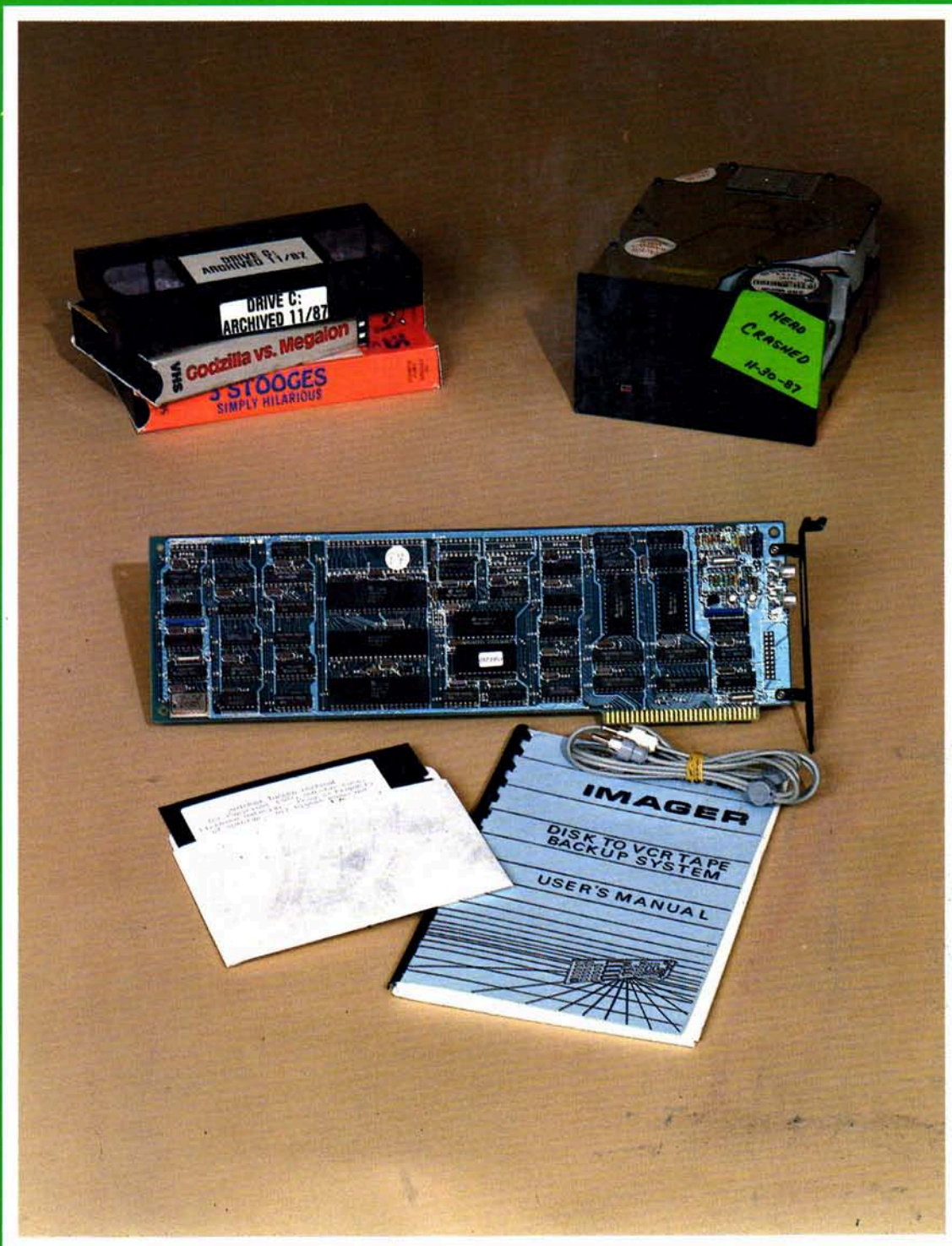


REMark®

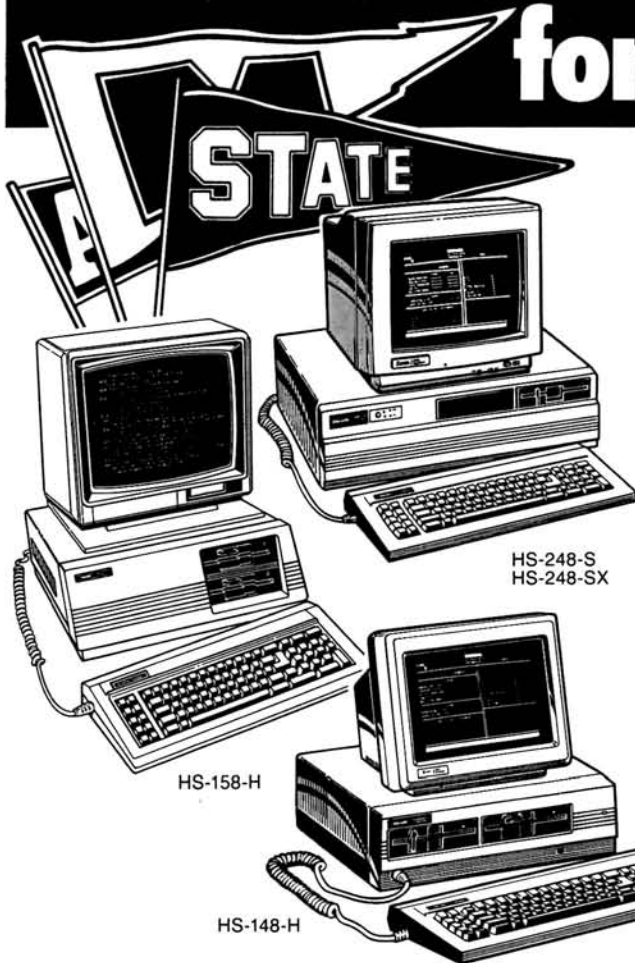
Volume 8, Issue 11 • November 1987

P/N 885-2094 Issue 94

\$2.50
Disk-To-VCR Tape
Back-Up System
See Page 45



BACK-TO-SCHOOL prices for kit computers



HS-248-S
HS-248-SX

HS-158-H

HS-148-H

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

And look what you get!

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

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

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

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

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

HUG Discount Does NOT Apply!

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

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

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

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

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

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

San Diego (La Mesa), 92041
8383 Center Dr.
619-461-0110

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

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

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

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

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

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

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

Jacksonville, 32211
9426 Arlington Expressway
904-725-4554

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

Tampa, 33614
4016 W. Hillsborough Ave.
813-886-2541

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

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

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

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

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

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

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

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

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

Rockville, 20852
5542 Nicholson Lane
301-881-5420

MASSACHUSETTS –
Peabody, 01990
242 Andover St. (Rt. 114)
617-531-9330

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Cleveland, 44122
28100 Chagrin Blvd.
216-292-7533

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

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

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

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

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

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

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

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

TEXAS – Dallas, 75218
12022C Garland Rd.
214-327-4635

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

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

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

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

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

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

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

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

**Phone orders
accepted.**

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

Heath® ZENITH®
Computers & Electronics

Units of Veritechnology Electronics Corporation

HZC-341



The other cats get to sing along!

That's because HEPCAT runs **with** your other programs, not **over** them. HEPCAT (HUG Engineer's and Programmer's Calculation Tool) is a powerful pop-up calculator for all Heath/Zenith MS-DOS and Z-DOS based computers. Unlike other pop-up calculators, HEPCAT does not stop the currently running program while it is popped up. That means that you can do calculations while your computer is busy with something else. For example:

- While Lotus (tm) is loading a huge spreadsheet, you can check your kid's math homework.
- While Dbase (tm) is sorting a large database, you can add up some grocery prices.
- While your computer is busy compiling one program, you can work on number base conversions needed for another program.

HEPCAT is safe to pop-up during just about any running program — even during disk activity. And HEPCAT has other features the other guys can't touch.

HEPCAT gets along with everyone . . .

HEPCAT supports more video configurations than any other pop-up, and always

pops up in the current video mode, rather than forcing the screen into a text mode as other pop-ups do. It also works properly with more programs than any other pop-up. You can pop up HEPCAT over Microsoft Windows (tm) and many other programs that other pop-ups can't work with, and even over some other pop-ups.

HEPCAT works harder . . .

HEPCAT provides a multi-function floating point calculator and a programmer's binary calculator that work together to do more than the basic four (+, -, *, /). The floating point calculator includes the following built-in functions: powers, pi, factorial, square root, sine, arc sine, cosine, arc cosine, tangent, arc tangent, log (natural and base 10), e^X and 10^X . It also includes the following conversions: degrees-radians, radians-degrees, Celsius-Fahrenheit, Fahrenheit-Celsius, centimeters-inches, inches-centimeters, meters-feet, feet-meters, kilometers-miles, miles-kilometers, grams-ounces, ounces-grams, kilograms-pounds, pounds-kilograms, milliliters-fluid ounces, fluid ounces-milliliters, liters-quarts, quarts-liters. The binary calculator works in these number bases: binary, tetral (base 4), octal, split octal, decimal, and hexadecimal; and it supports

these operations: MOD, AND, OR, XOR, SHL, SHR.

The HEPCAT floating point calculator supports 8 significant digits and can display numbers four ways: floating point, fixed point, scientific notation, and engineering notation. Numbers are handled internally in BCD format to eliminate binary round off errors in addition and subtraction.

HEPCAT eats less . . .

HEPCAT uses less than 16k of memory — less than any other pop-up calculator that we know of. It also uses less than 16k of disk space, so you don't have to worry about where to put it on a small system. The HEPCAT window uses less screen space, too. It shows you more real information than other pop-up calculator displays, but it doesn't waste space by showing you a keypad layout. You already know what your keypad looks like! HEPCAT is easier to learn, too, with commands that make sense.

If you are tired of pop-ups that can only sing solo, give HEPCAT a try. HEPCAT is available from HUG as part no. 885-3045-37 for \$35.00. It works on any Z-100 PC, Z-200 PC, or Z-100 (not PC) system and any version of MS-DOS or Z-DOS.

Heath/**ZENITH** Users' Group

Managing Editor Jim Buszkiewicz
(616) 982-3837

Software Engineer Pat Swayne
(616) 982-3463

Software Coordinator Nancy Strunk
(616) 982-3838

Production Coordinator Lori Lerch
(616) 982-3794

Secretary Margaret Bacon
(616) 982-3463

HUG Bulletin Board (616) 982-3956

Contributing Editor William M. Adney

Contributing Editor ... Harold W. Bauman

Contributing Editor Joseph Katz

Printer Imperial Printing
St. Joseph, MI

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

* U.S. Funds

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

Send Payment to: Heath/Zenith Users' Group
P.O. Box 217
Benton Harbor, MI 49022
(616) 982-3838

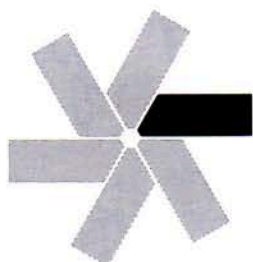
Although it is a policy to check material placed in REMark for accuracy, HUG offers no warranty, either expressed or implied, and is not responsible for any losses due to the use of any material in this magazine.

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



REMark®

Buggin' HUG
..... 7

Mainstream Computing
Joseph Katz 11

The Heath IC-1001 Logic Analyzer
Terry Perdue 21

C/80 — Eliminate Library Duplication
William M. Adney 29

The Heath Voice 2000
Terry Perdue 35

Expanded HUG Discount List
..... 39

HUG New Products
..... 40

HUG Price List
..... 42



Index of Advertisers

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

Real Men Don't Back-Up (Or Do They?)	
<i>Jim Buszkiewicz</i>	45
Making "Monkey See, Monkey Do" Livable Or How To Fix The H/Z-386 'Super' Keyboard	
<i>Jim Buszkiewicz</i>	48
An Introduction To TSRs	
<i>Pat Swayne</i>	51
Protected Input For C	
<i>Gary A. Appel</i>	57
C__Power — Part 8	
<i>John P. Lewis</i>	73
SPREADSHEET/DATABASE Corner Review #3	
<i>H. W. Bauman</i>	77
Heath/Zenith Related Products	
<i>Jim Buszkiewicz</i>	83

Reader Service No.		Page No.
102	Anapro	55
150	Dante Bencivengo	34
126	Al Davis	75
163	Domino Computer Services, Inc. ...	37
104	FBE Research Company	20
159	HEPCAT	3
106	HUG PBBS	76
161	Heath Company	46,47
143	Heath Education	50
107	Paul F. Herman	81
108	Hogware	9,39
137	Jay Gold Software	55
111	KEA Systems	34
114	Micronics Technology	41
117	Payload Computer Services	28
119	S&K Technology	20
121	Scottie Systems	19
122	Secured Computer Systems	22
124	Spectre Technologies, Inc.	38
153	Surplus Trading Company	37
131	Veritechnology Electronics Corp.	2
133	Bary A. Watzman	27
154	WindowDOS Associates	41

On The Cover: On page 45, The Light-Pen Company has a new product with a special deal. It's called "Imager", and lets you back up your disks onto VCR tapes. Photo by Jim Buszkiewicz.

BUGGIN' HUG

CONDOR Cataloging System

Dear HUG:

Just finished reading the May 1987 issue. As always very informative. I was especially interested in the article by Don Breslauer regarding CONDOR III. I use CONDOR III extensively both in my office, as well as at home. This has got to be one of the best, and more importantly, easiest databases to work with. While I am not a very good programmer, I have been able to turn some fairly fancy tricks with CONDOR.

Don Breslauer had a very good idea in establishing a cataloging system for REMark articles. Attached is my version of the same thing. I believe Don went a little too far in creating command files. To be sure they are necessary in the CONDOR system. However, he should have just let the system handle some of the work for which he created command files, such as ENTER, UPDATE and DISPLAY. By merely pressing "END" at these routines, I return to the menu. My exit procedure, allows me to return to the default rather than the logged on drive at the DOS.

I would appreciate hearing from others who use CONDOR. Also, how about more articles regarding this relational database system and as to how it runs on a Z-120.

Peter W. Koch
8-29B Pinehurst Drive
Lakewood, NJ 08701

Figure 1

```
***** REMark MENU *****
1. Add records [ENTER REMARK]
2. Update records [UPDATE REMARK]
3. Display records [DISPLAY REMARK]
4. LIST/PRINT REMark by Author
   [RUN AUTHOR]
5. LIST/PRINT Remark by Subject
   [RUN SUBJECT]
6. Exit REMark to CONDOR or DOS
   [RUN EXIT]
```

Figure 2

```
;Exit procedure from database to
CONDOR III or DOS
;Enter applicable default drive
i.e. A, B, C, etc. on line 12
;Syntax of RUN command is: RUN EXIT
*Message Select "C" to remain in
CONDOR or "D" to exit to DOS default
drive
```

```
*LET $1 = C
*LET $1 = D
*GET $1
*IF $1 = C
ABORT
*ENDIF
*IF $1 = D
A:
SYSTEM
*ENDIF
```

Figure 3

```
;Procedure lists REMark by author
;Syntax of RUN command is: RUN AUTHOR
*Message Enter author in quotation
marks
*GET $1
*Message select "L" for LIST or "P"
for PRINT
*LET $2 = L
*LET $2 = P
*GET $2
*IF $2 = L
SELECT REMARK WHERE AUTHOR IS $1
SORT RESULT BY TITLE MONTH YEAR PAGE
LIST RESULT BY AUTHOR TITLE MONTH YEAR
PAGE
*ENDIF
*IF $2 = P
SELECT REMARK WHERE AUTHOR IS $1
SORT RESULT BY TITLE MONTH YEAR PAGE
TITLE TOP,50,"REMark Articles by ",$1
PRINT RESULT BY TITLE MONTH YEAR PAGE
*ENDIF
HELP REMARK
```

Figure 4

```
;Procedure lists REMark by subject
;Syntax of RUN command is: RUN SUBJECT
*Message Enter subject enclosed in
quotation marks
*GET $1
*LET $2 = L
*LET $2 = P
*GET $2
*IF $2 = L
SELECT REMARK WHERE SUBJECT1 = $1 OR
SUBJECT2 = $1 OR SUBJECT3 = $1 OR
SUBJECT4 = $1 OR SUBJECT5 = $1
SORT RESULT BY AUTHOR TITLE MONTH YEAR
LIST RESULT BY AUTHOR TITLE MONTH YEAR
APPLICATION
*ENDIF
*IF $2 = P
SELECT REMARK WHERE SUBJECT1 = $1 OR
SUBJECT2 = $1 OR SUBJECT3 = $1 OR
SUBJECT4 = $1 OR SUBJECT5 = $1
SORT RESULT BY AUTHOR TITLE MONTH YEAR
TITLE "$1 As Printed in REMark Magazine"
PRINT RESULT BY AUTHOR TITLE MONTH YEAR
APPLICATION
*ENDIF
HELP REMARK
```

8" Drives And My Z-241

Dear HUG:

Learning how to use 8" drives on my Z-241 has been an interesting experience I thought other Huggies might find helpful.

I purchased a pair of Z-207-68" drives with my low profile Z-111 when I acquired it

four years ago. I found them to be much more useful than 5-1/4" drives for the large MS-FORTRAN programs I was writing. Believe it or not, I found having four floppy drives to be very convenient. Gradually, I upgraded my Z-100, adding a Z-205 board, a Z-217 10 meg Winchester drive, and a 256K RAM + 8087 board from Jim Hudson. By rough count, this computer has over 600 chips in it.

About a year ago, I got a Z-241. I thought, "Gee, wouldn't it be nice if I could use my 8" drives with my Z-241. Then, if something happened to my Z-100, I would not be without a place to use the very expensive 8" drives and the 8" diskettes I had been using as my primary backup medium." So I began to hunt for an 8" drive controller that could be co-resident with the existing floppy/hard disk controller in the Z-241. I had seen the ads for Floppy Disk Services in Computer Shopper and I called them. They said they had a combination 5-1/4" and 8" controller made by Maynard and a much more expensive one from Flagstaff Engineering. The Maynard board would require at least a custom device driver, they said, and it probably wouldn't work in the Z-241 because the Z-241 probably was not sufficiently AT-compatible. I had just acquired a copy of MS-DOS 3.2 and I thought, "Well, let's give the device drivers in it a try first." So I ordered the Maynard board.

As it turned out, Floppy Disk Services was right on both counts. I called Maynard and asked for some assistance. They referred me to Microtech Exports, 223 Forest Avenue, Palo Alto, CA 94301. Microtech, I learned, modifies the Maynard board and sells a custom device driver for it along with a program which enables one to read and write 8" and 5-1/4" CP/M disks. I sent them my board and ordered the software. In less than a week, I had it back. Now the 8" drives seemed to make the right noises, but all I got on the directory were zeros. Mr. Corwin Nichols of Microtech, after walking me through the diagnostics over the telephone, concluded that the Zenith disk controller did something different from the one used in the IBM PCAT. I checked the Technical Reference manual for the Z-241 and discovered that the disk controller was the only board in the computer for which there was not a schematic.

That's when the fun really began. I called the Zenith regional office and got the number for technical support in Chicago. For various reasons, "Jack" was never available to talk and never returned my calls. Mr. Jim

Miller, head of technical support in Atlanta, told me he had learned that the disk controller was a third party board made by DTC in Santa Clara and that was why they (Zenith) had no schematics for it. He gave me a number to call. I did. DTC was very pleasant, but said they could not give out the schematic. They did say that if Mr. Nichols would call them, they were sure they could answer his technical questions. I asked Mr. Nichols to call them. He did. Twice, in fact. He first talked to the marketing man to whom I had been referred and then to a technical person who said the board was done under contract to Zenith and that the schematic could not be released without Zenith's written okay. Back to square one. Mr. Nichols was convinced that differences between the disk controller IBM used and the DTC unit in my Z-241 were responsible for my difficulties. He assured me the Maynard controller with his device driver worked in a PCAT.

Before I went to the trouble of verifying that, I once again talked to Jim Miller and laid out for him all that had happened. He said he would send me an updated DTC controller to try. When it arrived, I found that the new board did not resemble the board in my machine in any detail. I exchanged boards and found that I could now format, write to, and read from 8" diskettes using Microtech's device driver (which is called by CONFIG.SYS) and their 8" diskette format software. CHKDSK reported a total of 1.25+ Mbytes on the disk. However, I could not read diskettes that had been formatted on the same drives when they were connected to the Z-100. In fact, an attempt to read the directory on such a diskette caused the computer to lock up. With CP/M diskettes, the machine made all the right noises when asked to read the directory, but displayed nothing on the screen. Upon learning of this development, Mr. Nichols concluded that the hardware was now satisfactory and that a patch to the software would be required. He asked me to send him the diskettes I had been experimenting with. I did so.

In about 10 days, I had the diskettes back with a new device driver, version 1.08. Based on a limited amount of testing, I have found that everything now works as it should. I could not be more pleased. It's the pleasant sort of sensation one receives after a difficult quest has ended successfully, like finishing a book or passing an examination. I would like to take this opportunity to gratefully acknowledge Mr. Nichols' assistance, as well as that of Mr. Miller.

Yours very truly,

Frederick O. Smetana, Professor
Mechanical and Aerospace Engineering
North Carolina State University
Raleigh, NC 27695-7910

AutoCAD Does Support The HI-80

Dear HUG:

The cover picture of the July 1987 issue of REMark looks very much like one of the sample drawings of AutoCAD (EC-1305). It PROMPTS me to want to share a discovery with members of H/ZUG who (1) yielded to an impulse, like I did, and took advantage of Jade Computer's special price on the Epson HI-80 Plotter, and (2) also own AutoCAD.

Although its installation guide says otherwise, AutoCAD does support the HI-80, and it can plot the above mentioned picture. This is possible because Epson has provided two command systems. (AC0), which is Epson's own system, and (AC1), which is the Graphtec system.

AutoCAD directly supports the ROLAND DG model DXY-800 which also uses the Graphtec system. In my case, I use the parallel printer interface and so does the ROLAND. So, all I have to do is disconnect the cable from my printer and connect it to the HI-80 Plotter and it's ready to go.

Well, not exactly. There is a little more to do:

1. Configure AutoCAD for Plotter Option #14, Model 800. This is an 8-pen plotter, but the 4-pen HI-80 doesn't care. It will use pen 1 for pen 5, 2 for 6, 3 for 7, and 4 for 8.
2. Set up batch files to command the HI-80 to use command system AC1. (This can be accomplished by opening the plotter case and turning dip switch 5 ON).

I use the following two batch files:

```
GO.BAT  which is:  PRINT AC1.AC1  
                          ACAD
```

```
AC1.AC1  which is:  AC1
```

Copy these two batch files to the ACAD .EXE disk along with PRINT.COM.

To get started: Connect the plotter, insert paper, turn the plotter ON, type GO and then press RETURN. GO.BAT will send the command AC1 to the plotter and then load AutoCAD. It is as simple as that.

Well, not exactly. Nothing is perfect, according to the AutoCAD installation guide.

AutoCAD does not Cap the Pens when it has finished plotting. For this reason, it is very important to manually press the ON/OFF button and then the CAP/OFF button before turning off the power to the plotter. Failing to do this will leave the pens uncapped and they will surely dry out (and it will cost \$8 to replace them).

Sincerely,

Robert F. Hassard
3466 Tice Creek Drive, #4
Walnut Creek, CA 94595

Chi Writer For The Science And Engineering Community

Dear Jim:

Christopher Feuchter's "Equation Master" as he described it in the August 1987 issue of REMark, addresses a need felt by many of us who work in the science and engineering community; the need to be able to write mathematical expressions using a word processor. However, unless I am missing something, it seems to me that he is reinventing the wheel. I wrote everything appearing in his Figure 1 in less than 20 minutes using Chi Writer just as it came from the box with no special programming or character design needed.

It is a shame to deprive Chris of the joy of writing all of that code, but anyone who does word processing which includes a lot of mathematical formulas and equations should look at Chi Writer. It comes with 14 different fonts available at the press of a special function key, including the complete greek alphabet and more than enough math symbols to keep Schrodinger, Maxwell, or any mortal happy.

In addition, Chi Writer is a fairly well behaved word processor for such homely tasks as writing books or letters to editors. And finally, the price is right. It is free — provided you can endure no documentation and commercial designed to prick your conscience every few thousand keystrokes. You can get the program without the commercial and with a very good manual for under \$100. Write to Horstmann Software Design, P.O. Box 4544, Ann Arbor, MI 48106.

Sincerely,

Robert R. Ludeman
4665 Greenfield Drive
Berrien Springs, MI 49103

"What's On The Menu"

Dear HUG:

Enclosed you will find a letter from a REMark reader (*letter withheld -ed*) who has expressed some concern over the usability of the "What's On The Menu" program that appeared in the June 1987 edition of REMark. This is the second letter I have received on this subject. And although this particular reader was less than tactful, he may have brought to my attention a point which may not have been clearly identified in my article — the specific applicability of the MENU.BAS program. The program, as developed, was written in GW-BASIC ver. 2.02 on an H/Z-100. I believe some readers have attempted to use the program on the newer Heath/Zenith IBM compatibles, and in its existing form, the program will not function. There are significant differences when comparing H/Z-100 and IBM compatible GW-BASIC color and graphics command usage. Consequently, several modifications would be required to convert the program for IBM compatible use.

Although I mentioned (in the Program Development section) the H/Z-100, and discussed its character's size attributes — which differ from the IBM compatibles — this information may have been overlooked by the zealous reader wanting to use the program and then read about it later. As I look back, I must admit that the article could have been more clear on this point. However, I assure the readers that MENU.BAS works well on the H/Z-100.

Sincerely,

M. D. Zapolski, Sr.
226 West Avenue
Bridgeton, NJ 08302

MPI-99 Printer Ribbons

Dear HUG:

I am writing in response to Anthony Placzek's letter requesting a source for MPI-99 printer ribbons.

I would like to let all your readers know that Studio Computers has been selling ribbons for the MPI-99, MPI Sprinter and MPI-150 printers for about five years. Although we no longer sell the printers themselves, we do supply a lot of cus-

tomers throughout the country with accessories. We still do some repair work on the MPI-150 models.

The MPI-99 and Sprinter ribbons sell for \$10.50, while the MPI-150s are \$18.00 each. Shipping and handling is \$3.00 per order (not per ribbon). Orders are accepted by phone or mailed-in requests. Payment can be made by VISA/MasterCard, check or money order. Call us at (313) 645-5365 or write us at Studio Computers, 999 S. Adams, Birmingham, MI 48011.

Studio Computers sells exclusively Zenith Data Systems computers and selected compatible accessories from various popular vendors. A free catalog is available upon request. We have been in business since 1979 serving the Heath/Zenith community.

Thanks for providing a very informative publication to the Heath/Zenith community.

Sincerely,

Ray Massa
999 South Adams
Birmingham, MI 48011

How's That Again?

Dear HUG:

We've always heard that truth is sometimes stranger than fiction. The following is a good example of that. The paragraph was received at an Air Force System Program Office from a major contractor in England, sent in response to a comment made concerning errors in a flowchart in a Computer Program Product Specification document submitted for approval:

"The erroneous indicators were indeed inserted inadvertently, the peccadillo being the concomitant consequence of the utilization of a conceptually-sound although operationally-underdeveloped computer-based iconographic system rather than the employment of an inexperienced illustrator."

Perfectly clear, right? It was certainly not what was expected by the reviewer making the comment. The original comment? It was:

"There is a discrepancy with (two decision boxes). They both have three exit

points. How can this be? Hopefully, only erroneous indicators have been inserted inadvertently by an inexperienced illustrator."

Sincerely,

Edward W. Snyder
4045 Forest Ridge Boulevard
Dayton, OH 45424-4834

Minor Problems With The WH-8-64

Dear HUG:

The WH-8-64 Memory board modification listed in REMark, July 1987 has a few minor problems.

Two additional jumpers must be installed on the memory board. These go at:

U62 Pin 14 to U54 Pin 17
U62 Pin 16 to U54 Pin 18

The Trionyx X/2 Card must be modified also. It was originally hooked up to the Heath 64K Ram Card using a positive going signal. This must be changed to a negative going signal. The jumpers (or resistors) at A-E; B-F; C-H; and D-J must be removed and jumpers installed at A-K; B-L; C-M; D-N.

This completes the modification.

Rick Indiano
1080 Farnsworth Road S
Roch, NY 14623

Not Everyone Has Problems With The EasyPC

Dear HUG:

After reading William G. Nabor's letter in "Buggin' HUG", July 1987, I am forced to respond in defense of all those other people who have installed the EasyPC with minimum problems. These people have written articles or letters in REMark and are:

Jim Buszkiewicz	January 1986
Timothy J. Donovan	August 1986
Lt. Col. Myles P. Somers	August 1986
Louise Mezzatesta	October 1986
Charles E. Wiley	November 1986
John Luongo	November 1986

I also installed the UCI EasyPC in my H-100 without any problems whatsoever. It

has been running well over six months with IBM software and no problems. No problems were encountered with pins during assembly. No problems were encountered with extraction of the ICs, since I used the extractor furnished by Heath when I originally assembled the H-100 kit in 1983.

I can sympathize with Mr. Nabor's need for Heath's flawless instructions instead of the UCI instructions furnished with the EasyPC kit. One must remember, however, that the UCI kit was released for use about the same time that the H-100 was no longer the shining jewel in the crown of computers offered by Heath/Zenith. If Heath had received directions to improve the instructions it probably would have been done. I can agree with him, that the H-100 was a superior machine for its time.

His other complaint about having trouble with his H/Z-100 software when he returns to the H-100 mode amazed me. I just have to press the "Z" key and the flawless beauty of the original H/Z-100 instrument bursts forth from the UCI constraints. Absolutely no problems with old software, except to reboot in DOS.

