously is Benchmark. It uses one variable, x, which is defined as an integer type variable. The program prompts for a starting value to begin counting down from. This value is placed in the variable, x, by the read statement. Next, there is a repeating operation, in this case it is the one line equation that subtracts 1 from x and stores the result back in x. Then, a check is made to determine if the remaining value of x is less than 1, if it is, then the message "done" is displayed. The program is then terminated. If the remaining value of x is not less than 1, the process is repeated.

The only things that are not immediately obvious to a non-Pascal programmer are the following: 1) the significance of the term "var" on the second line, 2) what the destination of the write command is, and 3) what the source of the read command is. However, these are logically defined and easily learned. The "var" is obviously an abbreviation for variable. In Pascal, a variable and its type — integer or real, for example — must be known before they can be used. The process of identifying the variables and their types is performed early in the program. In this case, that is the first thing that is done.

Pascal also has a default condition of using the keyboard as the input device and the terminal screen as the output (display) device. Consequently, since no other devices are mentioned in the read or write commands these are the devices used.

The repeat-until loop definition does exactly what it says, it repeats the instructions between the 'repeat' and 'until' statements until the condition after the 'until' is met. Then, the loop is exited.

However, any real program is going to be more complex than this example. A more general version of this example would be the following:

```
Program Benchmark;
var
  x: integer;
  done: boolean;

procedure decrement(var y:integer);
begin
  y:= y - 1;
end;

begin (* main program *)
  done:= false;
  write('input the value to be counted down to zero');
  read(x);
  repeat
    decrement(x);
    if x < 1 then done:= true;
  until done;
  write('done');
end.
```

This generalized version is very easy to understand. The procedure, "decrement", will be used by the main program to perform a decrement operation and is defined before it is used, just like any variables are. The test for the end condition is kept separate from the 'until' statement in order to facilitate a complex completion condition. If the end condition is very complex, it would be handled in a procedure or function probably called "end__check" and would return the boolean variable, done.

Since one of our major goals is to write code that is essentially self-documenting, and this version is a good example of a self-documenting program, it will be the base line program.

Program Versions

V0 — Base Line Version

Definition

```
Program Benchmark;
var
  x: integer;
  done: boolean;

procedure decrement(var y:integer);
begin
  y:= y - 1;
end;

begin (* main program *)
  done:= false;
  write('input the value to be counted down to zero');
  read(x);
  repeat
    decrement(x);
    if x < 1 then done:= true;
  until done;
  write('done');
end.
```

This version, as discussed above, is very easy to understand and is a good example of a structured self-documented program. However, it violates one of our rules for performance tuning our programs. Do you see what the violation is? The call to the decrement procedure on every pass through the repeat-until loop destroys any benefit from the pipeline design of the processor. Since decrement is very short this impact is very large. This is a worst case situation, but since the general philosophy of using function and procedures is to limit them to the performance of one general operation, you shouldn't have really large procedures. Certainly, longer than one line, but according to most people working in this area of the programming field, probably less than 100 lines and preferably a maximum of about 65 lines.

This program also uses a parameter passing technique known as parameter passing by reference. The significance of this is that the parameter that is passed from one procedure to another, in this case x, is actually updated by the procedure that it is passed to. Thus, when control is returned to the main program, the value of x has been decremented by one. This is necessary for this particular version of the program to work correctly. The other method of passing parameters is by value, which will be discussed later in the article. Another method of using the same variable with free access to all procedures is the next version that will be discussed.

Performance

The base line version required .048 seconds to execute if x is 1000. This will be the base line value for x in all of the variations that are studied here.

V1 — Version One

Definition

```
Program Benchmark;
var
  x: integer;
  done: boolean;

procedure decrement;
begin
  x:= x - 1;
  if x < 1 then done:= true;
end;

begin (* main program *)
  done:= false;
  write('input the value to be counted down to zero');
  read(x);
  repeat
    decrement;
  until done;
  write('done');
end.
```

This version is different from the base line program in only one aspect, which is the use of a global variable instead of a passed parameter. Global variables are just slightly more efficient than passing parameters. It is clearer what variables the procedure uses if they are passed, and hence, this is the preferred method of coding.

Performance

By using a global variable instead of passing a parameter improved the performance slightly, the required execution time is .041 seconds. This is not enough to be worth the loss in clarity to me and I would stay with the base line version instead of this one.

V2 — Version Two

Definition

```
Program Benchmark;
var
  x: integer;
  done: boolean;

procedure decrement;
begin
  repeat
    x:= x - 1;
    if x < 1 then done:= true;
  until done;
end;

begin (* main program *)
  done:= false;
  write('input the value to be counted down to zero');
  read(x);
  decrement;
  write('done');
end.
```

This version is slightly misleading when you read through the code, however, when compared to the base line program. Reading the "main program" portion of the program gives the false impression that decrement is executed only once. It is true that decrement will only be called one, but it will not return until x is less than 1. A better name for the decrement procedure in this case would be "zero_it" or "count_down", something that more accurately describes what the called procedure does.

Performance

The performance is affected drastically by this change in the logic. The repeated procedure calls are avoided, thereby saving

the overhead associated with a call. If the name was changed to more accurately reflect the actual operation of decrement, this would be an improvement over the base line version. The time required to execute version V2 is .023 seconds.

V3 — Version Three

Definition

```
Program Benchmark;
var
  x: integer;
  done: boolean;

begin (* main program *)
  done:= false;
  write('input the value to be counted down to zero');
  read(x);
  repeat
    x:= x - 1;
    if x < 1 then done:= true;
  until done;
  write('done');
end.
```

This version is clear and acceptable only because it is a simple program. It is included only for the purpose of comparison with the base line program. It is essentially the same as Version 2, without a procedure call.

Performance

The performance is also essentially the same as Version 2 requiring .023 seconds.

V4 — Version Four

Definition

```
Program Benchmark;
var
  x: integer;
  done: boolean;

procedure decrement(y:integer);
begin
  repeat
    y:= y - 1;
    if y < 1 then done:= true;
  until done;
end;

begin (* main program *)
  done:= false;
  write('input the value to be counted down to zero');
  read(x);
  decrement(x);
  write('done');
end.
```

This version uses parameter passing. This is parameter passing by value. Which means that the value of x is passed to the procedure decrement. Therefore, any changes to x in decrement will not be passed back to the main program. As a consequence then, decrement must perform all required operations on x to complete the process. Passing parameters by value is useful when only the initial value of the variable is needed by the called procedure and the final value will not be needed by the calling program. When parameters are passed by value, a copy of the variable is made and that is what is used by the called procedure to operate on the variable. This is fine for small variables, but a large array or long strings would use extra memory each time the procedure is called. Consequently, parameter passing by value is usually avoided for all variables except integers and real numbers.

Performance

The performance, however, is essentially the same as the others where the repeat-until loop is included in the called procedure. The time required to execute this version is .023 seconds.

V5 — Version Five

Definition

```
Program Benchmark;
var
  x: integer;
  done: boolean;

procedure decrement(var y:integer);
begin
  repeat
    y:= y - 1;
    if y < 1 then done:= true;
  until done;
end;

begin (* main program *)
  done:= false;
  write('input the value to be counted down to zero');
  read(x);
  decrement(x);
  write('done');
end.
```

This is the same as the previous version with the exception that the parameter is passed by reference. This is included for comparison between parameter passing by value and by reference. Passing parameters by reference is useful when the final value of the variable is needed by the calling program. When parameters are passed by reference, the master variable is the one that is modified. Therefore, it is essential that the called procedures operate correctly. They should probably contain special error detecting code and range checking on the variable to prevent unexpected results which will permanently destroy the data.

Performance

The performance, however, is only slightly longer than the others where the repeat-until loop is included in the called procedure. The time required to execute this version is .027 seconds.

V6 — Version Six

Definition

```
Program Benchmark;
var
  x: integer;
  done: boolean;

function decrement(y:integer): integer;
begin
  y:= y - 1;
  decrement:= y;
end;

begin (* main program *)
  done:= false;
  write('input the value to be counted down to zero')
  read(x);
  repeat
    x:= decrement(x);
    if x < 1 then done:= true;
  until done;
  write('done');
end.
```

This is a version of the base line program using a function call instead of a procedure call. The function always returns one vari-

able, in this case it is the updated value of x after subtracting one from it. The use of a function sometimes presents a clearer picture of what is happening in a program and in those cases would be the preferred method of programming.

Performance

There is a slight price to pay for the use of a function, however, the time required for this version is .060 seconds.

V7 — Version Seven

Definition

```
Program Benchmark;
var
  x: real;
  done: boolean;

procedure decrement(var y:real);
begin
  y:= y - 1.0;
end;

begin (* main program *)
  done:= false;
  write('input the value to be counted down to zero');
  read(x);
  repeat
    decrement(x);
  until done;
  write('done');
end.
```

This is a version of the base line program which uses a real variable for the count down variable. Use of this real variable instead of an integer doubles the size of the internal storage requirements from 2 bytes to 4 bytes, and consequently, doubles the number of fetches that are required to access the data. The message here is very clear, don't use reals unless you have no choice. If you can use integer numbers between -32767 and +32767 or between 0 and +65534, then don't use reals. If your application requires numbers outside these ranges, then you don't have a choice, you must use real numbers. Sometimes though you can scale and manipulate the numbers so no intermediate results are outside of these ranges, but the logic gets very complex. However, if you really need the speed, and you can handle your application by scaling, the effort to implement scaling will probably be worth it.

Performance

The time required for the real version of the program is .440 seconds. Now do you see why you should avoid real numbers at any cost?

V8 — Version Eight

Definition

```
Program Benchmark;
var
  x: real;
  done: boolean;

procedure decrement(var y:real);
begin
  y:= y - 1
end;

begin (* main program *)
  done:= false;
  write('input the value to be counted down to zero');
  read(x);
  repeat
    decrement(x);
```



```

    if x < 1 then done:= true;
until done;
write('done');
end.

```

This is a version of the real program above which uses a real variable for the count down variable. Only in this case, we tried to recoup some of the speed by not using reals all the way. Notice in the actual decrement equation, the number subtracted is an integer one (you can tell because it doesn't have a decimal point). The message here is very clear, don't mix variable types in an equation. This is even worse than all reals. The reason is that each time the integer must be converted to a real. This takes longer than just using a real in the first place.

Performance

The time required for the mixed real/integer version of the program is .680 seconds. Now do you see why you should avoid mixing different types of variables.

V9 — Version Nine

Definition

```

Program Benchmark;
var
  x: integer;
  done: boolean;

procedure decrement(var y:integer);
begin
  y:= y - 1; (* 1st decrement *)
  y:= y - 1; (* 2nd decrement *)
  ...
  y:= y - 1; (* 1000th decrement *)
end;

begin (* main program *)
  done:= false;
  write('input the value to be counted down to zero');
  read(x);
  decrement(x);
  write('done');
end.

```

This version, which is the brute force approach, is designed for speed without regard for how much memory is used. This is the pure fall through code method. This version maximizes the benefits from pipelined structure. However, it is only useful if you can determine ahead of time how many loops are going to be needed so you can build exactly that many statements into the program. This version is also very clear on what is going on, it is just very long in this particular case. I used the cut and paste features of my editor to create the procedure with 1000 statements in it.

Performance

The fall through version required .008 seconds.

Summary

The following table summarizes the results of the study.

Version	Description	Time, Seconds Per 1000 Loops
V0	Base line	.048
V1	Global variable	.041
V2	Loop in called procedure	.023
V3	No procedure call	.023
V4	Variable passed by value	.023
V5	Variable passed by reference	.027
V6	Function call	.060
V7	Real variables	.440

V8	Mixed real and integer	.680
V9	Fall through	.008

As you can see there is a wide variation in the results, from 8 milliseconds to 680 milliseconds, a factor of 85 to 1. By observing two basic rules, you can drastically affect execution speed while still maintaining readability of the code. The primary rule is don't use reals unless you have to. This will not affect the readability of the code in any way. The second rule is write "fall through" code as much as possible. You do this by 1) avoiding loops to perform fixed numbers of operations and 2) by avoiding procedure or function calls in loops.

Conclusion

Every person's programming style is a little different, however, with the information in the table, you can evaluate the impact of various options on your particular application. If your application is an interactive one that responds so quickly that the human operator can't tell the difference between these options, then ignore them when you write your code. Don't ignore readability though, regardless of speed considerations. If your application is affected by these options, then I hope they are useful to you in your programming.

Happy tuning.



Reasonably-Priced COMPUTER FORMS

- Checks, Invoices, Labels, Letterheads, Envelopes, etc.
- Small Quantities and Up
- Prompt Free Delivery

Why not send us a sample and ask for a price?

LACLEDE PRINTING COMPANY

10702 Manchester Road, St. Louis, Missouri 63122. (314) 821-3313



Did you know that HUG has a small business accounting package? Its unique name is **Accounting System**. As with most HUG software, it is user-friendly, double entry, can handle up to 999 separate accounts during any calendar year, and is available for ANY Heath/Zenith computer with a double density disk drive. The different versions available are as follows: **CP/M — P/N 885-8047-37, Z-DOS/MS-DOS — P/N 885-8048-37, MS-DOS — P/N 885-8049-37.**



WILDFIREtm

An INCREDIBLE PERFORMANCE IMPROVEMENT from Software Wizardry

Wildfire has spirit! It will spur your Z-151/161 to run faster than the 8mHz Z-158! Wildfire uses the reliable NEC-V20 chip. It outdoes the 80286 processor chip additions for compatibility with the IBM(tm)PC and costs up to \$600 less!

The heart of Wildfire is a daughter board that mounts on your processor board, saddled between it and the video board. Several higher-speed chips are included to replace socketed chips on your processor board. A high speed/low speed switch that mounts on your machine's front panel allows you to trot at normal speed, or break into a real gallop when you really want to ride!

Software Wizardry brings you Wildfire, complete with front panel, installation instructions, and reliable daughter-board design, for only \$249!

If you order now, you can try Wildfire with a 15-day guarantee. If not satisfied, you can return it within 15 days for a full refund.

Contact your local dealer, or order direct from Software Wizardry by calling (314)724-1738. Dealer inquiries invited.

Software Wizardry
THE MAGIC TOUCH

1106 First Capitol Dr.
St. Charles, MO 63301
(314)724-1738

Also from Software Wizardry...

For the Z-151/152/161

RamPal

The Original Memory
Upgrade Kit **\$39.95**
for only...

RamPal allows you to put up to 640K RAM on your standard memory board. This simple one-chip replacement requires no modification to existing hardware. One bank of 41256 chips is required, two for full 640K memory.

Z-100 Upgrade Package Deal

Reliable Memory/
Speed Upgrade **\$299**
for only ...

Our special package includes the RamPal 100 Memory Upgrade, 27 RAM chips, and the 8mHz Speed Upgrade. Now you can have compatibility at a price you can afford. Order part number FCC-MEM100FULL.

8mHz Speed Upgrade

Reliable Z-100 Speed Upgrade
for true 8mHz **\$149**
Operation.

This upgrade will bring your Z-100 up to the operating speed of the latest Z-100 version. Plus, you can switch down to 5mHz for programs that require it. Includes over 25 chips, circuit board, and back panel for 5 or 8mHz operation.

On The Leading Edge

William M. Adney

P.O. Box 531655

Grand Prairie, TX 75053-1655

DOS Error Messages, '241 Hardware Problems

Some years are worse than others, and so far, this one has turned out to be a real beaut! I have seriously considered renaming this place "DisasterCentral" — not because I specialize in handling disasters, but because a disaster of one kind or another seems to centralize here. In fact, this is the first REMark article that I have written in 1987 which may give you a clue as to what kind of year it's been so far. That also explains why I have missed the last two issues of REMark as far as this column is concerned.

As this year started, I was in Indiana helping to take care of my mother who was extremely ill with emphysema. That took a few weeks, and since that trip was not scheduled, it played absolute havoc with all of my writing deadlines. Fortunately, my mother is much better now. But at least one good thing came out of all that. I did manage to stop smoking last November. I won't bore you with stories about my climbing the walls as a result, but I think I have managed to stop for good. Much of the credit for that goes to a number of students in my last semester's classes for their support and encouragement. It wasn't easy.

As far as this column is concerned, it is more to the point to tell you about a rather odd problem (and its solution) that I have had with my '241.

Computer Problems

For one reason or another, I have gradually been moving some of my writing projects

over to the '241. Most of that movement involves files related to my FlipFast books, but I have also started to use the '241 for original writing projects. One such article appeared in the first issue of Kit Builders' Journal.

Here's what happened. I wrote the article on the '241 as usual. No big deal about that since I saved the article on the hard disk and took the usual backups. Since KBJ also likes to have articles submitted on disk, I copied the article to a 5-1/4" disk and sent it to Rick Simpson, the editor of KBJ. Rick called me a week later to tell me that the disk was BLANK — no files. That was really strange, since the last thing I do is check a disk to make certain that the file was copied correctly. When I use my H-100 with CP/M WordStar for writing, I check the directory on the disk, as well as use the TYPE command to check the file. Since I used Microsoft Word version 3.0 to originally write the KBJ article, I only checked the directory on the disk, since trying to TYPE a Word file results in all kinds of wondrous displays and beeps due to the unique non-ASCII file format. The DIR command showed that the file was on the disk, so I sent it to Rick.

But the fact remained that the disk was blank. How did that occur when I actually checked the disk with DIR? The file was listed correctly in the directory, so it was copied properly — Right? WRONG, of course, but how could that possibly hap-

pen? Some investigation revealed a fact that I "knew", but had forgotten.