There is no rationality for me to go out and buy an IBM PC. My UCI EasyPC answers all my needs. The money I saved also allows me to buy lots of inexpensive IBM software.

I enjoy reading REMark despite its thinness. Dr. Roberts' assembly language

program "Scroll" is a gem in the July issue. I would like to see a good article on how to convert portions of the 768k of memory now on my H-100 for RAM disk purposes to store, for instance, all of the Peachtext Word Processor for faster letter writing. Since I use both a matrix and a daisywheel printer, it would be nice to have just a simple second port conversion on how to add on to the single port furnished with the EasyPC.

Jim Eilers
1800 Rolling Hills Court
Bartlesville, OK 74006

Would Like To Donate Software

Dear HUG:

I have some Z-100 software that I would like to donate to a charitable/educational institute. The software includes Condor rDBMS, Peachtext 5000 and Microsoft Pascal.

Ralph Seiler
377 E. Leslie Avenue
Salt Lake City, UT 84115

Needs A Terminology Dictionary

Dear HUG:

I would like to suggest that someone, in the near future, write a short article ex-

plaining some of the terminology used in REMark. I use a Z-248 in a government installation and I simply do not have the time to become familiar with all of the terminology associated with such a complex system. About 50-75 percent of the time, I cannot follow an article because of terminology. For example, in the current issue (Sept. '87), on page 11 is a letter which talks about "Program Segment Prefixes", "Memory Control Blocks" and commands, such as "SET COMSPEC=". I have no idea what these terms mean.

Another comment. I see very little in the way of discussion of real-life application problems. I use dBASE III to manage several data files of more than 6,000 records and have managed by experimentation to write several fairly sophisticated menu-driven retrieval and editing programs. We are literally at the point where our work could not be done with any degree of efficiency without this database, which is installed on ten separate computers. In some cases, I have spent hours trying to solve certain programming problems because the dBASE III manual simply does not adequately cover certain commands. As for the Zenith customer support center (since their reorganization, a separate line has been set up for government users), they have yet to return any phone calls. I have thus written them off as a joke.

I am not totally negative. I am simply airing some long-standing gripes. REMark is a good magazine and I have on occasion

FOR H/Z COMPUTERS

* LOGITECH MOUSE AND PLUS SOFTWARE \$95
(includes programable popup menus)

* LOGITECH CADD SOLUTION \$153
(Logitech Mouse, Plus, & GenericCADD)

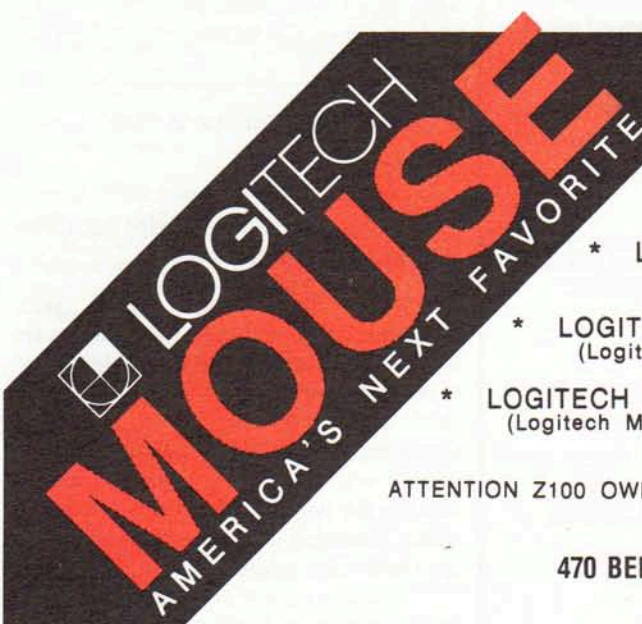
* LOGITECH PAINT SOLUTION \$119
(Logitech Mouse, Plus, & PC Paint)

* LOGITECH PUBLISHING SOLUTION \$148
(Logitech Mouse, Plus, & Publisher)

MC VISA check (add \$4 s/h)

ATTENTION Z100 OWNERS: Use the Logitech mouse with ShowOff and MS Windows

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



used some tip or program from various issues. I also appreciate the use of the HUG BBS at no extra cost.

In closing, if anyone is using dBASE III and experiencing programming problems, drop me a line. Maybe I have already experienced the same problem and know the fix for it.

Sincerely,

D. R. Cool
Defense Electronics Supply Center
DESC-ECS
Dayton, OH 45444

Won't Operate Properly

Dear HUG:

I have an H-89 with the HDOS operating system.

Recently, I purchased a Radio Shack DMP-130 printer that will interface serial or parallel.

I installed an HA-88-3 serial interface card in the computer.

Problem is, I have not been able to get the printer to operate properly.

I would appreciate hearing from anyone who may have successfully accomplished the above and how he did it.

Sincerely,

Harold G. Quillen
5603 McGregor Court
Fayetteville, NC 28304

Heath/Zenith Stands Behind Products

Dear HUG:

Recently, I had a problem with a ZDS program and some communications with the company which seemed to have fallen in the crack.

But it hadn't, and the problem was overwhelmingly resolved by Heath, more than meeting my expectations.

For you newcomers, you'll be glad you joined HUG and the Heath/Zenith family. They really do stand behind their products and their customers, and that's a rari-

ty today, particularly in the computer business!!

Thank you Heath. I'm glad I came aboard, you HUGGIES will be, too.

Kindest regards,

Rodney E. Cavin
P.O. Box 507
Altamonte Springs, FL 32701

A Solution To DEC Printers

Dear HUG:

From the latest REMark issue, the problem with the DEC LQP02 is common to all users of DEC and IBM PC equipment. The problem is that the LQP02 (and all DEC printers) only recognizes XON/XOFF protocol and the PC compatibles only recognize hardware handshake.

The cheap solution is to slow down the serial port to 300 baud and use continuous form paper so the buffer doesn't overflow. The best solution is from a company called GOLD KEY ELECTRONICS who make numerous converters that work!

Their converter hooks up between the IBM parallel cable and the RS-232 cable to the DEC printer. It includes a print buffer (16-64k) and plugs into a power outlet. So far everything I have tried works perfectly, and I'm now running the LQP02 at 9600 baud, 8 bits, no parity. The price for this is \$149-229, and it's saved my sanity.

It works with any DEC printer, and I'm reasonably sure it will work with any XON/XOFF device and save your serial ports, eg. HP Laser, etc.

Their address: P.O. Box 186, Goffstown, NH 03045. Phone: (603) 625-8518.

Richard L. Gristak
7474 E. Arkansas #15-5
Denver, CO 80231

The C-7 Logitech Mouse, AutoCAD, And The H/Z-100

Dear HUG:

A mouse is a mouse is a mouse is simply just not true. I am an H/Z-100 orphan,

and I have a problem with a Logitech C-7 LogiMouse, AutoCAD version 2.18, and Microsoft Windows (MS-3063-30).

The problem is that the LogiMouse will not work with either of these programs. Although Logitech advertises that the C-7 mouse supports AutoCAD, they also say that they do not support the H/Z-100. Autodesk says that they do not support any mouse that misuses the serial port by sapping power from it. And they may be correct in as much as the 1488 line driver has a cutoff at 10 ma.

Heath says "bring your computer in and we'll fool around with it and see what we can come up with". As for Windows, naturally, the manual (page xviii) says you must use a Microsoft Mouse.

Incidentally, my C-7 LogiMouse works beautifully with Doodler V.

Before I invest more money on mice and accumulate a useless harem, I would very much appreciate hearing from any HUG H/Z-100 member who is using a mouse with AutoCAD version 2.18, or Microsoft Windows. Naturally, I would prefer to know how to use my C-7 mouse, but I would settle for advise on what mouse is best suited for the H/Z-100. I was intrigued with the idea of no separate power supply, but if I must, I'll give up on that idea.

Sincerely,

Robert F. Hassard
3466 Tice Creek Drive, #4
Walnut Creek, CA 94595

Emulator Board For The Z-158?

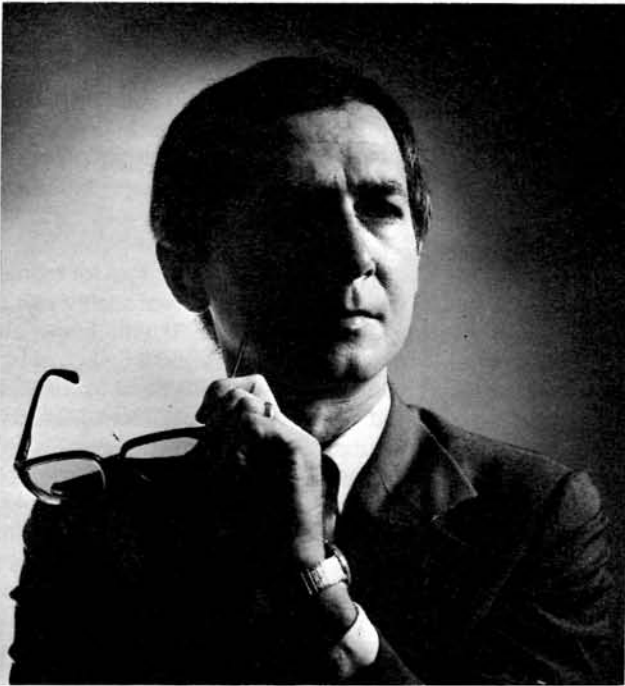
Dear HUG:

I hope that the many HUGgies out there can help me with this one.

Quadram manufactures and sells Quadlink, a plug-in board that enables an IBM PC or XT to emulate an Apple II+/e/c and run most Apple software under DOS and ProDOS. When I called Quadram to see if the Quadlink was compatible with my Z-158, they said it was not. (By the way, Quadram was very good about returning all phone calls until my concerns were addressed).

The question is: Does anyone manufacture an Apple emulator board that will run

Continued on Page 27



Mainstream Computing

Joseph Katz

103 South Edisto Avenue
Columbia, SC 29205

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

I have to relate this tale our neighbor Strothers Pope, a retired gynecologist who has seen many interesting things, just finished telling me. It's about how he saved the life of possibly the only elephant ever to reside in Camden, South Carolina. There's a moral to this story, of course, but I couldn't resist sharing it even if there wasn't.

Cotton is grown here. Some years ago an imaginative farmer in Camden had the bright idea to use one elephant instead of several mules to cultivate his fields. After all, he figured, an elephant is stronger and has more endurance. But a time came when the farmer decided that the elephant was more trouble than it was worth. It ate too much, was unruly, and left a wake of things and trampled plants as it walked.

That's what the farmer confided to Dr. Pope in a telephone call. Since the good doctor is well known in these parts for his interest in Africa, might he perhaps have an elephant gun? He did. Could the good doctor then perform the farmer a small service by journeying up to Camden with that elephant gun and perhaps killing the elephant? He could.

"Instead I saved that beast's life," drawled Dr. Pope while we were standing on our lawns this morning chatting.

"How?" I asked, with an odd feeling I would be sorry whether I did or didn't.

"I simply inquired of the gentleman where he intended to bury such a very large elephant. It turned out that he had put no more planning into the elephant's disposal than he had into its acquisition. I counseled with him until he concluded that it made more sense to give the live animal to a traveling circus. Which the farmer did. So you see, I saved that elephant's life."

I excused myself and trudged across the lawn, around behind the house, to my small office so I could immediately preserve this elephantine bit of South Carolina lore.

The Camden Elephant has a great deal to do with computers. Dr. Pope told me its story at one of those times when several seemingly unrelated incidents turned out to point in the direction of Camden. A few months ago at COMDEX in Atlanta, for example, I met Adam Obsome. He's now the President of Paperback Software, but several years ago was the President of Osborne Computer Corp., which introduced the first real portable microcomputer. If you use a Zenith Z-171 or any other portable micro, you owe homage to Osborne. A more important contribution he made then, I think, was a

statement on the order of "Sometimes the merely adequate is what you really need." I've paraphrased from memory, but I think the sense of his statement is right enough. Osborne was ridiculed by many people for that remark then. But a lot of things have been happening lately that make me think he was absolutely right.

I'm thinking specifically about database managers now, but the point applies to other things too. Sometimes "the best"--like the Camden Elephant--is so powerful for a specific situation that it's just not as good as something less ambitious but more focused. Of course I'm not praising mediocrity. What I'm saying is that there are many times when you'll get more accomplished in less time with a needle than with an icepick. Let me put some new software on the table to show you what I mean.

Buttonware's PC-File+

Of course Ashton-Tate's dBASE III Plus--the most recent offering in the dBASE line of database management programs--is good, and powerful, and justly among the classics of microcomputer software. I've nearly always had a dBASE, almost from the time Wayne Ratliff first introduced it under the name "Vulcan," long enough so I can claim to have grown up with it.

And often dBASE is exactly the right tool for me, especially when I take into account time I save because it's a familiar tool. I still like it for custom applications.

But I've come to realize (thanks to a number of you who have been nagging me so persistently) that in many situations dBASE is like the Camden Elephant. There I am programming a complete application from the ground up using dBASE III Plus when something smaller, more agile, and more sharply focused would do what I need in much less time and better too. But for a long time I didn't believe it, and wouldn't have believed it even if you painted the lesson on both sides of the Camden Elephant and paraded the beast through my bedroom. Friends I respect had told me to look seriously at Jim Button's PC-File database managers, and I did once to placate them, then came away with my own prejudices ratified. Now I know better and I'm happy to apologize to my good and patient friends whose advice I ignored for so long.

Free evaluation copies of PC-File and its descendants have been distributed

widely so you can explore the program and pay a registration fee only if you decide to continue using it. A few years ago I got one of those "shareware" copies of PC-File, gave it a try, and decided that the thing simply didn't have the features I needed. I continued doing everything in dBASE II then. But I was not right. Part of what led me to be not right about whichever version of PC-File I evaluated then was that Button supplied only a partial manual on those disks. The full manual would come only after you registered, which I was not about to do because I didn't see that the program had the features I needed, because the shareware disks didn't have the full manual. Circular, right? Well, now either Jim Button or I have become wiser.

The full manual is on the evaluation disks of PC-File+ (the latest version), so I didn't have my old excuse to resist my persistent friends. (You get support and the manual in book form when you register, which you ought to do for your own sake as well as the benefit of Jim Button's family.) With the full manual and the program in front of me, I spent a little time working

through the building of a database I desperately needed for real right then. A few months ago would have been better, but I hadn't had the time to do it in dBASE III Plus.

My PC-File+ database management "program" was complete in about five minutes and I nearly keeled over in disbelief. Five minutes. And it worked just the way I wanted it to. It still does and I'm still using it.

PC-File+ is an excellent tool for managing either a flat file or a reasonably complex relational database. If you know about database managers you'll be bored by my explanation that a flat file is like a stack of cards that are preprinted forms for recording data: name, company name, address, city, state, and ZIP code, for example. The cards are separate from one another.

Add relational capabilities and you link the cards according to data they have in common—the "relations." Let's say you need to build a mailing list of employees in two different companies located in two different states. After a while you'll get

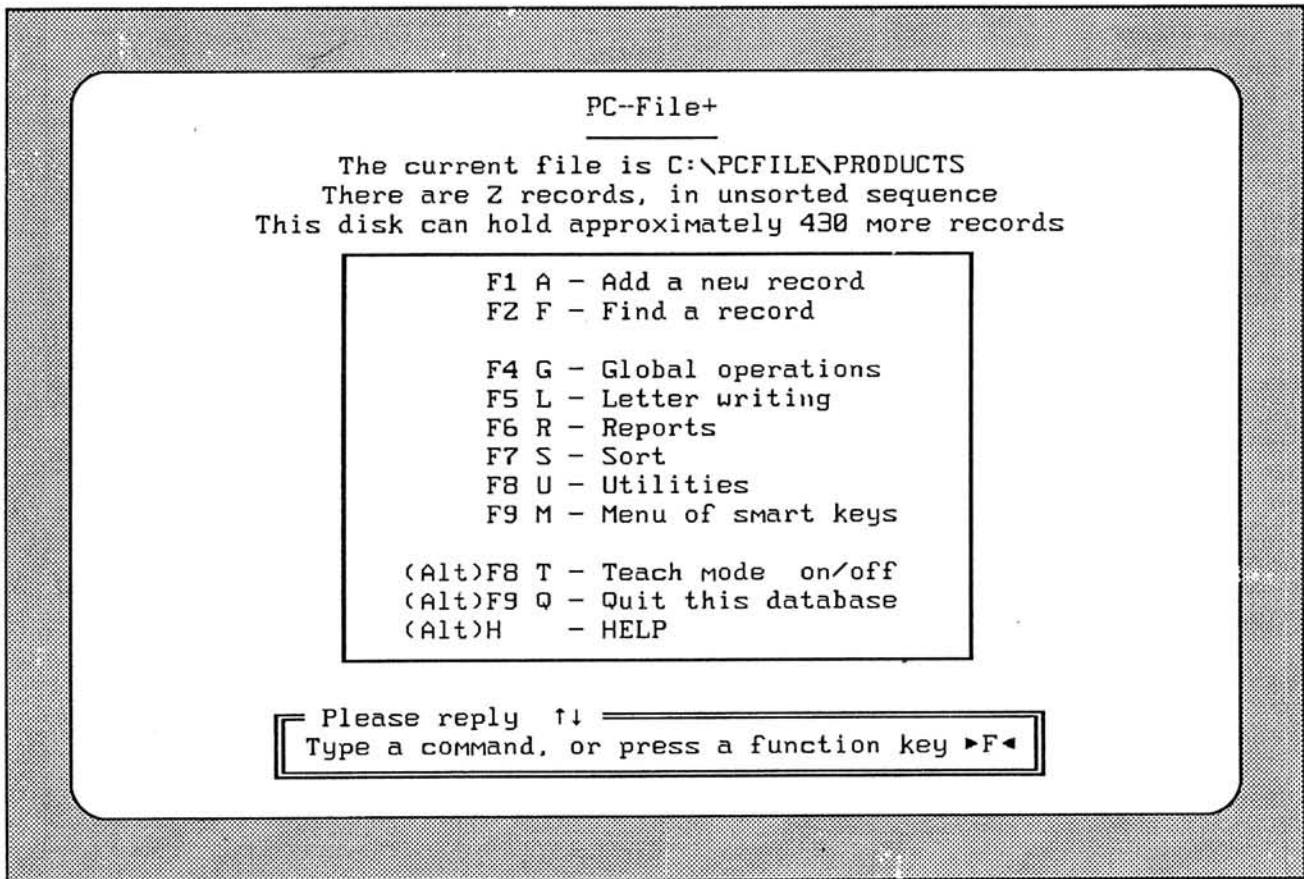


Figure 1. Main menu for a PC File+ database. (I did this screen shot and all others this month with SymSoft's HotShot, and I produced camera ready copy with PageMaker.)

kind of irritated about having to type into your database anything more than the person's and company's name: everyone in that particular company is going to have the same address, city, state, and ZIP code, so why in the world do you have to enter all that stuff over and over again for each company? What you need is a database manager that will let you establish the name of each company as a relation to its address, city, state, and ZIP code. You enter that information only once for each of the two companies, and the information is stored in a file of its own, perhaps called something unimaginative like "COMPANY." (I've learned to place a premium on unimaginative, and therefore easily remembered, names for everything in a database. The cute stuff trips me up later.) Then, if you're using a relational database manager, you type only the name of a person's company when you're entering data for the person. The database manager recognizes the relationship (because you've programmed it to do so) between the company name and the rest of the stuff, which you therefore never

have to type again. A nice bonus, obviously, is that if people on your list change companies, all you need do is change the company name. The relational database manager links those mobile people to their new addresses and such. If your database manager had only flat-file capabilities, with no relational capability, you'd probably be tempted to do frequent mailings about the virtues of stable employment. With a relational database manager, though, it's no skin off your nose if those guys bounce back and forth between the two companies every other day. The relational database manager does the real work.

Of course you can do a great deal with a database manager that has extraordinarily powerful relational capabilities. You could add each employee's social security number, for example, and link it to other files containing salaries, withholding taxes, vacation schedules, race, religion, political leanings, and all sorts of other stuff that really are none of your business. You need a pretty sophisticated relational database manager to do all that, and PC-

File+ will.

Because PC-File+ is menu driven and extremely well designed, it's simple to set up a new database on the fly. You can either "paint" an entry screen with your field names or enter the names in a list and have PC-File+ arrange them for you. The latter is the "fast" method of creation, and it is indeed fast: immediately afterwards you get the main menu for using your database (see Figure 1). At this point your database is ready to accept data. I'm sorry if I went too fast for you: it's so simple a process I can't go much slower.

You do fancier stuff, if you'd like, by then choosing the "Utilities" option from the main menu and editing the resulting database definition for some simple but useful "masks" through which you can control data entry or formatting. For example you can force selected fields to accept only alphabetic data, and even only uppercase alphabetical data. As another example, you can format the entry of dates into one of several conventional

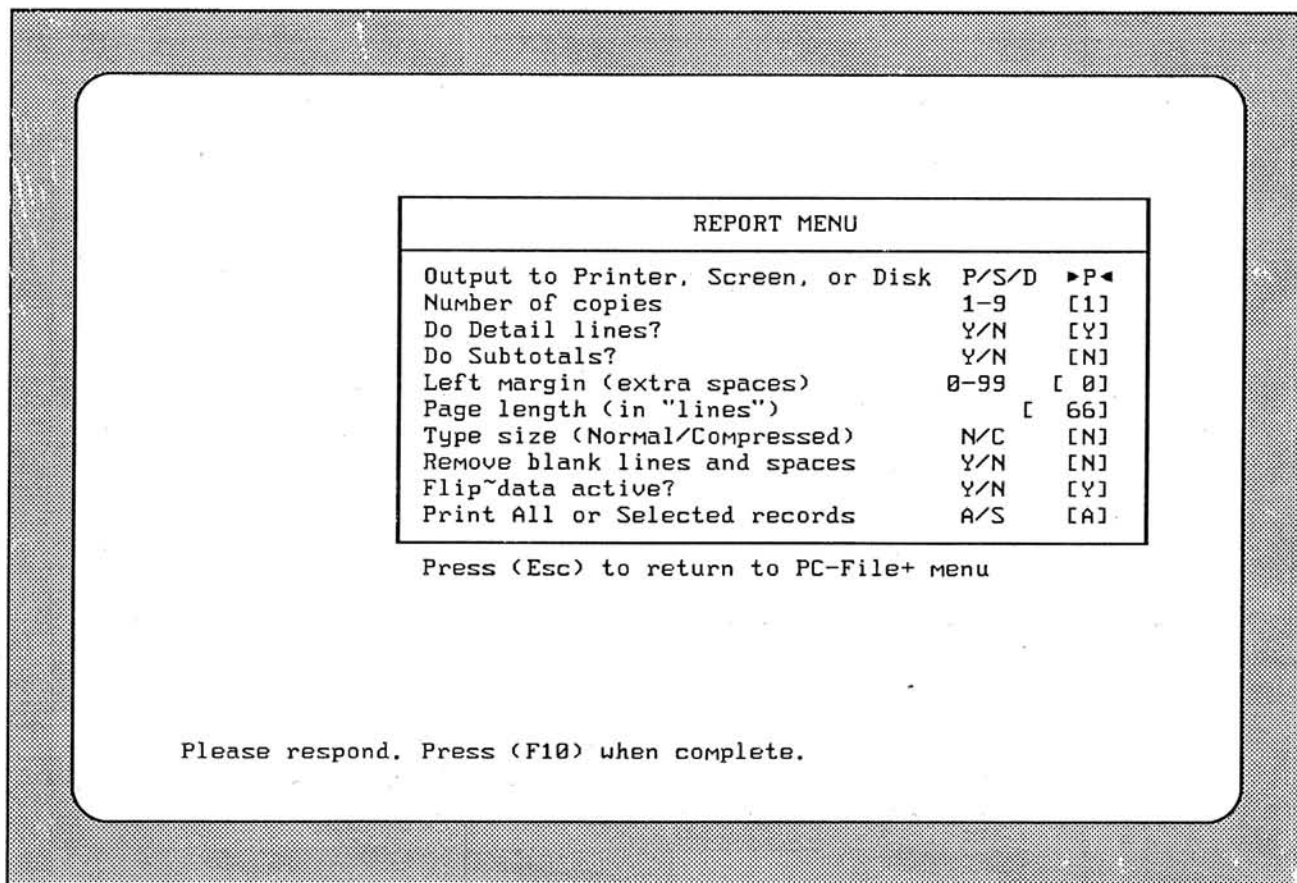


Figure 2. Note that this PC File+ report menu gives the option to "flip" data separated by a tilde ("~") in a field: "Katz~Matthew L." can be printed as either "Matthew L. Katz" or "Katz, Matthew L."

formats: month, day, and year; year, month, and day; and day, month, and year. (I miss having a predetermined mask for telephone numbers. Although Buttonware's Tech Support came up with a clever workaround, I still miss it.) You also can edit the database definition to make some fields contain "constants" (default data, such as the State in which you sell most of the Widgets your company makes) that nevertheless can be edited if the operator encounters an exception. That's handy. So is the ability to make a field "relational," perhaps to look up in a separate database the single-unit price of your Widgets. Then, since you also can make certain fields "calculated," you can have your database calculate how much to charge for the three Widgets ordered by a customer. Lo and behold, you have the basis for an automated invoicing system. As I've said, you can do fancy stuff with PC-File+.

You also can do fancy stuff after your database is operational. One thing that really annoys me about many database managers, including dBASE III Plus, is the need to decide in advance how you want

names entered: first and last, or last and first. You can play that game with PC-File+, or you can choose to ignore it entirely. Many of my on-the-fly databases take the latter way out and simply have a field called "Name." An entry in it might look like this: "Katz~Matthew L." The tilde character ("~") is a "flip" signal. When the time comes to print a report, you have the option to flip data on either side of the marker, making the field produce "Matthew L. Katz" (see Figure 2). I vote knight-hood to Jim Button for that slick feature.

Your reports from PC-File+ can be either slick or functional. Those are relative terms because the functional report formats from PC-File+ are similar to those I labor to produce in dBASE III Plus. But in PC-File+ they're standard reports generated by choosing one of a few menu options. (They're the equivalent of the "fast" option for building the database itself.) Or you can slick things up by painting your desired report format on the screen. Or you can choose to get your hands dirty and use the PC-File+ command language to do something really

complex. Or you can cheat by having PC-File+ generate one of those other report formats and then run your own editor on the file to modify the thing to suit your own baroque tastes. But I would advise against doing the hard stuff until you are familiar with PC-File+ and its command language. It won't take long--maybe a few hours--if you know your way around microcomputer database managers from the inside.

PC-File+ has such modest requirements that only the minimum of 384KB of RAM and version 2 of MS-DOS need mentioning. Its specifications are worth skimming: up to 71 databases open simultaneously (you can do lots of relations); up to 1,665 characters in a field; up to 70 fields per database; up to 65,533 records per database; up to 10 sort fields. One splendid feature of PC-File+ that doesn't appear in the summary is the program's provision for a "superfield"--which means "a big field"--the maximum size of which depends on the number of fields you use for the record. With ten fields in the record, for example, your superfield can be up to 945 characters--

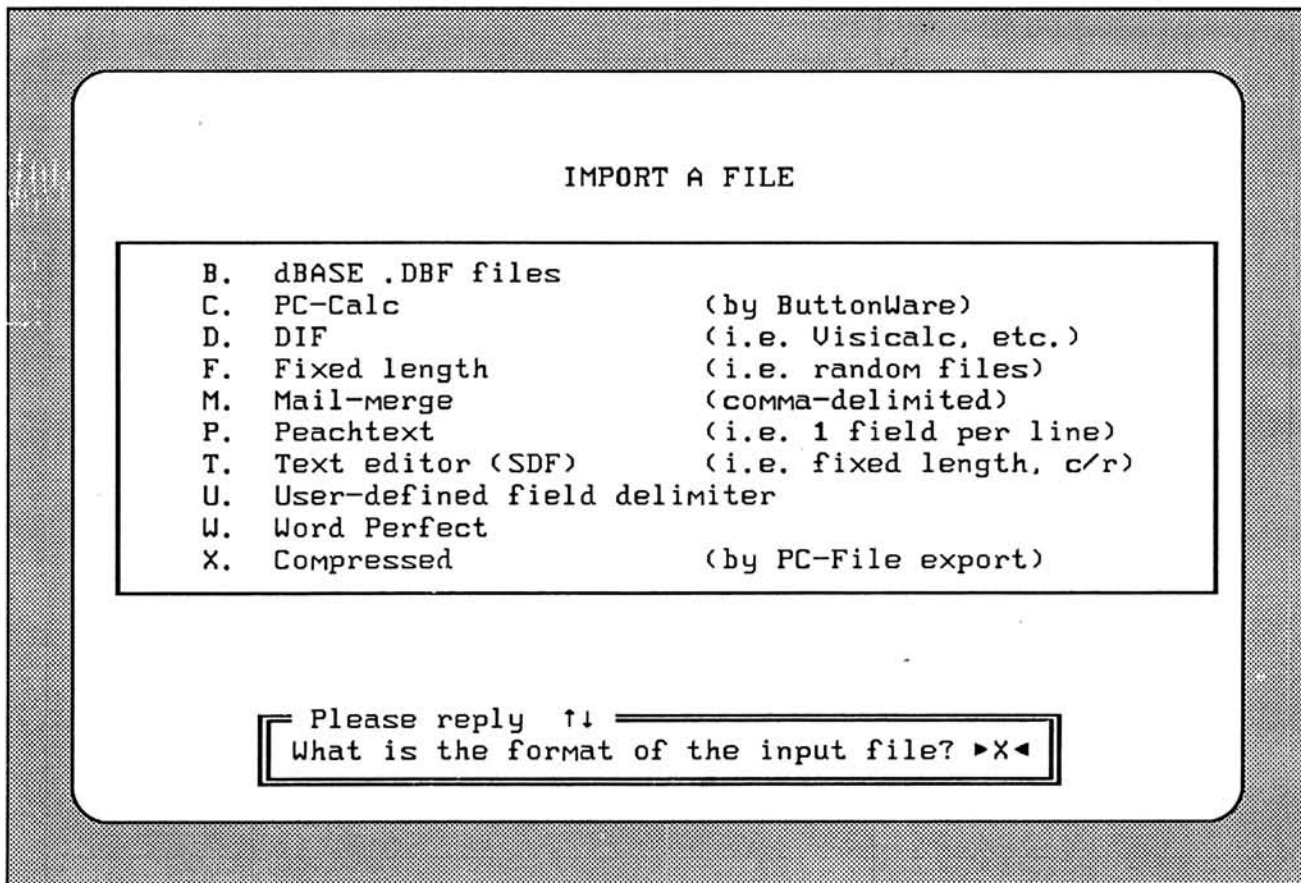


Figure 3. The PC-File+ data import options allow it to accept data from practically any program.

which makes a nice block for recording comments. PC-File+ makes fixed length databases, same as dBASE III Plus, so empty fields take up the same disk space as fields filled to capacity in every record. Buttonware bills PC-File+ as "The Most Popular Database in the World." I can believe it. The program is a full-featured, powerful database program that at the same time is easier to use than any other of its kind I've seen. It's not merely "user friendly:" it's downright easy-to-use. Everyone I know who uses the program praises its online help, "teach" mode, and manual. I think the online help is okay, but not great. It's not really context sensitive, so you get the same overview of what to do no matter where you are in a menu when you ask for help. The "teach" mode simply turns on the online help so it shifts appropriately as you move through the menus. You might like the feature more than I do: I tend to reach for a book when I need help. (Consider my profession.) I think the manual is okay too, but also not great. There's a great deal of good information, but often there's a trickle when I want a torrent. Nowhere, for example, is there even a sug-

gestion about potential uses for a superfield. Nope, I haven't forgotten how bad the dBASE manuals used to be and that many of us survived them nevertheless. I guess I'm holding Jim Button to an extremely high standard because he deserves to be judged by that standard. PC-File+ is a bargain at the \$69.95 registration fee. Only a churl would use the program without registering it. All others (except churls) get support, full access to Buttonware's bulletin board (206/454-2629) from which evaluation copies of the latest Buttonware programs can be downloaded, and the printed, paperbound manual.

Powerline Systems' Jupiter

I can't believe Dr. Pope also told John Preusse of Powerline Systems about the Camden Elephant, but when a copy of Jupiter arrived the very next day I wondered if he had heard the story too. Although Jupiter is a much more specialized database management program than PC-File+, the two reflect similar drives towards a sharply-focused tool instead of a lumbering giant.

Powerline Systems bills Jupiter as "a Records Management System"--a program optimized to manage certain kinds of information about people. You can't use Jupiter to keep an inventory of Widgets you make. Switch the focus from your Widgets to the customers who buy them, however, and Jupiter comes into its own. Jupiter is prefabricated for its purpose of managing data about people. Every part of it is efficient, well organized, and easy to use. The main menu reproduced as Figure 5 shows what I mean.

The way Jupiter works is to key on the people in your database. There are three "frames" (Jupiter's term) per person. Think of the frames as a group of three index cards paperclipped together.

Frame one (Figure 6) is where you put basic data, with slots for six categories you define as you please and one "miscellaneous" slot you can vary from person to person. Jupiter is programmed to help you slide data into this frame. For example it capitalizes the initial letters of last and first names (so you can simply bang data into your database without

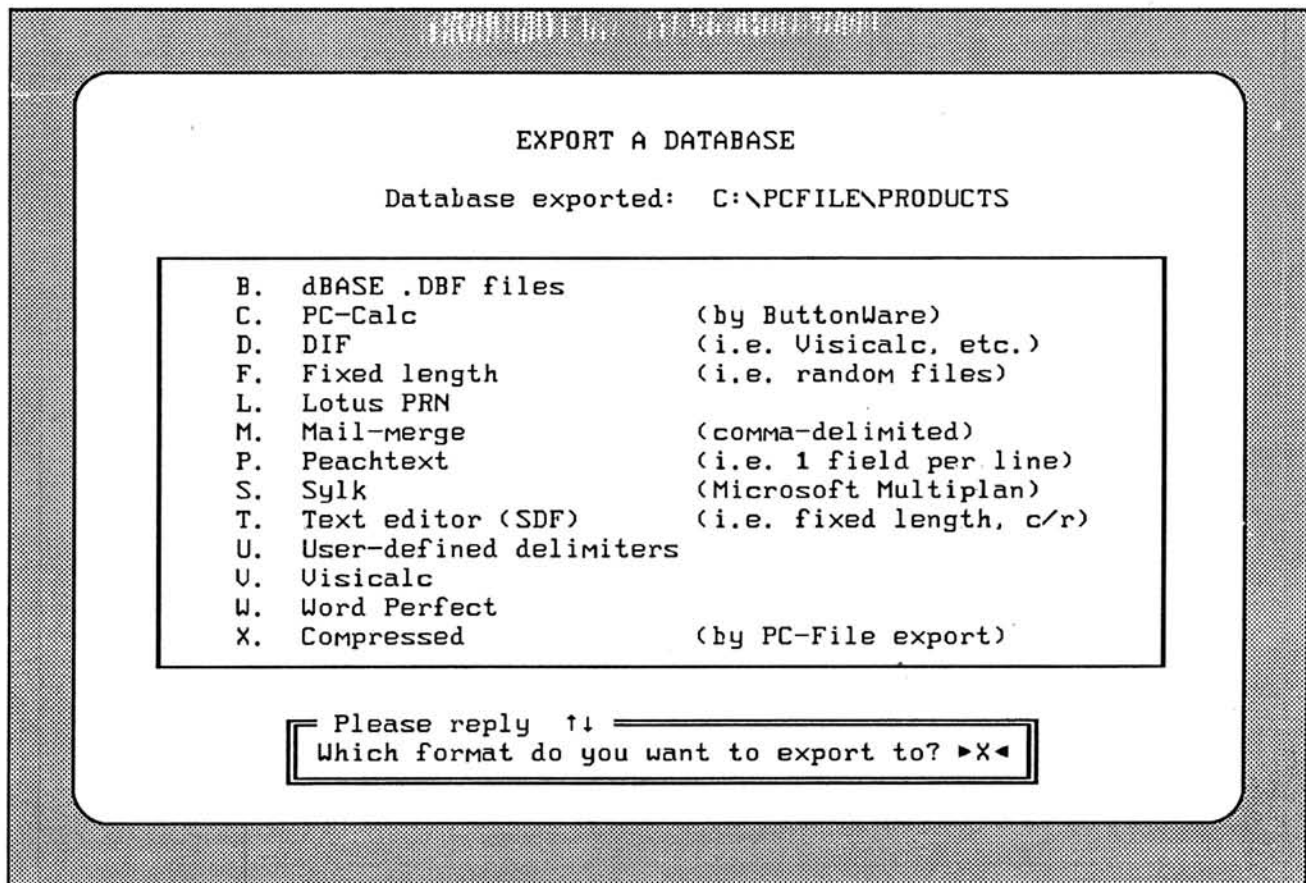


Figure 4. Options to export a PC-File+ database include "user-defined" delimiters, which allow use of data with unlisted programs such as XyWrite III Plus.

worrying about the shift keys), and validates phone numbers and zip codes as they are entered.

Frame two is space for a memorandum of the kind in Figure 7. It's a free-form field, and one of several ways to retrieve records is by searching for words in these memos. One of the few awkwardnesses in Jupiter is the relatively unsophisticated editing features in this memo frame. For example there's no automatic word wrap (which is not apparent in my illustration), so words often will spill disconcertingly from the end of one line to the beginning of the next.

Frame three (Figure 8) is Jupiter's homage to Mammon: here's where you keep track of up to twelve financial transactions for each person. In addition to the usual information, Jupiter also offers space for a "Mark"-- which means anything you want. It's a Boolean field in which an asterisk signifies "True/False," "Yes/No," "Paid/Unpaid," or whatever else you think important and that can be represented by a flag.

Jupiter does what it does, and does it extremely well. I don't like its

requirement for the ANSISYS driver. Powerline uses it to make the same program run on various computers by avoiding any programming techniques that would limit it to IBM compatible computers. That makes the screen handling a bit too sluggish for my tastes, and I don't like the ANSI driver anyway. But this area involves opinion and taste, and is not worth much bother. I do like almost everything else about Jupiter. One thing I like most especially, and you will too, is the way it stores data. It's dynamic storage, so there's no space wasted on fields you don't use. Nice. You can build a whopping database with Jupiter.

GDI's FormEasy

I've never seen anything else like Graphics Development International's FormEasy. I'd be astonished if there was anything else like it for a microcomputer. And yet it's such a great idea that I can't really believe no one has done something like it before.

FormEasy is specifically directed at the needs of those who have to process large quantities of forms and formal

information. You know the drill if you have to face the job of filing periodic reports about large numbers of people. The manual way starts with a stack of forms. Get one from the top and fill in the name, address, social security number, and so on for the first person. Get the next blank and fill it in with data for the next person. And so on and on. I did it in the army (ours) many years ago and that experience taught me the true meaning of the expression "War is hell." When computers came along (go ahead and joke about my age), the job was made easier on occasion by having it make reports from a database on preprinted forms. One thing today and yesterday have in common, though, is the need to order and store boxes of forms. That's expensive. It's even more expensive when you have a stock of forms that becomes obsolete.

Here's why FormEasy is such a splendid concept. It's a combination graphics package and database management report system. You draw the form, code it for positioning data to be drawn from your own database manager, and have it zip the completed forms to your laser printer. Right now FormEasy is available

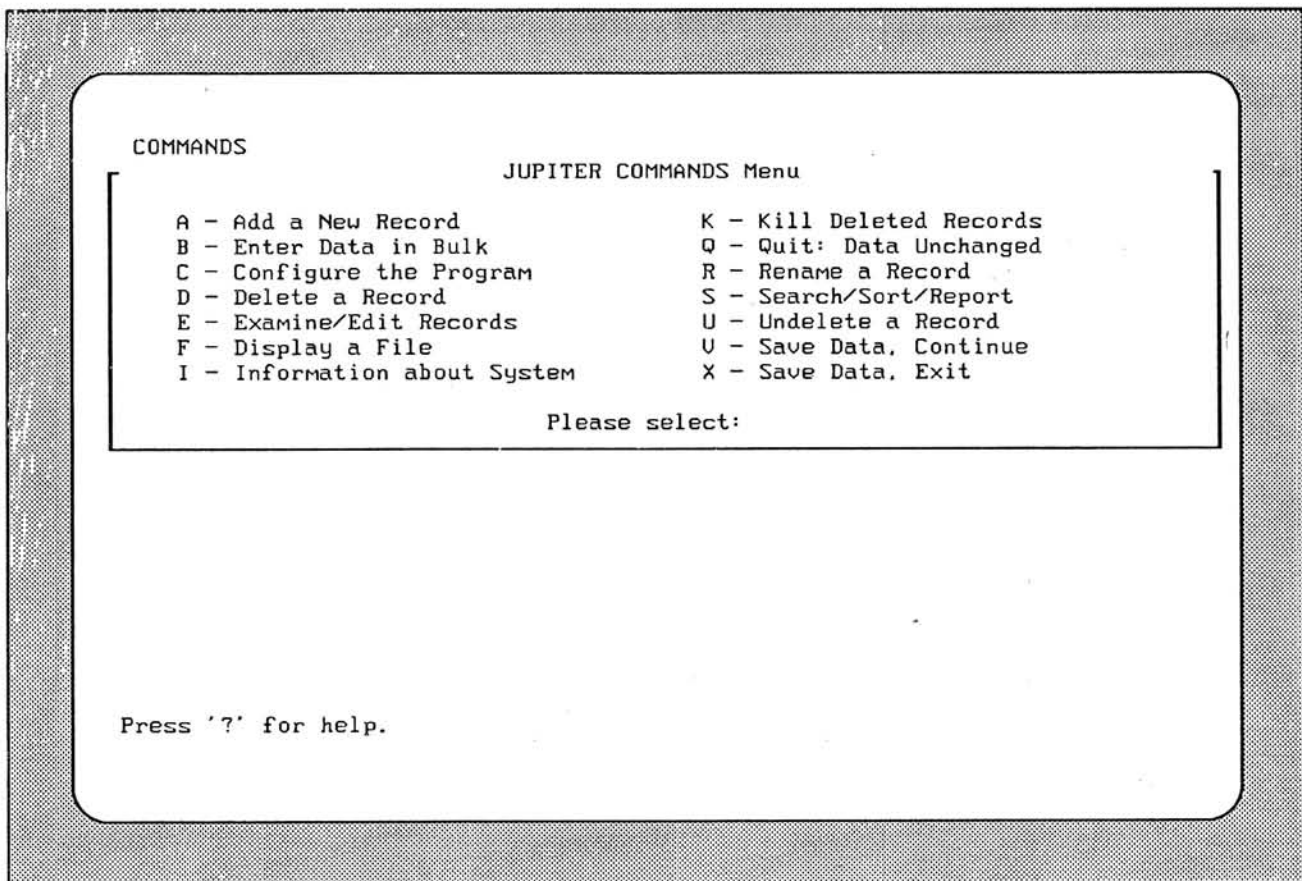


Figure 5. Jupiter's main menu reflects the program's ease of use.

for the entire series of Hewlett-Packard LaserJets and the Okidata Laserline 6, Kyocera, Canon, and Cordata laser printers. I pried out of GDI the information that they're working on a driver for PostScript printers like the Apple LaserWriter, and I put in my bid for a copy right away.

Even though I'm not in the forms completion business, I was fascinated at the Spring 1987 COMDEX by the display of FormEasy and the companion FormScan (which allows you to scan an existing form so you can work database magic by completing it). Now I've tried FormEasy myself, I thought I'd be doing a good deed this month and tell you about it on the chance you can save time, money, and space. You can get a demo from GDI, and if forms are your business, you ought to.

Don't let anyone tell you that there no longer are great new ideas for microcomputer software. FormEasy is one.

Bill Adney's FlipFast Guide to MS-DOS

I feel foolish, and I'm not going to feel any less foolish for writing this. It's going to

look as if I'm repaying the pat on the back Bill Adney gave me last month, and for that reason I considered keeping quiet about Adney's FlipFast Guide to MS-DOS. Then I thought it would be as wrong to keep quiet for that reason as it would be to speak well of the book for that reason. And then I decided that I'm just too old to play the adolescent game of exploring my psyche. Even this little bit was pretty sickening, wasn't it? So here goes.

I don't think you can tell from the title of Bill's book that it's really the one important book about MS-DOS for owners of any Zenith MS-DOS computer. It covers not only the mainstream computers we focus on here, but also the Z-100. And it's not only a thorough guide to the basics, but it's also an incredibly valuable reference tool for advanced users and programmers.

Don't think it's just another introduction to MS-DOS. Once you've passed the stage of needing that sort of thing and need instead a handy deskbook, you need Bill's book. It's where you look up specifics about things ranging from Wait a second. I needed an example to fill

in that last sentence, so I looked up "DIR" in Bill's book, and found there a parameter I knew I had seen elsewhere and just couldn't remember. It's "DIR/P," which has the directory listing pause at the end of each screen. Because I couldn't remember it, I'd been using CTRL-S to stop and CTRL-Q to start again, and I always hate doing that. Thank you, Mr. Adney.

Okay, back to what I was saying. Bill's book is where you look up specifics about things ranging from basic commands such as "DIR" to advanced topics like the formatting details of various disks.

The reason why you need it is that Bill's book is the only place you'll find such information especially for Zenith computers. Remember: although Heath's and Zenith's IBM compatibles do about the same things as others of the breed, they sometimes differ in important details. For example, the CONFIGUR.COM program you have on Zenith's MS-DOS is something special you won't find on versions for other computers. If you say that the treatment of such things in Zenith's manuals is all you really need, you either haven't looked closely at the manuals or

MCTEAGUE, JOHN		Basic Information		Frame 1 of 3
Individual Data Record				
Name	Dr. John McTeague			
Address[1]	Polk Street			
Address[2]				
City State ZIP	San Francisco	CA	94710	
Country	USA			
Salutation	Dr. McTeague			
Phone	(415) 777-2305			
Misc	Sucker			
Category[1]	C.O.D. no checks, no cards			
Category[2]				
Category[3]				
Category[4]				
Category[5]				
Category[6]				
		RandomCode	006	
		Updated	1 Sep 1987	

ESC: SAVE	Ctrl-C: Cancel	Ctrl-P: Print	Ctrl-U: Erase Field
Enter: Next Field	Keypad 3: Next Frame	Keypad 9: Prev. Frame	
Ctrl-S: Set Preset	Ctrl-E: Erase Preset	?: Specific Help	

Figure 6. The first frame of a Jupiter database records basic information about a person.

you're an advanced computerist and an exceptional reader.

As for me, at first I put the inscribed copy Bill sent me onto a shelf in my house where I keep precious gifts like that from friends. Then, when I realized how valuable its contents were, I started carrying it to my office for quick use and back to the house for safekeeping. Now it's on the handiest shelf in my office, right next to my other frequently-used reference books. I guess I'll have to buy a copy now so I can have it for use and get this inscribed copy back into the house where it belongs. Trust me. Get a copy of Bill's book. You need it.

Hmm . . . I don't feel foolish at all. I think I've done you a favor.

My HUG award

There I was chattering away at the HUGCON banquet when Dale Wilson poked me in the ribs and Tom Jorgenson tried getting my attention and Matthew said "Dad!" and Janet said "Shut up, Joe!"

"Huh?" I said.

I had almost talked my way past the award Jim Buszkiewicz was trying, patiently, to give me.

If you were among the 750 other diners that evening you witnessed a rare event. I was too astonished to say anything at all. In case you missed it, Jim poked me and said, "Make an acceptance speech. But a short one." It took all the wits I could muster to say, "For the only time in my life I am at a complete loss for words. I don't know what to say except 'Thank you.'"

I still am too flabbergasted to say anything except "Thank you." I am proud and happy and I don't feel the least bit silly taking that lovely plaque everywhere I go.

You know, you're a really grand group of people and you're very kind to me. Thank you.

See you later.

Products

PC-File+. \$69.95 (plus \$5 shipping)
ButtonWare, Inc.
P.O. Box 5786

Bellevue, WA 98006
800/J-BUTTON; 206/454-0479

Jupiter. Version 2.0. \$99.95.
Powerline Systems
P.O. Box 97
Lincroft, NJ 07738-0097
201/747-2063

FormEasy. \$495.
Graphics Development International
Suite 4
20-C Pimemtel Court
Novato, CA 94947
415/382-6600

FlipFast Guide to MS-DOS. Paper. ISBN 0-931472-32-6. \$24.95.
By William M. Adney.
S-A Design Books
Building E
515 West Lambert
Brea, CA 92621-3991

My volume of mail prevents me from answering all letters about this column, but I try to answer most of those that include a stamped, self-addressed envelope. I may publish your letter (perhaps in edited form) unless you specifically tell me not to.

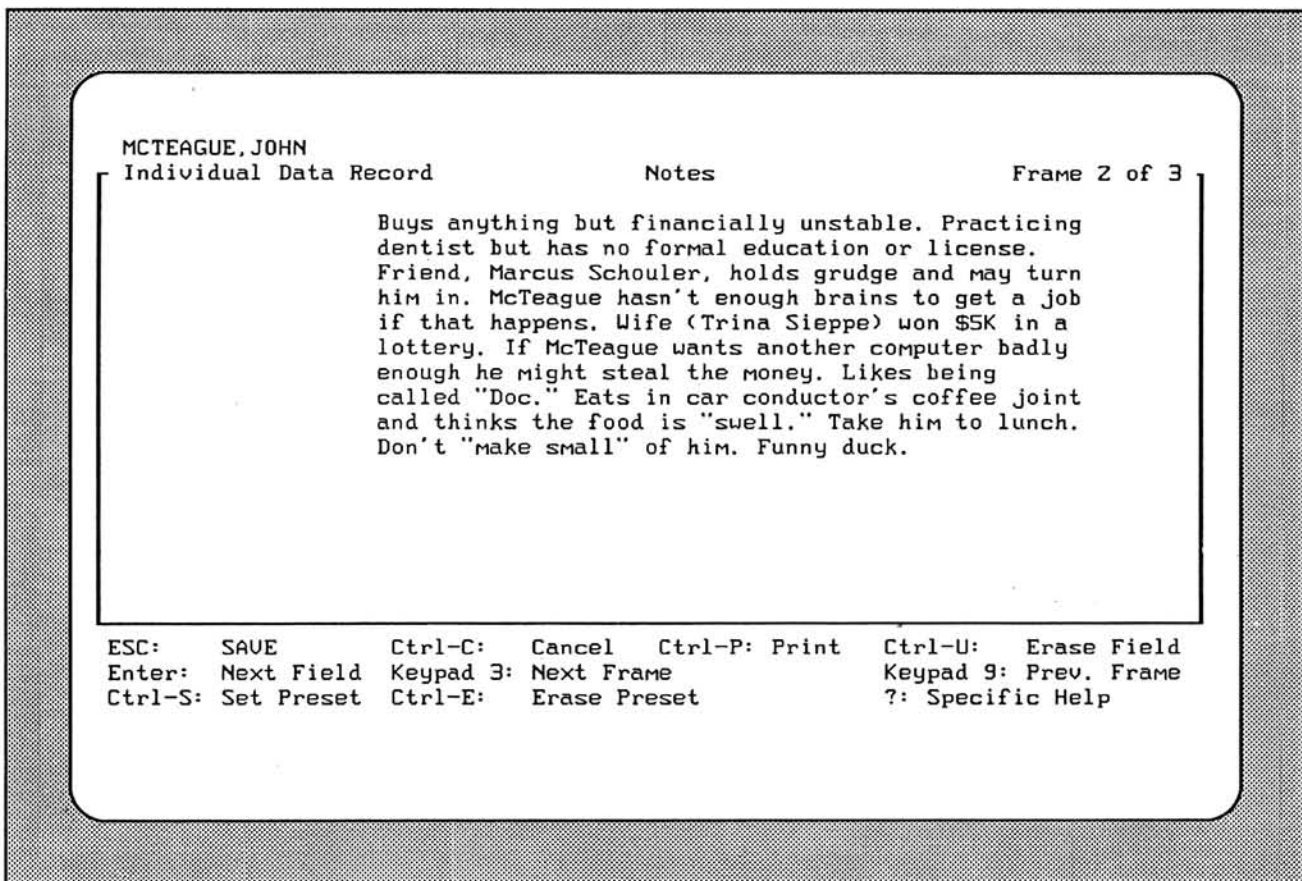


Figure 7. The second frame of a Jupiter database is space for a memo about the person.

MCTEAGUE, JOHN

Individual Data Record

Transactions

Frame 3 of 3

Date	Ref	Item	Amount	Mark
23 Dec 1986	1432	H-8 w/ZUM 131	1,768.00	*
3 Jul 1987	1524	ZP-150 w/5MB winchester	3,926.47	*
24 Aug 1987	1563	Z-100 w/QMS PS-800+	6,243.13	*

Total 11,937.60

ESC: SAVE Ctrl-C: Cancel Ctrl-P: Print Ctrl-U: Erase Field
Enter: Next Field Keypad 3: Next Frame Keypad 9: Prev. Frame
Ctrl-S: Set Preset Ctrl-E: Erase Preset ?: Specific Help

Figure 8. Frame three of a Jupiter database records financial transactions linked to the person. *



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

Seagate

20mb ST225\$259. • (w/cont.) \$319. • (w SCSI cont.) ... \$410.00
30mb ST238\$279. • (w/cont.) \$365. • ST4038\$555.00
Z / 100 ST225N + CDR-11B Interface Board\$725.00

MiniScribe

20mb M8425\$275. • (w/SCSI cont.) \$425. • M3425 ... \$264.00
30mb M8438\$299. • M3438\$289.00
Z / 100 M8425S + CDR-11B Interface Board\$745.00

Call or write for information on other sizes & brands.

CDR-ZS100 Speed Module\$45.00
GRAYMATTER Application Software Z-MAX\$49.00
FBE Research Company, Inc. ZMF100a\$59.00

SPECIAL ANNIVERSARY OFFER

SL WABER: 6 Outlet Power Strip
15AMP • Surge & Noise Suppression\$12.50

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

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.

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.

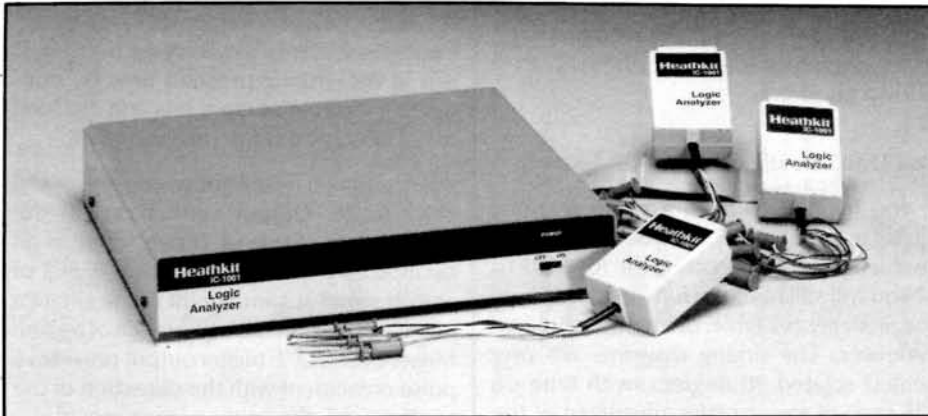
Demo disk for Strike and The PC Resident Speller **\$ 2.00**

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

The Heath IC-1001 Logic Analyzer



Terry Perdue
Heath Design Engineer

If you've ever tried to troubleshoot digital circuitry using only a multimeter, logic probe, and/or one- or two-channel oscilloscope, you've likely encountered a situation in which you wished you could observe the logic levels on a number of points in the circuit simultaneously. Without that ability, you're unable to determine the time relationships between the signals on a number of points in a circuit.

There is a piece of test equipment available that was designed to do just that — the Logic Analyzer. Most logic analyzers are prohibitively expensive for the home experimenter/designer or small service shop. They look similar to oscilloscopes, with built-in CRT displays, and an array of controls and switches to configure them appropriately for the job at hand, and they typically cost thousands of dollars. (And unfortunately, they do not double as oscilloscopes.)

Heath Company realized that many people who are involved in electronics, whether as hobbyists or professionals, already have a personal computer that could replace much of the traditional logic analyzer's circuitry. The PC's video monitor is ideal for displaying large amounts of data, and the keyboard is perfect for entering configuration information. Therefore, a logic analyzer designed to interface with a PC could eliminate much of the expensive display circuitry, switches, controls, and

power supply that are normally required. This would not only allow a dramatic reduction in the price, but the unit could be much smaller and more convenient to use.

The result of that realization is the new IC-1001. It is a 16-channel logic analyzer in a package the size of a typical library book. Its power is derived from a wall cube or external battery supply. With nothing but a power switch and LED on the front panel, one might be lead to believe that there is little inside the box. Yet inside, a single easy-to-build circuit board holds 45 ICs and other assorted components. The rear panel has a socket for the power cube, a couple of BNC sockets that can provide triggering for external equipment, a 9-pin RS-232 connector to interface to the PC, and three multi-pin connectors that are used to acquire data via three 'pods'.

There are two data pods and a clock pod. Each of the data pods provides eight data input leads, plus a ground lead; and the clock pod has a clock input and two qualifier inputs, plus a ground. The pods contain high impedance buffers to minimize loading on the circuit under test, and all connections are made with convenient spring-loaded hooked clips.

The cables between the pods and the rear panel connectors are flat ribbon types, so the analyzer can sit out of the way on a shelf

above your work area, with the cables routed under the unit.

Since the unit is a separate box, it does not tie up a slot in the computer, and may be easily moved from site to site. This also means that it may be used with a laptop PC, such as the '181 or '183, to provide a portable logic analysis system.

Operation

Operation of the IC-1001 is all menu-driven, with several pages of on-screen help available. The screen also displays current settings, error messages and configuration prompts. To configure the instrument, you select the active clock and qualifier levels, positive or negative logic polarity (whether a '0' represents a logic LOW or logic HIGH), Delay or Non-Delay mode, and a trigger word. In Delay mode, you also select a delay count; in Non-Delay mode, you select the number of words to be stored after occurrence of the trigger word.

You then 'arm' the Analyzer. Each time a clock pulse occurs coincident with the proper levels on the two qualifier inputs, the Analyzer compares the logic levels on the 16 data inputs with the trigger word you selected. This trigger word is made up of 0's, 1's, and X's. (An X tells the Analyzer to accept either a 0 or a 1 on the corresponding data lead.) If a match is detected, the Analyzer 'triggers'. In Delay mode, data

acquisition is delayed by the number of clock pulses you specified, up to 50,000. Then, the level on each of the 16 data inputs is saved on every subsequent qualified clock pulse. This continues until 2K words of data have been saved, or until you press the ESC key. The acquired data is then available for observation.

Non-Delay mode is similar, except that as soon as you arm the Analyzer, it begins saving data. When 2K words have been saved, the internal RAM overflows, but the last 2K words are always valid. When the specified trigger word occurs, the Analyzer continues storing data, but it also counts the clock pulses. When it reaches the count you specified, it stops. If, for example, it acquired 2K words of data prior to the trigger word, and you had specified that 1K clocks be counted after it, the Analyzer would hold 1K words of data after the trigger, and 1K words prior to the trigger, allowing you to effectively look back in time from the trigger event!