The key to that DIR situation is that DOS only reads the actual disk directory after about two seconds have elapsed. If you enter a DIR command before that two second limit, DOS assumes that you could not have possibly changed the disk in the drive, and therefore, it uses the "disk directory image" that it has in memory. It is no big deal for a touch typist to be able to beat that time. I was watching for completion of the COPY command, and entered DIRA: as soon as it had completed successfully. And the DIR showed the expected file listing, since it actually displayed the memory image of the disk directory, not the actual directory on disk.

The moral of this story is to always wait at least two seconds before checking the directory on a disk to make sure that a file was actually copied. That explains how I was able to "verify" a blank disk, but how did that happen in the first place?

A Hardware Problem?

One of my technical experts told me that he had had a similar problem that was due to a bad I/O board in the '241. I won't pretend that I understand how that was possible, since I am not an expert on the '241 hardware. In any case, we had a deadline to meet for the KBJ article, so Rick Simpson helped me get a new I/O board (from the factory) so that I could get the article submitted on disk. I installed the new I/O

board, and that did not fix the copy problem. The '241 still would not copy a file to the A drive, which is the 360 KB drive in my system. You may recall that I suggested swapping drives because of problems with copy protected software in a previous column. That is, the A drive should be the 360 KB drive and the B drive should be the 1.2 MB high density drive.

When all else fails, I can always rely on help from one of the local Heath/Zenith Electronics Centers, so I made the 27 mile trip to the Dallas store. After checking out a number of things, we found out that I simply had a bad 360 KB disk drive that was causing some rather odd problems, including the copy problem. So we replaced the drive with a new one, and that fixed the problem — Right? WRONG AGAIN!

Because the '241 operates at 6 MHz with no wait states, it turns out that some 360 KB drives don't work properly in that system. I presume that the same is true for the '248, but I don't know that for sure. In any case, how do you know what drive works and what doesn't? Simply TRY it! Apparently, some brands and types of 360 KB drives work fine and others don't. I hope that all Heath/Zenith Electronics Centers and Zenith repair facilities are aware of this (the Dallas store was), but I thought you should know about it, just in case.

At this point, I had a new I/O board and a new 360 KB disk drive in the '241, so I was able to get the file copied and off to Rick Simpson. Everything was back to normal and working correctly — Right? Well, not exactly . . .

A Strange Problem

For the most part, my '241 was now working. At least I could copy files to a floppy disk drive. But I had another type of problem that only occurred with two application software packages. I could not print a SuperCalc spreadsheet or a Reflex database. Both packages would display a "Printer out of paper" error message and refuse to print anything on my C. Itoh C-310 printer. And I could not get the Microsoft Bus Mouse to work at all, with Reflex, although it worked fine with Word and other software.

Since I had just purchased the C-310, I suspected some kind of a hardware problem with the printer or cables, but that did not make any sense. The printer worked just fine with Microsoft Word, the DOS PRINT command, and all of my other software, except for SuperCalc and Reflex.

I had just upgraded SuperCalc to version 3, release 2.1 as a result of an update card

from Zenith. I had previously used version 3 release 2.0 for printing, so I checked that release. Oddly enough, 2.0 printed a spreadsheet with no problems. But 2.1 would not. So I called Zenith Software Consultation to report the problem, and gave them all the appropriate data to duplicate it. One of the consultants promptly called me back to report that they were not able to duplicate the problem. Their analysis was that I MUST have some kind of a hardware problem. My response to that was: "Oh YUCK! — not ANOTHER One!!!" (My students tell me that "yuck" is an appropriate response to this kind of situation.) Now I knew beyond any shadow of a doubt that it was going to be one of those years!

I suspected that the new I/O board might be the culprit, so I swapped it with the old one. No change — SuperCalc 3/2.1 and Reflex still would not print anything. So it was back to the Heath/Zenith Computer & Electronics store . . .

In order to check everything, I took my entire system, including the printer and cables. We checked my software on a store '241, and both SuperCalc and Reflex worked properly with my C-310 printer and cable. That eliminated everything, except the '241 hardware. It was time to play musical boards.

I swapped out the CPU board, and there was no change. Then I swapped the I/O board, and that FIXED both SuperCalc and Reflex. In addition, the mouse now worked with Reflex. Obviously, the I/O board that I received from the factory had the identical problem as the original that came with the '241.

Whatever the problem was, the other interesting point is that the Z-200 Disk Diagnostics reported that everything was fine. I ran all the detailed tests on both the CPU and I/O components, and everything checked out okay. Despite the satisfactory checkout by the diagnostics, the I/O board (both of them) still had a bad component that the diagnostics failed to find. Although I still believe that the Z-200 diagnostics are quite good, I am realistic enough to know that they cannot check out every possible hardware problem in the system. Perhaps that also explains how a bad board could be obtained direct from the factory.

All of this took place over a period of weeks. Perhaps the worst problem created by all of this is that a number of my hard disk backups were bad. That is, they were absolutely blank. Whatever component failed on the disk drive and/or I/O board,

DOS failed to detect that there were any problems. As a result, I actually "lost" two files since the disks are blank. Fortunately, I have printouts of both — one was a program and the other was a text file. But I am not looking forward to reentering all of that by hand.

The Moral Of The Story Is . . .

Always check the files written to a disk, particularly backup files. Even though I keep two sets of running backups for my hard disk, both were bad because of timing. Some specific backups for data (like my program) were also blank for the same reason. Fortunately, neither file was critical, so I can reenter that stuff in a leisurely fashion.

The most disturbing thing about this was that DOS failed to detect an error when writing to a bad disk drive, but I know that hardware problems can cause strange situations. And the fact that the Z-200 Disk Diagnostics also failed to detect a problem on the I/O board is cause for concern.

From now on, you can bet that I will wait a couple of seconds before checking a disk drive for a successful file copy. And I will also take a couple of minutes to check all backup disks, too. You might want to consider doing that as well.

As a result of all this, I thought there was another aspect of the DOS that would be useful information for you.

DOS Error Messages

If you use your computer very much, there are few things that you will run into as often as the DOS error messages. Like most error messages that are generated by operating systems, these messages tend to be rather cryptic, and many are not very helpful to say the least. Some are downright obscure, and more than a few are actually misleading.

It is no wonder that many people believe that computers take on a life of their own when we try to use them. As I mentioned a couple of months ago, at least one of my students thinks that DOS (specifically PC-DOS) is a good example of user hateful software. It is too bad that the design of DOS makes it unlikely that it will ever be user friendly in its present form.

But error messages are a way of life it seems, and even experts manage to find ways to cause them. Since many of the error messages tend to use terms that are unique to computer systems, I thought it might be worthwhile to try to explain some of the common, and obscure, error messages that

you can get during the operation of MS-DOS or PC-DOS. Since it does not matter (for the most part, anyway) whether you are using MS-DOS on a Z-100 or a PC compatible, I will use the term "DOS" to indicate the operating system.

Types Of Error Messages

I have found it helpful to divide the various DOS error messages into four distinct types: BIOS, Device, Syntax, and Command specific. This distinction is based on exactly where in the DOS the error message is generated.

The BIOS error messages are physically located (i.e., programmed) in the Basic Input/Output System code developed by each manufacturer for a specific computer system. The BIOS is a special type of "program" that performs the interface or translation between the computer's hardware and the system software. It is a critical type of error message which means that, when one occurs, you can expect to lose data.

The BIOS error messages are contained in a system/hidden file that has a generic name of IO.SYS. The IBM PC series and many PC compatible computers generally use a file name of IBMIO.COM or equivalent. In addition, the IBM PC and compatibles also have a ROM-BIOS that contains error messages. The exact file name and location of the error messages are not important to this discussion.

BIOS error messages are usually generated as the result of a major failure in DOS, such as being unable to read the boot disk to obtain system files. Recovery from a BIOS error message usually requires rebooting of the system and the loss of any data that was not saved to disk. In extreme cases, the IBM PC and compatibles may require a power-off/on sequence to reset the computer hardware. In the Z-100, a CTRL-RESET sequence performs that function with no problem.

Perhaps the most famous Z-100 BIOS error message is the infamous "WILD INTERRUPT". In general, that means you tried to run a program intended for an IBM PC on a Z-100.

My standard recovery procedure from a BIOS error message is simple. Panic about the possible loss of data is the first step. Then, I simply do a CTRL-RESET, and go back to whatever testing I was doing. It is worthwhile to note that I have never seen the WILD INTERRUPT error message during the operation of a normal Z-100 program.

The second type of error messages are the DEVICE error messages. These are elusive

little devils. In general, they are generated as a result of an attempted illegal access to a device for one reason or another. In this context, I am specifically talking about a problem in reading from or writing to a device.

Device error messages can be easily identified by their unique "Abort, Retry, Ignore" error message. This type of error will cause the DOS to stop performing the requested action and can result in the loss of data. Because of the possible loss of data, you should always identify the problem so that you can avoid the problem in the future. The Device error messages are generally located in the command interpreter, COMMAND.COM.

The third type of error message is the SYNTAX type of error message, which is caused by incorrect command line syntax for a built-in DOS command, such as DIR, DEL, CHDIR or PATH. These error messages are also generated by the command interpreter, COMMAND.COM. They are generally not as disastrous as the first two types, since they are usually caused by a simple command line error that normally does not cause loss of data.

The last type of error message is what I call the Command Specific errors. These error (and sometimes information) messages are generated by a specific DOS disk resident command, such as FORMAT, DISKCOPY, or BACKUP. From a technical perspective, these error messages are physically coded in the disk resident program. As a matter of fact, I have developed the ERROR MESSAGE section for the FlipFast books by reviewing the actual error messages in each and every DOS program. After obtaining the exact wording of the error message, I then did some extensive testing on each command to determine what kinds of "mistakes" cause that message to appear.

BIOS Error Messages

As I was working on the new FlipFast book for version 3 MS-DOS, I got the bright idea that I should include all of the BIOS error messages, including those generated as a result of the ROM code. After some investigation, I found that there are so many ROM versions around that it simply did not seem to be a good idea, since there was no effective way to ensure the accuracy of the messages because of frequent changes. I, therefore, decided to document only those that were contained in the BIOS file, such as the WILD INTERRUPT that I mentioned before. This should not be a problem for most Heath/Zenith users, since vir-

tually all of the ROM-based error messages are documented in at least one of the technical manuals for a specific computer. Perhaps the most distinguishing characteristic of the BIOS error messages is that they occur VERY infrequently for most users. Since many users never see these messages, it did not seem to be worth a couple of weeks' time to investigate each one.

Device Error Messages

On the other hand, Device error messages are a relatively common occurrence for most computer users. They are generated by the DOS when an error is found in reading from or writing to any of the hardware devices (e.g. printer, disk drives, etc.) on your system. DOS will pause to allow a selection of one of the available responses or it will redisplay the prompt until a valid response is entered. Device error messages take the following form:

```
<type> error <function><device>
Abort, Retry, Ignore?
```

The <type> errors are quite specific and each individual error message is listed in the FlipFast book under the COMMAND command. The <function> error indicates a problem in "reading" or "writing". The <device> is the name of the problem device; i.e., "PRN" for the system printer or "drive B" for a disk drive.

One example of this kind of error is the usual one that many of us see on occasion:

```
Not ready error reading drive B
Abort, Retry, Ignore?
```

This specific message nearly always means that we incorrectly entered the drive letter (i.e., nothing more than a typo), the disk drive door is open or that the disk was improperly inserted for a command, such as DIR B:.

When a message of this type occurs, NEVER change the diskette in a drive before entering a response to this error message. It is highly likely that you will destroy any files on the new diskette. Responses to these error messages should usually be made in the following order:

- R — Retry the read or write function again after correcting the problem. This is the Microsoft preferred response, and I also recommend that you always use the Retry option first.
- A — Abort the program or operation in progress. This response will normally return you to the system prompt, but it may result in the loss of data.

- Ignore the error and continue the program or operation in progress. Use caution when selecting this option; data is often lost since you have instructed DOS to ignore the error.

For those of you who are technically inclined, these error messages (actually numeric codes) can be directly associated with device driver problems. As I mentioned before, each of these specific error messages is included in the FlipFast books in Section 4 under COMMAND.

Syntax Error Messages

Syntax error messages are those vague and obscure ones that are generated by COMMAND.COM (or any disk resident program) as a result of any number of problems. If I were to provide nominations for the absolute worst error message in DOS, it would be a toss-up between "Bad command or filename" or "Invalid parameter". Both of them are obviously quite vague. And the "Bad command or filename" message can take some time to figure out what caused it in the first place.

One of the specific problems with these error messages is that they are a "catch-all" for problems that a programmer was simply too lazy to do better. My solution for that is to fire a programmer who perpetrates that kind of problem, since it is an incredibly lazy, unprofessional, and clumsy approach to programming. I should specifically note in passing that I have not found any of that kind of error message that was developed by any of the programmers at Zenith Data Systems. I know a couple of them and have been impressed by their knowledge, dedication, and professional approach to programming. This particular problem is one of Microsoft's, since PC-DOS has the identical error messages.

"Bad Command Or filename"

Let us take a look at the possibilities for the "Bad command or filename" message first. Because of the clumsy message, I will split the message into two parts and discuss the "Bad command" part first.

This one is a real dilly. When you see the "Bad command", it means the previous command (e.g., FORMAT) was not found in the current working directory or subdirectory or simply stated: "Command not found". The obvious problem is that you may have simply made a typo in the command line. Even if DOS does not have any difficulty locating the FORMAT command, I suspect that DOS will not find the FORMAT command. Be sure that you check the spelling of the command first.

If you have used the PATH command to enable DOS to search other directories, it also means that DOS could not find the command in any of those directories either. The first thing to do is to use the DIR command to check the current directory to verify that the disk resident command actually resides in the directory. If you have used the PATH command, then simply enter PATH to verify that DOS "remembers" the path and drive specification. While that sounds simple, that problem took me about a half hour to find one time, since I normally provide a PATH specification in my AUTOEXEC file. I was doing some testing, and forgot that I had deleted the AUTOEXEC for test purposes. There was no path defined in my system. In retrospect, that sounds like a simple problem, but it was quite difficult at the time.

So far, we have two possibilities for the "Bad command" part: A simple typo on the command line, or DOS could not find the command in the current directory/path. Now, let us take a look at the possible problems with a bad file name.

A "Bad filename"

There are three possibilities for this error message: A misspelled file name, the file name does not exist in the current or specified path, or the file name contains invalid characters.

Like the previous explanation, this error message may also mean that you have simply made a spelling error in the file name on the command line. You can use the DIR command to check the spelling of all file names.

If a path is not specified in a command line (or with the PATH command) for a file, DOS will always expect to find the file in the current working subdirectory first. Otherwise, DOS will attempt to find the file as defined in the path specification preceding the file name. Regardless of how the command line is constructed, you can again use the DIR command to verify the existence of the specified file name in the current subdirectory or other path.

Last, but certainly not least, is the fact that you may have used invalid characters in a file name. Remember that all of the special characters used in a command line have special meanings and cannot be used within file names. Characters that cannot be used within completely specified (i.e., unambiguous) file names include the following:

```
" < > , ; : = [ ] + !  
 \ / space tab . (period)
```

Other special characters have unique meanings, such as the wild card characters, * and ?. And, of course, the period is used to separate the file name from the file type.

Some application programs, like word processors and spreadsheets, have additional restrictions on file names. For example, at least one word processor does not allow the use of hyphens in a file name. My recollection is that Samna does not allow a number of normally valid characters to be used in a file name, but it may be another one. Samna, in particular, has so many strange ways of doing things that I stay as far away from it as I can. Maybe they fixed all of that when they removed the copy protection.

I have picked on the "Bad command or filename" error message to show how ambiguous a message can be. Any error message that has at least five possible interpretations does not help a user determine a problem very well — particularly, a new user. No wonder a lot of people do not like computers when programmers use this kind of rather obtuse logic to identify a problem. It is too bad that many people blame computers for this kind of problem when it is really caused by an individual programmer. But let us take a look at another very poorly worded error message.

Invalid Parameter

My experience is that a lot of people seem to have difficulty with this error message. Although I am used to that word, I think that the biggest part of the problem with that error message is that many computer users do not understand what a parameter is. So I will define it before I go on.

A Parameter is a variable item of information that controls the operation of software (i.e., a program) by defining program functions, selecting options, establishing limits or otherwise modifies program operation. Although this is a general definition that applies to computers, it specifically includes all information provided in a command line (excluding the command itself), such as drive identifiers, path names, file names, switches, etc.

For example, consider the following command: DIR B:*.COM/W. Parameters in this command line include the B: (drive identifier), *.COM (file name), and the /W (optional switch for the wide directory display). Each of these items or elements of information must be exactly correct for the command to function properly. If one or more of these items is incorrectly entered, the "Invalid parameter" error message may be displayed. As a matter of fact, the easiest

way to get the "Invalid parameter" message is to simply omit the colon following the drive letter so that the command becomes: DIR B *.COM/W. DOS sees that there is more information on the command line, but does not know how to process the command since the colon is missing. Is "B" a drive letter, file name or what?

This error message can occur at unusual times when the command appears to be correct at first glance. Be sure that the character following the drive letter is a colon — it is very easy to enter a semicolon instead.

I think this error message could have been improved by replacing it with some specific error messages. Messages like "Invalid switch" or "Invalid file name" would be much more helpful, even though some of the logic would have been more complicated to program.