The Analyzer assigns the trigger word an address of zero, and each clock after the trigger is addressed in ascending order. In Non-Delay mode, the clocks prior to the trigger word have negative addresses.

The IC-1001 can present data in either a 'state' format of 1's and 0's, or a timing format, similar to what you might see on a 16-channel oscilloscope (see photos). The data may be displayed appropriately spaced in groups of 3 bits or 4 bits, for easy interpretation as octal or hex data, respectively. The included PC software allows you to zoom in or out when in the timing mode, to view as few as 4 clocks or as many as 2048. The trigger word is identified by a cursor line, and another cursor line may be moved around through the data, (or the data moved under the cursor) during examination. You can instruct the Analyzer to search for specific patterns in the acquired

data, and there are a number of other ways to conveniently position your 'window' on the data.

Both state and timing modes offer additional information, such as binary, hex, octal and ASCII equivalents of the data at each clock, and the position with respect to the triggering event. You can also save the acquired data to disk for later examination or comparison.

A 'checksum' function is also provided to allow you to easily compare data between two acquisitions or different sections of one. This can be useful when comparing a questionable piece of equipment to a known good one.

You Don't Really Need A PC!

If you happen to have a video terminal sitting around neglected, you can make it useful again by connecting the IC-1001 to it. You will still have the full functionality of the analyzer available, but with a few compromises. The timing diagrams are presented rotated 90 degrees, with time on the vertical axis, to take advantage of the normal scrolling capabilities of the terminal, and only 24 clocks may be viewed at a time. Yet a wide variety of positioning options is provided for selecting the section of data you want to examine. Of course, you don't have the direct capability of saving files on disk, but a means is provided for dumping the data to a computer for storage via a terminal emulation program. (You can select a pre-dump time delay to provide time to switch cables, etc.) On-screen help is still available, modified to be specific to terminal operation.

In this terminal mode, the Analyzer can automatically retrigger itself at intervals you select. You can position the display on an area of interest, and use this feature to watch for differences that might occasion-

ally occur in a marginal or intermittent system.

Uses

The IC-1001 Logic Analyzer may be used to check parallel printer interfaces, microprocessor-based appliances and amateur radio equipment, test equipment such as counters and DMMs, or any other circuitry that contains 5-Volt logic. The clock source may be derived from the circuit under test, or from a separate source, up to a frequency of 10 MHz.

If you design microprocessor-based equipment, the Logic Analyzer may be of use in monitoring program flow by connecting it to the address bus and clocking off of READs from the program ROM.

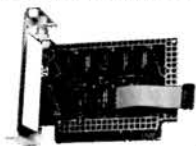
If you connect a cable between one of the BNC Trigger Output connectors on the back and the External Trigger input of an oscilloscope, you can observe a digital or analog signal at some point in the circuit a precise time after the occurrence of a given trigger event. One trigger output provides a pulse coincident with the detection of the trigger word, the other output provides a pulse when the delay count you specify is reached. The polarity of each of these trigger outputs is jumper selectable.

Summary

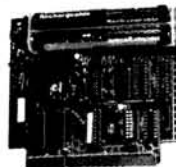
The Logic Analyzer is one of those test instruments that you may not use every day, but for which there is no convenient substitute. If you have one available when you need it, you're sure to save yourself time and frustration. At \$269, you can now afford to have one on your bench the next time the need arises.

HEATH/ZENITH 88, 89, 90 PERIPHERALS

16K RAM EXPANSION CARD



Only \$65.00
Shipping &
Handling \$5.00



REAL TIME CLOCK

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

2 PORT SERIAL/3 PORT PARALLEL I/O CARD

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



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
BUSHA PARK, CA 90620
(714) 952-3930

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

Specs In A Nutshell

Data RAM	16 bits wide by 2046 deep
Input impedance	1 Megohm shunted by 10 pF
Delay count	2 to 50,000 clocks
Clock frequency	Up to 10 MHz.
Setup time	< 5 nS. typ.
Hold time	< 5 nS. typ.
Channel-channel skew	< 5 nS. typ.
Baud rates	150 to 19,200 - manual or auto selection



C/80

Eliminate Library Duplication

Dave O'Meara

3204 W. Concord Way, #468
Mercer Island, WA 98040

All standard C/80 programs require the inclusion of **CLIBRARY** for input/output and other functions. Programs using floats(MATHPAK) require **FLIBRARY** and where applicable **MATHLIB**. When compiled, **CLIBRARY** accounts for approximately 2k bytes, **FLIBRARY** 3k bytes, and **MATHLIB** 2k bytes. The following article will attempt to describe an environment where C/80 users can write and execute C/80 programs without library overhead in every program.

The startling idea of eliminating the library overhead cannot be accomplished without the proper tools. First we require the proper commercial software; a relocatable assembler and linker (we will use Microsoft's Macro80 and Link80); and of course Software Toolworks' C/80 and C/80 MATHPAK. The remaining software must be user designed and written or is available from the author under the name **UMAN**(User Manager).

The general idea is to write a C/80 program that loads and executes subroutines written in C/80. The subroutine can then make use of the libraries linked into the calling program. The program(let's call it the master program or MP) must be written and linked to accomplish the following; 1) input the subroutine name in an interactive mode; 2) load the subroutine from disk; 3) execute the subroutine; 4) make available to the subroutine **CLIBRARY**, **FLIBRARY** and **MATHLIB**. It is interesting to note; one positive side effect from this arrangement is that on completion of a subroutine, control is returned to the MP and not the operating system, thus eliminating the inevitable warm boot that accompanies all standard C/80 programs.

The MP might look like the following;

```
#include "d:scanf.h"
#include "d:printf.h"

main() {
  static char command[20] = { 0 };
  static unsigned int address = 0x5000;

  printf("\nEnter Subroutine Name or ^C to Quit");
  scanf("%14s",command);

  /* attach default extension of SRT(subroutine) */
  if (index(command,".") == 1) strcat(command,".SRT");
  if (load (command,address)) call (address);
```

```
else printf("SUBROUTINE NOT FOUND %s",
command);
main ();
}

load () { ... user written functions

call () { ...
listing 1.
```

The load() function is easy to write but the call() function might require a little more expertise. Note that the load() function should return zero if the subroutine is not found.

Now we must compile and link the MP so that a subroutine can make use of the MP's I/O library(**CLIBRARY**). The trick is to link the C/80 libraries into user selected memory locations. Let's assume that MP is 5k bytes in length without **CLIBRARY**. Using Microsoft's Link-80 we link the MP in the following manner;

```
ALink80
*/d:170/p:300,SCANF,PRINTF,MP,STDLIB/S
*/d:103/p:1500,CLIBRARY,MP/n/e
```

The first Link-80 command line locates the **DATA** section of the MP at hexadecimal 170 and the **PROGRAM** portion at hex 300. Notice that **SCANF** and **PRINTF** are linked first. This allows enhancements to the MP without affecting the memory locations of **SCANF** and **PRINTF**. The second command line links the **CLIBRARY DATA** section at hex 103 and the **PROGRAM** section at hex 1500, and generates the command file MP. The **CLIBRARY** requires 109 (0x170 - 0x103) bytes for a **DATA** section while a liberal 384 is given to MP. Now that we know where **CLIBRARY** will be when MP is active, subroutines loaded and executed by MP can make use of the same library. Remember that **CLIBRARY** must be located at the end of a C/80 program since dynamic space allocation begins at the end of the **CLIBRARY** and proceeds upwards in memory. Therefore, care must be taken not to overwrite subroutines by allocating excessive space. Since the end of MP will be about 1C00 hex, we will begin the subroutines at 5000 hex allowing 13056 bytes for dynamic space allocation(0x5000 - 0x1C00).

The next step is to write and link a C/80 subroutine. Listing 2 is the source for a basic scientific calculator complete with five memory locations. When compiled as a standard C/80 program this program is 11k bytes in length(including

CLIBRARY, FLIBRARY and MATHLIB). As a subroutine under our MP program above it will be 9k bytes in length. If we expanded the MP to include the MATHPAK libraries the subroutine would be a mere 4k bytes (it's size under U-MAN). In this example, SCANF and PRINTF are also made available to subroutines. The DATA and PROGRAM sections of these two routines are located at hex 0x170 and hex 0x300. Note that if only PRINTF is required in the subroutine, both routines must still be linked in sequence.

To compile and link a C/80 subroutine we do the following;

1) C/80 Compile - C -a -m -w60 CALC

The -a switch forces the data and program sections to be compiled and located in sequence, since this is required, the first statement in the program must be JMP main, to bypass any data variables at the beginning to the subroutine;

The -m switch tells the C compiler to generate Macro-80 code;

The -w switch allocates extra space for the switch and case statements (required for this example);

2) Macro-80 Compile - M80 =A:CALC

generate the relocatable module CALC.REL;

3) Link-80 Linker - L80

*p:5000,CALC,STDLIB/S,MATHLIB/S,FLIBRARY

*u

*d:103/p:1500,CLIBRARY/e

The first command line links the calculator module and required libraries, for simplicity, we are including the MATHLIB and FLIBRARY libraries in the subroutine; the subroutine is linked at hexadecimal 5000 or decimal 20480; the second command line displays the undefined globals and the size of the subroutine, if the size is already known this step is not necessary;

Link-80 displays the following information:

Data FFFF FFFF 99 where 99 is the size of the subroutine in decimal; 99 / 256 determines the number of 256 byte blocks in the subroutine; (for this example the value is 34)

The third command line resolves the references to required CLIBRARY routines and is placed at the same location as the CLIBRARY for the MP program, this is what allows the subroutine the ability to use the MP CLIBRARY, the linker is then exited and an image of the subroutine is left in memory;

If the subroutine required SCANF or PRINTF, the third command line would have been: /d:170/p:300,SCANF,PRINTF

The problem now is to get the memory image on disk as an executable module. One method is to use the CP/M utility DDT and move the image to 0x100 and use the CP/M SAVE command to write it to the disk;

DDT

*M5000,9000,100

*^C

Save 34 CALC.SRT where 34 is # of 256 blocks as calculated above;

Another method is to write your own utility that reads the memory image and writes it to disk, the syntax might be as follows;

4)SAVMEMO(5000)P(34)N(CALC.SRT)

where O is the program origin, P is the number of blocks and N is the subroutine name (this utility is available with U-MAN);

The procedure to compile, link, and save a C/80 subroutine, can be illustrated as a CP/M submit file. There is one peculiar problem with Macro-80, it will not execute properly when

Xsub is active, thus Xsub must be placed in the submit file after Macro-80.

```

-----
                                904.668000000
A=1/x      B=x^2      E=SQR  C=CLR  D=CE
F=lnx      G=e^x      H=log  I=10^X J=y^X
K=MM1      L=MM2      M=MM3  N=MM4  O=MM5
P=STO      R=RET      S=SUM  T=ABS  X=MMC
-----
      U=sin  V=cos  W=atan  [ = ]
      [ 7 ]  [ 8 ]  [ 9 ]  [ * ]
      [ 4 ]  [ 5 ]  [ 6 ]  [ / ]
      [ 1 ]  [ 2 ]  [ 3 ]  [ - ]
      [ 0 ]  [ . ]  [ Z ]  [ + ]
-----
MEMORY1=2.5860000000
MEMORY2=5.1720000000
                                MEMORY3=1.0344000000e+01
                                MEMORY4=1.8446
                                MEMORY5=-4.523

```

Calc.c sample output

```

/* REL 1.0 FEB. 08, 1986
by Dave O'Meara
U-MAN Scientific Calculator
Compile: C -w60 calc.c
*/
#define NULL '\000' /* ASCII zero */
#define ESC '\033' /* escape */
#define BELL7 /* bell */
#define EXIT24 /* ctrl-x; program terminator */
#define DELF "\033K" /* delete to end of line */
#define CLEAR "\033E" /* clear the screen */
#define BRIT "\033p" /* reverse video */
#define NORM "\033q" /* normal video */
#define CURON "\033y5" /* cursor on */
#define CUROF "\033x5" /* cursor off */
/* #define UM 0 */ /* define for U-MAN inclusions */

#ifdef UM /* jump to beginning of program */
#asm /* when using U-MAN; a product available */
JMP main /* from the author of this program; */
#endasm /* U-MAN eliminates the duplication of */
#endif /* CLIBRARY, MATHLIB, & FLIBRARY in C/80 */
/*
/* programs(a great space saver!) */

float sin(), cos(), atan(), sqrt(), exp(), pow(), pow10(), ln(),
log(), fabs();
float atof(), ftoa();

/* globals for the calculator */
char buf[80],buf2[20]; /* input buffer & memory display
buffer */
int err,nbr,pnbr,mem; /* error switch; size of buffer; */
/* number of decimals; memory subscript */
float fnem[5],sgn,y=0.; /* memory array & sign switch */

#ifdef UM /* U-MAN communication structure */
#include "d:struct.c" /* beyond this exercise */
#endif

main() {
#ifdef UM
static int fir_tm = 0;
if(! fir_tm) { /* U-MAN - load MATHLIB & FLIBRARY */
fir_tm = um-auto = 1;
strcpy(um-x_file,"MFLIB");
return;
}

```



```

else um-x_file[0] = 0;
#endif

init('A'); /* initialize screen and working variables */
map(); /* display calculator */
inpt(); /* driver */
prts(24,1,CURON);
}

inpt() {
static int in;

while(bdos(6,0xFF)); /* clean up excess input */
while(! (in = bdos(6,0xFF))); /* input a console character */

if(in == ' ' && in != '\n') in -= 32; /* convert to upper case */
if(err) { /* initialize if error occurred */
err = 0;
init(3);
}

switch(in) {
case EXIT : return; /* clt-X; terminate */
case 'C' : init(1);break; /* clear the calculator */
case 'D' : init(2);break; /* clear current input # */
case 'Z' : sign();break; /* sign switch (-/+) */
case '.' : if(index(buf, ".") = 0) /* multiple decimals? */
break;
default : calc(in); /* process the input */
}
inpt();
}

calc(in) int in; {
static int s,cal;
static float f,x,x2;

if(isdigit(in) || in == '.') { /* input 0 - 9 or . */
if(cal && !nbr) init(3); /* if +,-,/, * clear display */
out(in); /* display the new number */
}
else {
if(index(buf, ".") = 0); else strcat(buf, ".");
x = atof(buf) * sgn; /* convert displayed number to float */

switch(in) {
case '=' : /* calculate answer for +,-,*,*/
switch(cal) {
case '*': f = x2 * x; break;
case '/': f = x2 / x; break;
case '+': f = x2 + x; break;
case '-': f = x2 - x; break;
}
case NULL: cal = 0; break;
case '*':
case '/':
case '+':
case '-': x2 = x; /* save value */
nbr = buf[0] = 0;
cal = in; return; /* save operator */
case 'A': f = pow(x,-1.0); break; /* functions */
case 'B': f = pow(x,2.0); break;
case 'E': f = sqrt(x); break;
case 'F': f = ln(x); break;
case 'G': f = exp(x); break;
case 'H': f = log(x); break;
case 'I': f = pow10(x); break;
case 'J': if(y == 0.) { /* Y to the X power */

```

```

y = x;
init(2);
return;
}
f = pow(y,x); y = 0.;
break;
case 'K': /* select memory */
case 'L':
case 'M':
case 'N':
case 'O': mem = in - 'K';
dismem(); return;
case 'P': fmem[mem] = x; /* save into memory */
dismem(); return; /* display memory */
case 'S': fmem[mem] += x; /* SUM into memory */
dismem(); return;
case 'R': f = fmem[mem]; break; /* retrieve memory */
case 'T': f = fabs(x); break; /* absolute value */
case 'U': f = sin(x); break;
case 'V': f = cos(x); break;
case 'W': f = atan(x); break;
case 'X': init(0); /* clear memory */
dismem();
default : return;
}

pnbr = pnbr 6 ? 6 : pnbr; /* minimum decimal point */
/* convert the floating point answer to a string */
if((s = f_a('f',f)) 20 || s 2) s = f_a('e',f);
if(s 2 || s 30) {
init(1); /* error has occurred */
prts(2,29,"ERROR");
err = 1;
}
else {
init(3); /* display the answer */
out(0);
if(sgn 0) sign();
}
}

f_a(typ,f) float f; char typ; { /* convert float to ASCII
string */

ftoa(typ,pnbr,f,buf);
return(strlen(buf));
}

sign() /* calculate sign of the input value */

sgn *= -1.0;
if(sgn 0.) prts(2,2," "); else prts(2,2,"-");
}

out(in) int in; { /* display the answer or the input string */
static int i,j;

j = nbr = strlen(buf); /* get size of current buffer */
if(nbr 30 && in) putchar(BELL); /* value too large */
else {
buf[nbr] = in; /* display the new string in reverse */
buf[++nbr] = 0;
for(i = 34; j = 0 && i 1; j--,i--) {
prts(2,i,"");
putchar(buf[j]);
}
if(index(buf, ".") = 0 && in) pnbr++; /* # of decimals */
}
}

```

```

}

dismem() /* display memory values */
static int row,col,i;

for(i = 0,row = 23,col = 1; i 5; ) {
  ftoa('e',10,fmem[i],buf2);
  prts(row + i,col,"MEMORY");
  prts(0,0,DELF);
  putchar(i + 49);
  putchar('=');
  if(i++ == mem) prts(0,0,BRIT);
  prts(0,0,buf2);
  prts(0,0,NORM);
  if(i == 2) { row = 20; col = 41; }
}

map() { /* display the calculator */
static int i;
static struct {
  int rw,cl;
  char *lne;
} l[9] = {
4,2,"A=1/x B=x^2 E=SQR C=CLR D=CE",
6,2,"F=lnx G=e^X H=log I=10^X J=y^X",
8,2,"K=MM1 L=MM2 M=MM3 N=MM4 O=MM5",
10,2,"P=STO R=RET S=SUM T=ABS X=MMC",
12,5,"U=sin V=cos W=atan [=]",
14,5,"[ 7 ] [ 8 ] [ 9 ] [ * ]",
16,5,"[ 4 ] [ 5 ] [ 6 ] [ / ]",
18,5,"[ 1 ] [ 2 ] [ 3 ] [ - ]",
20,5,"[ 0 ] [ . ] [ Z ] [ + ]";

map2('-',35,1,1);
map2('-',35,3,1);
for(i = 0; i++ 21; ) { map2('l',1,i,1); map2('l',1,i,35); }
for(i = 0; i 9; prts(l[i].rw,l[i].cl,l[i++].lne));
map2('-',35,21,1);
}

map2(ch,num,r,c) char ch; int num,r,c; {

```

```

prts(r,c,BRIT);
while(num--) putchar(ch);
prts(0,0,NORM);
}

prts(row,col,p) int row,col; char *p; { /* print string at
row/column */

if(row == 0 || col == 0); else {
  putchar(ESC); putchar('Y');
  putchar(row + 31); putchar(col + 31);
}
while(*p) putchar(*p++);
}

init(lvl) int lvl; /* initialization */
static int i,col,cnt;

switch(lvl) {
case 'A': prts(0,0,CLEAR);
prts(0,0,CUROF);
#ifdef UM
prts(25,1,DELF);
um-swt = 1;
#endif
case 0 : for(i = 0; i 5; fmem[i++] = 0.0);
case 1 : col = 2;
cnt = 33;
pnbr = 1;
sgn = 1.0;
y = 0.;
case 2 : buff[0] = 0;
case 3 : prts(2,col,"");
for(i = 0; i++ cnt; putchar(' '));
cnt = 32;
col = 3;
}
}

```

listing 2.

*

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.**

**EXPLORE
NEW WORLDS
WITH
HUG
GAME
SOFTWARE**

Continued from Page 10
in my Z-158? Does or has anyone
modified a Quadlink to run in Zenith PCs?

Sincerely yours,

Mark D. Weiss
11 Alan Road
Spring Valley, NY 10977

**Microsoft WORD 3.1
And FANSI-CONSOLE 2.00H**

Dear HUG:

Just a note to your readers about a small incompatibility I discovered between (among?) Microsoft WORD 3.1, FANSI-CONSOLE 2.00H, and the Z-150 CGA video card. If you are running WORD in the graphics mode (character attributes displayed as WYSIWYG), and spell check a document with the Library Spell command, Spell will come up in the color mode, and when finished, will return you to WORD in the color mode (character attributes shown as different colors).

Has anyone else run up against this? Is it peculiar to the Z-150 CGA card, or does it occur on the Z-158 or Z-240 series, as well? I have written Hersey Micro Consultants (the authors of FANSI-CONSOLE) to see if they can find and correct the bug.

FANSI-CONSOLE is otherwise an excellent, inexpensive replacement for ANSYS. It not only speeds up most screen writing (directory displays, TYPE command, and applications that use ROM BIOS for screen writes), but emulates DEC VT100 and DEC VT52/Heath H19/Z100 console, recalls lines scrolled off screen, and contains several other utility-type enhancements. It is not compatible with the Z-100.

Sincerely,

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

Michael Leblanc's Letter Of September

Dear HUG:

I read Mr. Michael Leblanc's letter in the September REMark with considerable interest, since FBE has attempted to go this route.

**CHUGCON SPECIAL !
PERKS
ONLY 49.95!**

This year we are making our \$49.95 CHUGCON show special price for Perks and Perks-PC available to our mail order customers nationwide. Offer applies only to mail orders for Perks and Perks-PC (reg. \$69.95) which include payment in full (check) with the order. Offer expires 12/31/87 and does not apply to PO's, COD or credit card orders, etc. Please add \$4.00 S&H, Michigan residents please add 4% sales tax also. Send order with payment to:

**Barry A. Watzman
560 Sunset Rd.
Benton Harbor, Mi. 49022
(616) 925-3136**

Last year, I added an EGA Card to my '151 system and decided to remove the Video Card. I developed a decoder PROM that would allow me to install a RAM chip on the CPU Card if I squished the two BIOS ROMs into one larger EPROM. The same method was described by Mr. Robert Maskasky in the May issue of REMark.

It occurred to me that this would make a nice product for FBE to produce. The package would include the decoder ROM, a RAM chip, and a squished version of the BIOS ROMs. All I needed to do was get permission from Zenith to produce the squished BIOS ROM.

After many telephone calls through "proper" channels to Zenith, I gave up and called a contact inside Heath. He gave me the number of a person in the ZDS Software Group to contact. After several non-returned calls, I gave up totally. By then, I had seen the ad for Dante Bencivengo's VMM150 Video Card Eliminator.

To salvage something from the effort, FBE has been selling the decoder PROM (Part number RM-150, \$6 for the first one and \$3 each thereafter, UPS shipping and in-

stallation documentation included) for several months. Of course, you do need an EPROM Programmer to squish the BIOS ROMs.

Regarding the copyright question, I believe that squishing the BIOS ROMs would be considered "fair use", as long as the squished EPROM ends up in the same machine as the original ROMs.

It would be nice if ZDS would produce a one-chip ROM BIOS for the '150 (like in the '138/148, '158, etc.), but if they don't want to, I'd sure be willing to do it. If any ZDS execs are reading this, you can reach me (or my machine) at (206) 246-9815. Thanks!

Sincerely,

Dave Brockman
FBE Research Company, Inc.
11648 Military Road South
Seattle, WA 98168

Another Patch For ZPC

Dear HUG:

Please add the following patch to Mr. Swayne's file of patches.

Continued on Page 44

FINAL Z-100 SOFTWARE CLOSE OUT

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

PART NUMBER	DESCRIPTION	LIST PRICE	SALE PRICE
MS-463-1	Z-Basic (16 bit) interpreter	\$175.00	\$10.00
MS-463-7	Multiplan	\$195.00	\$10.00
CB-463-11	Z-Chart	\$150.00	\$10.00
CD-463-2	Condor File Manager	\$299.00	\$10.00
PK-100-4	All 4 listed above	\$819.00	\$38.00
MS-253-1	Microsoft BASIC-80 (8-bit)	\$175.00	\$10.00
OS-53-2	CP/M-85 (8 bit)	\$150.00	\$15.00
OS-63-4	Z-DOS	\$150.00	\$20.00
CB-463-9	PECON Peachtree to Condor	\$99.00	\$15.00
RS-463-5	Peachtree Inventory Management	\$499.00	SOLD
WI-463-1	Remote Batch Terminal Emulator	\$899.00	SOLD

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

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

COLOR GRAPHICS UPGRADE. All memory chips (16 pieces 150 ns) required to update Z-100 Series computer for color. Installation instructions included. Order Memory Kit #100-64-16\$21.50

MEMORY UPGRADE. All memory chips (9 pieces 150 ns) required to upgrade from 128K to 192K RAM. Installation instructions included. Order Memory Kit #100-64-9\$12.50

ZMF100A by FBE Research. A modification package which allows 256K chips to be used on the old-style motherboard to reach 768K. Simple assembly with no soldering or trace cutting. Compatible with Easy PC and Gemini Emulator. \$60.00 alone or \$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 \$578.00, 31 MEG \$598.00 System without Drive and Controller\$239.00

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

*** PANASONIC DOT MATRIX PRINTERS ***

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

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

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

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

Winchester Hard Disk Drive Internal Set-up. Includes Winchester drive, controller/interface card, cables and all hardware. With 20 MEG (formatted) drive \$339.00 30 MEG \$359.00. May be installed in Z-148 using an Expansion Card sold below.

PTZ-148 Expansion Card for Z-148. Includes 2 expansion slots plus a clock/calendar. \$118.00

Winchester Hard Disk Drive External Set-up. Includes Winchester drive, controller/interface card, power supply and case with fan. With 20 MEG (formatted) drive\$528.00
With 30 MEG (formatted) drive\$548.00

ZFL-181-93 LAPTOP PORTABLE with the amazing Supertwist LCD backlit screen 640K RAM, two 720K 3.5" disk drives, clock, P & S Ports, 8 MEG. Retail \$2399.00. Payload\$1739.00

*** HALF HEIGHT FLOPPY *** DISK DRIVES

MITSUBISHI M2896 8"	48 TPI DS/DD	1.2 MEG\$375.00
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

*** SEAGATE HARD DISK DRIVES ***

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

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

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

Single 5.25" unit for Full Height Drive\$68.00
Dual 5.25" unit, for Half Height Drives\$92.00
Dual 8" unit with fan, for Half Height Drives\$188.00
Face plate for single drive in dual case\$8.00
For WINCHESTER Drive, with fan\$191.00

*** 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\$3.28 each
256K Dynamic RAM, 120 ns\$3.87 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



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.

On the Leading Edge

HUGCON 87, Keyboards, Special Awards, MACE Utilities 4.10, Microsoft Word 4.0

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

There were a number of interesting goodies at HUGCON this year, but nothing that I thought was spectacularly new. Perhaps the best new product was Enable for the Z-100. It was particularly appropriate that The Software Group, manufacturers of Enable, received the "Vendor of the Year" award for the Z-100 version of Enable. As I mentioned last month, the Z-100 Enable is almost certainly the last major software package to be released for the Z-100.

Of course, the new Z-386 computer system was displayed, and it has a new twist -- it can talk. The voice sounds suspiciously like the HERO robots, but I am not sure that I want my computer to talk back to me yet.

From my perspective, attendance at HUGCON was somewhat disappointing with respect to both members and vendors. In general, there was a distinct appearance that member attendance was down in the display area. At times, it was almost deserted. Vendor attendance was also clearly lower than in past years, and some vendors were conspicuous by their absence. Software Toolworks, one of the biggest supporters of Heath and Zenith, was particularly conspicuous by their absence. UCI (Easy PC) did not attend either.

As far as I could tell, there still was considerable interest in the discussion groups, and most of the discussions (including mine) were pretty well attended. Although part of my discussion was directly related to the new features of MS-DOS 3.20, I also included some informa-

tion about hard disks that was a direct result of your letters. Later in this article, I will tell you about the hard disk discussion and a way to increase your effective hard disk capacity by as much as 50%. It may even save you the cost of a hard disk.

I did manage to find something new that I thought was particularly interesting. CDR had a new floppy disk drive that has a 10 megabyte capacity. I have read about that for over a year in various Kodak advertisements, but I had not seen one until the HUGCON. The drive resembles a normal half-height 5.25" floppy drive in size, but it uses a cartridge that is quite similar to the 3.50" disks. This 10 megabyte cartridge is about the same physical size as a 5.25" disk and about twice as thick. Since this technology is so new, the prices are still artificially high -- around \$900 for the drive and about \$40 or so for each cartridge. Marc Brooks of CDR told me that "everyone" expects these prices to be about half that by the first of next year, so I will look forward to that. He also told me that he thought that these drives had an access time on the order of 100 milliseconds, so they are quite a bit slower than today's hard disks. Still, that is not too bad when I consider that my Z-100 has about a 75 ms hard disk. My general thought is that this new floppy drive with the 10 megabyte cartridge is a perfect medium to use for hard disk backup. I will try to get one when the prices become a little more reasonable. If you cannot wait for the prices to come down, you might check with CDR to find out about the current pricing and availability. As usual, their address and phone number is listed at the end of this article.

If you have had a chance to read other computer-related publications, you may have seen that John Frank is the new president of Zenith Data Systems. I always look forward to the Awards Dinner because there are a variety of interesting speakers, and this year was no exception. As you may know, John Frank was in marketing, and he has been credited with ZDS's astounding success in selling computers to the military and government. Of course, Bill Johnson, president of Heath Company, and Joe Schulte, president of Veritechnology Electronics Corporation (VEC -- the Heathkit stores) attended as well. It is nice to know that the presidents of all three companies take such an interest in HUG that they always attend the HUGCONs. I look forward to their comments each year.

I thought the comments of all these men centered on a "new spirit of cooperation" between ZDS and the two sister companies. Although that is good news, it is not new news. I have heard that before, but I thought that John Frank was much more "believable" than Bob Dilworth was. Even though I do think that this relationship has probably improved in the last year, I am quite hopeful that it will get even better. The primary impact on the user group is that we will see new products in the Heathkit stores much sooner. Who knows, we might even see them when they are announced!

John Frank did make one comment about an observation that IBM listens to their microcomputer user groups. I disagree. My experience is that IBM does not listen

to user groups much more than most companies, and in some cases, less. Although there is some evidence that IBM does listen to a few comments from the mainframe computer groups (e.g. SHARE and GUIDE), I have not seen that in the microcomputer arena. In fact, there is one specific area where it appears that IBM has ignored all users -- the area of keyboard design.

Keyboards

The keyboard is the most important aspect of any computer. Data entry is by far the most important computer function since, without it, we would have no need for computers in the first place. The lowly keyboard therefore assumes an importance beyond a simple piece of hardware.

Today, a number of people are making a big deal about the user interface to a computer. New operating systems, like OS/2, are supposed to have a better interface than DOS-based systems. I will believe that when I see it. The point is that the keyboard IS the user interface. We use the keyboard for commands and data entry, and if a keyboard is clumsy, that will increase the number of typos and other errors.

IBM essentially established the standard for keyboard layout with the Selectric typewriter keyboard. Many touch typing students learned how to type on that keyboard. One could also make a pretty reasonable argument that IBM established a standard for computer terminals with the 3270 display terminals that are used for mainframe computers. However, IBM took all of that background and experience, and dumped it into the trash can when they designed the microcomputer keyboards. These keyboards do not resemble either the Selectric or the 3270 layouts.

Actually, there is nothing wrong with change, but this is getting ridiculous. To date, IBM has designed four incredibly poor keyboards: the original PC, the PC Jr., the AT ("enhanced" keyboard), and the "new enhanced" 101-key keyboard for the PS/2 computers. All of these keyboards appear to be specifically designed to strike terror into any touch typist's heart. Each one of them is "non-standard" when compared to the Selectric. And each has its own specific problems. There are apparently a lot of people that agree with this view because one company, Key Tronic, has become

well-known for their keyboards that fix these problems.

I found that the original PC keyboard had two serious problems: the Enter key and the Shift key location. The original Enter key was two keys high, but there was an "extra" key on that line. If you were used to the L-shaped enter key, I found that I nearly always hit that "extra" key instead of the Enter key. That plays havoc when you are trying to use a word processor. The location of the Shift key was clumsy, but I found that I finally got used to that. I never did get used to the shape and location of the Enter key on the original keyboard. My typing speed was reduced by half as a direct result of that.

The original keyboard on the PC Jr. contained buttons instead of keys which made it virtually impossible to touch-type on that keyboard. Each key required a "key press", much like the keys on many inexpensive calculators. This keyboard was so bad that IBM eventually issued a "recall" and replaced it with a keyboard that had keys instead of buttons.

Then we come to the "enhanced" keyboard. This version was the first to have twelve function keys. In this version, the function keys were moved from the left-hand side of the keyboard to a row along the top. The other significant change was that the ESCape key was moved to the keypad area instead of staying in the standard, upper left-hand corner of the main keyboard. The good news was that this version of the keyboard had an L-shaped Enter key that was much easier to use. But it is clear that the designer of this keyboard did not use much microcomputer software and was not a touch typist because of the location of the ESCape key. This keyboard was the best of the lot because, with the exception of the ESCape key, most of the rest of the key locations were pretty usable. A lot of people object to the change of the function key locations, and I have found that it is difficult to get used to despite the fact that they are similar to my Z-100.

Last, but certainly not least, is the "enhanced" 101-key keyboard that is part of the new PS/2 computer series. This one is really a beaut! The ESCape key had to gather up its traveling togs again, and go to the left-hand side of the function key row. There are now two CTRL keys and two ALT keys. Both are located on the

bottom row with the Space Bar. And the Caps Lock key has been moved to the old CTRL key location. I learned this the hard way when I tried to interrupt the COPY command with a CTRL-C and found that a Caps Lock-C did not quite handle the problem. There were quite a few comments about this rather significant change when IBM announced this keyboard, and all of them were not complimentary. IBM responded that they had spent a lot of money on consulting for the new keyboard design, but it is clear that they did not talk to anyone who really USES a keyboard. Any touch typist who spends a considerable amount of time at a keyboard could have told them about keyboard design and its relationship to the popular microcomputer software.

I suggest that it is high time that all companies, especially IBM, quit trying to design the "ultimate" keyboard and standardize on one design. As I said before, I do not particularly object to change, but I do have considerable problems with non-standard hardware and software. For example, it is difficult to see why we are limited to 12 function keys. IBM could take a lesson from another one of their divisions and provide 24 function keys like some of the mainframe 3270 terminals. That is not a pipe dream because a 24-function key keyboard has existed for some time -- it is part of the 3270/PC system, so the idea is not even new.

Turkey of the Year Award

Contrary to what I said last month, I have decided to give a more than a little belated "Turkey of the Year" award to IBM for poor microcomputer keyboard design. Moreover, they have also earned a bronze, silver, and gold turkey cluster in lieu of second, third, and fourth awards respectively. All four of these keyboards have significant layout problems for touch typists who use microcomputer software. There has been little evidence that IBM is really concerned with the user based on their rather clumsy attempts at keyboard design.

Zenith Keyboards

I think that one of the best features of the Heath and Zenith computers has always been an excellent keyboard design. Their keyboards have generally had the keys in their standard locations using the standard design. Even the '150 had an L-shaped Enter key that was standardized in the Selectric. And the Z-100 keyboard is

probably the best keyboard design ever in terms of feel and key location. Even though my '248 is faster, and in some respects better, than my '100, I have been more than a little reluctant to convert files to the '248 because I still like the other keyboard better. I can type like a rocket on the Z-100. I still have not developed that kind of speed on the '248.

As I said earlier, I saw the Zenith 386 system at HUGCON. Unlike previous systems, it appears that Zenith decided to use a virtual copy of the IBM keyboard layout for the 386 machine. The rationale for doing that is to clearly make the Zenith 386 system more "compatible" with the IBM version. Nonsense and garbage! The "new" locations of the Caps Lock, two CTRL, and two ALT keys are not helpful at all.

The good news is I found out that a simple ROM change in the keyboard will "remap" the Caps Lock key and one of the CTRL keys (the one to the right of the Space Bar) so that at least that problem can be fixed. The disadvantage of this approach is that the old Caps Lock key (the new CTRL key) still has the LED showing the status of the Caps Lock toggle no matter what the ROM does for keyboard mapping. I can live with that as long as the keys are in the "normal" positions. If you have any kind of interest in a ROM like this, I suggest you write to Mr. Chas Gilmore at Heath Company at the address listed at the end of this article. I discussed this subject with him at HUGCON, and he would like to hear from you if you have an interest in a new keyboard ROM. If there is enough interest in this, perhaps Heath or HUG will make an updated keyboard ROM available for the 386 keyboard. It's nice to know that Heath Company still has an interest in a user's opinion in general, and my thanks to Chas for his assistance.

Lest you think I have forgotten about Zenith's involvement in the keyboard issue, I have a special award for ZDS.

Lemming of the Year Award

The lemming is a curious beast. Where one goes, others follow, even to the extent of going over a cliff and crashing into the sea. And so it is with keyboard design. ZDS has managed to earn the first annual "Lemming of the Year" award for their version of the IBM keyboard design on the 386 system. Enough on keyboards until the next IBM design comes out.

MACE Utilities 4.10

Although new technology has supposedly given us more speed, that is a double-edged sword. Even though the faster speeds allow us to do more work faster, it is unfortunately true that we can make mistakes faster too. Accidents, typos, and hardware failures can occur in the blink of an eye, and destroy days', weeks' or months' worth of work.

Each component of a computer system, hardware and software, must work together to help you accomplish a task. If one component fails, then all kinds of bad things can happen. Unfortunately, it is usually the human that is the weakest link in the system because we do make mistakes. Mistakes can range from deleting one important file to formatting a complete hard disk causing a loss of megabytes of data. Hardware fails too -- bad sectors develop over time, and occasionally a hard disk will completely fail with no notice at all, such as the original hard disks that consistently failed on the older ATs. Even some software has problems of one kind or another that can cause data loss.

For that reason, I always recommend a good set of recovery utilities. I like the MACE Utilities because they provide a wide variety of programs that are useful to all computer users, but are especially useful if you have a hard disk. In fact, I suggest that if you are planning to get a hard disk, you should also have some room in your budget for the MACE Utilities. This set of programs can be a real lifesaver as I discussed in the December 1986 REMark, and the new 4.10 release is even better. This release includes everything you need to maintain your hard disk system and recover from many kinds of accidents.

Release 4.10 includes three disks and manuals: Utilities disk (red), Hot Rod disk (yellow), and db Fix recovery utilities (green). All of the manuals have been revised, and they have been improved in the process. Since there has been at least one reported problem with the cache utility that I will discuss later, I have tested most of these programs on my '248 with a 40 MB and a 20 MB hard disk using Zenith MS-DOS 3.20 (BIOS version 3.29).

The MACE Utilities Disk

The red Utilities disk contains the recovery utilities as well as the menu and the on-screen help information. All of the

utilities can also be run direct from the command line.

Perhaps the most frequently used utility is UnDELETE which allows you to recover erased or deleted files. I have checked this one out pretty thoroughly (not always as part of a test), and it has recovered each file with no problems. The only real caution about any utility that can recover an erased file is that you MUST use it immediately after a file has been erased. If you do not, there is a good chance that DOS will reuse that space, and the data will be overwritten so there is no chance of recovery.

If you have ever formatted a hard disk, you will appreciate the UnFORMAT program that will recover most of the files from a formatted disk. The reason that this is possible is because some FORMAT programs do not overwrite data on existing disks -- they usually just initialize the disk directory and the File Allocation Tables. In general, it is usually not possible to recover all files on a formatted disk, but this program works quite well. I recovered all files in subdirectories from a formatted disk at UTA last year as I discussed in the December 1986 REMark.

RXBAK is a special program that can help you recover just about any file that has been erased or just about everything on a formatted disk. It creates a file in the root directory called BACKUP.M_U that contains a copy of the boot sector, root directory, and File Allocation Table (FAT). If this program is run BEFORE a file is erased or a disk is formatted, you can usually recover everything on the disk. My practice is to run this program for each partition on my hard disks with a batch file before I power-off the system. The manual suggests that RXBAK be run before and after applications such as word processing in the event that floppy disks are switched at the wrong time. DOS will very cleverly scramble the FAT when floppy disks are changed unexpectedly which is, by the way, one of the most important reasons that the CHKDSK command was developed.

The REMEDY program can diagnose a disk for bad sectors and fix them. If you are not using the Zenith DETECT command for a hard disk, the REMEDY program can be quite important. REMEDY runs the CHKDSK program, and then performs a non-destructive read of all sectors on a disk. If the sector cannot be successfully

read on the first try, then REMEDY marks the sector (cluster actually) as bad. Since heat can cause additional bad sectors to develop after a disk is formatted, it is a good idea to run REMEDY after the disk has been in use for a while. This program works for both floppy and hard disks. If you do not have the Zenith MS-DOS with the DETECT command to check your hard disk periodically, the REMEDY program is an excellent alternative, particularly if you use PC-DOS.

Two format programs are also included: one for floppies and one for hard disks. These are "safe" format programs because they do not overwrite the data clusters on a disk like some format programs do. Many format programs overwrite any data on a floppy disk. I checked my version of the FORMAT program (in MS-DOS 3.20), and it does indeed wipe out all data on a floppy disk. Both of the MACE format programs do not overwrite disk data, and it can be recovered with UNDELETE and UNFORMAT as already discussed.

The MACE Hot Rod Disk

This disk is truly worthy of the name "Hot Rod" because it contains a number of programs that you can use to speed up the performance of your system. It also contains a number of other useful utilities that most users will find extremely helpful.

FRAGCHK checks all files on the disk to see if they are fragmented, and if so, how many fragments there are. While this may seem, to the uninitiated at least, to be a technical discussion, a modest understanding of the fragmentation problem, and its solution, can significantly improve your system performance. Although I have discussed this before, I will briefly go through the basics again.

DOS allocates and frees file space in clusters. When you initially create the first file on a newly formatted disk, the clusters are allocated sequentially so that the cluster numbers may be in the order of 02, 03, 04, 05, and so on depending on the size of the file. Because these clusters are sequential, they are also contiguous which means they are physically "next to each other" on the disk. Because the file is contiguous, the read/write head on the disk drive does not have to move much (if at all) to access the clusters. As you update the file (and perhaps the software creates automatic backup files), DOS re-

leases the previously used clusters and allocates different (not necessarily new) clusters to store the file.

At some point, the file clusters are no longer contiguous (i.e. non-contiguous), and the drive's read/write head may have to move any number of times to access the file. In other words, the file has been fragmented (by DOS), and the file's data may be stored anywhere on the disk. Since the drive's read/write head has to move a number of times to access all of the data, you may notice that your system is "slowing down" as a result of all this drive head movement.

There are a couple of ways to fix this. First, you can copy the files to a newly formatted disk because the COPY (or BACKUP/RESTORE) command will unfragment the files as they are copied. This works well with a floppy disk system, but it tends to be awkward on a hard disk system. The easy way to fix this is to use a utility program that "unfragments", "defragments" or "optimizes" the disk.

Although you can use the "CHKDSK *.*" command to see how many non-contiguous files are in each subdirectory, you will need to check EACH subdirectory. The MACE Utilities includes a FRAGCHK program that checks each file in every subdirectory for fragmentation.

The UNFRAG program is also included which allows you to "fix" all of the fragmented files on the disk or partition. This is particularly useful if you have a number of large files as I do that tend to get fragmented all over the place. I think that this particular program is one of the most important that any hard disk owner can have aside from the UNDELETE and UNFORMAT programs. I run this program regularly (sometimes weekly or more) on both the 40 MB and 20 MB hard disks. The manual also suggests that file recovery is much easier when the files are not fragmented, so that is another good reason for this program.

Another useful program (SQZD) removes deleted directory entries from a current directory or an entire disk. This can reduce the number of places (and time) that DOS must search for the PATH command because all directory entries in the path must be searched up to the first unused entry.

If you do not have one of the various programs that list a sorted directory, you

will find that the SORTD (sort directory) program is a big help. It allows you to sort directory entries by filename, file type (extension), date or size. The program always sorts the subdirectories first, followed by the file names in the subdirectory. SORTD does not actually display the directory; it just sorts the existing directory entries. It certainly makes the DIR/P command fairly useful.

There are a number of things that can slow down a computer system, and disk I/O is one of the major ones, particularly on a data base file search. The utilities package contains four cache programs that can help reduce this disk I/O time by keeping a certain amount of file data in memory. One program, VKETTE, uses 26 KB of memory for floppy disk caching. The CACHE program allows you to specify the amount of conventional memory for caching hard disk or removable cartridge drives. CACHE-AT uses extended memory for the cache on a Z-200 or similar system. And CACHE-EM uses expanded memory for the cache.

Some users have reported problems with the MACE caching programs, and my information is that this is a bug in Zenith MS-DOS, not the MACE programs. This problem appears to have appeared in version 3.1 only based on the information I have. Various symptoms seem to have occurred, but the most common is a system freeze when the caching command is entered. That appears to be a specific problem with the programming of a special interrupt used for disk access in the BIOS for version 3.1. All of the caching programs seem to work just fine with version 3.2 that I am currently using.

Although you may find that caching will significantly improve some performance, it is important to know that it does NOT seem to improve everything. For example, you may not notice much improvement with a word processing program, although the COPY command and some types of data base access will improve significantly.

This disk also contains two of my favorite programs that I have added to the AUTOEXEC.BAT file: VKEYRATE and VSCREEN. VKEYRATE is a "keyboard accelerator" that changes the typematic speed and delay on the keyboard. The typematic speed is the speed at which a key repeats when you hold it down to duplicate characters. VKEYRATE allows

you to change that from 10 to 30 characters per second. I have mine set at 30 cps. The delay is the time that a key must be held down before it begins to repeat. VKEYRATE has a range of 250 to 500 milliseconds, and I have mine set at 500. It really improves the responsiveness of the keyboard.

The VSCREEN utility is a simple program that speeds up the screen display. It is useful when programs use the video BIOS routines, and it essentially replaces those functions with an optimized screen handler. The documentation says that programs using the video BIOS routines (such as DOS) will run 2-3 times faster, and I believe it. I have noticed a significant performance improvement with word processors such as WordStar 4.0 and MicroSoft Word.

The MACE DBFIX Disk

There are a number of ways to have a lot of fun with a computer system; however, one of them is not spending a weekend trying to recover a corrupted dBase file. There are many ways that a dBase file can be corrupted that includes everything from a user programming error to an actual bug in dBase. This is one of those programs that you probably will not need often (hopefully), but when you need it, you need it NOW.

This disk contains all of the programs necessary to help you recover a dBase file that has been corrupted. Since I really do not use dBase, I did not test this program except with a practice file that was included on the disk. DBFIX allows you to recover (or define) new headers as well as recovering corrupted records.

A Summary of MACE Utilities

The MACE Utilities were designed to run on a PC or compatible system, and I think that it is one of the best utility selections available. I have been impressed with the quality of their technical support, but I think that there is at least one more important factor. They use a Zenith system (specifically a Z-248), so I suspect that they can be much more helpful than many other vendors of similar utilities. Since the utilities were developed and tested on a Zenith system, you will not have to explain how partitions are assigned with the ASGNPART command in case you have a problem. They also know about any "hiccups" in Zenith MS-DOS (like the disk I/O problem in 3.1), so you do not need to go into any detail about that either. In short, I have more confi-

dence in the MACE Utilities than any of the others because of their specific knowledge of Zenith systems. Although they obviously have other systems as well, it is comforting to know that they understand Zenith computers.

All of the MACE Utilities provide something for just about every computer user. Although this new version is about \$20 higher than the one I discussed in the December 1986 REMark, it has additional features and programs that still make the package an excellent value. If you have a floppy disk system, I think that you should seriously consider MACE Utilities since you can speed up your system's performance as well as recover files. If you have a hard disk system, MACE Utilities provides just about everything you need to maintain optimum system performance, and this package should probably be your next software purchase. All in all, the MACE Utilities software is highly recommended for all computer system users.

Microsoft Word 4.0

Since I reported on some of the problems that I had with Word 3.10 last month, I thought I would give you an update now that Microsoft has announced version 4.0. I checked with Microsoft to see if any of the problems that I mentioned last time had been fixed in the 4.0 release. They have not.

Although Word may be adequate for many purposes, I find that the 19 K memory limitation for indexing, autosort, and other similar features is just too small for my use. I was told that, although they were aware of the problem, nobody seemed to know when or if this limitation would be changed.

The other major problem was the fact that I was severely limited in the size of the file that I can edit because of all of the formatting that is stored in memory. In one case, I had a FlipFast book file that was only 60 K that Word had extreme difficulty in editing despite the fact that I have 640 K of memory. Somehow I expected more from a word processor that sells for a list price of \$450 and is supposed to be state-of-the-art.

For what it's worth, it appears that Word 4.0 consists primarily of enhancements from all reports. I do not have 4.0, and I do not expect to get it because I can see no logical reason for spending the money for the update when it does not fix any of

the previous problems. In general, I understand that the user interface (i.e. menu) has been streamlined, and that Word is now much faster. Although I am sure that there have been other good changes, that seems to be the thrust of the advertising that I have seen.

In the Future

As usual, there was more to write about this month than there is space to do it. I still have a lot of information to share with you about hard disks as a result of a number of your letters. I guess that will have to be deferred to next month. I still appreciate your letting me know about your questions and interests -- it helps a lot.

If you have any questions about anything here, be sure to include a self-addressed, stamped envelope (business size preferred) if you would like a personal reply to your question, suggestion or comment. I'll look forward to hearing from you.

Products Discussed

Mr. Chas Gilmore (see text)
Heath Company
Hilltop Road
St. Joseph, MI 49085

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

Enable
PC Series \$695.00
Z-100 \$300.00 (Introductory price \$195.00)
The Software Group
Northway Ten Executive Park
Ballston Lake, NY 12019
(518) 877-8600

FlipFast Guide to Z/H GW-BASIC \$21.95
FlipFast Guide to Z/H MS-DOS 24.95
S-A Design Books
515 W. Lambert, Bldg. E
Brea, CA 92621-3991
(714) 529-7999

Word Version 3.10 \$450.00
Microsoft
13221 SC 26th St., Suite L
Bellevue, WA 98005
(800) 426-9400 (Orders only)

10 MB floppy drive (see text)
CDR Systems, Inc.
7210 Clairemont Mesa Blvd.
San Diego, CA 92111
(619) 560-1272





Use your Z181/183

as a portable VT220/240 workstation with:

- TRUE DOUBLE HIGH/DOUBLE WIDE
- "TRUE" 132 COLUMN MODE (128 actual)
- TRUE SMOOTH SCROLLING
- KERMIT & XMODEM FILE TRANSFERS

ZSTEMpc™ VT220 Emulator \$150
ZSTEMpc™ VT240 Emulator \$250

including 4014 and REGIS graphics.

also available **PS220™ Keyboard**

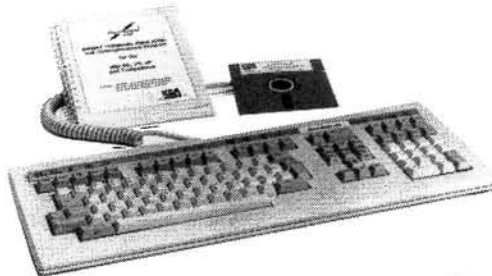
(see our *PowerStation* advertisement)

KEA Systems Ltd.

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

PowerStation™

A Complete VT220 / VT240 Work Station Upgrade
 for Zenith PC's, AT's and Compatibles



"You'll never know
 you are not using
 a real DEC terminal
 unless you take
 advantage of the
 many extended
 features."

PowerStation™ 240 \$389
 VT240 style keyboard and ZSTEM VT240 Emulation Software.

ZSTEM pc™ - VT240 Emulator Emulation Software only. \$250
 VT240/241 Emulation software with all the features of ZSTEM VT220 plus
 ZSTEM 4014 and REGIS graphics.

PowerStation™ 220 \$289
 VT220 style keyboard and ZSTEM VT220 emulation Software.

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

EGAmate™ \$39
 Daughterboard option for 132 columns on most
 standard EGA adaptors.

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

ZSTEMpc™-VT100 Emulator \$99
 High performance COLOR VT100. True double
 high/wide, smooth scrolling. ISO and attribute
 mapped color. XMODEM and KERMIT, softkey/MAC-
 ROS, DOS access. Available to the A/F and Navy on
 AFCAC 254.

KEA Systems Ltd.

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

See us at
 COMDEX &
 DEXPO

Z150/160 HARDWARE ENHANCEMENTS

TURBOPLUS V2.0

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

TurboPlus V2.0 WORKS Where Others Fail!

Price \$125 ppd.

VMM150 VIDEO CARD ELIMINATOR

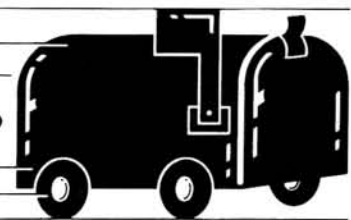
Use ANY Video Card in 150 (even GB-1)
REMOVE the Heath/Zenith Card
 Free One Standard Size Slot
 Use Less Power, Run Cooler
 Simple Plug-In Installation
 Includes PC Board, Decoder and New Back Panel

Price \$45 ppd.

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

ABSOLUTE 15 DAY MONEY BACK GUARANTEE

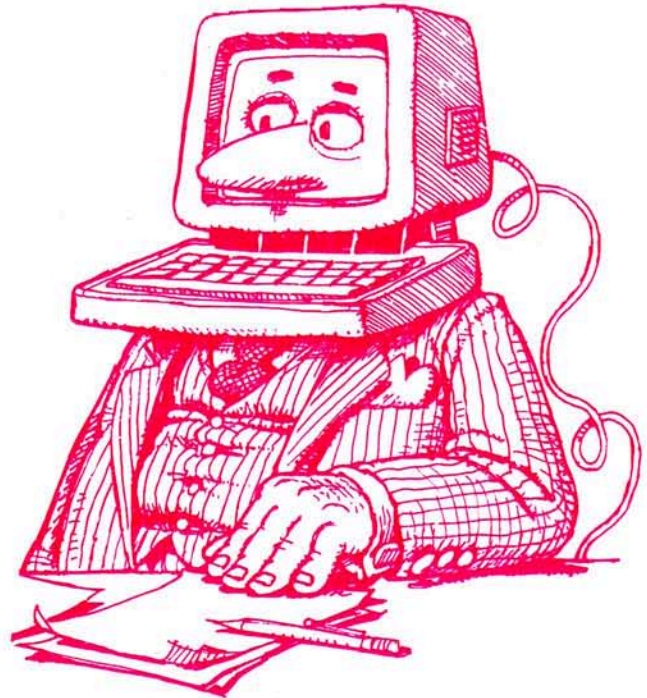
MOVING?



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

The Heath Voice 2000

Terry Perdue
Heath Design Engineer



Can We Talk?

How would you like to give your PC-compatible computer a voice of its own? Now you can! Heath has recently introduced an accessory which allows you to add speech, sound effects and even music capabilities to your PC at a very attractive price.

The new HV-2000 consists of a small circuit board that plugs into an expansion slot in the PC, an attractive external speaker, and the interface software. Assembly of the kit takes only a couple of hours.

Once the board is plugged into the computer and the supporting software is installed, your computer will be ready to speak any ASCII text you'd like it to. The text can come from an MS-DOS command line, a disk file, even from some external device connected to the computer through a serial port!

The software consists of a VOICE.SYS device driver and several utilities. To install the device driver, you need a CONFIG.SYS file on the disk or partition that you boot from, containing a command line of the following form:

```
DEVICE=VOICE.SYS 300
```

This will automatically install the device driver when you boot your computer. The '300' in the above example specifies the address (in hex) that you have jumped

the voice card for. Once installed, the device driver software remains resident in memory, ready to speak any ASCII text that is written to it, whether from DOS or a high-level language, such as BASIC.

The device driver's name is HV. From DOS, you could cause a text file to be spoken by using one of the following resident DOS commands:

```
COPY <filename> HV
TYPE <filename> > HV
      (using I/O re-direction)
```

Similarly, typing DIR > HV would use I/O re-direction to speak to the directory.

Tell Me More!

Files created in the Document mode of WordStar are compatible because the VOICE device driver ignores the additional spaces that WordStar adds to justify lines (these spaces have the high bit set) and the control characters that are used to cause parts of text to be printed in boldface, underlined, etc.

The software normally accumulates and converts text until a linefeed character is encountered (or until 255 characters have been written without a linefeed), before speaking the string. An exception occurs if a word is found to be hyphenated at the end of a line. In this case, the driver waits for an appropriate point in the next line of text, so that the hyphenated word is converted properly.

The driver also watches for punctuation that implies that an extra pause is appropriate, a short pause after a comma, and a longer pause after a period, for example. This makes the speech sound more natural, and therefore, adds clarity.

By the way, when you reach the INITIAL TESTS section of the HV-2000 Assembly Manual, you are instructed to plug in the board, make the software preparations outlined earlier, re-boot your system, and type SPEAK SPEAKME.TXT. If you did everything right, your computer will give you a nice verbal pat on the back!

The SPEAK Command

A SPEAK.COM transient program is provided that has three useful forms.

```
(1) SPEAK <filename>
```

This causes the specified file to be spoken and displayed.

```
(2) SPEAK "<string>"
```

This speaks the text string in the command line. For example, you could have your computer greet you when you turn it on by including the following line in your AUTO-EXEC.BAT file:

```
SPEAK "Good morning. Have a nice day."
```

```
(3) SPEAK COMn: <baudrate>
```

This form speaks and displays text sent to a serial port by an external device. XON/

XOFF handshaking is provided, as is a 60K buffer, in the event that the external source does not respond to handshaking. The baud rate may be between 150 and 38,400.

The NEWORDS Utility

Although the device driver contains well over 500 rules with which to determine the proper pronunciation of each word, some words may be pronounced incorrectly. Another program that is furnished, NEWORDS.EXE, allows you to create and maintain an exception file containing unique words, and to specify phonetically the pronunciation you desire for each word. This is an easy-to-use interactive program. The driver looks for the exception file, WORDS.EXC, at boot-up. If it is found, it gets loaded as part of the driver. Each word encountered in a text string is looked for there first — if not found, the standard rules are applied.

The device driver is not 'write-only'. Reading from it returns a string consisting of the mnemonics corresponding to the phonemes* chosen for the last-spoken string of text. This feature may be used to provide a starting point when you want to create mnemonic strings, and is in fact used for this purpose by the NEWORDS utility. When you enter a word, the program speaks the word using its built-in rules, displays the mnemonics of the phonemes used, displays a list of all valid mnemonics, and prompts you to enter a new string of mnemonics. It then speaks the new mnemonic string, and asks if it sounds okay. If not, the process continues until you're satisfied, at which time it adds the word and mnemonic string to the WORDS.EXC file and returns to the main menu.