Command Specific Error Messages

In addition to all of the error message types that we have discussed so far, DOS commands can also generate specific error messages based on the command input. The FORMAT command, for example, may display a "Format failure" or "Insufficient memory for system transfer" error message under certain conditions. These error message literals are coded directly in a specific program and are not part of the DOS internal error detection and display scheme. Each disk resident program — FORMAT, DISKCOPY, etc. — must check for possible errors and display an appropriate error message. That is one reason that you may find differences in error messages between the Zenith MS-DOS and the IBM PC-DOS.

Without going into the details of everything, Microsoft generally furnishes virtually all of the source code to original equipment manufacturers (OEMs) except for the BIOS. In some cases, it is the manufacturer's responsibility to customize the code for some DOS programs (like FORMAT), so that they will run on a specific computer configuration. In other cases, such as the MORE filter, there is no reason for the manufacturer to change anything, since that code is "standard". Programs that must be customized are part of the DOS changes that each manufacturer must make, and this explains why some programs (specifically FORMAT) run a little differently depending on whether you are running the Zenith MS-DOS or IBM PC-DOS.

In the particular case of the FORMAT program, it appears that Microsoft released some bad code in early releases of the version 2 software. Some people have blamed Zenith for difficulties in formatting hard disks on the PC series computers even though IBM had similar problems. For those of you familiar with the technical details, I am talking about the "famous" 64K boundary problem. Even Microsoft makes mistakes, although I think that their biggest one to date is the premature release of the long awaited MS-DOS Technical Reference Encyclopedia. There are so many mistakes in it that it has been "re-called" by Microsoft — not a very good recommendation for a book that sells for \$134.95. In case you are wondering, I am returning mine for a refund.

Closing REMarks

There are a number of items that I am waiting for right now, so we will see what

develops for next month. I presume (hope!) that things are somewhat under control at this point. And I trust that the rest of the year will be better because of the way it started off.

I'll be glad to answer any of your questions about information in this column if you enclose a stamped, self-addressed envelope with your letter. A business sized (#10) envelope is best, since my replies are sometimes quite lengthy.

Products Discussed

Software

HyperACCESS (PC only) (PM-160)	\$149.95
MS-DOS Version 3.2 PC only (OS-63-32)	\$150.00
Programmer's Utility Pack (CB-3163-30)	150.00
Disk Diagnostics Z-100 only (CB-463-13)	\$ 79.00
Z-150 only (CB-5063-28)	79.00
Z-200 only (CB-3163-31)	79.00
PC SuperCalc3 (SC-5063-3)	395.00

Hardware

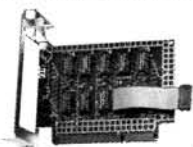
Advanced Personal Computer (HS-248)	\$2499.00
20MB Winchester (ZD-200)	1199.00
40MB Winchester (ZD-400)	1699.00
Winchester Cable (Z-417-1)	20.00

Heath/Zenith Computer Centers
Heath Company Parts Department
Hilltop Road
St. Joseph, MI 49085
(800) 253-7057 (Heath Catalog
orders only)



HEATH/ZENITH 88, 89, 90 PERIPHERALS

16K RAM EXPANSION CARD



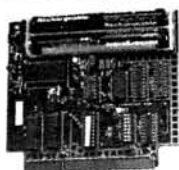
Only \$65.00
Shipping &
Handling \$5.00

2 PORT SERIAL/3 PORT PARALLEL I/O CARD

Price
\$199.00
2nd Oper.
System
Driver
\$25.00
Ship. &
Hdlg \$10



REAL TIME CLOCK



Price \$130.00
with
Batteries
Shipping &
Handling \$5.00
\$114.00
w/o Batteries



PRICES ARE LESS SHIPPING &
TAX IF RES. OF CALIFORNIA.
MAIL ORDER: 12011 ACLARE ST.
CERRITOS, CA 90701
(213) 924-6741
TECHNICAL INFO / HELP:
8575 KNOTT AVENUE, SUITE D
BUENA PARK, CA 90620
(714) 952-3930

TERMS & SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE
VISA & MASTER CARD GLADLY ACCEPTED



Want New And Interesting Software?
Check Out HUG Software

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

*** CHIP SPECIALS ***

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

64K Dynamic RAM, 150 ns\$1.35 each
256K Dynamic RAM, 150 ns\$3.30 each
256K Dynamic RAM, 120 ns\$3.78 each

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

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

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

SmartWatch from FBE Research. Installs in ROM Socket on CPU Board in Zenith computer series Z-100/150/148/158. 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 (formatted) drive \$448.00. May be installed in Z-148 using an Expansion Card sold below.

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

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

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

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

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

SC-837-1 SuperCalc \$195.00 \$40.00

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

INTERNAL DUAL DRIVE SETUPS. Includes two half height double sided disk drives and all hardware and connectors required to mount inside your H-89. Steel mounting shield/case included.

MITSUBISHI MF501 Setup, 48 TPI, 6 MS seek, 320K\$262.00
MITSUBISHI M4853 Setup, 96 TPI, 3 MS seek, 640K disks ..\$269.00

*** HALF HEIGHT FLOPPY *** DISK DRIVES

MITSUBISHI M501 5.25" 48 TPI DS/DD 320K/360K\$105.00
MITSUBISHI M504 5.25" 96 TPI DS/DD 360K/1.2 MEG ..\$152.00
MITSUBISHI M4853 5.25" 96 TPI DS/DD 640K\$109.00

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

*** SEAGATE HARD DISK DRIVES ***

ST-225 20 MEG Winchester Hard Disk\$339.00
With Western Digital Controller & Cables\$424.00
ST-238 30 MEG, Requires RLL type controller\$363.00
With Adaptec RLL Controller & Cables\$468.00
ST-4038 30 MEG High Speed for Z-200\$619.00
ST-4051 40 MEG High Speed for Z-200\$739.00
ST-4096 80 MEG High Speed with Software\$1425.00

PAYLOAD COMPUTER SERVICES

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



Please **MAIL** or **PHONE** your order today and expect prompt service. **MASTERCARD** and **VISA** gladly accepted with no additional charge. All hardware carries a 90 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.

Convert Your Z-100's Keyboard To Dvorak

Dr. Glenn F. Roberts

7734 Marshall Heights Court
Falls Church, VA 22043

As many people now know, the standard "Qwerty" keyboard (nicknamed after the first 6 letters on its top row) was deliberately designed to be slow and inefficient, since early typewriters tended to jam when typists typed too fast. A superior keyboard layout was designed by August Dvorak in the 1930's, but the "Qwerty" layout was so firmly entrenched in the minds and habits of American business that Dvorak's new approach has never been widely adopted.

Recently, however, there has been renewed interest in the Dvorak layout because of the ease with which it can be implemented on today's microcomputers. If you have a Z-100 computer, you can set up a Dvorak layout fairly easily using the keyboard remapping function of the FONT program provided by Zenith with the MS-DOS 2 software distribution.

Zenith designed keyboard remapping into the Z-100 primarily to support multiple language keyboards. Using this remapping capability, you can reassign any key code to any other key code. Beginning with MS-DOS 2, Zenith provides a utility to allow users to create their own keyboard remapping. This capability is part of the FONT program, which can also be used to design character fonts.

Instructions for using FONT are in the MS-DOS manual, and brief help menus are

available from within the FONT program by hitting the Help key. The FONT key remapping option lets you remap any key by changing the code that is produced when the key is pressed. Table 1 below shows the 62 key changes that must be made to change the standard "Qwerty" keyboard on the Z-100 to a Dvorak keyboard. After you have made these changes using FONT, save the file as "DVORAK.CHR", and place the command FONT DVORAK.CHR in your AUTOEXEC.BAT file. From then on, the Dvorak setup will be installed automatically at bootup.

You can physically rearrange the key caps by (gently!) prying them off the key switches and relocating them as shown in Figure 1. The Figure shows the "{/[\" key placed where the \"_/-\" key normally is, however, I have seen other Dvorak keyboard utilities that redefine \"_/-\" to be \"[/\" and redefine \"{/[\" to be \"_/-\". The placement of these keys was not considered by Dvorak, who did his original work in the 1930's when brackets and braces were not available on standard typewriter keyboards (see *Typewriting Behavior*, by A. Dvorak, N.L. Merick, W.L. Dealey, and G.C. Ford, American Book Co., 1936).

I have not shown the remapping for control keys. If you are going to use the Dvorak keyboard in an application which needs control keys, you must remap the 24 alpha-

betic (not 26, since A and M are in the same location on both layouts) control keys also. The codes produced by holding down the control key are hexadecimal 40 less than the equivalent upper-case character codes, thus control-A is 01 (upper-case A is 41), control-B is 02, etc. Remapping the "Qwerty" control codes to Dvorak control codes, therefore, involves mapping control-B (02) to control-X (18), control-C (03) to control-J (0A), etc., as calculated by subtracting 40 from the upper-case entries in Table 1.

Unfortunately, the Caps-Lock key will not work properly with some of the keys when they're remapped this way. Normally Caps-Lock is designed to lock only alphabetic characters. The keyboard hardware is pre-programmed to decide which keys are alphabetic and which are not, thus the Q, W, and E keys (which change to ", <, and >, respectively) will be shifted, and the <, >, and ? keys (which change to W, V, and Z, respectively) will not be shifted when Caps-Lock is down. True hackers will note that there is a way to solve this problem by writing a keyboard interrupt handler that places the keyboard in "event driven" mode, however, this is not a trivial task.

If you are using one of the newer (IBM compatible) Z-100 PC computers, you should be able to use the key remapping feature built into the ANSI driver to remap your keyboard. To do this, make sure you

Qwerty	Hex	Dvorak	Hex	Qwerty	Hex	Dvorak	Hex	Qwerty	Hex	Dvorak	Hex
"	22	—	5F	K	4B	T	54	g	67	i	69
'	27	-	2D	L	4C	N	4E	h	68	d	64
—	5F	{	7B	N	4E	B	42	i	69	c	63
,	2C	w	77	O	4F	R	52	j	6A	h	68
-	2D	[5B	P	50	L	4C	k	6B	t	74
.	2E	v	76	Q	51	"	22	l	6C	n	6E
/	2F	z	74	R	52	P	50	n	6E	b	62
:	3A	S	53	S	53	O	4F	o	6F	r	72
;	3B	s	73	T	54	Y	59	p	70	l	6C
<	3C	W	57	U	55	G	47	q	71	'	27
>	3E	V	56	V	56	K	4B	r	72	p	70
?	3F	Z	5A	W	57	<	3C	s	73	o	6F
B	42	X	58	X	58	Q	51	t	74	y	79
C	43	J	4A	Y	59	F	46	u	75	g	67
D	44	E	45	Z	5A	:	3A	v	76	k	6B
E	45	>	3E	[5B	/	2F	w	77	,	2C
F	46	U	55	b	62	x	78	x	78	q	71
G	47	I	49	c	63	j	6A	y	79	f	66
H	48	D	44	d	64	e	65	z	7A	;	3B
I	49	C	43	e	65	.	2E	{	7B	?	3F
J	4A	H	48	f	66	u	75				

`ESC[n;mp`
to remap ASCII key `n` to ASCII key `m`. The values substituted for `n` and `m` must be in decimal. For example:
`ESC[65;81p`
remaps the A key (hex 41, decimal 65) to produce Q (hex 51, decimal 81). You can

One Note Of Caution: The manual for the MS-DOS programmer's utility pack warns that "if you exceed 250 bytes of ANSI definitions, you might obtain unexpected results", however, it is not at all clear whether they are talking about the total number of characters copied to the driver (e.g. the string "ESC[65;81p" is 10 characters long), or the number of keys that can be re-

Readers who want to know more about the Dvorak keyboard layout should see the article "Keyboard Efficiency", by Donald Olson and Laurie Jasinski in the February 1986 issue of *BYTE*, or subscribe to a newsletter called "Dvorak Developments", P.O. Box 717, Arcata, CA 95521. According to the *BYTE* article, the newsletter costs \$6 for two sample issues and includes additional material.

!	@	#	\$	%	^	&	*	()		+	~
1	2	3	4	5	6	7	8	9	0		=	
"	<	>	P	Y	F	G	C	R	L	?	/	
	.		p	y	f	g	c	r	l			
A	O	E	U	I	D	H	T	N	S			Return
a	o	e	u	i	d	h	t	n	s		-	
:	Q	J	K	X	B	M	W	V	Z			Shift
:	q	j	k	x	b	m	w	v	z			

More On Two Terminal Games!!!

Steven W. Vagts

9509 Gray Mouse Way
Columbia, MD 21046

Since my last article, "Two Terminal Games", I have found an answer to a question, found two problems with "POKE" and have made an exciting discovery.

I hadn't planned on writing another article, but when I continued experimenting with my two terminal game routines, I found a problem that I felt you needed more information on. If you haven't had the opportunity to read my article, "Two Terminal Games", appearing in the April 1987 issue of "REMark", you may wish to do so before reading this article, as I am building upon subject matter contained in that article. Others may wish to review it.

One Terminal Operation And POKE

The programs I have been converting to two terminal operations have the option of only using one terminal and this has worked quite successfully. I ask a question at the beginning of the game, if two terminals are to be used, and load a flag according to the response. I then test this flag in the program each time I reach a point where information would need to be sent to the other terminal. If the flag indicates only one terminal is being used, nothing is sent.

One of the programs I was converting, however, used all available memory — so much so, that I couldn't just run it using the MBASIC interpreter; it had to be compiled, then run. So I certainly didn't have the space to test a flag each time it came to sending information to a second terminal. Well, I just let the program send the information to the port anyway. If nothing was connected to the port, or if the other terminal was off, then so what? This worked fine when I was using the MBASIC "OUT" command, as we discussed last time. During the conversion to the use of LPRINT statements, which by the way, took more precious space but was done because of the speed problems, I always had the other terminal on, loaded with "TXCVR", a program discussed in my last article, and ready to go. It worked fine.

During the development of "TERM2", my demonstration program also discussed last time, I had always had the other terminal ready also. After all, it was designed to be run on only one terminal.

Well, the other day I had the desire to use this massive program on one terminal. Lo and behold, when it came to a point where information was normally passed to the other terminal, it gave the same indications as a crash! After returning from orbit (I thought this program was finally done), I set about trying to figure out why. Well, any of you who, for one reason or another, attempt to run "TERM2" on only one terminal will find the same problem.

It turns out to be the POKE 3,41 command. Making line 190 a REM statement, or removing it entirely, has the effect of leaving the IOBYTE set at 169. For some reason, there must be a difference in signals between these two settings, thus causing the problem. The cure? Test the flag that I mentioned earlier to determine whether you wish to change the IOBYTE or not. Leave the IOBYTE alone if you only plan to use one terminal. I hope this helps anyone who has run across the same problem.

POKE 3,169? A Word To The Wise

While I discussed the purpose of changing the IOBYTE last time, I neglected to explain where the numbers 41 and 169 come from. If you PEEKed the 0003H location on your system, you may have had an entirely different number there than 41 or 169! POKE 3,169 at the end of your program and you may change the entire port configuration of your system!

Without overly reviewing what the IOBYTE is (we discussed this last time), let's discuss how the numbers 169 and 41 were obtained and go from there.

Pulling out the CP/M 2.2 Alteration Guide (its been getting a lot of use (abuse?) from me recently) will help during this discussion. The IOBYTE is split into four distinct fields of two bits each, called the CONSOLE, READER, PUNCH and LIST fields and is where the logi

cal and physical device matches determined when running the CONFIGUR program are stored. Each field's two bits can consist of any of four choices; 00, 01, 10, or 11, depending upon your configuration.

For my system:

- The CONSOLE (bits 0 and 1) is set to 01, the CRT: device
- The READER (bits 2 and 3) is set to 10, the UR1: device
- The PUNCH (bits 4 and 5) is set to 10, the UP1: device
- The LIST (bits 6 and 7) is set to 10, the LPT: device

In binary, then, this represents:

"LIST+PUNCH+READER+CONSOLE" or "10+10+10+01" or 10101001 or 169 decimal.

When we change the LIST device to TTY:, it is set to 00. This then gives a binary number, 00101001, or 41 decimal.

Therefore, if your system was configured differently from mine, the IOBYTE may be composed of any binary combination. The last thing we want to do is exit a program with a completely changed configuration. Life is hard enough, without trying to figure out why your high speed reader no longer reads!

The FIX: Going back to the "TERM2" program printed in my last article, we need to make some minor adjustments. Line 190 had read:

```
190 POKE 3,41: WIDTH LPRINT 255: REM ....
```

To find out what the IOBYTE was — and storing that information for later use — before we change it, we need:

```
190 IOB=PEEK(3): POKE 3,41: WIDTH LPRINT 255
```

Unless you are writing some complicated program that requires a different number from 41, such as using a different CONSOLE, READER or PUNCH, 41 should work fine. Our main concern is how we exit the program. Therefore, we need to change line 1200, which had read:

```
1200 WIDTH LPRINT 80: POKE 3,169: STOP
```

Recalling our saved variable IOB, we write:

```
1200 WIDTH LPRINT 80: POKE 3,IOB: STOP
```

That should cure any inadvertent misrouting of electrons following an exit from our programs.

Change Device Ports Without Running "CONFIGUR"?!!

As you recall, in my last article we discussed how we can make LPRINT use the TTY: device, but we needed to run the CONFIGUR program on the master terminal to configure TTY: to the 330Q port. I didn't know at the time how to make this change via my program rather than using CONFIGUR, so I threw out a challenge for ideas. It's one of those things that keeps programmers awake at night, you know what I mean?

Well, I get some of my best programming solutions while trying to sleep — I also get some of my worst. Anyway, I accepted my own challenge and began a search for the answer. Let's compare notes.

It would make sense to place the port and baud rate information in the same vicinity as the IOBYTE, but according to section 9 of the CP/M Alteration Guide, "Reserved Locations in Page Zero", there is nothing that relates to the information needed. In fact, the IOBYTE itself appears to be some kind of anomaly. Section 6 states, "No CP/M systems use the IOBYTE (although they tolerate the existence of the IOBYTE at location 0003H), except for PIP

which allows access to the physical devices, and STAT which allows logical-physical assignments to be made and/or displayed." Well, so much for that idea — I'm glad some programmer thought of the IOBYTE though, as you will see when we finally do find the solution to our problem.

Reviewing the STAT command showed it could only be used to monitor and temporarily control the assignment of physical input/output devices to the appropriate logical device names. Although entitled, "The Utility that Temporarily Sets Printer Characteristics", SETLP only enables you to change the operating system's baud rate setting for a serial printer and determines whether the LPT device will be used to run a serial printer or a parallel printer. Close, but no cigar. Besides, we want software control from within a program.

Another couple of "nights" went by; then the obvious hit me. I've been looking in the wrong place. It's the BIOS that holds the information we need!! All we have to do is find it!

A piece of cake, for anyone knowing how to use CP/M's Dynamic Debugging Tool (DDT). DDT and I were still total strangers though. Well, I thought I'd just PEEK around a little, if you will pardon the pun, and try a little luck at MBASIC...

We all know (?) that CP/M is logically divided into several distinct parts:

BIOS — Basic I/O System; of which we have the source code.
BDOS — Basic Disk Operating System
CCP — Console Command Processor
TPA — Transient Program Area

The BIOS and BDOS are logically combined into a single module with a common entry point, and are referred to as the FDOS. This is located at the highest portion of computer memory, which, for a 64K terminal, extends up to 65,536 bytes.

By using a simple MBASIC program (Figure 1), and a known value to search for, we should be all set. What known value? Looking into the BIOS LISTING FOR CP/M 2.2.04, lines 970 through 982 hold the secret. The one constant that most of us use — our console — is located at port 350Q, 232 decimal, and use 9600 baud, expressed as two bytes, 12 and 0 decimal, immediately following.

```
10 FOR I=50000 TO 65500
20 A=PEEK(I)
30 IF A=232 THEN B=PEEK(I+1): IF B=12 THEN 50
40 NEXT: STOP
50 PRINT: PRINT I: FOR J=I-10 TO I+15
60 PRINT PEEK(J);
70 NEXT: GOTO 40
80 END
```

Figure 1

We could continue line 30 with a test for the third byte also, but there won't be that many listings that we can't find that combination in the list ourselves. I ran the program and found the following information:

HEATH/ZENITH CP/M 2.2.04 BIOS CODE — 64K RAM

ADDRESS	DECIMAL VALUES		PORT VALUES:
	PORT:	BAUD RATE:	
60984	232	12 0	CRT: 350Q, 9600 baud.
60987	216	12 0	TTY: 330Q, 9600 baud.
60990	224	24 0	LST: 340Q, 4800 baud.
60993	208	128 1	UR1/UP1: 320Q, 300 baud.

C.D.R. CP/M 2.2 BIOS CODE — 64K RAM

DECIMAL VALUES				PORT VALUES:			
ADDRESS:	PORT:	BAUD RATE:					
57399	232	12 0		CRT:	350Q	9600	baud.
57402	216	12 0		TTY:	330Q	9600	baud.
57405	224	24 0		LST:	340Q	4800	baud.
57408	208	128 1		URI/UP1:	320Q	300	baud.

Again, the information found at each address may be different from mine, depending upon how your system is configured.

So, what does this do for us? Well, first it allows us to PEEK or, if we wish to actually change it, POKE an address to change the information, rather than be forced to run the CONFIGUR program. For example, if we were running Heath/Zenith CP/M 2.2.04 and we wanted to change the TTY: port to 320Q and the baud rate to 300 for some reason, we could POKE 60987,208: POKE 60988,128: POKE 60989,1 and it would temporarily be changed until we rebooted or changed it back through additional POKE commands.

This would be all well and good, IF this port data were given in a location that remained constant for all dialects of CP/M, like the IOBYTE. But, alas, as shown above, this isn't to be. Whoever the programmers were that allowed a common place for the IOBYTE didn't think of this other information.

This, then, causes a problem. For individuals, such as myself, that operate both H/Z BIOS and C.D.R. BIOS programs, simply POKING these locations won't work. Anyone contemplating this technique in their games MUST REMEMBER to specify the BIOS that the program is meant to run with. As good an idea as the POKE command may seem for this, I think I prefer to change the port via the CONFIGUR program.

Two Computers And One Disk Drive!!!

When I bought my internal double-side, double-density (DSDD) drives and controller board, I installed my old hard-sectored, single-sided, single-density (SSSD) drive in an external enclosure and left the old controller board installed. This allowed me the convenience of readily placing my work, and articles, to the standard H/Z SSSD disk for sending to other H/Z-89 users or to this magazine. Well, the other day this external drive died.

My daughter had just completed her 4-H minutes on my other H-89 and was ready to print them, but the parallel printer is only connectable to this terminal! Oh, grief. I was faced with a number of options — retype her minutes or exchange the external drive with the internal drive of the other terminal, neither of which would be enjoyable.

Then I thought of a possible third option; use the other drive where it was — installed in the other H-89! I had thought about this earlier, but hadn't needed to try it before. Theoretically, it should work the same way any external drive should work, and whether it receives its commands from its own controller board or another one shouldn't matter, as long as it is powered by its host computer. Well, to make a long story short — it worked, and better than I had expected. Here's how:

Prepare a 34 conductor drive cable long enough to go from the female drive connector of one computer to the female drive connector of the other. One outside edge of the cable should be stripped or marked so the cable ends aren't swapped inadvertently. Check inside both computers to ensure the marked edge of the ribbon cable going to the connector on the backplate are both installed in the same manner. Connect your new cable so the

marked edges align — misalignment may not be conducive to the long life of your controller boards.

Power to the drives is provided separately by its enclosure or host computer. Therefore, both computers must be turned on before use. Simply use the disk drive as though it were installed in a separate cabinet.

This opens up the possibility for other uses. Both computers can be used at the same time without interference as long as disk accesses are not from both computers at the same time. It is also best to perform a warm boot (CTRL-C) before accessing the drive, if a change had been made to the disk's directory since the last access. I've tried reading, writing, copying and deleting files from either computer and have had no problems.

Well, that brings us to the close of another session. I hope I have brought some light on the various subjects I've mentioned during the past several months and have encouraged others to participate and perhaps share some of their thoughts and ideas. Casually flipping through "REMark" and "SEXTANT", you can see that the H-89 articles are getting fewer and are rapidly following the way of the H-8. Yet, there are so many areas that remain unexplored and certainly unwritten, perfect for the hard working H-89, but perhaps too menial for the newer, more capable computers.

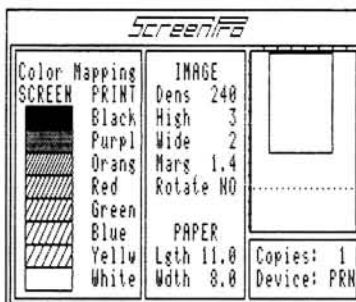
My next area of interest, unless I get sidetracked, is to explore uses of the recently added parallel port — for other than the printer. I haven't seen any articles on controlling household appliances or electric trains. Yet, I can't believe that, out of the thousands of you out there, no one has done this. It sure would save some of my midnight oil if I didn't have to reinvent the wheel.

Once again, if anyone has questions or suggestions on any of my articles, feel free to drop me a line.



ScreeniFo[®] Copyright 1987 Paul F. Herman Inc.

A new concept in memory resident utilities from the publisher of DOODLER and MOUSE PACK



- Print any portion of the screen. Graphics or Text.
- Page editing and layout features.
- Copy screen images from one program to another.
- Over 100 different printers supported.

- Select print density, height, width, and margin.
- Multiple copy capability is great for mailing labels.
- Color printer capability with complete color mapping.
- 'Frame Grabber' lets you copy images from one graphics program to another, or do step-and-repeat.

Available only for the Z-100 \$59.⁰⁰



Software Graphics Tools
3620 Amazon Drive, New Port Richey, FL 33553
C.O.D. Orders accepted - Call 813-376-5457



TO ORDER
CALL 714-540-1174
or WRITE

M/C, VISA, CHECK or MONEY ORDER
CALIF. RESIDENTS ADD 6% Sales Tax

SHIPPING-ADD
UPS Ground \$1.50
2nd Day Air \$4.00
Next Day Air \$12.00

Zenith is a registered trademark of Zenith Data Systems Corporation.

ZP-150 RAM EXPANSION 32K as low as \$45

The ZP-150 is a great laptop. It lacks only one thing—sufficient on-line memory. Our 32K low-power CMOS modules plug into existing sockets in the computer to provide up to 416K of on-line, non-volatile memory. Upgrading your machine with from one to twelve modules takes a matter of minutes.

We are in our third year of providing quality memory upgrades similar to the 32K to thousands of satisfied portable computer users. Our memory modules are 100% factory tested and carry a one year materials and workmanship warranty. Because your satisfaction is our goal we offer a full refund 30 day return policy.

**Priced at: \$59 for 1 or 2
\$49 for 3 to 9
\$45 for 10 or more**

Illustrated step-by-step instructions included.

**AI AMERICAN
CRYPTRONICS INC.**

(Formerly Cryptronics, Inc.)
1580 Corporate Drive, Suite 123
Costa Mesa, California 92626
(714) 540-1174

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.

VAX & PC Users

ZSTEMpc™-VT220 Emulator for the PC Series
High performance COLOR VT220. Double high/wide, 132 mode, smooth scrolling, downloadable fonts, user defined keys, 8-bit and full national/ multi-national modes. XMODEM and KERMIT, softkey/ MACROS, DOS access. **\$150.00**

EGAmate™ — option for true 132 column mode **\$39.00**

ZSTEMpc™-VT100 Fastest, most complete emulation **\$99.00**

ZSTEMpc™-4014 Emulator for the PC Series
Use with ZSTEMpc-VT100, VT220, or stand-alone. Interactive zoom and pan. Save/recall images to/from disk. Keypad, mouse, printer, and plotter support. **\$99.00**

ZSTEM™-VT100 Emulator with XMODEM FOR THE Z-100 **\$148.95**

DECKHAND™ Utilities for the Z-100 and PC Series
MS-DOS Utilities with VAX/PDP-11 switch processing, DIR, COPY, DELETE, RENAME, TYPE with wild-cards, full DATE processing, attribute processing, query, backup and more. **\$49.00**



KEA Systems Ltd.

2150 West Broadway, Suite 412
Vancouver, B.C. Canada V6K 4L9
Technical Support (604) 732-7411

Order desk (800) 663-8702 Toll Free

Telex 04-352848 VCR
VISA and Mastercard accepted.
30 day money back guarantee.
Quantity and dealer discounts available.

Trademarks
VT100, VT220, DEC, VAX — Digital Equipment Corp. ZSTEMpc
ZPAL, COMPAL, EGAmate, DECKHAND — KEA Systems Ltd.

Z-100 PC Expansion with ZPAL™

Z-151/Z-161 Users
Use 256K bit memory chips on the original memory board. Extend the memory to 640K or 704K bytes with ZPAL-2 Decoder **\$29.00**

Z-158 Users
Extend your system to 768K bytes with 3 banks of 256K bit memory chips and ZPAL-158 decoder. ZPAL-158 Decoder **\$36.00**

Z-138/Z-148 Users
Extend your system to 768K bytes with 3 banks of 256K bit memory chips and ZPAL-148 decoder. ZPAL-148 Decoder **\$36.00**

Z-100 PC Expansion with COMPAL™

Z-151/Z-161 Users
Instead of disabling COM2 when using an internal modem or multifunction board, MAP it to COM3. Supported by ZSTEMpc-VT220 or the driver software included. COMPAL-3 **\$39.00**

How To Convert HDOS MBASIC Programs To Other Microsoft BASICs

Kirk L. Thompson

#6 West Branch Mobile Home Village
West Branch, IA 52358

In an erratic series of articles, I am discussing a collection of programs written in HDOS MBASIC for the H8 and H-89/90. While most of these are oriented toward Clark Systems' old "MAILPRO" mailing list package, a few relate to Hoyle and Hoyle's "QUERY!3" database management system, such as TXT2Q3 (REMark, publication date). Versions of the latter are available for HDOS, CP/M-80, and MSDOS.

But I recently stumbled across a problem which demands some attention if you plan to adapt these, or indeed, any HDOS MBASIC program, to another operating system and Microsoft BASIC. I shall discuss the crux of the problem and a method for solving it.

Delimiting Physical Lines

A Microsoft BASIC line of code, no matter which interpreter you have, may be broken into logical and physical lines. The former is the entire code which is preceded by the line number. For example:

```
300 IF INSTR(F$, ".") > 0 THEN 280
```

is a logical line. But most MBASIC interpreters have a maximum number of characters which can appear in a logical line of 255 characters. This many characters make the line difficult to read, so delimiting characters may be inserted, at your discretion, which break the logical line into physical lines. This delimiting character varies from version to version of MBASIC on Heath/Zenith systems.

Under the HDOS version, this is the at-sign (@). Hence, a long line of code could look like

```
300 IF INSTR(F$, ".") > 0 THEN PRINT@
      "ERROR--Extension not allowed!":@
      PRINT:GOTO 280
```

In this example, I've inserted the at-signs and strings of four spaces. The former breaks the logical line into three physical lines, and the spaces immediately following the at-signs line up the physical lines to start in a single column for pretty-printing.

But the way HDOS MBASIC inserts this physical line delimiter (@) creates problems when moving the code to other operating systems and MBASICs. The reason is the hard carriage return which follows the at-sign. If you attempt to load the ASCII code of an HDOS MBASIC program into, say, MBASIC 5.2 for CP/M-80, the latter will choke on the physical line immediately following the delimiter. It will spit out an error message of the type:

```
DIRECT STATEMENT IN 300
```

If you list the code, you will discover that reading in the program also terminates at the carriage return following the first at-sign in the code.

Multiple Editing

The fix to this problem requires two-stage editing and a text editor which is capable of deleting carriage returns or merging separate lines which have carriage returns between them. Most good word processors

allow you to do this. For example, under CP/M-80, you can use "Magic Wand" or Newline Software's "Text Processor." Although I have no experience with it, "WordStar" should also work. Under MSDOS, "PeachText" will do the job. Regrettably, a simple editor like Software Toolworks' "PIE", will not.

The task can be broken into four stages:

1. Print out the HDOS MBASIC ASCII file (optional, but recommended).
2. With your text editor, search for ALL appearances of the at-sign (@) and delete the trailing carriage returns or merge the physical lines. If your editor displays some character which represents the carriage return (as "Magic Wand" or "PeachText" do), turn it on to make it easier to perform this operation. When complete, save the file.
3. Now load your edited file into MBASIC and list it. You will observe that all of that pretty-printing of the physical lines has disappeared. The lines will look something like:

```
300 IF INSTR(F$, ".") > 0 THEN PRINT@
      "ERROR--Extension not allowed!":@
```

and so forth. Refer to your printout and edit each of the lines which contains the at-sign. What you want to do, here, is CHANGE the "@" to whatever your MBASIC uses to delimit physical lines. Under CP/M, for example, change the

"@'s" to line feeds. Save the program file when done.

4. Now you will have to convert some of HDOS MBASIC's reserved words to the equivalent used in your version and the terminal codes to your system if you aren't running an H/Z-19, or equivalent, terminal. These, however, are beyond the scope of this short essay.

CNVRTHMB.PRN

But by the time this appears, I expect to upload to HUG's PBBS (616-982-3956) a file which describes in excruciating detail, the ins and outs of converting HDOS MBASIC programs and the H/Z-19 terminal codes. It resides in one of the HDOS catalogs and bears the name at the head of this paragraph.

If you don't have access to a modem, I would be happy to send you a hardcopy of the file. Just send me a postage-prepaid (2-oz.), self-addressed manila envelope (minimum dimensions of 5-1/2" x 8-1/2") to the address above, and I will mail it to you.

There is a lot of public domain HDOS MBASIC software which those of you who are running other operating systems may want to use. This procedure will give you a first handle on converting this material to your Microsoft BASIC without having to rekey the entire source code.

HUG Software Engineer's Note: The HTOC program that comes on the HRUN HDOS Emulator disk (885-1223[-37]) can automatically convert HDOS MBASIC files

to the standard MBASIC format (if they were saved in ASCII), when it is used to transfer them from HDOS disks to CP/M disks.

In the CP/M and MS-DOS operating systems, text lines are delimited by a carriage return character (0D hex) followed by a line feed character (0A hex). Microsoft BASIC languages for these operating systems (including ZBASIC and GW-BASIC) delimit broken logical lines by reversing the carriage return and line feed characters (0A hex followed by 0D hex) at the break. The HTOC program, if so instructed, will replace @ signs with the line feed/carriage return sequence in HDOS MBASIC programs.

✱

S & K Technology, Inc.

Quality Software for Heath/Zenith Microcomputers

For the Z100...