* *Phonemes are the basic 'building block' sounds making up the words of a given language. A mnemonic name is assigned to each of the 64 phonemes* to refer to the sounds they represent. For example, to phonetically specify the words 'voice card', you might use the mnemonic string v o i e s p a k a h e r d. The p a mnemonic represents a pause between the words.*

If You Want To Get Fancy

In addition to plain text, a file or string may contain mnemonic strings and/or voice attribute specifiers, enclosed in braces ({ }). This allows you to spell unique words phonetically, and change any of the voice attributes at any point within the text. The voice attributes include:

- 64 phonemes,
- 4 phoneme durations,

- 16 speech rates,
- 4096 discrete inflection levels,
- 32 transitioned inflection levels with 8 transition rates,
- 4 octaves of musical notes,
- 16 amplitude settings,
- over 250 vocal tract filter settings, and
- 8 articulation rates.

With a little practice, very natural-sounding speech is possible, as well as some interesting sound effects, music, and singing. (The musical notes are very accurate.)

The voice normally defaults to a monotone, but you can add automatic inflection by either following the command line in CONFIG.SYS with a /I switch, or by preceding a line of text with an !. Although the inflection changes are essentially random, they are not instantaneous, rather gliding from one pitch to the next. This makes the speech more natural and less monotonous. Other software switches are available to:

- add a brief pause between words for better clarity,
- cause punctuation to be spoken (the words used may be changed via NEWORDS),
- cause numbers to be spoken digit-by-digit (normally 12345.6789 is spoken as twelve thousand three hundred forty-five point six seven eight nine '!'),
- select the default voice attributes,
- disable use of the exception file, and
- lock the options in a specific state.

What Can I Use It For?

There are countless ways to enhance your computing with the HV-2000. You can add voice prompts to your batch files to announce their progress, add verbal warnings to your programs to replace beeps or error messages that might be missed, add prompts and sound effects to game programs, etc.

The ability to speak ASCII text entering a serial port will be of interest to Hams and others who have the equipment to copy Morse code and teletype off the air. (If a group of characters contains at least one letter and one digit, it will be spoken character-by-character, so station call signs are spoken correctly.)

There are also many applications in which the HV-2000 will be especially beneficial to the blind population.

The Undocumented Feature

Although it is not documented in the manual, there is another, secret 'switch'

that may be specified in the command line!

When the Naval Research Laboratories developed the basic algorithm and rule set incorporated into the device driver in the early 1970's, it contained around 330 rules. Many additional rules were added during the course of the HV-2000's development. To aid in this process, an additional option was written in. This option was left in for those who are interested in seeing the rules that are being chosen to speak a particular string of text. This option is enabled by including an R switch in the command line.

The rules are written to the display beginning at the current location of the cursor, one rule per line. The next section offers a brief description of the algorithm and the significance of the characters displayed when this option is enabled.

Text-To-Speech Converter Operation

A text string is scanned from left to right. As each character is encountered, it is compared with a section of the rule table that contains rules beginning with that character. A rule consists of two sides — the input and the output. The output side contains the hex values corresponding to the required phonemes. The input side consists of a bracketed quantity, which must exactly match a character or group of characters in the text string (disregarding case), and in most rules, one or more 'tokens' which represent the context of the bracketed quantity. The tokens used are as follows:

! = punctuation or space
= 1 or more vowels
+ = a front vowel
^ = a consonant
: = 0 or more consonants
. = a voiced consonant
\$ = a consonant that influences a following 'u'
& = a sibilant
@ = a suffix

For example, if the letter l is encountered while scanning a string, the letter l and the characters preceding and following it are compared with a section of the rules table that begins as follows:

```
![I]!      ahl ie
[ING]!     e n
![IN]      i n
![I]'      ahl ie
[IN]D      ahl ie n
[IER]      ie er
#:R[IED]!  ie d
```

If the 'l' stands alone in the source text, (spaces or punctuation on both sides of it),

it will satisfy the first rule, and the phonemes required to speak the name of the letter will be appended to the output. If the 'l' is part of the suffix 'ING', it will match the second rule, and the appropriate phonemes will be appended to the output. In the latter case, the brackets enclose 'ING', so the next character to be scanned will be whatever follows the 'ING' suffix. To match the last rule shown above, the 'l' would have to be followed by an 'ED', then by punctuation or a space, and be preceded by an 'R', which is optionally preceded by a consonant(s), which is preceded by at least one vowel. The word 'carried' would match this rule.

Therefore, once the characters within the brackets on the input side of a rule are found to match a portion of the string exactly, the characters and tokens on the left and right of the brackets are compared with characters on either side of the matched characters in the string, from the center toward the outside. When every-

thing matches, the phoneme values from the output side of the rule are appended to the output buffer, the characters in the string that matched the bracketed characters in the rule are skipped over, and the process continues until the end of the input string is reached. Finally, the phonemes are output to the hardware.

Since the last rule in each section of the table consists of the character with no tokens, in this example [l], a matching rule will always be found. The position of each rule within the table is significant. That is why the Rules option was incorporated — when adding a rule to cover some discrepancy in pronunciation, it is important to note the rules that were being used, in order to properly locate the new rule. Additional testing is then required to ensure that the new rule does not cause problems with words that were previously spoken correctly.

Parts of a string that are in braces, (mne-

monics and attribute specifiers), bypass the rules, and are handled differently.

Summary

Giving your computer a voice adds a new dimension to computing. And while you can take the time to get as fancy as you want with the speech, music and sound effects for games and other novelty uses, very little effort is required to add voice to your more serious applications. So give your computer a voice. It may thank you.



DOMINO COMPUTERS

Z89/90 SOFTWARE

ELECTRONIC TYPING	\$ 79
CONDOR DBMS	671
CBASIC (CP/M)	111
PEARL III	359
MICROSTAT	199
MAGIC SPELL	283
DATASTAR	236
WORDSTAR	316
MAILMERGE	108
MICROSOFT BASIC	140
MICROSOFT COBOL	316
MICROSOFT FORTRAN	156
CP/M OS	140
MULTIPLAN	279
MAGIC WAND	236
PROPERTY MGMT	746
GENERAL LEDGER	316
INVENTORY CONTROL	396
SALES INVOICING	236
ACCTS RECEIVABLE	316
SUPERCALC	159

Z100 SOFTWARE

Z-CHART	\$ 78
CONDOR DBMS	338
CONDOR FILE MGMT	155
LOTUS 1-2-3	350
SUPERSORT	103
WORDSTAR 3.3	325
MAILMERGE 3.3	200
SPELLSTAR 3.3	100
MBASIC (CPM-85)	125
MULTIPLAN	125
CP/M-85 OS	125
PEACHTREE GL	295
PEACHTREE INVENTORY	295
PEACHTREE INVOICING	195
PEACHTEXT 5000	250
MICROSTAT FOR Z-DOS	200

CALL FOR SPECIAL HUG DISCOUNTS
ON NEW PRODUCTS.

WE STOCK MOST ZENITH PRODUCTS

WE RENT COMPUTERS, TOO!

GET YOUR ZENITH CHARGE CARD
THROUGH DOMINO COMPUTERS



AUTHORIZED DEALER

108 N. Hickory Avenue, Arlington Hts., IL 60004

312/870-8707

SURPLUS COMPUTER PARTS

- Largest Surplus Electronics Dealer in Western Michigan
- In Business over 40 Years
- Over 7,000 square foot Store and Warehouse

Examples of Zenith Salvage Products:

Z-100 Motherboards	\$20.00
Z-100 Power Supplies	\$25.00
Z-100 Winchester Controller	\$50.00
Z-151 Disk Controller	\$25.00
Z-171 CPU Boards	\$35.00
Z-158 CPU Boards	\$25.00
Z-241 CPU Boards	\$15.00
Z-200 Power Supplies	\$40.00

"Style Writer" Software for H-89
Disk Drives — 1/3 and 1/2 Height

Many other Zenith Salvage Available
All Surplus/Salvage Sold **AS IS - WHERE IS**

No Catalog — Please Call for Quotes

We Ship U.P.S. — C.O.D. (ONLY)

Sorry — No Foreign Shipments

Surplus Trading Company

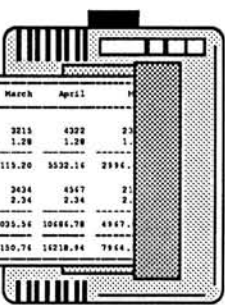
"THE HOUSE OF EVERYTHING, ALMOST!"

609 Paw Paw Avenue P.O. 1082 Benton Harbor, Michigan 49022-1082

CALL: LES TURK · 616 926-6391

LONG & LOUD!

Sideways & Banner Printing Utility for Dot-Matrix Printers



	January	February	March	April	
REVENUES					
Widget Sales (units)	2634	2345	3215	4322	23
Price each	1.28	1.28	1.28	1.28	1
Widget Sales (\$)	4677.12	3001.40	4115.20	5522.16	2336
Kadget Sales (units)	4221	3432	2424	4547	2
Price each	2.34	2.34	2.34	2.34	2
Kadget Sales (\$)	9877.14	8077.48	8025.56	10484.78	4947
TOTAL REVENUES	14554.26	11078.88	12140.76	16006.94	7984

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.

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%)

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**

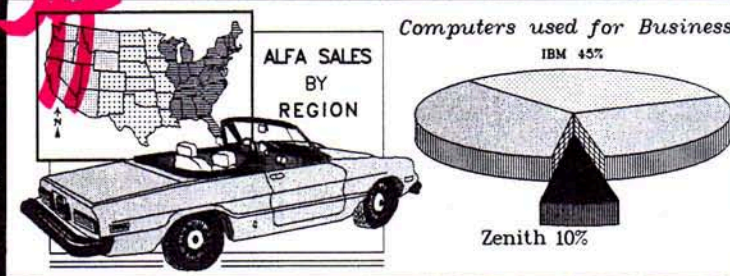
REMBRANDT runs on H/Z-89's, 90's, 100's and H/Z-19 equipped machines.

ShowOff

WILL IMPROVE YOUR IMAGE

Now, with the new ShowOff Art/Utility disk you can:

- Capture CGA graphs
- Use .MAC art
- Print ShowOff graphics on laser printers
- Convert ShowOff pictures to MAC format



With ShowOff, your Z100 will produce outstanding hires graphics.

ShowOff offers:

- 640 x 480
- 92 fill colors
- 92 patterns
- 25 text styles

AND, graphic compatibility unmatched by any other system

ShowOff \$79 ShowOff+Logitech Mouse \$174
demo disk \$3 ShowOff Art/Utility disk \$15

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

Versions of ShowOff available for Logitech, MS and PC Mouse, HIPAD and Summagraphics digitizers.
order direct: VISA MC check min requirements: Z100 Color RAM 384K RAM MS-DOS 2.0



Expanded HUG Discount List Featuring Products Eligible For Discount To HUG Members Only

HK-200	10%	Z-516	10%	ZDE-1217-BO	20%
HS-158-H	10%	Z-525	10%	ZDH-1211-BO	20%
HS-248-2	10%	ZA-170-1	10%	ZDH-1217-BO	20%
HS-248-SX	10%	ZA-170-3	10%	ZF-148-42	20%
HS-317-20	10%	ZA-170-4	10%	ZF-171-42	20%
HS-386-A	10%	ZA-180-20	20%	ZF-248-81	20%
HWD-20	10%	ZA-180-21	10%	ZFL-181-93	20%
HWD-20-AT	10%	ZA-180-35	10%	ZMM-149A	20%
TM-140	10%	ZA-180-40	10%	ZMM-149P	20%
TM-158	10%	ZA-180-45	10%	ZMM-1470-C	20%
TM-159	10%	ZA-181-4	10%	ZSS-100-27	20%
TM-170	10%	ZA-181-5	10%	ZVM-135	20%
Z-205-4	10%	ZA-181-7	10%	ZVM-1200-1	10%
Z-207-7	10%	ZA-181-8	10%	ZVM-1220-A	20%
Z-304	10%	ZA-181-9	10%	ZVM-1230-A	20%
Z-316-8	10%	ZA-181-17	10%	ZVM-1240-A	20%
Z-319	10%	ZA-181-18	10%	ZVM-1300-1	10%
Z-405-1	10%	ZCM-1390	20%	ZVM-1330	20%
Z-416	10%	ZCM-1400-1	10%	ZVM-1380-C	20%
Z-416-2	10%	ZCM-1490	20%	ZW-248-82	20%
Z-416-C	10%	ZD-12	10%	ZW-248-84	20%
Z-417	10%	ZD-200	10%	ZWL-183-92	20%
Z-445	10%	ZD-372	10%	All software on Pages 56, 57, and 58 of the	
Z-449	10%	ZD-400	10%	Christmas Catalog (#208)	20%
Z-505	10%	ZD-800	10%		
Z-515	10%	ZDE-1211-BO	20%		

Products and prices appearing on list subject to change without notice.



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

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

NOTES

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

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

KEYMAP To KEYMAC Upgrade

If you own the Z-100 KEYMAP program (885-3010-37), you can upgrade to the new KEYMAC keyboard macro processor for the Z-100 (885-3046-37) for only \$10.00. Just send your original KEYMAP disk and \$10.00 to HUG, Attn: Nancy Strunk, P.O. Box 217, Benton Harbor, MI 49022-0217. For a description of KEYMAC, see the August 1987 issue of REMark.

HUG P/N 885-3040-37 HADES Update

Murphey's second postulate states: "There's always one more bug." Unfortunately, such WAS the case with HADES, HUG P/N 885-3040-37. The problem corrected, only occurs with a 32-meg hard disk and only in the file mode. If you have version 1.01 or lower, return your original disk to Nancy Strunk, Heath Users' Group, P.O. Box 217, Benton Harbor, MI 49022-0217, and your disk will be updated to version 1.02, FREE!

HUG P/N 885-3043-37 SCREENDUMP Update (Correction)

Original owners of SCREENDUMP, HUG P/N 885-3041-37, can update their disk

set to the new version of SCREENDUMP, HUG P/N 885-3043-37 for \$10 and BOTH original disks to Heath Users' Group, Attn: Nancy Strunk, P.O. Box 217, Benton Harbor, MI 49022-0217.

HUG P/N 885-8005 HDOS MAPLE UPDATE

Version 2.0.8d of MAPLE for the HDOS operating system is now available. This version will allow 2400 baud operation. Previous owners of MAPLE can update their disks free by returning their original to Nancy Strunk, Heath Users' Group, P.O. Box 217, Benton Harbor, MI 49022-0217.


HUG P/N 885-8040-37 HELP Update

John Stetson's very popular HELP program, HUG P/N 885-8040-37 has been updated by the author. Now included on two disks, are separate versions for the H/Z-100 and PC Compatibles. Although the H/Z-100 version is fixed, the PC Compatible version is ever changing, and now includes, the ability for color output for color monitors, Zenith's version 3.2 of MS-DOS, and a summary of ANSI escape sequences, to name a few. Updates for current owners of HELP can be obtained for \$5 and the return of the original disk to: Heath Users' Group, Attn: Nancy Strunk, P.O. Box 217, Benton Harbor, MI 49022-0217. For a description of the original HELP program, see the March 1986 issue of REMark.

HUG P/N 885-8046-37
MS-DOS
Assembly Language Utilities
UPDATE

John Stetson, the original author of the HUG MS-DOS Assembly Language Disk, has updated this product with additional utilities and corrections, now making it a 2-disk set. Some of the additional features include: 1) An overview of MS-DOS V3.2 features, 2) TSR program to change the CPU speed on an H/Z-200, 3) BIOS modifications to the H/Z-100 MS-DOS 3.1 BIOS and PC MS-DOS 3.2 BIOS to allow exchange of 5-1/4" 96 tpi diskettes (not the 1.2 Mb high density type). 4) Sorted Directory utility program version 5.7, and 5) Miscellaneous changes and corrections. Original owners of this product can update their disk by returning it along with a check for \$5 (made out to HUG), to Nancy Strunk, Heath Users' Group. P.O. Box 217, Benton Harbor, MI 49022-0217.

✱

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 Software Select 4.77/6.67 MHz Hardware Reset. Satisfaction Guaranteed.
 - Prices: Assembled \$34.95, Kit \$24.95

H/Z 89 20 Meg Winchester ONLY \$595
 Boots from Hard Disk. ST225. For H150 - \$350

SOFTWARE for H/Z 150/160, 100, 89, 8!
 Perfect Funds \$29.95 Perfect Money \$19.95
 Paycheck \$39.95 Perfect Printer \$19.95
 Turbo Pascal \$69 ** SMARTWATCH for 150 \$39

Micronics Technology (904)-897-4257
 449 Barbados Way, Niceville, FL 32578
 Checks, VISA, MC. Shipping \$2. Hard Disk \$15

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

**If You Don't Have WindowDOS 2.0,
 You're Wasting Time!!**

Once you've experienced *the convenience of instant access to DOS commands*, you'll never be satisfied with returning to DOS to list files, format diskettes, or copy, rename, or erase files. Nor will you be happy with a DOS shell, because shell programs are just as inaccessible as DOS when you are using an application program. **Only one program combines memory-residency with the power of a full-featured disk manager: WindowDOS Version 2.0.**

Features Not Found In DOS

- ◆ Sort directories in 8 ways
- ◆ Copy, erase, or move groups of files
- ◆ Find any file in seconds
- ◆ Display default directory of any drive with a single keystroke
- ◆ Global copy & erase commands
- ◆ Copy function prompts you to insert another disk if necessary
- ◆ Display hidden files/subdirectories
- ◆ Display file contents in various formats and page forward/backward

- ◆ Display graphic tree
- ◆ Unique RAM Environment function shows name, size, location, and interrupts of every program in memory
- ◆ Rename subdirectories for instant reorganization
- ◆ Hide & unhide subdirectories
- ◆ See & change file attributes
- ◆ Send control codes to printer
- ◆ Switch default printer
- ◆ Password "lock" your system
- ◆ 5-minute screen-blanking function

Enhances These DOS Functions

- ◆ Format disks (faster than DOS)
- ◆ Make or erase subdirectories
- ◆ Copy, rename, or erase files
- ◆ Copy files to printer or COM ports
- ◆ Display disk free space

Other Information

- ◆ Not copy protected
- ◆ Uses only 51K of memory
- ◆ Supports EGA & Hercules
- ◆ Uninstall command
- ◆ For PC/XT/AT/100% Compatibles
- ◆ **Order Today--Only \$49.95**

WindowDOS Associates • Box 300488-B • Arlington, Tx 76010 • 817-467-4103

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!

HEPCAT is here! **HEPCAT** is here! **HEPCAT** is here! So what is **HEPCAT**, you may ask? Why it's just another Pat Swayne **SUPER-UTILITY**. **HEPCAT** is an acronym for **HUG Engineer's and Programmer's Calculation Tool**. Just what we don't need, another memory resident calculator, right? Wrong! With **HEPCAT**, you can throw away the rest and use the best. **HEPCAT** only uses two partial lines on your screen, and best of all, does NOT cause existing programs to stop executing! That means, while your computer is grinding numbers internally, you can be grinding them externally. Order **HUG P/N 885-3045-37**.

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

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
ACCOUNTING SYSTEM	885-8047-37	CPM	BUSINESS	20.00
ACTION GAMES	885-1220-37	CPM	GAME	20.00
ADVENTURE	885-1010	HDOS	GAME	10.00
ASCRTY	885-1238-37	CPM	AMATEUR RADIO	20.00
AUTOFILE (Z80 ONLY)	885-1110	HDOS	DBMS	30.00
BHBASIC SUPPORT PACKAGE	885-1119-37	HDOS	UTILITY	20.00
CASTLE	885-8032-37	HDOS	ENTERTAINMENT	20.00
CHEAPCALC	885-1131-37	HDOS	SPREADSHEET	20.00
CHECKOFF	885-8010	HDOS	CHECKBOOK SOFTWARE	25.00
DEVICE DRIVERS	885-1105	HDOS	UTILITY	20.00
DISK UTILITIES	885-1213-37	CPM	UTILITY	20.00
DUNGEONS & DRAGONS	885-1093-37	HDOS	GAME	20.00
FLOATING POINT PACKAGE	885-1063	HDOS	UTILITY	18.00
GALACTIC WARRIORS	885-8009-37	HDOS	GAME	20.00
GALACTIC WARRIORS	885-8009-37	CPM	GAME	20.00
GAMES 1	885-1029-37	HDOS	GAMES	18.00
HARD SECTOR SUPPORT PACKAGE	885-1121	HDOS	UTILITY	20.00
HDOS PROGRAMMERS HELPER	885-8017	HDOS	UTILITY	16.00
HOME FINANCE	885-1070	HDOS	BUSINESS	18.00
HUG DISK DUPLICATION UTILITIES	885-1217-37	CPM	UTILITY	20.00
HUG SOFTWARE CATALOG	885-4500	VARIOUS	PRODUCTS THRU 1982	9.75
HUGMAN & MOVIE ANIMATION	885-1124	HDOS	ENTERTAINMENT	20.00
INFO. SYSTEM AND TEL. & MAIL SYSTEM	885-1108-37	HDOS	DBMS	30.00
LOGBOOK	885-1107-37	HDOS	AMATEUR RADIO	30.00
MAPLE	885-8005	HDOS	COMMUNICATION	35.00
MAPLE	885-8012-37	CPM	COMMUNICATION	35.00
MICRONET CONNECTION	885-1122-37	HDOS	COMMUNICATION	20.00
MISCELLANEOUS UTILITIES	885-1089-37	HDOS	UTILITY	20.00
MORSE CODE TRANSCIVER	885-8016	HDOS	AMATEUR RADIO	20.00
MORSE CODE TRANSCIVER	885-8031-37	CPM	AMATEUR RADIO	20.00
PAGE EDITOR	885-1079-37	HDOS	UTILITY	25.00
PROGRAMS FOR PRINTERS	885-1082	HDOS	UTILITY	20.00
REMARK VOL 1 ISSUES 1-13	885-4001	N/A	1978 TO DECEMBER 1980	20.00
RUNOFF	885-1025	HDOS	TEXT PROCESSOR	35.00
SCICALC	885-8027	HDOS	UTILITY	20.00
SMALL BUSINESS PACKAGE	885-1071-37	HDOS	BUSINESS	75.00
SMALL-C COMPILER	885-1134	HDOS	LANGUAGE	30.00
SOFT SECTOR SUPPORT PACKAGE	885-1127-37	HDOS	UTILITY	20.00
STUDENT'S STATISTICS PACKAGE	885-8021	HDOS	EDUCATION	20.00
SUBMIT (Z80 ONLY)	885-8006	HDOS	UTILITY	20.00
TERM & HTOC	885-1207-37	CPM	COMMUNICATION & UTILITY	20.00
TINY BASIC COMPILER	885-1132-37	HDOS	LANGUAGE	25.00
TINY PASCAL	885-1086-37	HDOS	LANGUAGE	20.00
UDUMP	885-8004	HDOS	UTILITY	35.00
UTILITIES	885-1212-37	CPM	UTILITY	20.00
UTILITIES BY PS	885-1126	HDOS	UTILITY	20.00
VARIETY PACKAGE	885-1135-37	HDOS	UTILITY & GAMES	20.00
VOLUME I	885-1008	N/A	SOFTWARE LISTINGS	9.00
VOLUME II	885-1013	N/A	SOFTWARE LISTINGS	12.00
VOLUME III	885-1015	N/A	SOFTWARE LISTINGS	9.00
VOLUME IV	885-1037	N/A	SOFTWARE LISTINGS	12.00
WATZMAN ROM SOURCE & DOC	885-1221-37	CPM	H19 FIRMWARE	30.00
WATZMAN ROM	885-4600	N/A	H19 FIRMWARE	45.00
WHEW UTILITIES	885-1120-37	HDOS	UTILITY	20.00
XMET ROBOT X-ASSEMBLER	885-1229-37	CPM	UTILITY	20.00
Z80 ASSEMBLER	885-1078-37	HDOS	UTILITY	25.00
Z80 DEBUGGING TOOL (ALDT)	885-1116	HDOS	UTILITY	20.00

H8 - H/Z-89/90 - H/Z-100 (Not PC)

ADVENTURE	885-1222-37	CPM	GAME	10.00
BASIC-E	885-1215-37	CPM	LANGUAGE	20.00
CASSINO GAMES	885-1227-37	CPM	GAME	20.00
CHEAPCALC	885-1233-37	CPM	SPREADSHEET	20.00
CHECKOFF	885-8011-37	CPM	CHECKBOOK SOFTWARE	25.00
COPYDOS	885-1235-37	CPM	UTILITY	20.00
DISK DUMP & EDIT UTILITY	885-1225-37	CPM	UTILITY	30.00
DOCUMAT & DOCULIST	885-8019-37	CPM	TEXT PROCESSOR	20.00
DUNGEONS & DRAGONS	885-1209-37	CPM	GAMES	20.00
FAST ACTION GAMES	885-1228-37	CPM	GAME	20.00
FAST EDDY & BIG EDDY	885-8018-37	CPM	TEXT PROCESSOR	20.00
FUN DISK I	885-1236-37	CPM	GAMES	20.00
FUN DISK II	885-1248-37	CPM	GAMES	35.00
GAMES DISK	885-1206-37	CPM	GAMES	20.00
GRADE	885-8036-37	CPM	GRADE BOOK	20.00
HRUN	885-1223-37	CPM	HDOS EMULATOR	40.00
HUG 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
REMARK VOL 7 ISSUES 72-83	885-4007	N/A	1986	25.00
RF CAD	885-8020-37	CPM	UTILITY	30.00

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
SEA BATTLE	885-1211-[37]	CPM	GAME	20.00
UTILITIES BY PS	885-1226-[37]	CPM	UTILITY	20.00
UTILITIES	885-1237-[37]	CPM	UTILITY	20.00
X-REFERENCE UTILITIES FOR MBASIC	885-1231-[37]	CPM	UTILITY	20.00
ZTERM	885-3003	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
HUGPBBS	885-5006-37	CPM86	COMMUNICATION	40.00
HUGPBBS SOURCE LISTING*	885-5007-37	CPM86	COMMUNICATION	60.00
ICT 8080 TO 8088 TRANSLATOR & HFM	885-5008-37	CPM86	UTILITY	20.00
KEYMAC	885-3046-37	MSDOS	UTILITY	20.00
KEYMAP	885-3010-37	MSDOS	UTILITY	20.00
KEYMAP	885-5001-37	CPM86	UTILITY	20.00
KEYMAP CPM-85	885-1245-37	CPM	UTILITY	20.00
MAPLE	885-8023-37	CPM	COMMUNICATION	35.00
MATHFLASH	885-8030-37	MSDOS	EDUCATION	20.00
ORBITS	885-8041-37	MSDOS	EDUCATION	25.00
POKER PARTY	885-8042-37	MSDOS	ENTERTAINMENT	20.00
SCICALC	885-8028-37	MSDOS	UTILITY	20.00
SKYVIEWS	885-3015-37	MSDOS	ASTRONOMY UTILITY	20.00
SMALL-C COMPILER	885-3026-37	MSDOS	LANGUAGE	30.00
SPELL5	885-3035-37	MSDOS	SPELLING CHECKER	20.00
SPREADSHEET CONTEST PACKAGE	885-3017-37	MSDOS	VARIOUS SPREADSHEETS	25.00
TERM86 & DSKED	885-5004-37	CPM86	COMMUNICATION & UTILITIES	20.00
TREE-ID	885-3036-37	MSDOS	TREE 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
HEPCAT	885-3045-37	MSDOS	UTILITY	35.00
HUG BACKGROUND PRINT SPOOLER	885-3029-37	MSDOS	UTILITY	20.00
HUG BINDER	885-0004	N/A	REMARK BINDER	5.75
HUG EDITOR	885-3012-37	MSDOS	TEXT PROCESSOR	20.00
HUG MENU SYSTEM	885-3020-37	MSDOS	UTILITY	20.00
HUG SOFTWARE CATALOG UPDATE #1	885-4501	VARIOUS	PROD 1983 THRU 1985	9.75
HUGMCP	885-3033-37	MSDOS	COMMUNICATION	40.00
HUGPBBS SOURCE LISTING	885-3028-37	MSDOS	COMMUNICATION	60.00
HUGPBBS	885-3027-37	MSDOS	COMMUNICATION	40.00
ICT 8080 TO 8088 TRANSLATOR	885-3024-37	MSDOS	UTILITY	20.00
MATT	885-8045-37	MSDOS	MATRIX UTILITY	20.00
MISCELLANEOUS UTILITIES	885-3025-37	MSDOS	UTILITIES	20.00
REMARK VOL 5 ISSUES 48-59	885-4005	N/A	1984	25.00
REMARK VOL 6 ISSUES 60-71	885-4006	N/A	1985	25.00
REMARK VOL 7 ISSUES 72-83	885-4007	N/A	1986	25.00
SCREEN DUMP	885-3043-37	MSDOS	UTILITY	30.00
UTILITIES II	885-3014-37	MSDOS	UTILITY	20.00

PC Compatibles

ACCOUNTING SYSTEM	885-8049-37	MSDOS	BUSINESS	20.00
CARDCAT	885-6006-37	MSDOS	CATALOGING SYSTEM	20.00
CHEAPCALC	885-6004-37	MSDOS	SPREADSHEET	20.00
CP/EMULATOR II & ZEMULATOR	885-6002-37	MSDOS	CPM & Z100 EMULATORS	20.00
DUNGEONS & DRAGONS	885-6007-37	MSDOS	GAME	20.00
EZPLOT	885-6003-37	MSDOS	PRINTER PLOTTING UTILITY	20.00
FAST EDIT	885-8033-37	MSDOS	TEXT PROCESSOR	20.00
GRADE	885-8037-37	MSDOS	GRADE BOOK	20.00
HAM HELP	885-6010-37	MSDOS	AMATEUR RADIO	20.00
KEYMAP	885-6001-37	MSDOS	UTILITY	20.00
LASERWRITER CONNECTION	885-8050-37	MSDOS	PRINTER UTILITY	40.00
RF CAD	885-8038-37	MSDOS	UTILITY	30.00
SCREEN SAVER PLUS	885-6009-37	MSDOS	UTILITIES	20.00
SKYVIEWS	885-6005-37	MSDOS	ASTRONOMY UTILITY	20.00
TCSPELL	885-8044-37	MSDOS	SPELLING CHECKER	20.00

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

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

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

ORDERING INFORMATION

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

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

Real Men Don't Back Up... (Or Do They?)