WatchWord® \$100.00

The ultimate in word processing with speed and power. See subscripts, superscripts, underlining, and boldface directly on the screen. Create your own fonts and special characters. Other features include centering, formatting, automatic horizontal scrolling with long lines, large file capability, split screen, macros, color, and an extensive configuration facility. See reviews in Remark (July 1985) and Sextant (Jan-Feb 1985, Sep-Oct 1985). Requires 192K RAM.

The Resident Speller™ \$100.00

Spelling checker for use with WatchWord. Checks as you type from inside WatchWord or checks a file. Includes a 50,000 word expandable dictionary. Requires 192K RAM to check a file. Requires 300K RAM to check as you type.

Demo disk for both \$ 3.00

For IBM compatibles including the Z150 and Z200 series...

PC WatchWord® (New) \$ 99.95

The ultimate in word processing for the sophisticated user. Most of the features of the Z100 version except for screen fonts. Requires 256K RAM.

PC Resident Speller™ \$ 99.95

Spelling checker for ASCII files such as those created with WatchWord, WordStar, WordPerfect, PeachText, and VolksWriter. Includes Strike. Requires 256K RAM.

Strike™ \$ 49.95

Adds as-you-type spelling checking to your word processor. Works with the word processors above and also with DisplayWrite, MultiMate and PFS:Write. Requires 100K RAM in addition to that used by your word processor.

Demo disk for Strike and The PC Resident Speller \$ 2.00

Demo disk for PC WatchWord \$ 2.00

Texas residents please add state sales tax.

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

C__Power

Part 4

John P. Lewis

6 Sexton Cove Road
Key Largo, FL 33037

Hi Huggies, welcome to Part 4 of "C__Power". This issue will provide the code to complete menu option one; "1. Search for a record". The included source code will allow you to upgrade your existing project to a directory program, capable of high speed search and record recovery. We will leave two submenu options incomplete for now, namely the (R)eplace and (P)rint (hard copy) options.

Before we embark on a journey into evolution, let me digress for a moment and tell you a story about program development. The subject of this story is the venture we are working on.

This particular enterprise first saw the light of day as a rather primitive undertaking written in BASIC. The best thing I can say about this early effort is that I learned quite a bit about random records while trying to make it do the job I wanted. It went through quite a number of versions during its evolution before finally becoming quite useful in its present form. That was before I discovered "C". The first rendition of this program, written in "C", was nearly as bad as the first version written in BASIC. In their present form, they are both quite useful programs, worth all the effort that went into their development. The point that I'm trying to make here is that very few worthwhile programs are a resounding success in their first edition. A good case of persistence mitigated by a dose of determination

is probably the biggest asset a programmer can have.

Those of you with a penchant for experimentation have probably already put our latest function "jindex" to work. Possibly, you have even outstripped this series of articles and are using your new random records program. If that is the case, go back to your computer, this is not for you. The rest of you will likely find the new code very interesting. I have made a number of additions, besides the new functions. You will note an #include statement at the bottom of the listing too, which you might find a bit strange.

Before we go on to the listing, let me make an apology for not mentioning the compiler switches necessary for the compilation of version three of "CPOWER". You should invoke the compiler with the command: `c -c2000 cpower`. Without the period. The `-c2000` switch forces the compiler to make room for 2000 bytes of string space. I hope this has not caused too much consternation among those of you who are new to "C".

Please Note: The included listing is for INSERTION into the existing source code for "CPOWER". One line of global variables has been appended and one change has been made in the variables following "main" which is noted in the listing. The rest of the code (except where obviously

redundant for the purpose of clarity) is to be inserted using your word processor or text editor.

As before, we will look at the new code and then explain what is happening from the top down.

As you can see, I omitted all but the new code in this listing, except for a few lines near the bottom. This is an effort to cut down on the space used in the pages of REMark for this article. The listing is still rather long, but none of the code is very complicated or hard to understand. Hang in there, when you have this on disk and tested, you will have the "heart" of the program.

Before I go into explaining the code, let me tell you how it works. When you select "1. Search for record" from the menu, the program directs the computer to open the data file and read the first two fields (compny & name) of the entire data file into memory at location (string[size]). When this is done (about two seconds for a data file with forty records and an H-89 running a 4 mhz clock), the compny name fields have been concatenated and are now one long string (2000 bytes). Each record uses fifty bytes of memory and is ended with its companions. Now the user is asked for a "key" which is "handed off" to "jindex". The key should be a substring which will match with one or more of the record

fields. For instance: Smith, (without the comma) if we are looking for John Smith or any other Smith in the data file. This program will find every occurrence of Smith in the file, if the user invokes the "(C) continue" option. If the search key is unique, the computer will display only the one record where a match is found.

This search sounds as though it might be a bit slow since it uses a character at a time retrieval, but it is in fact, very fast. You might be asking, "What has the location of a substring in memory got to do with the location on disk of a record within a data file?" Remember, the string we are searching through is an image of the first two fields of all the records (of our data file) on disk strung together. If we know how many bytes from the beginning the substring match occurs, we can do a little integer arithmetic and come up with the offset of our record within the data file. This same offset can be used to find and display a record and/or edit the same. I will explain the arithmetic used by the program later as we are going through the listing.

Back to the listing and starting at the top, we find three new #define statements. Again, this is just a ploy to make it easier for the programmer to make changes in the code at a later date. LEN1 and LEN2 represent the length of the company and name fields. RECORD is defined as 128 which delineates the length of a data record in this file. These are values which might well be changed at a later date to suit the needs of the next user.

Moving down, we encounter two new variables declared as unsigned (integers), as well as an integer called init and four variables declared as type "char". You will notice that one of these (string[SIZE]) has been moved from 'main' to its new location, which makes it a "global" variable. While we are here, be sure you delete this variable from 'main' when you add it to the 'global' variables.

Next, we discover another new function called "search". Actually, it doesn't do any searching, but does read two fields into memory starting at location string[0] and reading our data file into memory until it encounters the character DEL in byte zero of a new record.

Looking a little closer, we see two variables declared inside the function. This indicates that these variables are known only to this function, we could use these same variable names within another function with no conflict. We then open the data file for reading and test the value of init which will

be 0, if this is the initial usage of this function after program acquisition or after any record writing activity. The user is informed of what is happening, and then the program directs the computer to retrieve fifty bytes of each record (LEN1+LEN2) and read them into memory starting at string[0]. After reading fifty bytes, the offset is incremented by 128 (RECORD) bytes and the process is repeated until byte 0 of the record being read holds a DEL character. This will cause the program to fall out of the loop and execute the last line of the search function. Here, init is made to equal one, q (a global value) is made to equal k (the continuous loop counter), fifty is subtracted from this variable and offset is reset to zero.

We have just read the first two fields of each record in the data file into memory, concatenating them while doing so. And then set a global variable (q) to point to the end of the string (for use by jindex).

The main reason for the existence of "srch2" is merely to get the search substring from the user, call another function (jindex), and then do some integer math with the result (mentioned earlier).

As you can see, we use "gofor" to retrieve the character string, "ptr", while limiting the length of this string to twenty-four characters (LEN1-1). Then "jindex" is called by asking for the value of "loc" and passing the variables (string, ptr, offset). Our function (jindex) returns the value of "loc", which is the number of bytes from the beginning of string[SIZE] where a match was found with the characters in either the "compny" or "name" fields from our data file. The program uses this value to determine the offset which is needed to access this record holding our substring match. A little integer math is performed to arrive at this value. If "loc" is divided by fifty (LEN1+LEN2), and the result is multiplied by 128 (RECORD), we have the offset needed to access this record. For instance, if 'loc' were equal to two hundred and this was divided by fifty, the result would be four. This value is then multiplied by 128 to yield the offset (512). Now we can access the fourth record in our file where our data lives. I have simplified the math by using small numbers, but the computer is not intimidated by big numbers as I am. It will yield the result, position the read/write head, and retrieve the record before I could grab a pencil.

Now we come to one of my favorite functions, "jindex". The credit for this gem belongs to "The Software Toolworks", since that is where it originated. I merely

modified it to be better suited to the job at hand. This entire program depends on this routine for its proper functioning, hence its value. You might ask why it was modified, since we haven't done anything yet that it wouldn't do before I butchered it. I thought you would never ask! You are permitted to look at the code under "case '1':" which follows the "menu". When you find the submenu, look for the (C)ontinue option. Notice that we call jindex from here using 'loc' as our third parameter instead of offset. We are incrementing the pointer (loc) with each call, using the current value of 'loc' and then doing some integer math (again). More on this later.

Looking very closely at the jindex function, notice how it compares a character at a time using nested "for" loops. The first one has the value of "p" (our third parameter) passed to it. This is either the value of offset or loc, depending on which routine called jindex. Next, i is made equal to p, i is then compared to q (our pointer to the last field in string[SIZE]), and if less than or equal to q, the nested for loop is executed. Here, we pass the value of i to j, set k equal to zero, compare the character from our substring t[k] to the '\0' character (string terminator in "C") and to character s[j] in our concatenated string. If there is a match, the counters j,k are incremented and another comparison is made until either there is no match or the string terminator is found in t[k] (our substring). The latter action indicates that we have found a match for our substring in string[SIZE]; thus returning the value of i. If we have no match, we fall out of the inner for loop and execute the outer loop, vacillating back and forth until i exceeds the value of q or we have a match. A "no match" condition will cause a (-1) to be returned for the value of loc. I hope I have explained this function so you can understand it. I had to study it (index) for some time before I was able to comprehend all of its facets; even longer before I was able to alter it to become its new entity (jindex).

Now we have come to "main" which should be on your disk in the proper format and tested so I won't dwell on that except to point out that case '1': is the first routine after the menu and must be "inserted" in your existing program.

This brings us to case '1': where we will find some code that utilizes the functions we just discussed and some routines of its own. After clearing the screen and locating the cursor, we test init for (1). If init == 0, we call search, otherwise we relocate the cursor and call srch2. This function (srch2)

asks for the value of loc when utilized, so we test for (-1) to see if we had a match. If loc == -1, we print a user message and enable a return to the menu. Assuming a match, the program causes the computer to locate the read/write head at offset (seek(fd,offset,0)) and proceed to retrieve the record a character at a time. Remember, the data file was opened previously by "search" and it was left open in anticipation of this routine. Now we print the record number using offset and some math to come up with the answer. Then the headings for all the record fields are printed to the screen, followed by a routine which dissects the 128 character string (formed during record retrieval) as it prints the fragmented string to the screen. These string segments are printed a field at a time next to their respective headings.

This brings us to an escape sequence which places the cursor on row 16, clears the screen from the cursor position down and then prints a submenu. We'll first choose (C)ontinue from the menu, since this will follow our top down perusal of this listing. Notice the code following the gofor instruction. The instruction: ch=men[0]; ch=toupper(ch);switch (ch); forces the character retrieved by gofor to be upper case. This is done to avoid additional instructions and/or confusion to the user. We only have to test for upper case as a consequence of this action. The code found at case 'C': does some more of that integer math with loc. First, dividing by 50 (the length of each corresponding fragmented string within string[SIZE]), then adding 1 to the result before multiplying this by 50 again. Sounds like a lot of work for a simple operation, but this is done to assure that our loc value falls on a field boundary (byte zero). For instance, if our substring match is with the last name in a name field, the value returned by jindex will not fall at the first byte of the record, rather nearer the far end. By dividing the loc value by fifty, we will arrive at a number which is an even multiple of 50. We add one to get to the next record and then multiply the result by fifty. This yields a value that is exactly fifty greater than the value of the first byte of the preceding LOCATION. Next, we will invoke jindex again with a call to: loc = jindex(string, ptr, loc);. Then, we do some creative arithmetic using the same logic as before to arrive at our new offset. This routine will return the offset of each record where a match is found for our substring. The user could enter a space when prompted for the search key and peruse through the file a record at a time, since a space will be found

in each record. This routine coupled with a little imagination will enable the user to access any part of the database needed, very rapidly.

Our next submenu option would be (E)dit. This uses much of the same logic as case '3'; the "Create a new record(s)" module. However, we will review it for the sake of clarity, and also to point out one of my mistakes. This code (edit module) is largely redundant and should have been included in a function, but historically this program did not start life with an editing capability. This was added after finding a need for it (not to mention, figuring a way to do it). We use the ESC J sequence, combined with locate, to clear the lower part of the screen (row 14 down) and print another submenu. The user is asked to enter a number corresponding to the field he or she wishes to edit. The program enters another switch case routine, and depending on the user's entry, will display an input field corresponding to the user's wishes. Gofor is used to retrieve the character string to be entered in the space occupied by the offending field. The value of offset has already been set to the first byte of this record when it was first displayed. All that remains to be done is to increment it to the field we wish to write over. Adding fifty to offset would have the effect of positioning the read/write head over the first character in the "Street" field. This field is then written using the printf function. The read/write head is sent to the end of the data file (seek(fd,0,2)); and the file is closed.


Before we go on to other things, let me show you another change and one addition we haven't covered. The first line of code under case '3': should read:

fclose(fd);. This is ADDED to the existing code at that point and should be the first line. Use your word processor to insert it. This is just a bit of insurance that the file is closed before it is opened for reading. A line of code has been added to the very bottom of this listing which reads: #include "stdlib.c". This is the library file containing the code for "toupper", and putting it at the bottom of the listing enables the compiler to find just the routine needed.

We have covered a lot of ground and we have a lot more to do, but now you should have some rather interesting code fragments and routines to play with. This portion of the program will stand alone and can be used "as is" for a directory. If you are having trouble with typos, or just lack the time to enter all this from the keyboard, I will transfer the source code and the corresponding .COM file to YOUR formatted blank disk (hard sector, single density for H/Z-89, or soft sector, double sided, double density for H/Z-100). These are both stock machines with no modifications to the drives or controllers, so please don't ask for anything exotic, I won't be able to help you. Please include \$5.00 to cover shipping and handling. Also remember, we still have a lot of ground to cover, so you may want to wait until the end of this series before sending for a disk. If you have any questions or problems that you are unable to resolve, write me with a description of your problem and enclose a SASE. I will TRY to answer your questions. If you have any suggestions or ideas you wish to pass along, I will be glad to hear from you.

"C" you in "C__Power — Part 5".



Micronics Technology 

SPEED MODES - H/Z 150/160 & H/Z 89

- H 89 Software Select 2/4 MHz. No Trace Cuts! Z80A and Software Included.
- H150/160 4.77/6.67 MHz. FREE Hardware Reset. Satisfaction Guaranteed.
- Prices: Assembled \$34.95, Kit \$24.95

H/Z 89 20 Meg Winchester ONLY \$595
Boots from Hard disk. ST225. H150 \$400

SOFTWARE for H/Z 150/160, 100, 89, 8!
Perfect Funds \$29.95 Perfect Money \$19.95
Paycheck \$39.95 Perfect Printer \$19.95

ORDER NOW by writing or calling:
Micronics Technology (904)-897-4257
449 Barbados Way, Niceville, FL 32578
Checks, VISA, MC. Shipping \$2. Hard Disk \$15

```

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

#include "printf.c"
#include "funcLib.c" /* routines cls, locate, getch and gofor
from "CPOWER" */
#include "seek.c" /* one of random access library files */
#define DEL 127 /* this char used for an "end of records" delimiter */
#define SIZE 2048 /* size of future "search" area */
#define DELAY 10000 /* number of iterations in delay loop */
#define LEN1 25 /* length of field for company & name fields */
#define LEN2 25 /* length of field for company & name fields */
#define RECORD 128 /* record size */

unsigned fd, offset, loc, q; /* global variables to be used
by functions and main */
int init;
char ptr[LEN1], string[SIZE], c[RECORD], dummy[2];

search ()
{
    int i, k; k=0;
    opnforrd ();
    if (init != 1)
    {
        printf("Please stand by. I'm initializing \n");
        while ( c[0] != DEL )
        {
            seek(fd,offset,0);
            for ( i=0; i < LEN1+LEN2 ; ++i, ++k)
            {
                c[i] = string[k] = getch(fd);
            }
            offset+=RECORD;
        }
        init = 1; q=k-(LEN1+LEN2); offset=0;
    }
    srch2 ()
    {
        printf("Please enter the search 'Key' ");
        gofor(ptr, LEN1-1);
        loc=jindex(string, ptr, offset); offset = (loc/(LEN1+LEN2))*RECORD;
    }
    jindex (s,t,p)
    char s[], t[];
    unsigned p;
    {
        int i, j, k;
        for ( i=p; i <= q ; i++)
        {
            for ( j=i, k=0; t[k] != '\0' && s[j] == t[k] ; j++, k++);
            if (t[k] == '\0')
            {
                return (i); /* i=pointer if match is found */
            }
            return (-1); /* no match */
        }
        opnforrd ()
    }

```

```

fd=fopen("data", "r"); /* open "data" file in read mode */
if ( fd == 0 )
{
    fd=fopen("data", "w"); seek(fd, 0, 0); /* if file does not exist, create */
    fprintf(fd, "%c%c", DEL, EOF); seek(fd, 0, 2); fclose(fd); /* one & insert */
    fd=fopen("data", "r"); /* delimiter */
}
}

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

main ()
{
    int i, k, on, j;
    char men[2], compny[LEN1], name[LEN2], street[35], city[33];
    char phone[11], ch;

    do
    {
        cls (); putchar(27); putchar(70); locate (4,5); /* enter graphics mode */
        printf("pppppppppppppppppppppppppppppppppppppp");
        printf("pppppppppppppppppppppppppppppppppppppp");
        putchar(27); putchar(71); /* exit graphics mode */
        printf("\n\n\t Please ....");
        printf("\n\n\t 1. Search for record.");
        printf("\n\n\t 2. Review existing record names.");
        printf("\n\n\t 3. Create new record(s).");
        printf("\n\n\t 4. Print entire mailing list (to printer).");
        printf("\n\n\t 5. Sort list (on specified field).");
        printf("\n\n\t 6. Exit to operating system.");
        printf("\n\n\t Enter the number corresponding to your choice ");
        gofor(men, 1);
        switch (men[0])
        {
            case '1':
                cls(); locate (4,10); offset=0;
                if (init != 1)
                {
                    search ();
                    locate (8,8);
                    srch2 ();
                    label: /* we jump here from the (C)ontinue routine */
                    if (loc == -1)
                    {
                        cls (); locate (6,4);
                        printf("No match found, press return for menu ");
                        gofor(men, 1); break; /* force return to menu */
                    }
                    seek(fd, offset, 0);
                    for ( i=0; i < RECORD ; ++i) /* get a record, a char at a time */
                    {
                        c[i] = getch(fd);
                        cls(); locate (2,6); printf("Data for record no. %d", offset/128+1);
                        locate (4,8);
                        printf("Company: "); locate (6,8); /* print headings to screen */
                        printf("Name: ");
                        locate (8,8); printf("Street: ");
                        locate (10,8); printf("City, State, Zip: ");
                    }
                }
            }
        }
    } while (men[0] != '6');
}

```



```

        locate (12,8);printf("Phone: ");
        locate (4,18);
        for (i=0; i <=24; ++i) /* insert data under correct heading */
            putchar(c[i]);
        locate (6,15);
        for ( i=25; i <= 49; ++i)
            putchar(c[i]);
        locate (8,17);
        for ( i=50; i <= 84; ++i)
            putchar(c[i]);
        locate (10,27);
        for ( i=85; i <= 116; ++i)
            putchar(c[i]);
        locate (12,16);
        for ( i=117; i <= 127; ++i)
            putchar(c[i]);
        do
        {
            locate(16,1);putchar(27);putchar(74);locate(16,4);
            printf("You may (E)dit, (R)ep lace, (P)rint, (C)ontinue\
            or (M)enu ");
            gofor(men,1);ch=men[0];ch=toupper(ch); /* allow for either */
            switch(ch) /* upper or lower case entry */
            {
                case 'P':
                    /* lprint (); (hard copy) future implementation */
                    break;
                case 'C':
                    loc /= 50;loc +=1;loc *= 50;loc = jindex(string,ptr,loc);
                    offset=loc/(LEN1+LEN2)*RECORD; /* bump pointer */
                    goto label; /* and establish new offset */
                    break;
                case 'R':
                    /* future implementation */
                    break;
                case 'E':
                    init=0;
                    locate(14,1);putchar(27);putchar(74);fclose(fd);o_pen ();
                    locate(16,4);printf("Please indicate the field you wish to edit\n");
                    printf("\n(1) Company, (2) Name, (3) Street, (4) City-State-Zip, (5) Phone ");
                    gofor(men,1);
                    switch(men[0])
                    {
                        case '1':
                            locate(20,4);printf("Company: _____");
                            locate(20,13);gofor(compny,LEN1-1);seek(fd,offset,0);
                            fprintf(fd,"%-25s",compny);
                            goto done;
                        case '2':
                            locate(20,4);printf("Name: _____");
                            locate(20,10);gofor(name,LEN2-1);seek(fd,offset+25,0);
                            fprintf(fd,"%-25s",name);
                            goto done;
                        case '3':
                            locate(20,4);printf("Street: _____");
                            locate(20,12);gofor(street,34);seek(fd,offset+50,0);
                            fprintf(fd,"%-35s",street);
                            goto done;
                        case '4':
                            locate(20,4);

```

```

                            printf("City, State, Zip: _____");
                            locate(20,22);gofor(city,32);seek(fd,offset+85,0);
                            fprintf(fd,"%-32s",city);
                            goto done;
                        case '5':
                            locate(20,4);printf("Phone: _____");
                            locate(20,11);gofor(phone,10);seek(fd,offset+117,0);
                            fprintf(fd,"%-10s",phone);
                            done: seek(fd,0,2);fclose(fd);break;
                    } /* end of edit switch */
                } /* end of menu switch */
            } /* end of do-while switch */
            while (ch != 'M');
            break;
        case '2':
            /* future implementation */
            break;
        case '3':
            fclose(fd); /* close file if open */
            cls();locate(6,6);printf("Please be patient. I'm looking for the");
            locate(8,6);printf("the end of this file ");init=0;
            for( i=0; i <= DELAY ; ++i );
            opforrd();offset=0;
            seek(fd,offset,0);
            while ( string[0] != DEL )
            {
                for (i=0; i <=127; ++i)
                {
                    string[i]=getc(fd);
                }
                offset +=128;seek(fd,offset,0);
            }
            offset -= 128;fclose(fd);string[0]='0'; /* Purge string[0] */
            for ( ; )
            {
                cls ();locate (4,10);printf("Mail List\n");o_pen ();locate(6,6);
                printf("Enter Company (if applicable) _____");
                locate(8,6);
                printf("Enter name _____");
                locate(10,6);
                printf("Enter Street _____");
                locate(12,6);
                printf("Enter City, State, Zip _____");
                locate(14,6);
                printf("Enter Phone _____");
                locate (6,36);gofor(compny,24);locate(8,17);gofor(name,24);
                locate(10,19);gofor(street,34);locate(12,29);gofor(city,32);
                locate(14,18);gofor(phone,10);
                seek(fd,offset,0);
                fprintf(fd,"%-25s%-25s%-35s%-10s",compny,name,street,city,phone);
                seek(fd,offset+128,0);
                fprintf(fd,"%c%c",DEL,E0F); /* insert delimiter & end of file marker */
                seek(fd,0,2);fclose(fd);locate (16,4);
                printf("Enter 0 to exit, any other character to continue ");
                gofor(men,1);
                offset +=128;
                if ( men[0] == '0') /* test for more input activity */
                    break;
            }
        }
    }
}

```

```

        } /* end of for loop */
break; /* two "break" statements necessary here, one for "for" loop
and one to delimit the "case" */
case '4':
    cls();locate(4,1);
    printf("You have selected the 'Print' option which has not\n");
    printf("yet been implemented. Press RETURN for menu ");
    gofor(men,1);
    break;
case '5':
    cls();locate(4,1);
    printf("You have selected the 'Sort' option which has not yet\n");
    printf("been implemented. Press RETURN for menu ");
    gofor(men,1);
    break;
} /* end of switch loop */
} /* end of menu do - while loop */
while ( men[0] != '6');
} /* end of main */
#include "stdlib.c"

```

```

/* ---- "CPOWER" by John P. Lewis ---- */
/* ***** NOTICE! ***** */
/* this listing is not a stand alone program but for insertion into
program from "C POWER, PART 3 */

```

```

#define LEN1 25 /* length of field for compny & name fields */
#define LEN2 25
#define RECORD 128 /* record size */

```

```

unsigned fd, offset, loc, q; /*global variables to be used
by functions and main */

```

```

int init;
char ptr[LEN1], string[SIZE], c[RECORD], dummy[2];

```

```

search ( )
{
    int i, k,k=0;
    opnforrd ( );
    if (init != 1)
    {
        printf("Please stand by, I'm initializing \n");
        while ( c[0] != DEL )
        {

```

```

            seek(fd,offset,0);
            for ( i=0; i < LEN1+LEN2 ; ++i,++k)
            {

```

```

                c[i] = string[k] = getc(fd);
            }
            offset+=RECORD;
        }
    }

```

```

    init =1;q=k-(LEN1+LEN2);offset=0;
}

```

```

    srch2 ( )
{

```

```

    printf("Please enter the search 'Key' ");

```

```

gofor(ptr,LEN1-1);
loc=index(string,ptr,offset);offset = (loc/(LEN1+LEN2))*RECORD;
}
jindex (s,t,p)
char s[],t[];
unsigned p;
{
    int i, j, k;
    for ( i=p; i <= q ; i++)
    {
        for ( j=i, k=0; t[k] != '\0' && s[j] == t[k] ; j++, k++);
        if (t[k] == '\0')
            return (i); /* i=pointer if match is found */
    }
    return (-1); /* no match */
}

/* code for main begins here, note the change in string allocation
for "phone" */
char phone[11], ch; /* notice !! change phone[10] to phone[11] */

/* new code for case '1' */
case '1':
    cls();locate (4,10);offset=0;
    if (init != 1)
        search ( );
    locate (8,8);
    srch2 ( );
    label:/* we jump here from the (C)ontinue routine */
    if (loc == -1 )
    {
        cls ();locate (6,4);
        printf("No match found, press return for menu ");
        gofor(men,1);break; /* force return to menu */
    }
    seek(fd,offset,0);
    for ( i=0; i < RECORD ; ++i) /* get a record, a char at a time */
    {
        c[i] = getc(fd);
        cls();locate (2,6);printf("Data for record no. %d",offset/128+1);
        locate (4,8);
        printf("Company: ");locate (6,8); /* print headings to screen */
        printf("Name: ");
        locate(8,8);printf("Street: ");
        locate(10,8);printf("City, State, Zip: ");
        locate (12,8);printf("Phone: ");
        locate (4,18);
        for (i=0; i <=24; ++i) /* insert data under correct heading */
            putchar(c[i]);
        locate (6,15);
        for ( i=25; i <= 49; ++i)
            putchar(c[i]);
        locate (8,17);
        for ( i=50; i <= 84; ++i)
            putchar(c[i]);
        locate (10,27);
        for ( i=85; i <= 116; ++i)
            putchar(c[i]);
        locate (12,16);
    }
}

```

```

for ( i=117; i <= 127; ++i )
    putchar(c[i]);
do
{
    locate(16,1);putchar(27);putchar(74);locate(16,4);
    printf("You may (E)dit, (R)eplace, (P)rint, (C)ontinue\
    or (M)enu ");
    gofor(men,1);ch=men[0];ch=toupper(ch); /* allow for either */
    switch(ch) /* upper or lower case entry */
    {
        case 'P':
            /* lprint (); (hard copy) future implementation */
            break;
        case 'C':
            loc /= 50;loc +=1;loc *= 50;loc = jindex(string,ptr,loc);
            offset=loc/(LEN1+LEN2)*RECORD; /* bump pointer */
            goto label; /* and establish new offset */
            break;
        case 'R':
            /* future implementation */
            break;
        case 'E':
            init=0;
            locate(14,1);putchar(27);putchar(74);fclose(fd);open ();
            locate(16,4);printf("Please indicate the field you wish to edit\n");
            printf("\n(1) Company, (2) Name, (3) Street, (4) City-State-Zip, (5) Phone ");
            gofor(men,1);
            switch(men[0])
            {
                case '1':
                    locate(20,4);printf("Company: _____");
                    locate(20,13);gofor(compny,LEN1-1);seek(fd,offset,0);
                    fprintf(fd,"%-25s",compny);
                    goto done;
                case '2':
                    locate(20,4);printf("Name: _____");
                    locate(20,10);gofor(name,LEN2-1);seek(fd,offset+25,0);
                    fprintf(fd,"%-25s",name);
                    goto done;
                case '3':
                    locate(20,4);printf("Street: _____");
                    locate(20,12);gofor(street,34);seek(fd,offset+50,0);
                    fprintf(fd,"%-35s",street);
                    goto done;
                case '4':
                    locate(20,4);
                    printf("City, State, Zip: _____");
                    locate(20,22);gofor(city,32);seek(fd,offset+85,0);
                    fprintf(fd,"%-32s",city);
                    goto done;
                case '5':
                    locate(20,4);printf("Phone: _____");
                    locate(20,11);gofor(phone,10);seek(fd,offset+117,0);
                    fprintf(fd,"%-10s",phone);
                    done: seek(fd,0,2);fclose(fd);break;
            } /* end of edit switch */
        } /* end of menu switch */
    } /* end of do-while switch */
    while (ch != 'M');
    break;
}

case '2':
/* future implementation */
break;

case '3':
fclose(fd); /* close file if open (new code for case 3, to be
inserted at first line) */
break;
/* case 4 and 5 not yet implemented */
} /* end of switch loop */
while ( men[0] != '6' );
/* end of main */
#include "stdlib.c"

cls ()
{
    putchar(27);putchar(69);/* escape, "E" sequence for H89's */
    locate (row,col)
    int row, col;
    row+=31; col+=31;
    putchar(27);putchar(89); /* escape, "Y" sequence for direct */
    putchar(row);putchar(col); /* cursor control */
}

fetchc ()
{
    #asm
    MVI MVI E,0FFH /* "direct" call to CP/M Dos */
    MVI C,006H/* for I/O (input) */
    CALL 5
    CPI 0
    JZ DIR
    #endasm
    gofor (c,s)
    int s;
    char c[];
    {
        int i;
        for ( i=0; i <= s && c[i-1] != 13; ++i )
        {
            c[i]=fetchc();/* c[i]=toupper(c[i]) */ putchar(c[i]);
            if ( c[i]==8 ) /* this subroutine "calls" */
            {
                /* fetchc () for character I/O */
                i-=2;putchar(32);putchar(8);
            }
        }
        if ( i-1 == s ) /* a return or s+1 characters will */
        {
            c[i-1]='\0';/* cause this routine to return */
            if ( c[i-1]==13 )/* and will replace the return with a null */
            {
                c[i-1]='\0';/* ('\0'). This is a "C" requirement */
            }
        }
    }
}

```


Here Comes Help For WordStar

(And Other DOS 1 Applications)

Ami Atir
55 Zahal Street
Kir-On 55451
ISRAEL

"The last straw" that pulled me out of my natural laziness, and made me write this, was Mr. Howard Rubin's letter, "More On WordStar", in "Buggin' HUG" (REMark, November 1986 issue, page 9).

Like many other users, I liked the idea of subdirectories, introduced with MS-DOS 2, and was frustrated because my favorite application programs could not access files which were not in the current subdirectory.

As an enthusiastic reader of REMark, I was watching all references to this problem. I even copied and assembled Joseph Katz's CD.EXE, but this was only for the fun of trying it. For my real work I am using a far better solution . . .

I know, I know, I should have let you know about it sooner. I must admit that being lazy is one reason for this delay (and writing in English does not come easy to me). But I also noticed that some other solutions were mentioned in REMark, for instance, MSDOS DPATH in the HUG software library (885-8039-37). Even in the same issue of REMark (Mr. Eric L. Pang's article "Getting The Most Out Of Hierarchial Directories"), two more solutions were mentioned: FILEPATH, and SUPERPATH.

I can not judge those three programs. All I know about them is from reading short references in such articles. I understand that they are aimed to solve the same prob-

lem. I assume that they are useful. But I have never tested them, and probably will never try to. I am completely satisfied with SCOUT.

I first learned about SCOUT from a reader's letter in the January 1985 issue of BYTE. Thank You, Mr. Paul Crumley. This letter sparked my hope for a solution (it was long before DPATH was announced in REMark).

The letter mentioned using SCOUT on an IBM PC XT, so I wrote to COMPUTER INSIGHTS to ask them if SCOUT will run on my Z-100 (not PC). They hesitated to make a promise, but they stated that "SCOUT was using only the DOS programming interfaces". Based on their statement, I decided to risk \$29.95 (plus air mail fee to Israel), and ordered SCOUT.

I have been using SCOUT since June 1985, and it came out to be the best software investment I have ever made!! (The formula I use is very simple: Rating = Times__used/Dollars__spent.)

Using the example of directory structure given in Eric L. Pang's article, you can type the following line on DOS's command line:

```
SCOUT K=C:\PROJ_X\BUDGET
```

This will define the subdirectory C:\PROJ_X\BUDGET as an "imaginary drive K:". You can do with this drive almost every-

thing you could do with a real drive under DOS-1. You can't FORMAT it, or CHKDSK it, but you can make it your default drive (K:) or DIR K:, or COPY a file to/from it, or even invoke a program (COM, EXE or BAT) file residing in it from anywhere in your system (other default drive, other subdirectory), by prefixing the command name with the drive designation, exactly as you used to do under DOS-1. Example: K:CLOCK OFF

Assuming you have MULTIPLAN (version 1, like me) in C:\SS which is your current subdirectory, you will be able to access your MP worksheets in C:\PROJ_X\BUDGET, from within MP, just by prefixing the file name with the imaginary drive letter. For instance:

```
T(ransfer) L(oad) K:EMPL.SS
```

You can use any letter for your imaginary drive which is not used by DOS for a real drive. If you use an IBM XT with one hard disk, then the last letter used by DOS is C:, and you can use all the letters from D: to Z:.

On a Z-100, DOS reserves the letters A: through H: for its own use, even if you do not really have these drives installed, so you can use imaginary drives starting at I:. If you are using two RAMDRIVES (like me right now), which are I: and J:, then you can start with imaginary drive K:.

If you have (like me) MP worksheets that automatically read data from supporting

sheets [using X(ternal) C(opy)], and you are too lazy (like me) to go and change all the references to have the K: prefix, you can use the same command with a small addition like this:

```
SCOUT K=C:\PROJ_X\BUDGET /S
```

The /S switch puts the "K: drive" in an automatic SEARCH PATH, thus if the specified file is not found in the current directory, SCOUT will go and search for it in all the imaginary drives that you previously defined using the /S switch. (Did I forget to mention that you can define many imaginary drives to exist concurrently, by repeatedly typing similar commands on the DOS command line? Well, of course you can!)