Jim Buszkiewicz
HUG Managing Editor

I'd been pretty lucky when it came to hardware reliability. I had yet to experience a real catastrophic hard disk failure, or accidentally type "format c:/n"! Maybe my days were numbered, or just maybe the drives I used knew about my low tolerance level for silly failures, and what was in store for them should they fail! Well, apparently, one 20 Meg drive didn't get the message, or just refused to believe it. It decided to fail in such a way that it just couldn't be accessed anymore. My other hard disk, connected to the same controller card, worked fine. So, I was fairly certain that the problem was in the drive itself. Even though this drive contained the ONLY copy of an irreplaceable mailing list, I wasn't too upset, at least not just yet.

First, I had to deal with the rebel drive. There was a principal involved you know, and besides, an example needed to be set for the other drives. A small hole, strategically drilled in the sealed platter cavity, and beach sand blown into the hole while the drive was spinning, should do it!!!

Replacing, prepping, parting, and formatting the new drive was time consuming and boring as usual, so I won't go into details. Next, I went into my video library case, and retrieved a BETA video (VHS would have worked just as well, but I always use the best) tape, inserted it into my tape deck, and typed: "MRESTORE C:", and hit the play button. A few minutes later, my hard disk was restored to the same condition it was on the day it was archived!

It's called the "IMAGER", and it's distributed by The Light-Pen Company. Yes, those same people that brought us that high quality light pen, now have a way for everyone to back up their hard disks, quickly, and more importantly, economically.

Backing up large hard disks to floppies is not only impractical due to the large numbers of floppies needed, but also quite

time consuming. So, LPC figured that just about everyone nowadays owns a VCR (video cassette recorder), and if they don't, now there are TWO good reasons for buying one. If you're like me, you'll have the best of both worlds, VHS for great movie selections, and BETA for high quality video recordings. It doesn't matter which you use with the IMAGER, however, both formats work equally well.

The IMAGER itself consists of a PC style circuit board which uses a single backplane slot. The card easily drops into any Heath/Zenith PC compatible (including the H/Z-200 systems) and is connected to the VCR by way of two standard RCA phono jack (video-in and video-out) cables, which are included with the card. Also included is a 96-page manual which leads you step-by-step from installing the card to operating it.

The best hardware in the world wouldn't be complete without some controlling software, and the IMAGER is no exception. The support software is supplied on a standard 5" disk which comes with the IMAGER. These programs allow you to BACKUP, VERIFY, LIST FILES, and RESTORE data between the computer and VCR. There's even a program included for performing self-test diagnostics on the VCR/Computer system. The software operation is so reliable and simple, that the manual is almost not needed. The IMAGER can be controlled from the main menu-driven program, or a single function can be performed from an MS-DOS command line. For example, from the main program menu, you can press the 'F5' or the 'L' keys to perform the "List Files" operation. Alternatively, you could type "MDIR *.*" from the MS-DOS prompt, and obtain the same results.

During the 'Backing-up' process, the IMAGER apparently saves the data on the tape in blocks (or chunks). Each individual

block is written more than once! The reason for this being that if an error occurs during the 'Verify' or 'Restore' functions, the IMAGER can simply pick up the next block of data on the tape. After a 'Verify' or 'Restore' function is performed, the software displays the tape's figure of merit in the form of two numbers. The first number is an indication of the quality of the tape. 128 indicates a perfect tape, and 0 (zero) corresponds to 10% soft errors and indicates poor tape quality (negative numbers are possible here, indicating you should quit buying your tapes at the local supermarket). The second number indicates the actual number of soft errors encountered for all images found.

The IMAGER transfers data at the rate of 12,000 bytes per second. 10 Megabytes of data can be typically backed up in about 11 minutes. Backups should be made at the fastest possible tape speed, and a standard 2-hour tape can typically hold 110 megabytes. Besides the NTSC standard, the IMAGER also supports SECAM or PAL non-interlaced formats.

Once in a great while, a product comes along that is so practical and inexpensive it makes you wonder why someone hadn't thought of it sooner. The IMAGER from the Light-Pen Company is just such a product. The price of the IMAGER is \$295, however . . . HUG has made another deal for its members!! To HUG members only, the IMAGER can be purchased for \$195!! Yup, 'wheelin-dealin' Jim comes through again. Just tell them you're a HUGGIE, and it's yours for \$100 off the retail price! If you need more information, or would like to order, contact Frank Verdi at The Light-Pen Company, P.O. Box 45255, Los Angeles, CA 90045-0255, or call (800) 634-1967. California residents can call (800) 821-7807.



IF YOU WANT A BETTER COMPUTER BUILD IT YOURSELF.

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

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

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



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

Every INTEL 80386 microprocessor used in our H-386 is 100% tested for all functions. And you get superb graphics because one video port provides dazzling 640-by-480 color on Zenith's new flat



screen monitor and another port drives EGA, CGA and TTL monochrome monitors. Both ports automatically emulate common video formats for easy system configuration.

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

In the bargain, you get the satisfaction of having built a powerful computer system all by yourself, and the confidence that this exciting product will deliver all

the performance and dependability you expect. At a significant savings over comparable off-the-shelf brands.

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

So if you want a better computer, build it yourself. Impress your friends.

And save money at the same time.



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

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

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

Heathkit®

Heath
Company

Heathkit is a registered trademark of Heath Company, a subsidiary of Zenith Electronics Corporation.

©1987, Heath Company.

Continued from Page 44

Buggin' HUG. Instead of routing the INT LED directly to P1, run it through a toggle switch which you will want mounted where you can reach it. I have a RESET button, the speed indicator LED and my toggle switch in a box which is mounted on the front of my computer. The other thing you must do is leave R3 in place. Thus, the toggle switch will either allow the INT LED signal through, or it will block it and a default due to R3 will prevail. When the toggle switch is closed (on), the INT LED will control the speed. When the switch is open (off), the speed will be forced to slow no matter what the INT LED does.

We will then be left with a system which is software controllable, but also gives us manual override. This is truly wonderful!

There is one more thing we need to make the whole system complete (and quite professional). We need software control. I have written three programs to round out this whole speed-up package.

- A. SPEEDUP1.COM
- B. FAST.COM
- C. SLOW.COM

SPEEDUP1.COM implements three things:

1. Alt-MINUS and Alt-PLUS (on the keypad only) will select slow and fast speeds, respectively.
2. Alt-Prtsc will send a FormFeed to the printer (it will never hang though if the printer is not available). This particular method even works properly with the print spoolers I have tried. I added this in order to keep from having to load SPEEDUP1 and FORMFEED (a public domain program) to both intercept the keyboard interrupt.
3. Drive A and B diskette I/O is intercepted and all WRITES, VERIFYs, and FORMATs are done in slow speed. Whatever speed had last been chosen (by Alt-MINUS or Alt-PLUS) is restored immediately when the operation has completed. Note that Floppy READs and all drives C and up are not affected. The reason for this particular item is that DISK FORMATs and WRITES would never work on my system. The FORMAT or any COPY to a diskette would fail and a message would be printed telling me of the failure. With this program, I no longer have to worry.

Note that if you write a FAST or SLOW program, such as suggested in the April Buggin' HUG, then option (3.) will not know which speed has been recently selected (the LEDs cannot be read). Thus (3.) will restore what it remembers was last selected by (1.). This is why this program is all in one; the last selected speed is easily shared with the diskette I/O speed change software.

There are times when a FAST or SLOW program is definitely needed. Specifically, when you want to slow a program down and ensure that (3.) does not speed it up again. One such case is FASTBACK(tm), others include games which use the joystick (very timing dependent). Such programs may use a diskette and thereby have (3.) restore things. One option is to throw the toggle switch, the other is to use my SLOW.COM and FAST.COM programs which actually call the SPEEDUP1 resident program via INT 13h (AH=87) to order a speed change. A parameter of DL=0 orders the slow speed, while DL=1 orders the high speed.

Now we have quite a professional package, try to find these features elsewhere for under \$50.00:

- a. A speed-up modification, which has been upgraded to be controlled by software with a toggle switch override.
- b. An LED to indicate the actual current speed status.
- c. A reset button for those times when your new program goes bonkers.
- d. A resident program which will allow speed selection from the keyboard (Alt-MINUS and Alt-PLUS).
- e. A resident program which slows the system down automatically for disk WRITES and FORMATs and switches back to the previous setting when the diskette I/O is completed.
- f. A set of transient programs which (usually in batch files) can switch the speed to FAST or SLOW in coordination with the resident program.

All this can be yours! So get soldering!

Sincerely,

James R. Reinders
7551 Haley Road
Milford, MI 48042

Continued on Page 56

TALK IS CHEAP.

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

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



If you have a modem, the HV-2000

Computer Voice will allow your computer to recite reference and research information from time-sharing services. Or, speak radio transmitted ASCII information.

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

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

Heathkit®


Heath
Company

Benton Harbor, MI 49022

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

© 1987, Heath Company

Making "Monkey See, Monkey Do" Livable Or How To Fix The H/Z-386 'Super' Keyboard



Jim Buszkiewicz
HUG Managing Editor

When I was about 4 years old (can you picture that!), I was introduced to a boy who was about a year older than me. I didn't realize it then, but 'Pete' was to become my life-long friend. Pete and I did everything together. If Pete did something 'stupid', so would I. I, of course, would always get into trouble for it though. My mom would say, "Jim, why do you have to be so stupid? If Peter jumps off a bridge, are you going to follow him?" I guess Zenith never had a mom like that, because he (Zenith), is still following Pete's (IBM's) silliness. I'm specifically referring to key layout on the new ZKB-2 (H/Z-386) keyboard.

Among the many gripes I have with this keyboard, my two biggest include the placement of the 'ESC' and 'CTRL' keys. Because of the symmetrical nature of the keyboard, the designer was apparently a mathematician, and not a computer operator. Unfortunately, only the worst problem is easily repaired; the placement of the 'CTRL' key. Normally, you would expect to find it immediately to the left of the letter 'A', but for some reason, it gave way to the seldom used 'CAPS LOCK' key.

To make the switch-a-roo between the left 'CTRL' key and lighted 'CAPS LOCK' key,

you'll need the following items: Phillips Screwdriver, 1/4" Nutdriver, Needlenose Pliers, Sharp X-Acto Knife, Wire Cutters, Wire Strippers, Solder, Soldering Iron (low wattage), Solder Sucking Device, and about 2 feet of #30 gauge wire wrap wire. The mod is quite easy, and can be made by even the most fumble of fingers. What these steps are trying to accomplish is to simply interchange the physical locations of the two switches. Here's what to do:

1. Place the keyboard on your work surface so the keys are facing downward, and remove the two pop-up legs at the rear of the unit. These are removed by squeezing the two ends together, which allows the legs to snap out of their holder.
2. Remove the six phillips head screws from the bottom keyboard cover plate, and remove the plate.
3. Remove the six hex head screws which holds the keyboard assembly to the top cover, and remove the keyboard.
4. Turn the keyboard over so the keys are right side up, and remove the left hand 'CTRL' and 'CAPS LOCK' keycaps. These caps are press fit on the switches and can be easily lifted (pryed) up. The metal stabilizing arms should remain on the keyboard and snapped loose from the keycap as it is lifted off.
5. Turn the keyboard over to expose the foil side once again, and de-solder the 'CTRL' key contacts. A vacuum type solder sucker is recommended for this procedure to remove all the solder from the hole to free up the actual switch contact.
6. Next, de-solder the 'CAPS LOCK' key contacts. REMEMBER, since there is a light in this switch, there are four (4) contacts, not two. Two of these contacts are large, two are smaller (see the illustration).
7. Flip over the keyboard again, and remove the two unsoldered switches. These switches are snapped into the metal plate, and are held in place on the right and left sides of each switch. The switch is released by squeezing the two snaps together, or by pushing in (to release the snap), and prying up on each side individually.
8. Using a sharp X-Acto knife, cut a break in each foil going to the two largest foil

pads on each of the two switches (see illustration). You should make four cuts in all. **DO NOT** cut the foils going to the two smaller foil pads on the original 'CAPS LOCK' key locations.

9. Identify the original 'CTRL' switch. This is the switch with only two (2) contacts. Insert this switch into the location where the original 'CAPS LOCK' keyswitch was. This should be immediately to the left of the letter 'A' key. Observe switch orientation. The two switch contacts should go into the two larger contact pad holes. Snap the switch into place, and solder the contacts to the foil pads.
10. Cut two six inch (6") lengths of wire-wrap wire, and remove about 1/16" of insulation from one end, and 1/8" of insulation from the other end of each piece. Solder the shorter bare end of each wire to the LED (light emitting diode) contacts on the 'CAPS LOCK'

keyswitch. These contacts are the two smaller pieces of wire coming from the bottom of the switch. When you solder these wires, try to place them as close as possible to the very bottom of the switch, and be as quick as possible with the heat. Once soldered, cut off the contacts as close as possible to the solder joint so the switch can rest flush against the circuit board when it's re-inserted.

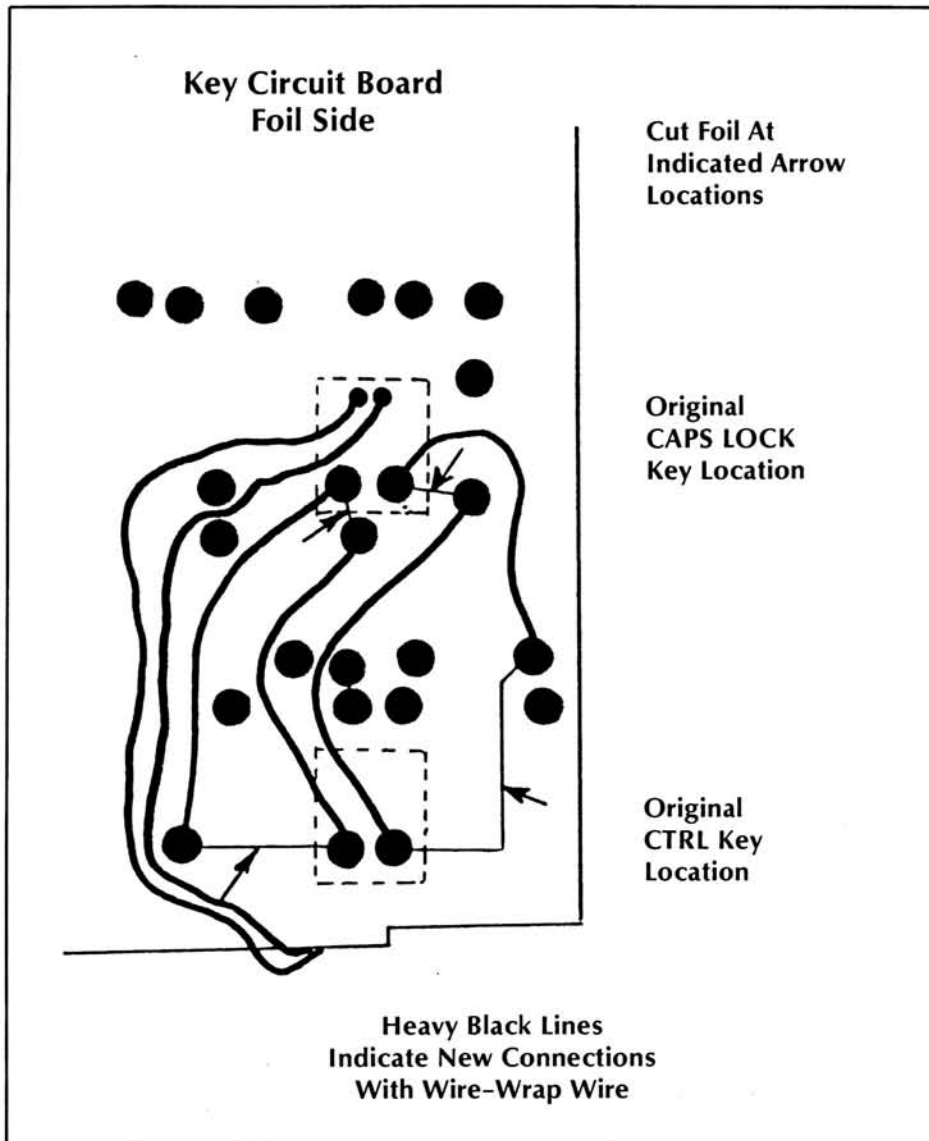
11. Route the two wires coming from the 'CAPS LOCK' switch through the top of the metal switch retaining plate, but between that plate and the main circuit board. This should be done at the original 'CTRL' key location; just below the left 'SHIFT' key. Try to keep track of which wire goes to which LED contact on that switch. The reason being, the LED is polarity sensitive, and the other ends of these wires have to get soldered back into the original LED solder pad holes. Don't worry if you

lose track, the LED just won't work, and you'll have to swap the wires. Nothing will get damaged.

12. Snap the lighted 'CAPS LOCK' switch into its new location (just below the left 'SHIFT' key), carefully guiding the two LED wires out the side of the keyboard as you do so. The two large contacts should fall into place, and protrude through the two foil pads on the foil side of the circuit board.
13. Solder the two large contacts to the foil pads.
14. Route the wires over to the two original smaller LED foil pads (where the new 'CTRL' key switch now resides), and, if you kept track of the polarity, solder them into place on the foil side of the circuit board. If you didn't pay attention to polarity, solder the two wires in place any way. If the 'CAPS LOCK' light doesn't work, just swap the two wires later on.

15. At this point, there should be four (4) foils (two from each original switch) which aren't connected to anything. Using the remaining wire, connect the two foils, which originally went to the 'CAPS LOCK' key switch, over to the switch contact foil pads on the new 'CAPS LOCK' key switch location.
16. Likewise, connect the two foils, which originally went to the 'CTRL' key switch, over to the switch contact foil pads on the new 'CTRL' key switch location.
17. Before re-assembling the keyboard, power up the computer and verify that both new keys, as well as the light on the 'CAPS LOCK' key are all working correctly.
18. Replace the two keycaps to their new locations, and re-assemble the keyboard housing.

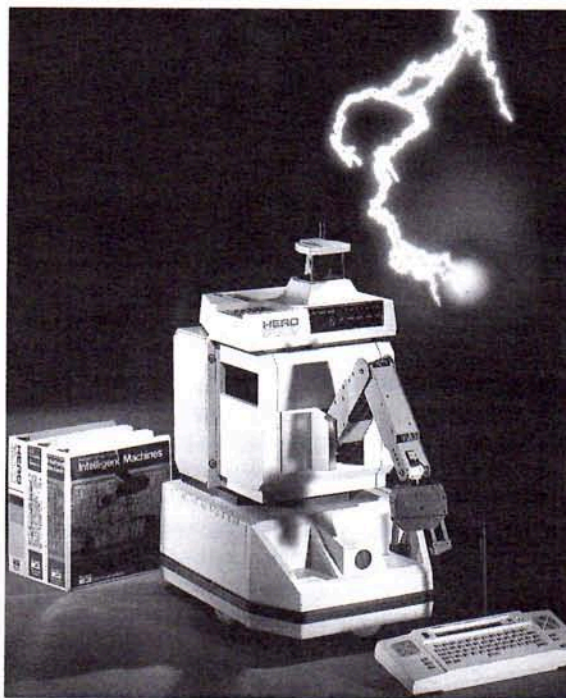
You'll find that the new 'CTRL' key cap was contoured for the lowest row of keys, and actually sticks up slightly from the keys in the third row. I've actually found this to be somewhat helpful. You can feel this key sticking up slightly with your left 'pinky', and doing so, helps align your fingers on the keys! *



MAKE LEARNING AN ADVENTURE

Heathkit/Zenith Educational Systems

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



HERO® 2000 Teaching Robot

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

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



The MACRO-86 Assembly Language Programming Course

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

The MACRO-86 Algorithms Course

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

AutoCAD computer-aided design software

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



The Computer Servicing Series

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

To order: Call TOLL-FREE
1-800-253-0570

Alaska and Michigan residents
call: 616-982-3411.

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



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

An

Introduction

to

TSR's

Pat Swayne
HUG Software Engineer

Note: This article is intended for readers with at least some knowledge of assembly language and the MS-DOS environment.

A TSR (or a TSR program) is a special kind of program that runs under MS-DOS. Normally, when you run an MS-DOS program, MS-DOS loads the program into memory and turns control over to it. When the program has completed its task, it returns control to MS-DOS, which returns the memory used by the program to the "pool" of memory available to programs. A TSR program uses a special code when it exits that tells MS-DOS not to free up some or all of the memory that it used. TSR stands for "Terminate but Stay Resident" (or "Terminate and Stay Resident"), which is the name of one of the special exit codes. The other special exit code that can be used, which is available in MS-DOS versions 2 and above only, is called "Keep Process", so I guess TSR programs could also be called KP programs.

Uses of TSR's

TSR's can be used to do three different jobs in a computer. One thing they can do is to extend the capabilities of the operating system or a program. If you are using a keyboard macro processor (such as KEYMAC) to make it easier to enter the commands for a particular program, then

you are using a TSR to extend the capabilities of that program. Another use for TSR's is to provide instant access to utility programs, and to allow you to use the utilities without quitting and then restarting another program. Sidekick and Perks and other "pop-up" utilities are TSR's of this type. The third use for TSR's is to allow the computer to do two jobs at once, which is called multi-tasking. This type of TSR can be further divided into two categories--non-interactive and interactive. Print spoolers and screen clocks fall into the non-interactive category. They do their job in the background without interaction from the user while he/she works with another program. HEPCAT is an example of an interactive multi-tasking TSR. Unlike other pop-up utilities, HEPCAT allows the currently running program to continue non-interactive processing while it is popped up.

Multi-tasking, Multi-user

Since we have brought up the term "multi-tasking", perhaps we should point out the difference between multi-tasking and multi-user, another term you may have heard. When you have two jobs being done at the same time on a computer, you have multi-tasking. When you have two users (two people) working at the same time with the same program, presumably with two different terminals con-

nected to the same computer, that is an example of a multi-user operation. If you have two users using two different programs at the same time on one computer, then you have a multi-tasking multi-user operation.

Although a TSR could be written that would support a second user for some jobs, multi-user operations will not be discussed in this article.

Keeping a Program Resident

The Terminate but Stay Resident exit code is a software interrupt (interrupt 27H) designed for use within .COM programs, similar to the normal exit interrupt, 20H. To use it, you must point the DX register to the last byte of code that is to stay resident plus one (or to the first non-resident byte) and then execute the interrupt, as in this example:

```
MOV  DX,OFFSET LAST_BYTE+1
      ;POINT TO LAST BYTE + 1
INT  27H
      ;EXIT AND STAY RESIDENT
```

The CS register must be pointed to the Program Segment Prefix, or PSP in such a way that CS:0 is the start of the PSP. This is the normal condition while a .COM file is running, but it would take some fancy register manipulation to achieve this condition. The Keep Process code is provided

in DOS version 2 of higher, which is function 31H of the system interrupt, 21H. To use this function, the DX register must be loaded with the amount of memory you want to keep resident in paragraphs. In a .COM program, you could do it like this:

```
MOV  DX,OFFSET LAST_BYTE+15 ;GET SIZE + 15
MOV  CL,4
SHR  DX,CL                  ;DIVIDE BY 16
MOV  AH,31H
INT  21H                    ;EXIT AND STAY RESIDENT
```

You may have noticed that this code is a little different from the example in the Programmer's Utility Pack manual. Their method will result in 16 bytes too much being kept resident if the size is an even multiple of 16 bytes, whereas this method will not. In an .EXE program, you have to add the size of all segments or portions of segments that you want to keep resident and convert that result to paragraphs. As the PUP manual suggests, don't forget to add in the space used by the Program Segment Prefix.

Accessing a TSR

Once a program executes one of the special exit codes that leaves it in memory, it releases control back to MS-DOS exactly as a normal program does. In order for the program to have some control over the system, it must install some kind of "hook" that will allow it to hang on to some control after it exits. The hooks are required to allow access to the program after it exits (for example, to activate a pop-up utility), and/or to allow the program to do some work while other work is going on. Fortunately, hooks are provided by the computer system in the form of interrupts that occur no matter what the computer is doing. A TSR program can vector one or more of these interrupts into itself, perform a little of its task each time the interrupt occurs, and vector the interrupt on to its intended destination when it is done with it.

When a program does its work by capturing interrupts and working a little each time they occur, it steals some of the processor's time that would normally be used for something else. That is why this way of making the computer appear to do two jobs at once is sometimes called "time slicing". In order to not hinder the main job of the computer too much, a multi-tasking TSR should only use time in small bursts, and it should try to avoid using

computer time when the computer is doing something that is time critical, such as reading a disk.

Among the interrupts commonly used by TSR's are the spooler interrupt, the timer interrupt, the keyboard interrupt, and the system (DOS) interrupt.

The Spooler Interrupt

The spooler interrupt (INT 28H) is provided by MS-DOS to support the PRINT utility. It is executed at the end of each DOS system call, and also continuously while the system is waiting for input during the execution of DOS function 10H. It is important to note, however, that it is NOT generated continuously if function 3FH is used for input (using the STD INPUT device).

The spooler interrupt is intended to provide a safe time in which MS-DOS functions such as disk reads can be done. The PRINT utility spools files to the printer while the computer is doing other jobs, and it reads the files from the disk as it goes. The reason why a safe time must be provided in which to perform MS-DOS functions is because MS-DOS is not re-entrant. In other words, while MS-DOS is in the process of executing a system function, another function request cannot be made.

Here is the reason why MS-DOS is not re-entrant. When a system call is executed, MS-DOS sets up an internal stack so that there will be no danger of overflowing the user's stack. If a TSR program executes a system call while another system call is in progress, the internal stack will be re-set, most likely wiping out the return addresses of subroutines being used by the first system call. This results in a system crash.

The spooler interrupt is generated at a time when the normal stack is active, making it a safe time in which a TSR can call MS-DOS. However, since the spooler interrupt is generated only at the end of system calls and during function 10H, a multi-tasking TSR driven by it will cease operation while a program that makes no system calls has control, or while a program is waiting for input using function 3FH.

The Timer Interrupt

The timer interrupt is generated by a timer chip in the computer. On both Z-100 (not PC) and PC-compatible computers, the BIOS processes the timer interrupt and generates a software interrupt during the processing. So there are actually two timer interrupts: a hardware timer interrupt, and a software timer interrupt. On a PC-compatible, the hardware timer interrupt is INT 8H and the software interrupt is INT 1CH, and the period is approximately 55 ms (18.2 ticks/second). On a Z-100, the hardware interrupt is INT 42H and the software interrupt is INT 51H, and the period is 10 ms (100 ticks/second). It is usually recommended that you use the software interrupt in an application, but in a TSR it might be better to use the hardware interrupt, as we shall see later. On a PC-compatible, it is not uncommon for an application program, especially a game, to re-program the timer chip to generate the hardware interrupt at a different rate (18.2 ticks/second is difficult to work with for timing things). These programs will usually take over generation of the software interrupt and try to maintain it at the normal rate. Because the timer interrupt is subject to being changed, you may want to avoid using it to time things within your TSR.

On a Z-100, the BIOS provides information during the software timer interrupt that you may find useful. The AX register contains the number of timer ticks that have occurred since the previous interrupt. This number will usually be one, but if interrupts have been disabled for a while, the count will indicate the number of ticks that occurred while interrupts were disabled. This makes it possible to use the software timer interrupt for accurate timing, by simply adding the AX register to your time counter at each interrupt (rather than simply incrementing the counter).

The Keyboard Interrupt

On a PC-compatible, the keyboard hardware generates an interrupt (INT 9H) that can be intercepted to monitor keyboard activity. Pop-up utilities usually use this interrupt to determine when their "hot key" has been activated. You can determine which key has been pressed by inputting from port 60H. The value you read will be the "scan code" of the key, not the ASCII value. You can absorb the key so that other processes using the keyboard interrupt will not see it by executing this code:

```

IN    AL,61H      ;READ STATUS
OR    AL,80H      ;SET ACKNOWLEDGE BIT
OUT   61H,AL      ;ACKNOWLEDGE KEY
AND   AL,7FH      ;RESTORE ORIGINAL STATUS
OUT   61H,AL      ;AND SET IT

```

Once you do this, you can return from the interrupt (IRET) rather than passing it on to the next process, but you must reset the interrupt controller before you return. To reset the interrupt controller, write a 20H to port 20H.

On a Z-100 there is a hardware key interrupt (INT 46H) and a BIOS generated software key interrupt (INT 50H). The hardware interrupt is shared with the video retrace circuitry and the light pen circuitry, and it is difficult to work with. The software key interrupt is much easier to work with. There is no need to read the key from a port, because it is provided in the AL register, and the AH register can be used as a status flag to let other processes know that you have absorbed the key. The status flag also lets you know if a process that had the interrupt before you absorbed the key. A zero in AH signifies that the key has not been used, and a non-zero indicates that it has been used.

The Keyboard BIOS Interrupt

On PC-compatible computers only (not Z-100's), the BIOS functions are accessed via software interrupts. The keyboard function interrupt is INT 16H, and you can capture this interrupt to perform such functions as key mapping. The only drawback is that programs that use the hardware key interrupt to read the keyboard will bypass your mapping routine.

The System Interrupt

The system interrupt (INT 21H) can be captured by TSR's to accomplish a number of functions. One thing you can do is to see what system calls are being made. You may want your TSR to cease functioning during certain system calls. Another reason for capturing the system interrupt is to allow your TSR to process some of the functions instead of letting DOS process them. Some TSR's process console input, so that they can be popped up during input. A third reason for vectoring the system interrupt is to create additional functions. You can use the system interrupt to control your TSR this way. The TSR can monitor all functions and pass the ones belonging to DOS on, and process the others itself.