The search facility is very useful and very flexible, more than you can guess at first thought, and I will return to discuss it later, but first let me present another switch:

```
SCOUT L=C:\DEVELOP\C\HEADER /S/R
```

This line will put the "L: drive" in the automatic search path, and will also define it as "READ ONLY". You can use the /R switch to defend your files from overwriting them by mistake, or even erasing them by using DEL L:*.* (or similar commands). Using the /R switch will protect you from such mistakes when you access the files through SCOUT (Using L:), but do not expect SCOUT to protect you if you try DEL C:\DEVELOP\C\HEADER*.*...

Before going back to continue the discussion of the search facility, let me say (before I forget) that you can remove a currently existing imaginary drive, by "undefining" it with:

```
SCOUT K=
```

Let's go back to the search facility. The currently defined search path includes all the imaginary drives that you previously defined with the /S switch. The search is performed in the same order you defined the imaginary drives. You can temporarily deactivate the search facility by using the following command:

```
SCOUT /D
```

This command will not remove the search path definition, it will just deactivate it. You can even add imaginary drives to the search path while it is deactivated. The following command will add the M: drive to the search path, and deactivate the search facility:

```
SCOUT M=C:\DEVELOP\C\LIB /S/D
```

The search path can be reactivated by the following command:

```
SCOUT /A
```

And, of course you can reactivate the search facility while defining a new imaginary drive, (putting it, if you want, in the search path), using the following command:

```
SCOUT N:\DEVELOP\C\SRC [/S]/A
```

I find this ability to have complete control of the search facility, (having it ON or OFF, having drives in the search path or out of it), and the ability to access files which are not on the currently defined automatic search path, very VERY useful.

As an example consider my case, where I have MP worksheet files in the following subdirectories:

```
B:\CBB      ;Check Book Balance
B:\BUDG     ;Budget planning
B:\CSFL     ;Cash Flow planning
```

The files in these directories may seem to be identical. Each is holding the following files:

```
JAN86
FEB86
MAR86
APR86 and so on...
```

They have the same name, but they are completely different worksheets. Being able to have the same names for different files in different directories, is part of what makes the idea of hierarchical directory so nice.

While in MP, and working on the MAR86 worksheet from the B:\CBB directory, I want it to automatically access the FEB86 worksheet, and get the LAST_BALANCE from there, but I do not want it, by mistake, to access the "same" file in B:\BUDG.

For this purpose, I define the following "drives" (if you think that I am doing it using a .BAT file, you are quite right):

```
SCOUT I=B:\CBB /S
SCOUT J=B:\BUDG
SCOUT K=B:\CSFL
```

That does it. Now, while inside MP, I can access all my files in the defined imaginary drives, and while working on my check-book balance, my worksheets can access the supporting sheets in the correct (B:\CBB) directory.

If you tend (like me) to forget what drives you have currently defined, you can enter "SCOUT" with no parameters, and SCOUT will show you, on your screen, a complete report like this one:

List of current imaginary drives:

```
I ==> b:\cbb
J ==> b:\budg
K ==> b:\csfl
```

Search path for default drive references:

```
b:\cbb
```

Of course, you cannot get this report while inside MP, so just before entering MP, I enter the following command:

```
SCOUT > PRN
```

You know, of course, about redirection!

And I have the same report on a piece of paper right in front of my nose...

There is more to tell about SCOUT's capabilities:

To my great frustration, I discovered that the DOS's PATH command does not help to access executable program files if you have them in a RAMDRIVE.

Yes! I am using right now two RAMDRIVES, I: and J:, created with MDISK.DVD from the MS-DOS 2 Programmer's Utility Pack. If I have ZDIR.COM on drive J:, and put drive J: in DOS's search path, I cannot invoke ZDIR unless J: is the default drive, or I specify J:ZDIR. (And I encountered the same problem while using the version of MDISK.DVD supplied with DOS 2).

Using SCOUT, I can overcome this problem. I just define my J: RAMDRIVE as an imaginary drive K:, and put K: in DOS's PATH, like this:

```
SCOUT K=J:\
PATH = K:\
```

Now if I try to invoke ZDIR, DOS will find it and run it from the K: drive. I find this trick very useful.

By now you should be able to guess what will be my proposed solution to Mr. Howard Rubin's problem with WordStar. Using the directory structure presented in Mr. Eric L. Pang's article, and assuming as suggested there, that WordStar and all its overlay files reside in the I:\BIN subdirectory, I will define the following imaginary drives:

```
SCOUT J=I:\BIN /S
SCOUT K=I:\DEVELOP\C\SRC
SCOUT L=I:\DEVELOP\C\SRC\MATH
SCOUT M=I:\PROJ_X\PROPOSAL
SCOUT N=I:\PROJ_X\CORRESP
SCOUT O=I:\PROJ_X\DOC
```

Assuming you have I:\BIN in DOS's PATH, you can invoke WS from anywhere in your directory structure. WS will not complain about not being able to find it's overlays, because SCOUT will find them (notice the /S switch in the J: drive definition). And from within WS, you will be able to access and edit your K:PROG1.C file or change the "logged drive" (using the L command from the main menu) to L: and edit your FIB.C file there. You can also copy (using the O command) M:RFP.WP to O:RFP.DOC, and so on.

I propose a similar setting for a two diskette drive system, where WS and it's overlays reside in A:\, and all your files reside (in subdirectories) in B:. Defining A:\ as an imaginary drive, and putting it in SCOUT's search path, can save you from "hanging", which can occur for instance when you exit from SpelStar while you are not logged to drive A:.

You may be surprised, but SCOUT can be of great help even when using "second generation" application programs. I mean programs that already know about subdirectories, and can use them, for instance LOTUS 1-2-3.

While inside LOTUS, to change subdirectory to I:\SS (to access a spreadsheet template) you have to type:

```
/FDI:\SS
```

And then to go back to your working directory you have to type:

```
/FDI:\PROJ_X\BUDGET
```

I find it much easier to type:

```
/FDJ:
```

and then, to go back type:

```
/FDK:
```

don't you agree?

Another bit of useful information before I come to conclusion: The nice booklet that accompanies SCOUT states that "SCOUT imposes a minimal but necessary overhead for its services", and suggests that you increase the number of BUFFERS by including BUFFERS=x in the CONFIG.SYS file. I recommend to follow this advice.

The overhead is hardly noticeable if you are using a hard disk, and if you think that it is objectionable while using diskette drives, you should hear my story:

I was using SCOUT with my two diskette drive system for quite a long time. I was satisfied with (almost) all aspects of its performance, but I tried to avoid using commands that would list all files in an imaginary drive.

Trying to use DIR K: instead of DIR I:\PROJ_X\BUDGET was causing the system to access the disk once for each directory entry. The time consumed was very long, and the whirring noise of the drive was tearing my heart...

No! It was NOT a bug in SCOUT. It was a bug in MS-DOS!!!

This time I was saved by a reader's letter ("A Patch For MS-DOS Version 2 Buffers") in "BUSS" #121. Thank you, Mr. Lawrence R. Steeger.

I wonder why ZDS software department did not bother to notify us MS-DOS users, about this bug and this patch, by sending a letter to each one of us. Not every user has access to bulletin boards, and even if you have, you must be lucky to come across such an important bit of information...

Patching MSDOS.SYS, as described in this letter, greatly improved the system performance, regarding disk access in general. And as to using SCOUT, the improvement was outstanding. The overhead on using SCOUT is barely noticeable, even while using a diskette drive system.

I purchased SCOUT from:

Computer Insights
P.O.B. 110097
Pittsburgh, PA 15232
U.S.A.

I sure hope they are still in existence and in operation.

I do not hesitate to recommend SCOUT to every MS-DOS 2 user. In any case, I think that it will be nice if one of REMark's readers or columnists could "test run" and compare both SCOUT and the other products mentioned, and report the results to REMark's readers.

General

Expanded HUG Discount List Featuring Products Eligible For Discount To HUG Members Only

HK-200	10%	Z-417-1	10%	ZF-148-42	20%
HS-148-2	10%	Z-445	10%	ZF-171-42	20%
HS-158-1	10%	ZA-170-1	10%	ZF-248-81	20%
HS-158-W	10%	ZA-170-2	10%	ZFL-181-93	20%
HS-248-S	10%	ZA-170-3	10%	ZMM-1470-G	20%
HS-248-SW	10%	ZA-170-4	10%	ZSS-100-27	20%
HS-317-20	10%	ZA-180-20	20%	ZVM-121-1	10%
HVM-1220-A	10%	ZA-180-21	10%	ZVM-135	20%
HWD-20	10%	ZA-180-35	10%	ZVM-135-1	10%
HWD-20-AT	10%	ZA-181-3	10%	ZVM-135-2	10%
TM-140	10%	ZA-181-4	10%	ZVM-1200-1	10%
TM-150	10%	ZA-181-5	10%	ZVM-1220-A	20%
TM-158	10%	ZA-181-7	10%	ZVM-1230-A	20%
TM-170	10%	ZA-181-8	10%	ZVM-1240-A	20%
Z-205-4	10%	ZA-181-9	10%	ZVM-1300-1	10%
Z-207-7	10%	ZD-12	10%	ZVM-1330	20%
Z-304	10%	ZD-200	10%	ZVM-1380-C	20%
Z-316-8	10%	ZD-400	10%	ZW-148-42	20%
Z-319	10%	ZDE-1211-AO	20%	ZW-248-82	20%
Z-405-1	10%	ZDE-1217-AO	20%	ZW-248-84	20%
Z-409	10%	ZDH-1201-AO	20%	All software on Pages 91, 92, and 93 of the Winter	
Z-416	10%	ZDH-1211-AO	20%	86-87 Heathkit Catalog	20%
Z-417	10%	ZF-148-41	20%		

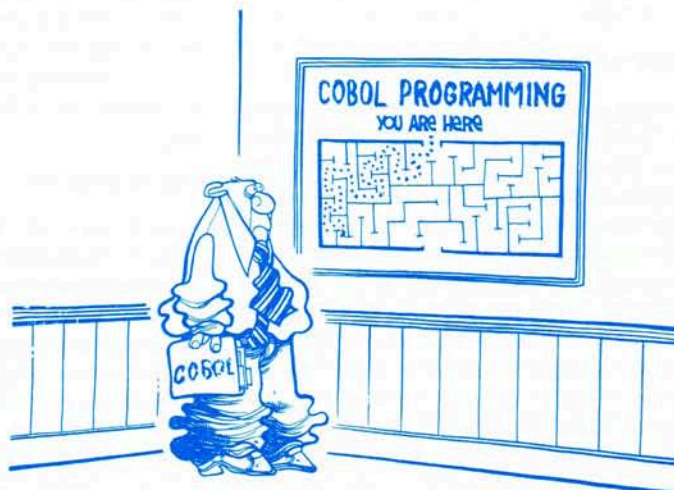
Products and prices appearing on list subject to change without notice.

COBOL Corner XVIII

Final

H. W. Bauman

493 Calle Amigo
San Clemente, CA 92672



Note! "COBOL Corner" readers, this is the LAST article in this series! This is because of the falling interest. Neither REMark or the author believe that with the limited space, the series warrants going on. I would like to have the readers and users of "COBOL Corner" read the questionnaire at the end of this article and supply your answers to the author for future use in other series.

Introduction

I hope that the users of "COBOL Corner" have completed and understand the article XVII Project. This project will be similar, using the same Input Disk Data and the same two hard-coded tables. Also, the Output Report should look the same. Therefore, if you obtained good results with the last program, the user will find that this project will be easy to code. There will only be a few changes in the WORKING-STORAGE SECTION and the PROCEDURE DIVISION. However, do NOT try to do this program until you have successfully completed the previous project. Again, please do not forget to do the Planning Phases before starting to code this program. You might think that you are saving time, but you will find that as you get into Advanced COBOL Programs, you will find yourself lost! So, take advantage of Planning these easier programs for the learning process BY DOING.

Program Differences

This program will use the Binary Search with INDEXES and the SEARCH statement.

The WORKING-STORAGE SECTION will require the following changes:

1. The MONTH Table, which is hard-coded, will not change.
2. The Input Disk Data will not be changed.
3. The hard-coded DEPT-NBR-NAME Table will have some small changes, starting with a REDEFINES statement:

```

05 DEPT-NBR-NAME-TABLE REDEFINES DEPT-NBR-NAME-DATA
10 DEPT-TABLE-ENTRY OCCURS 110 TIMES
                        ASCENDING KEY TD-DEPT-NBR
                        INDEXED BY ST-INDEX.
                        15 TD-DEPT-NBR PIC 9(04).
15 TD-DEPT-NAME PIC X(17).
  
```

these two items, you will know that the program is using the Binary Search. Can you find these statements or phrases in the above coding? Also, look up the KEY phrase and the SEARCH ALL statement in the COBOL manual.

2. Binary Search is much faster than a Serial Search if the Table has over thirty entries. Serial Search is more efficient for smaller tables.

The PROCEDURE DIVISION will have the following changes in the coding:

```

210 LOOKUP-DEPT-TABLE.
    SEARCH ALL DEPT-TABLE-ENTRY
      AT END
        MOVE "NO " TO TABLE-ENTRY-FOUND-SW
      WHEN
        TD-DEPT-NBR(ST-INDEX) IS EQUAL TO DP-DEPT-NBR
        MOVE "YES" TO TABLE-ENTRY-FOUND-SW.
  
```

Discussion

Let's summarize what we know about the Binary Search at this point:

1. The Binary Search requires use of the KEY clause in the WORKING-STORAGE SECTION, and the ALL phrase in the SEARCH statement (SEARCH ALL) in the PROCEDURE DIVISION. Therefore, if you look at a COBOL program and find
3. The Binary Search REQUIRES that the table arguments be in a sequential order.
4. When the ALL phrase is specified to obtain a Binary Search, the SEARCH statement handles the Index initialization. Therefore, the COBOL programmer need not code a SET statement, as for a Serial Search.
5. The WHEN phrase requires specifica-

tion of the INDEX-KEY field as the first entry, and the CONDITION TEST is limited to an EQUAL relationship. If multiple WHEN conditions are written, they MUST be specified with AND connectors. This means that all WHEN conditions MUST be satisfied to end the search prior to the END-OF-TABLE.

Binary Search

When a table has many (over thirty) entries, a Serial Search for arguments whose entries are located deep down in the table become time consuming. The Binary Search first comparison is made in the middle of the table (rather than the first argument in the table as what is done for a Serial Search). Then the table is split into two halves and either the top half or the bottom half is searched depending on the relationship of the search argument to the mid-point table argument. If the search argument is greater than the middle table argument, the upper half of the table will be used. Conversely, if the search argument is less than the middle table argument, the lower half of the table will become the search area. Next, the search argument will be compared with the mid-point argument of the above selected half of the table. Again, the chosen half will be split into two halves. Depending on the result of the comparison, the top or the lower half will be checked as above. This halving procedure will be repeated until an EQUAL table argument is found or until there are no remaining portions of the table to divide. Draw yourself a little sketch of this procedure to better visualize the process. Note: This Binary Search CAN-NOT be used with a table organized randomly. This is where the user must use an internal or external COBOL SORT. Not all COBOL compilers have this SORT statement. If this is true, an external SORT process must be used on the table first. Advanced programs must provide for this SORT function to ensure that the table will be in ascending or descending order. MS-COBOL has a SORT statement for this purpose, but another software MS-SORT package must be used.

New COBOL Binary Phrases

The KEY Phrase:

When a table is used in a Binary Search, the KEY phrase MUST be specified to indicate whether the table arguments are arranged in ascending or descending order. The format is shown below:

```
OCCURS integer TIMES
  ASCENDING
```

```
KEY IS data-name
  DESCENDING
  INDEXED BY index-name
```

Recognize that the use of the KEY phrase does not ensure that the table arguments are in the desired order. It is the COBOL programmer's responsibility to MAKE SURE that the table is arranged in accordance with the KEY phrase specification. (You will require further study into this detail if the user is planning to go on with advanced programs using the COBOL SORT coding).

The ALL Phrase:

Specification of ALL triggers the Binary Search Logic. Binary Search is somewhat harder or more complex to code than the Serial Search. The WHEN phrase requires specification of the Index-Key field as the first entry, and the Condition Test is limited to multiple WHEN conditions. They MUST be specified with AND connectors. This means that all WHEN conditions MUST be satisfied for the search to end prior to END-OF-TABLE. The format for the SEARCH ALL statement is shown below:

```
SEARCH ALL identifier-1
  AT END imperative-statement-1
    data-name-1 IS EQUAL TO identifier-2
    WHEN        literal-1
    condition-name-1 IS EQUAL TO arithmetic-expression-1
    AND data-name-2 IS EQUAL TO identifier-3
    condition-name-2 literal-2
    imperative-statement-2 IS EQUAL TO arithmetic-expression-2
  NEXT STATEMENT
```

When the ALL phrase is specified to obtain Binary Search, the SEARCH statement tells the COBOL compiler to handle the index initialization; thus, the SET statement is NOT required.

Notice that we have been working with more than one table, but they have been what is called Single-Level Tables. Single-level tables are used the most. There are certain applications that will require Multiple-Level Tables and they can have two or three dimensions. For example, a table containing a Part Number, Part Description, and Part Price is a single-level table with one table argument and two table functions. An example of a multiple-level table — two-level table — would have Pay Rates for various job classifications that vary during each of three shifts. The Hourly Pay Rate would depend on two factors — job classification and shift worked.

The two-level table would use two OCCURS clauses and, if indexes are used, two

INDEXED-BY clauses. If subscripts are being used, two subscripts would be required. The user will encounter these in advanced programming and best learned by doing. There is not enough article space to get into advanced programming that would teach a novice! At this point, I want to make the users aware of multiple-level tables to answer any questions that might occur.

Conclusion

For our preliminary exposure to tables, I want to tell the users about another type of programming logic. Sequential lookup coding can be simplified with the VARYING phrase of the PERFORM statement. I will let the user try another way of doing this project using subscripts and the PERFORM/VARYING statement logic. This program could use the same disk input data and it will again use two hard-coded tables. This means that if the user uses the new logic with the same data, he/she will obtain the same Output Report. I am sure that if the users have been working with all the information that "COBOL Corner" ar-

ticles and their COBOL manual have provided, they can do this project without further help. Be sure to try it. The user will then have three programs using the same data, but each of the three will use different logic. All three should produce the same Output Report. Use these three projects to compare the three coding methods for efficiency, program speed, and complexity of the coding. Please give this a try. If the user can do this, the effort of learning COBOL to this point has been worth while!

COBOL Corner Questionnaire

COBOL or COMmon Business-Oriented Language has been the mainstay programming language of large computer systems in government and corporate data-processing departments, and it does NOT appear that COBOL will soon become obsolete. Huge numbers of COBOL programs, worth many millions of dollars, have been written. Thus, many companies and

government agencies will continue to use COBOL. Now it is possible to duplicate COBOL programs on the PC type computers and they will make a great training ground for future jobs in industry. It is unlikely that American business will scrap this language just because it is not as elegant or compact (no longer a problem with low cost RAM memory and storage devices) as some of the high-level languages like PASCAL or C, for example. The government is trying ADA for the future. It will take many years to see if it goes. COBOL makes heavy use of English words in its instruction set because it allows word lengths of as many as 30 characters. This makes COBOL programs easy to understand and to maintain if the programmer provides enough comments in his coding and uses good structure. Its structure is not conducive to compact programs, but this is not a problem in the current computers used in industry. Just pick up the Want-Ads section of a newspaper and look at the help wanted section and find out how many organizations are looking for COBOL programmers.

Now that COBOL compilers are available for the desktop computer demonstrates that COBOL is alive and well in the micro-computer Real-World. It was "COBOL Corner" hope and desire that it would be used by the readers to get positions in the data-processing industry or promoted. After all, the real large data-processing programs are just many small programs all linked together to solve many types of computer problems. "COBOL Corner", as the users will know, consisted of a series of tutorial articles presented in a step-by-step arrangement to teach COBOL to any readers that were willing to work with each article. The series started with novice programs many months ago and these were followed by a series of intermediate programs that we are finishing now. The next series of advanced programs are getting difficult to fit into REMark's space so the staff and the author have decided that we would discontinue "COBOL Corner" at this point. We have covered a lot of ground. The articles have touched on all types of accounting programs, forecasting, planning, databases, and even a hint about spreadsheets.

The letters that I have received from the users tell me that the main reason for their interest was self improvement for their present job or for a promotion to another job. I have also received letters from college instructors telling me that they have used "COBOL Corner" in their computer science classes. What does this mean?

What am I leading up to? The next step for the "COBOL Corner" users would be to suggest that they do each and every project over and over until they can do each program without effort. When the user can do this, they will be ahead of most programmers starting out in the Real-World. The COBOL programs also prepare the user for other languages, as well. Every programmer should always start with a Planning Stage, followed by a Coding Stage, and finally a test run to debug the program. The logic of all languages is similar.

I have four questions I would like to have the readers answer and send me their answers. These questions will not take much of your time, but would help the author plan future series of articles. The answers will fit on a postal card. Will you supply the answers? The future of tutorial articles in REMark is in your hands by whether you DO answer and HOW you answer! Have you any questions that I have left out? If so, let me know. If you would like a personal answer from me, PLEASE provide a SASE (business size) to help me answer you. Here are the questions:

1. Do you read "COBOL Corner"? If so, do you work with the many projects? If not, why not?
2. The author and HUG offered a disk with the completed programs and data for this series of articles. I hear that HUG had a goodly number of sales. I know that the experienced users could do the projects without the disk. Would the readers like a similar disk for other tutorial articles?
3. Now is the time for REMark readers to express themselves. Do you like tutorial articles? How can they be made more useful? I am looking for both the PRO and CON suggestions. Have the users LEARNED anything about COBOL? Have the articles helped the users with their work or to get a new job?
4. Assuming that the readers want tutorial articles to continue, we need to know what type of computer set-ups the users have and if any changes in the set-up are being planned soon. I need to know the type of computer the user will have, the amount of working memory RAM, the type and capacity of disk devices, type of printer, and what software you have or what would you be willing to purchase.

I will be awaiting the users guidance! As I am sure most of the REMark readers know, these articles require a lot of time to prepare. As your author, I get my compensa-

tion from the users' letters and knowing that I am helping new computer users get started. Without this satisfaction, I could be doing other more rewarding things. I really like working with computers even though I have been in this field for many years. I only wish that these tutorial articles were available when I was starting out. I am lucky that I do not have to make a living doing this now.

Again, PLEASE provide the answers as soon as possible whether they are PRO or CON! These will help to make the pages of REMark available to the best advantage of the majority of the serious readers. This is what everyone wants, I am sure. I want to thank all the "COBOL Corner" users and the staff of REMark for making these articles rewarding! Thanks everyone!



HARDWARE HEATH/ZENITH

SURPLUS - SALVAGE - ETC.

Most Circuit Boards for the Heath/Zenith Line.
Z-100, Z-241, Z-158, Z-151, H-89, Z-171,
Z-148. Cabinets, Disk Drives, Power Supplies,
C.R.T.'s, Keyboards, Cables, H-DOS Operating
System. \$5.00 HTX-11 Keyboards \$2.50 Ea.
(NEW) 20 or More \$1.00 Ea. (NEW) L.C.D.
171 \$20.00 Ea. (NEW) 171AC Power Supply
\$25.00 Ea. (NEW) Modem 171 \$100.00 Ea.
Coaxial Cable (for Networking) 100 ft. Long
\$35.00 Ea. 25 ft. Long \$20.00 Ea. Call or
Write for a Price List - Continually Getting More
Stock Each Month - Shipped U.P.S. - C.O.D.

SAVE THIS AD!

Self Addressed Stamped Envelope to:
Al Davis
4894 Lake Chapin Road
Berrien Springs, MI 49103
Phone: (616) 471-1792



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?



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.



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.

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.



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



data systems

COMPUTERS

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

VIDEO

Paradise Auto Switch EGA	\$429.00
Video 7 EGA Deluxe	\$429.00
Video 7 EGA High-Resolution Color Card	\$369.00
Video 7 MGA High Res. Monochrome Graphics	\$189.00
NEC Multisync High Res. Color Monitor	\$599.00
C. ITOH CM1000 Color Monitor	\$399.00
Mitsubishi EGA High Res. Color Monitor	\$429.00
Premier Technologies Z148 1 1/2-Slot Board	\$119.00
ZVM-1230-A: Non-Glare, Green Monitor	\$ 99.00
ZVM-1240: Non-Glare, Amber Monitor, TTL	\$159.00
ZVM-1200-1: Tilt Swivel Base	\$ 17.00
ZVM-1330: High-Resolution Color, RGB	\$459.00

DYNAMIC RAM MEMORY

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

MODEMS

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

PC COMPATIBLE UPGRADE ACCESSORIES

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

Z100 UPGRADE ACCESSORIES

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

PRINTERS

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

Z200 UPGRADE ACCESSORIES

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



Box of 10 With Lifetime Guarantee

Nashua Diskettes: DS/DD, 48 TPI, Soft-Sector	
or Hard-Sector (Specify)	\$950
Nashua Diskettes: DS/DD, 96 TPI, Soft-Sector	
or Hard-Sector (Specify)	\$21.50
Nashua Diskettes: DS/HD, SS, 600 Oersted Media	
for AT and Compatible 1.2 MB Drives	\$27.95

HARD DISK DRIVES

With One-Year Replacement Warranty

SEAGATE ST 225 20MB Hard Disk	\$299.00
SEAGATE 20 MB HD w/OMTI	\$399.00
OMTI/LAPINES 20 MB Hardcard	\$449.00
SEAGATE ST238 30 MB w/OMTI RLL Controller	\$499.00
SEAGATE ST251 50 MB Half HT HD w/OMTI Controller	\$899.00
SMS 25 MB Tape Backup w/One Tape	\$99.00
SMS 60 MB Tape Backup w/One Tape	\$49.00

FLOPPY DISK DRIVES

With One-Year Replacement Warranty

FUJITSU M2551A: 5 1/4, Half Ht, 48tpi, DS/DD	\$112.50
MF501A: 5 1/4" HalfHeight, 48 TPI, DS/DD	\$119.00
MF504A: 5 1/4" Half Height, 96 TPI, 1.2 MB	\$159.00
5 1/4" 360K External Floppy Drive for ZF-181	\$199.00

SOFTWARE

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

OTHER HARDWARE

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

TERMS

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



RAM Technology

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

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

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



International HUGCON87

HUG • Hilltop Road • St. Joseph, MI 49085 • (616) 982-3463

10th Anniversary Celebration!

INTERNATIONAL HEATH/ZENITH USERS' GROUP CONFERENCE

O'Hare Hyatt Regency
Rosemont, Illinois
August 21, 22, 23, 1987

Name(s): _____

Company: _____
Address: _____
City: _____ State: _____ Zip: _____

Enclosed is \$27.00 for each of the individuals listed above to attend the International HUG Conference being held the weekend of August 21, 22, and 23, 1987. Please send tickets along with information regarding hotel reservations and transportation.

Amt. Enclosed: _____ No. Attending: _____

For Our Information:

Which Heath/Zenith computer do you now operate? _____

Are you a Non-User-Attendee? Yes No

Are you a computer related manufacturer? Yes No

If yes, would you like exhibit information? Yes No

Are you, or anyone in your party, interested in activities in or around the Chicago area other than the Conference? Yes No

If yes, please indicate any suggestions you may have: _____

Special Notice To Exhibitors:

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

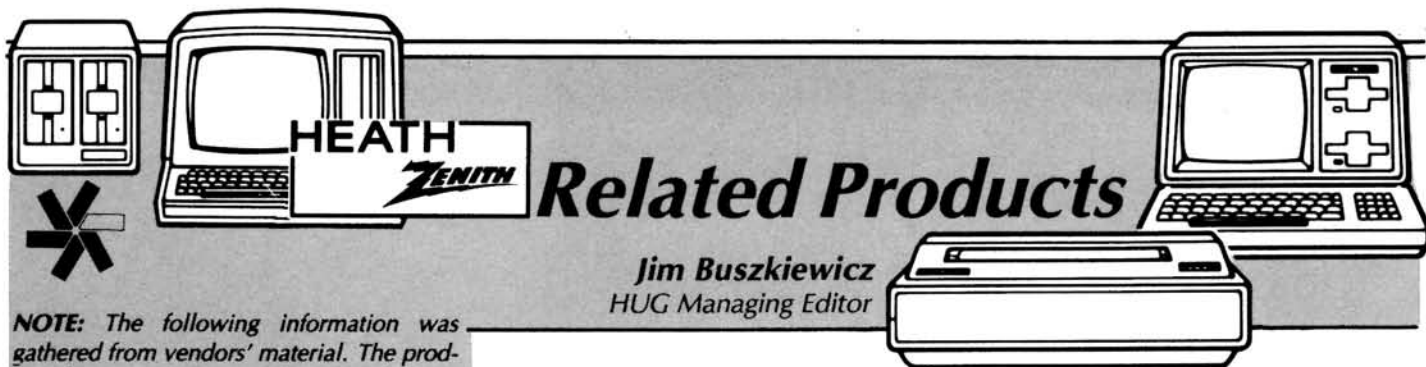
For Your Information:

The \$27.00 you are paying for your reservation to the International HUG Conference entitles you to all functions of the Conference. Visitor tickets, for those of you simply attending the seminars and exhibits, are available for \$12.00. Visitor tickets do not include eligibility for prizes or food while attending the Conference.

Please send your completed registration form or suitable copy to:

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

Registration(s) must be post marked no later than July 31, 1987. Cancellation will not be accepted after this date.
Sorry, We cannot accept purchase orders



NOTE: The following information was gathered from vendors' material. The products have not been tested nor are they endorsed by HUG. We are not responsible for errors in descriptions or prices.

SimpleWare of Glendale Heights, Illinois, has announced the introduction of a new interactive learning program called TUTOR. This program is designed to teach the beginning user how to use the Personal Computer and Disk Operating System. TUTOR takes a very simple, systematic approach in developing awareness and understanding as it uses plain and simple language to teach the user the complexity of the computer hardware and the Disk Operating System. TUTOR is comprised of 5 menu driven modules, hardware overview, DOS overview, DOS command line, DOS transient commands, and references.

TUTOR is a powerful education tool with such features as:

- Written in easy to understand language
- Menu Driven and very user friendly
- On-line HELP (F1 key) Glossary
- Operates on Color Graphics or Monochrome video cards
- Full color support - To highlight key areas
- Illustration - Pictures and Graphics
- Interactive Lessons
- Operates on all Heath/Zenith PC compatibles

TUTOR requires MS-DOS 2.0 or higher, and 256k of RAM. For more information, contact SimpleWare, P.O. Box 5146, Glendale Heights, IL 60138-5146, (312) 351-3238.

Now available from **MICRO 1** is the 286 SPEED CARD. This card lets your PC compatible system run at AT speeds without any loss of compatibility. This is the first accelerator card to achieve 100% compatibility with all hardware and software available. A speed change switch is provided on the back of the card to switch the 80286 processor between 7.2 MHz and 4.77 MHz. The card is compatible with Heath/Zenith PC compatibles, and sells for \$575. For more information, contact MICRO 1, 557 Howard Street, San Francisco, CA 94103, (415) 974-5439.

BV Engineering has just released version 3 of its' PCPLOT high resolution graphics program. Now PCPLOT not only makes linear and logarithmic plots, but it will also plot line graphs with error bars, stock market charts, bar charts and stacked bar charts. PCPLOT supports two independent Y-axes which can be scaled to different data sets. Data points can be connected with dotted, dashed or solid lines. Open plots, grid lines, or "tickmarks" may be specified. You can mix line graphs, bar charts, stacked bar charts, stock market charts and error bars on a single graph. X and Y data may be separately scaled. Legends can be specified on an individual data file basis. Alpha-numeric labels can be placed anywhere on the plotting surface. AUTO features enable PCPLOT to take instructions from a file instead of the keyboard. PCPLOT is compiled in machine code to run fast — a complete semi-log graph of two data files each containing 200 data points takes only one minute on a stock IBM PC.

PCPLOT version 3 sells for \$95 and requires the MS-DOS operating system. For more information, contact BV Engineering, 2200 Business Way, Suite 207, Riverside, CA 92501, (714) 781-0252.

Spectre Technologies Inc. of Woodland Hills, CA announces three new software products for Heath/Zenith computer systems. The first, is a pop-up type program for CP/M computers, called PRESTO. When a special 'trigger' key is pressed, PRESTO halts the current program, opens a window on-screen, and waits for a command. PRESTO offers the following features:

- A floating point calculator
- A programmer's calculator (hex, binary, octal, decimal)
- Screen dumps — to printer or to a file
- A perpetual calendar (with clock, time and alarm clock, if computer has a clock)
- A Rolodex-like function for cataloging names, addresses, phone numbers or appointments
- Access to CP/M's features — copy, erase and rename files and perform directories

- A complete keyboard macro program which lets you define their own special function keys at any time

Versions of PRESTO are now available for the Heath/Zenith H8/H19 and H/Z89/90 computer systems running CP/M. PRESTO is priced at \$39.95 and comes complete with a fully typeset easy-to-understand users manual.

Also available is a new product called LONG & LOUD!. Originally called TWIST & SHOUT, this program is a sideways and banner printing software package for all CP/M and MS-DOS computers. The following enhancements and improvements have been incorporated:

- Better screen handling
- Easier installation
- More printer support
- Four type sizes in sideways printing mode
- Five type styles in banner mode (Times, Sans Serif, Olde English, Script and Symbols)
- Foreign character set mode

LONG & LOUD! is available for all Heath/Zenith computers and is currently priced at \$34.95.

Finally available for the H8/H19 and H/Z89/90, is a software product called REMBRANDT. The REMBRANDT Complete Business Graphics Toolkit allows the CP/M computer user to:

- Draw freehand on the screen
- Create bar charts, pie charts, xy plots
- Print graphics on most dot-matrix and daisy wheel printers
- Create on-screen electronic slide shows
- Integrate text and graphics on the printed page

REMBRANDT is priced at a low \$39.95 and comes complete with a fully typeset easy-to-understand user manual. For more information about these three software products contact David Grenewetzki, Spectre Technologies Inc., 22458 Ventura Boulevard, Suite E, Woodland Hills, CA 91364, (818) 716-1655. *

Don't Get HYPER or
CROSS because your
modem software is
too complex to use,
get HUGMCP for the
no-hassle modem
connection.



Hilltop Road
Saint Joseph, Michigan 49085

BULK RATE
U.S. Postage
PAID
Heath Users' Group

POSTMASTER: If undeliverable,
please do not return.

P/N 885-2089