Using Interrupts

The general procedure for using an interrupt in a program, whether it is a TSR or not, is to read the vector currently assigned to that interrupt, store it within your program, and install a new vector pointing to the code in your program that will process the interrupt. Your processing code should execute a far jump to the vector previously assigned to the interrupt when it is done with its processing, except in special cases such as the keyboard interrupt mentioned previously.

To read and set interrupt vectors, you can use DOS functions 35H and 25H, or you can alter the data in memory directly. It is possible for a TSR to capture the DOS interrupt and watch for functions 25H and 35H to see what other programs, including future TSR's, do with interrupts. If you alter interrupt vectors by altering memory directly in your TSR, you will avoid this kind of scrutiny by other TSR's.

If you use the timer interrupt or the spooler interrupt, it is a good idea to limit the amount of work your TSR does during each interrupt. If there is a lot of work to do, divide it into pieces and do a little during each interrupt.

Avoiding Conflicts

As we have pointed out, it is most important that your TSR does not make any DOS calls while other DOS calls are in progress. It is also a good idea to avoid using slices of processor time during disk accesses, because it can slow them down. Two ways to avoid such conflicts are to use the spooler interrupt to drive your TSR, or to capture the system interrupt. However, there are drawbacks with both these methods. As we have already pointed out, the spooler interrupt is not always available. With MS-DOS version 2 and above, the use of file handles has made monitoring system calls to watch for disk accesses difficult.

One trick you can use in your TSR to make sure that there is no conflict with DOS is to use both the spooler interrupt and the timer interrupt to drive your program. During the timer interrupt, you can

see if a DOS call is in progress by checking the position of the stack. In my programs, I use codes like this:

```

MOV   AX,SS      ;GET STACK SEGMENT
MOV   BX,CS      ;GET CODE SEGMENT
CMP   AX,BX      ;COMPARE THEM
JB    IN_DOS     ;STACK LOWER, WE'RE IN THE DOS

```

If your program is to be run under MS-DOS 3.2 or higher, you must use the hardware timer interrupt in order for this to work. The reason is that MS-DOS 3.2 uses "stack frames" for all hardware interrupts. Stack frames work like this: MS-DOS vectors the interrupt into the RAM portion of the BIOS. Here, the current stack at the time of the interrupt is saved, and a stack unique to each interrupt is set up. Then the normal code that processes the interrupt is executed using a code like this:

```

PUSHF           ;PREPARE FOR IRET
CALL  DWORD PTR [ADDRESS] ;EXECUTE THE ROUTINE

```

The normal interrupt code for the timer interrupt will therefore issue the software timer interrupt with the stack in the hardware interrupt's frame, and my test will not work. Using the hardware interrupt in your TSR avoids this, because the TSR gets the interrupt before DOS does.

If you use this test, your program should monitor system calls and watch for function 3FH, and then check for standard device handle numbers in the BX register. It should skip the test if those conditions are met, so that the TSR will not halt when a program uses standard devices. If you want to study what I have discussed above in more detail, examine the file SCRNCCLK.ZSM (Z-100 screen clock source) on the HEPCAT disk (885-3045-37).

Removing a TSR

Due to the large number of TSR programs available now, it is sometimes not possible to load all of the ones you want to use into memory at once. This can be because you do not have enough memory, or because some of them conflict with each other. It is therefore often a good idea to make your TSR removable. Under MS-DOS version 2 or above, you can remove a TSR from memory using system function 49H (Free Allocated Memory) and following this procedure. 1) Restore all captured interrupts by ensuring that the vectors for them point to the same lo-

cations they did before your TSR started. 2) Point the ES register to the first segment used by the program, and execute function 49H. 3) Get the address of the environment segment. It is found in the PSP, at address 2CH. In a .COM file, the CS register points to the PSP, and in an .EXE file, the ES register points there at the start of the program. There is also a function under MS-DOS version 3.1 and above (function 62H) that can retrieve the address of the PSP. 4) Point the ES register to the environment segment, and execute function 49H. 5) Use INT 20H or function 4CH to exit.

A Sample TSR

The assembly listing accompanying this article is a sample TSR program for PC-compatible computers that provides a "concurrent" beep. (Sorry about that, Z-100 guys, but at least your computer already has a concurrent beep.) On a PC-compatible, when a command is sent to the BIOS to make a beep via function 10H (the screen function), the BIOS keeps control during the time it takes to make a beep rather than just starting a beep and then returning control to the caller. To illustrate this, enter the following at the DOS prompt:

```
ECHO ^G^G^G^G^G^G^G^G^G^G
```

The ^G means Control-G, and you can just hold down the Ctrl key and the G key until it repeats a few times. When you hit Return, you will hear a beep for each Control-G, and the DOS prompt will not return until all beeps have occurred. Do the same thing on a Z-100 (not PC) if you have access to one, and notice the difference. Now assemble the listing into CBEEP.COM, run it (nothing will appear to happen), and enter the ECHO test line again. Now you will hear just one beep, and the DOS prompt will re-appear immediately, even before the one beep is finished. The CBEEP source listing is well commented, and if you study it, you should be able to see what it is doing.

Listing

```

PAGE          ,132
:             CBEEP -- CONCURRENT BEEP ROUTINE.
:             THIS PROGRAM IS A SAMPLE TSR FOR PC-COMPATIBLES
:             THAT PROVIDES A CONCURRENT BEEP.
:
:             BY P. SWAYNE, HUG SOFTWARE ENGINEER 18-SEP-87

CODE          SEGMENT
ASSUME       CS:CODE,DS:CODE,ES:CODE,SS:CODE
ORG         100H

START:       JMP      SETUP          ;INSTALL PROGRAM IN MEMORY

:           STORAGE AREA -- VECTORS AND FLAGS

INT10V      DW      0,0             ;INT 10H VECTOR
INT1CV      DW      0,0             ;INT 1CH VECTOR
BEEPFLG     DB      0              ;BEEP PROGRESS FLAG
BEEPCNT     DB      0              ;BEEP COUNTER

:           SCREEN INTERRUPT (INT 10H) PROCESSOR
:           DURING THIS INTERRUPT, WE CHECK FOR
:           THE COMMAND TO BEEP, AND JUST SET A FLAG
:           WHEN A BEEP IS REQUESTED.

SCREEN:      CMP      AX,0E07H      ;TIME TO BEEP?
:             JNZ      SCRNX         ;NO
:             MOV      CS:BEEPFLG,1 ;ELSE, FLAG IT
:             IRET                    ;SKIP NEXT PROCESS
SCRNX:      JMP      CS:DWORD PTR INT10V ;GO TO NEXT PROCESS

:           TIMER INTERRUPT (INT 1CH) PROCESSOR
:           DURING THIS INTERRUPT, WE MAKE THE BEEP
:           IF IT IS NEEDED.

TIMER:      PUSH     AX             ;SAVE AX
:             PUSH     DS           ;AND DS
:             MOV      AX,CS
:             MOV      DS,AX        ;PUT DS HERE
:             MOV      AL,BEEPFLG   ;GET BEEP FLAG
:             CMP      AL,1         ;TURN BEEP ON?
:             JZ       BEEPON       ;YES
:             CMP      AL,2         ;TIME BEEP?
:             JZ       BEEPTIM      ;YES
:             JMP      TIMEX        ;ELSE, EXIT
BEEPON:     INC      AL
:             MOV      BEEPFLG,AL   ;MARK BEEP ON
:             CMP      BEEPCNT,0    ;BEEP ALREADY ON?
:             JNZ      CONBEEP      ;IF SO, CONTINUE BEEP
:             IN      AL,61H
:             OR      AL,3
:             OUT     61H,AL        ;TURN ON SPEAKER
:             MOV      AL,0B6H
:             OUT     43H,AL        ;CHANNEL 2, MODE 3
:             MOV      AL,0C5H
:             OUT     42H,AL        ;LSB FOR 600 HZ
:             MOV      AL,7
:             OUT     42H,AL        ;MSB FOR 600 HZ
CONBEEP:    MOV      BEEPCNT,4     ;SET BEEP COUNTER
:             JMP      TIMEX        ;DONE, EXIT
BEEPTIM:    DEC      BEEPCNT       ;DECREMENT BEEP COUNTER
:             JNZ      TIMEX        ;NOT DONE BEEPING, EXIT
:             MOV      BEEPFLG,0    ;ELSE, FLAG BEEP OFF
:             IN      AL,61H
:             AND     AL,0FCH
:             OUT     61H,AL        ;TURN SPEAKER OFF
TIMEX:     POP      DS             ;RESTORE REGISTERS
:             POP      AX
:             JMP      CS:DWORD PTR INT1CV ;ELSE, PASS INTERRUPT ALONG

:           SET UP VECTORS AND EXIT WITH SOME OF THE
:           PROGRAM RESIDENT

SETUP:     MOV      AX,3510H       ;FUNCTION 35H, INT 10H
:             INT     21H           ;GET INT 10H VECTOR
:             MOV      INT10V,BX    ;SAVE IT

```

```

MOV     INT10V+2,ES
MOV     AX,351CH
INT     21H           ;GET INT 1CH VECTOR
MOV     INT1CV,BX    ;SAVE IT
MOV     INT1CV+2,ES
MOV     AX,2510H     ;FUNCTION 25H, INT 10H
MOV     DX,OFFSET SCREEN ;POINT TO OUR PROCESS
INT     21H           ;INSTALL OUR VECTOR
MOV     AX,251CH     ;DO INT 1CH
MOV     DX,OFFSET TIMER ;OUR TIMER PROCESS
CLI     ;KILL INTERRUPTS
INT     21H           ;INSTALL NEW TIMER VECTOR
STI     ;RESTORE INTERRUPTS
MOV     DX,OFFSET SETUP ;SAVE ALL UP TO SETUP
INT     27H           ;EXIT AND STAY RESIDENT

```

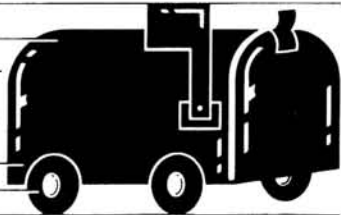
```

CODE    ENDS
END      START

```



MOVING?



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

VERSION UPDATE

HOME FINANCE SYSTEM VERSION 3

—HFS-III keeps track of multiple checking accounts, asset accounts, (cash, savings, IRAs, CDs) and regular bill payments. Prints checks, including payee address, on any business-sized check.

—Define up to 100 expense codes. Separate tax flag.

—Fast. Written in 100% assembly language.

—Pop-up menus and help windows at a keystroke.

—Frequently used transactions stored for easy use.

—Accepts HFS-II data.

Hardware: HZ-100, all Heath/Zenith PCs or any other PC/XT/AT compatible computer (256K), 2 disk drives. Any printer.

Software: MS-DOS 2 or higher.

Price: \$99.00 (includes shipping).

MasterCard/Visa accepted, please include phone number.

Available in December



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



EMULATE — Break the format barrier!

Read and write these disk formats:

Actrix	DEC Rainbow	Magnolia	Royal/Triumph
Altos	Eagle II	Morrow MD	Sanyo 1100
AMPRO	Epson QX-10	NCR DecMate5	Superbrain
Beehive Trp	Fujitsu CP/M86	NEC PC-8001A	Televideo
CDR Systems	IBM CP/M86	Osborne 1	TRS80 CP/M
Cromemco	IMS 5000	Otrona	Xerox 820
DEC VT180	Kaypro II	PMC MicroMate	Zorba

H37 version includes formatting capability.

H89 or H8 with H37 \$59

For CDR BIOS 2.91 \$49

CPC CP/M ↔ PCDOS

Transfer files between PCDOS and CP/M disks. Includes access to subdirectories. For H89 or H8 with soft sector controller.

CPC — (H37, CDR or Magnolia) \$35

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



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

Diskettes — Box of Ten:

5" DSDD hard sector \$10

5" DSDD soft sector \$10

8" SSDD (limited quantity) \$ 5



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 (HDOS or CP/M) — 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

Check or Money Order — Visa, M/C — C.O.D.

Add \$4 per order for shipping and handling

California residents add 6% tax

Call or write for catalog, Monday through Friday, 9-5

ANAPRO

805/239-1273

205 Lake Nacimiento Drive
Paso Robles, CA 93446

FAST

```

title FAST
;
FAST.ASM -> FAST.COM
;
April 16, 1987
by James R. Reinders
;
Assumes SPEEDUP1.COM is installed
See SPEEDUP1.ASM for comments.
;
code segment
assume cs:code, ds:code
;
cr equ 13
lf equ 10
bell equ 7
dosexit equ 20h
dos equ 21h
romtime equ 1Ah
outchr equ 02h
pstring equ 09h
getdate equ 2Ah
setdate equ 2Bh
gettime equ 2Ch
settime equ 2Dh
buff equ 80h
;
org 0100h
;
main proc far
mov dl,1
mov ah,87
int 13h
int dosexit
;
main endp
code ends
end main

```

SLOW

```

title SLOW
;
SLOW.ASM -> SLOW.COM
;
April 16, 1987
by James R. Reinders
;
Assumes SPEEDUP1.COM is installed
See SPEEDUP1.ASM for comments.
;
code segment
assume cs:code, ds:code
;
cr equ 13
lf equ 10
bell equ 7
dosexit equ 20h
dos equ 21h
romtime equ 1Ah
outchr equ 02h
pstring equ 09h
getdate equ 2Ah
setdate equ 2Bh
gettime equ 2Ch
settime equ 2Dh
buff equ 80h
;
org 0100h
;
main proc far
mov dl,0
mov ah,87
int 13h
int dosexit
;
main endp
code ends
end main

```

SPEEDUP1

```

title SPEEDUP1
;
SPEEDUP1.ASM -> SPEEDUP1.COM
;
Speed-up; version 1; by James R. Reinders; April 16, 1987
;
This is a resident routine, run it once and you're all set
until you boot up again (put SPEEDUP1 in your AUTOEXEC.BAT).
;
This program is for use with a Z150/H150 computer which has
been modified by the addition of a software controllable speed-
up circuit. Such a speed-up circuit is based on the following:
;
(1) The article: D. Bencivengo, "H-150 Speed-up Modification",
REMark, vol. 7, no. 6, pp 55-59, June 1986.
(2) The article: D. Bencivengo, "H-150 Speed-up Modification
Update", REMark, vol. 8, no. 4, pp 51-53, April 1987.
(3) The letter: R. J. Maskasky, "Buggin' Hug", REMark, vol. 8,
no. 4, pp 9-10,83, April 1987.
;
Basically, the circuit required is one which uses the circuit
from (1) and redoes a few things in order to control the speed
via the INT LED on the CPU card. This program assumes that
the HIGH speed is selected when the INT LED is turned off and
the LOW speed is selected when the INT LED is turned on. Unless
these hold true, most of what you have here is a program which
turns the INT LED on and off under keyboard control.
;
There is one further modification which I would suggest, in
addition to those offered in (3). Instead of routing the
INT LED directly to P1, run it through a toggle switch which
you will want mounted where you can reach it. I have a RESET
button, the speed indicator LED and my toggle switch in a
box which is mounted on the front of my computer. The other
thing you must do is leave R3 in place. Thus, the toggle
switch will either allow the INT LED signal through, or it
will block it and a default due to R3 will prevail. When
the toggle switch is closed (on), the INT LED will control
the speed. When the switch is open (off), the speed will be
forced to slow no matter what the INT LED does.
;
We will then be left with a system which is software
controllable but also gives us manual override. This should
be perfect!
;
---
;
This implements three things:
;
(i) Alt-MINUS and Alt-PLUS (on the keypad only) will select
slow and fast speeds respectively.
;
(ii) Alt-Prtsc will send a FormFeed to the printer (it will
never hang though if the printer is not available). This
particular method even works properly with the print
spoolers I have tried.
;
(iii) Drive A & B diskette I/O is intercepted and all WRITES,
VERIFYs and FORMATs are done in slow speed. Whatever speed
had last been chosen (by Alt-MINUS or ALT-PLUS) is restored
immediately when the operation has completed. Note that
Floppy READs and all drives C and up are not affected.
The reason for this particular item is that DISK FORMATs
and WRITES would never work on my system. The FORMAT or
any COPY to a diskette would fail and a message would be
printed telling me of the failure. With this program,
I no longer have to worry.
;
Please note: if you write a FAST or SLOW program such as
suggested in (3), then option (iii) will not know which
speed has been recently selected (the LEDs cannot be read).
Thus (iii) will restore what it remembers was last selected
by (i). This is why this program is all in one...
the last selected speed is easily shared with the diskette
I/O speed change software.

```

Continued on Page 81

Protected Input For C

Gary A. Appel
1318 Old Abbey Place
San Jose, CA 95132

Introduction

A couple of years ago, when my wife decided to take a programming course in C, I purchased a copy of the Software Toolworks C80. After a quarter of homework was completed, the C80 compiler got put away in favor of the Turbo Pascal compiler I was more familiar with. Every so often, I was tempted to pull it out again and try some serious C programming. And the one major disadvantage of Turbo Pascal, the inability to separately compile modules, kept bothering me. Eventually, the C80 compiler got pulled back off the shelf, for a serious look.

Before writing any serious programs in any language, I feel the need to develop a good, foolproof input routine. The version of scanf supplied with C80 does not provide adequate error checking or handling. In fact, I've had problems just getting scanf working with floating point input. As a result, this protected input module was developed for use in additional C development.

I assume that the reader is somewhat familiar with the C programming language. In particular, I will make no attempt to explain or define the operation of individual C statements. I will attempt to describe any unusual or confusing techniques that I have used. In addition, I will describe the techniques involved in compiling a separ-

ate module for C80 to be combined with the main program during link time.

I will also describe some techniques I have used in attempting to avoid both erroneous input and computations in my programs. The end result is an input module which should prove to be useful for additional program development in C.

The Functions

This input module consists of three main functions. Together they allow the program to look for and detect the following forms of input.

1. Integer constant
2. Floating point constant or expression
3. Character string to match a supplied list
4. End-of-line
5. End-of-file
6. No input
7. Error

Several of these cases need further definition.

Integer constant is obvious. Floating point constant or expression allows the input of a floating point constant, or an expression consisting of the four operators +, -, *, and /, along with parentheses, and floating point constants.

Option 3 is included to allow the calling function to make a selection based on a

string supplied from the user. For instance, the program might respond to the string "quit" by exiting to the operating system. The calling function passes a list to the input routine. This list contains pointers to several different strings. Should the user type a string matching some entry in the list, its location in the list is returned. A simple switch statement can then be used to determine the proper action. This option will be described in more detail later.

End-of-line and end-of-file are just that. A value is returned by the function to indicate that the end of a line or file has been reached. The appropriate action can then be taken.

No input is an option that will probably be of only occasional use. If the user at some point types two consecutive commas, indicating the lack of any input between them, the input function will return a value indicating no input.

And finally, what if the input doesn't make sense. This can occur if numeric data is entered wrong, an expression is entered wrong, or a string was supplied that does not match any string in the list provided. In this case, the function returns a value to indicate an error condition.

The first of the three functions is gettkn (get token). This routine is the heart of the input module. This function is passed the list of

strings to be searched for a match, and returns an integer 'token' for each individual string indicating which of the seven types of data described was received as input.

The other two functions are twins. One is used for floating point input, and the other for integer input as described above. They can be called after `gettkn` has detected the presence of numeric data, or they can be called to force `gettkn` to look for a numeric input. The routines are named `getreal` and `getint`.

Some Definitions

In this discussion, several terms will be used which may require some definition in order to avoid confusion. I will attempt to define these terms before proceeding.

Token: I will use this term to describe the integer value returned by the three input functions. For example, the integer value returned when an end-of-line is detected will be described simply as an integer token. I may also use the term token to describe the type of data to which the input functions must respond, including each one of the strings in the supplied list.

Module: This entire file will be compiled into a single module. A module is a function or group of functions that are compiled at the same time, but separately from any additional functions. The module can then be added to the C library.

Function/Variable Declaration: The declaration of a function or variable informs the compiler of several different characteristics or 'attributes' of the function or variable. For instance, what type of variable, and how the variable is stored, as well as which functions know of the existence of the variable. For a function, the declaration may inform the compiler what type data the function returns, and which other functions may know of the existence of this function. In C, the declaration does not tell the compiler how many or what type of parameters are passed to the function.

Function/Variable Definition: Note that the declaration only informed the compiler of the characteristics of the function or variable. It does not define the function, or reserve storage for the variable. This is the job of the definition. In practice, the definition usually occurs with the declaration, but not always. In several cases, we will be declaring functions before we define them, for reasons explained later.

My Compiler Environment

I am using the C80 compiler from The Software Toolworks, along with the op-

tional mathpac. I am using the compiler on an H-89, with two quad density drives, so disk storage is of little consideration. The compiler is configured to produce Assembly language code for the Macro 80 assembler, which supports the use of separately compiled relocatable modules.

The C80 compiler outputs Assembly language source code from the compiler. While this requires an additional step in the compilation process, it does allow the user the rarely used opportunity to optimize the code. More often, I find the ability to inspect the code to be advantageous, as it sometimes quickly points out areas in the source code where I have assumed I told the compiler to do one thing, while I actually told it to do something else. Very often the resulting source code is so different, that the mistake becomes obvious.

One very appealing feature of the C80 compiler is that all the run time source code is included. Modification of the source code to improve the run time module is possible. An example will be shown later where examination of the source code has resulted in the solution of a problem that appeared to have no reasonable solution. Unfortunately, the solution is C80 specific.

I tend to place compilers in three categories. These are: fancy, usable, and toys. Both C80 and Turbo Pascal are in my usable category. What this means to me is that usable programs can be written using these compilers. The execution speed is acceptable, and the error handling characteristics are acceptable. The toy compilers are usable only for educational purposes. They are slow, have inadequate error handling characteristics, or just have bugs. The fancy compilers are nice, if you can afford them.

The use of separately compiled modules is appealing to me for several reasons. First, the compilation time is reduced, as only short sections of code are compiled at one time. Second, it saves disk space, as the entire source code file need not be present on the disk during compilation. And most important, it allows a very clean interface with the calling function. Only the required function entry points are passed to the linker. Anyone who has used Turbo Pascal can appreciate the number of variables that get globally defined simply because they need to be recognized by more than one procedure. When separately compiled, these variables, as well as minor functions, can be declared as global within the module, and yet not be passed to the linker, therefore, becoming invisible to any additional modules.

While these listings are specifically for the C80 compiler, they should be usable with little modification on any C compiler supporting floating point, as C is one of the most portable languages around. Any C80 specific code has been labeled as such.

The `gettkn` Function

The actual layout of a function is dictated by deciding what the function should do. Before writing any function, its actions have to be well defined. For instance, we must decide exactly what type of input the function must respond to, and how. What should the function do if an error is detected, and how do we define an error? The definition must be precise enough to allow the function to be written down to its own function calls. Additional definition is then required before writing those functions, until eventually the entire definition is established.

In the case of the `gettkn` function, we have already defined the list of tokens that must be responded to. In the case of the list and numeric data, we have been vague so far. Their definitions can wait.

At this point, we must know something about the input routine used in the C80 runtime package. C80 accepts an entire line at a time from the CPM environment, using the conventional CPM control codes. Once an entire line has been received, as indicated by the receipt of a carriage return, C80 will pass the received characters one at a time using `getchar`, with the last character passed a newline character. As a result, we don't have to worry about line editing, CPM will handle it for us. The important point to remember here is that when a newline character has been received, the program will have to pause for additional input. If the newline character has not been received, additional data will be available without requesting additional input from the user.

As we scan through the input string, we must separate the input into the various token types. The end-of-line and end-of-file tokens are easy to define. Error, while more difficult to define, is conceptually quite simple. If we can't make sense of the input, we return an error token. No input has already been defined as the empty space between two commas. We will extend the definition to include the possibility of a null string between two commas, if two commas are entered next to each other. We still have to define the string and numeric inputs.

For simplicity, we will define the string tokens as any string that begins with an

Listing 1

```

/* Listing #1, a demonstration for the input module */
#define C80 /* Delete if not compiled using C80 */
#ifdef C80
#include "fprintf.h"
#endif

#define NOINPUT 0
#define EOF -1
#define EOL -2
#define ERROR -3
#define FLTVAL -4
#define INTVAL -5

/* Definitions */
/* Define token values */

main()
{
    /* Variable definitions */
    static char *list[] = /* Our List of options */
    { "Help", /* Option # 1 */
      "List", /* Option # 2 */
      "Look", /* Option # 3 */
      "QUIT", /* Option # 4 */
      ""
    };
    int token; /* token returned by gettkn */
    float fdata; /* data retrieved from getreal */
    int idata; /* data retrieved from getint */

    printf("test routine\n");
    for (:(token=gettkn(list)) != EOF;) /* Continue looping unless EOF */
    { switch(token) /* Select appropriate action */
      {
        case 1: /* Help Option */
            printf("I'm too busy to help you now\n");
            break;
        case 2: /* List Option */
            printf("You have choosen the list option\n");
            break;
        case 3: /* Look Option */
            printf("You have choosen the look option\n");
            break;
        case 4: /* Quit Option */
            printf("Thank-you for testing\n");
            exit();
        case NOINPUT: /* No Input */
            printf("No input received\n");
            break;
        case ERROR: /* Input Error */
            printf("I don't understand, sorry.\n");
        case EOL: /* End of line */
            printf("\n");
            printf("Continue...");
            break;
        case FLTVAL: /* Floating Point Input */
            if (getreal(&fdata, -500.0, 500.0) != ERROR)
                printf("Floating point value is: %8.2f\n", fdata);
            else
                printf("Floating point value out of range\n");
            printf("\n");
            printf("Try again...");
            break;
        case INTVAL: /* Integer Input */
            if (getint(&idata, 0, 1000) != ERROR)
                printf("Integer value is: %5d\n", idata);
            else
                printf("Integer value out of range\n");
            printf("\n");
            printf("Try again...");
            break;
    } /* End of switch */
    } /* End of for */
    printf("End of file condition detected\n");
}

```

alpha character, upper or lower case. Any input that does not match any of the above definitions will by default be assumed to be numeric.

The `gettkn` function must be passed a list. Actually, it will be passed a pointer to a list. For now, that's all we need to know. It will simply pass the pointer on to another function that will attempt to match strings.

We have already stated that `gettkn` will return a token value. We really don't have much choice here, as the only reasonable type to return is integer. Each token type will be assigned an integer value.

For starters, there is already a definition for end-of-file, or EOF. This is a value returned by `getchar`, and is equal to `-1`. For compatibility, we will return the same value. If the user supplies a string which matches some entry in the list, we will return the location of that string in the list, from 1 to `n`. The remaining tokens will be assigned values of zero, or less than `-2` to avoid any conflict with the previous tokens.

At this point, we need to define some variables.

First of all, we need an integer variable to hold the token value to be returned. While it may be possible to return a value without an explicit variable to hold the value, the resulting program would probably not be clean. By defining a variable, we may set its value at any time, and return it later. We will call this variable 'type'.

We also need a character variable to store the character just received from `getchar`. We will call this variable 'nextchar'.

We need a string buffer for storage of strings during the table lookup function, and during numeric and expression evaluation, since these both involve operations with multiple characters. We will call this character array 'buffer'.

And finally, we need a pointer into the array buffer. We will simply call this variable 'i'.

One of the tokens returned represents an end-of-line detected. We will call this token 'EOL'. If an end of line is detected we must read another line. But we don't want to hang up waiting for another input before the EOL token has been returned, because the calling function won't know to request input until after it has seen the EOL token. We have to look for new input on the first invocation of `gettkn` after the EOL token has been returned.

Although we have not elaborated on the error token, it needs some additional con-

sideration. What do we do if we return an error token? Again nothing, until the next invocation of `gettkn`. At this time, we must discard the remainder of the line and look for additional input. It is very difficult to make sense of input after an error has been detected.

The operation of `gettkn` can now be paraphrased as follows:

```
if (Error was detected)
    Discard the line;

if (End of line)
    Read a new line;

switch
{
    case Comma was detected:
        /* No Data */
        return(no-data token);

    case New line
        return(end-of-line token);

    case End of file
        return(end-of-File token);
}

if (Alpha character was detected)
    look for matching string in supplied list;

else
    attempt to read a numeric input;
```

The module listing shown in Listing 3, should now be consulted to see the actual construction of the `gettkn` function. In converting the above description into C, a few problems arise which must be taken care of.

The first problem that comes up is how to handle the end-of-file token. This will probably only happen during input redirection, and the calling function should not attempt to obtain any additional input. But let's assume it may. We will return the EOF token, and discard any additional input on that line, a purely arbitrary decision. In order to do this, we will return EOF but set our own private copy of type to ERROR.

Second, how do we decide to get additional input? We can't do this when `nextchar` is the newline character, as mentioned above. So, we will set `nextchar` to EOL to signify the completion of a line, and the requirement to look for additional input. In effect, EOL is a noncharacter that lets us know that `nextchar` does not contain a valid character. One must be obtained by another call to `getchar`, which will force the input of an entire new line, as required by the C80 run-time library.

Third, in the case of an error, we will also set `nextchar` to EOL to force the input of a new line after discarding the old one. This is

redundant information, as the ERROR token already informs us that new data is required, but we already look at `nextchar` for this purpose.

Finally, how do we start cleanly? We must initialize `nextchar` to EOL to signify that `nextchar` does not yet contain a valid character, and we must set type to something other than ERROR. We must also assure that it is not set to INTVAL or FLTVAL, the integer and floating point tokens, so that our functions do not assume that valid data has been received. I have chosen to set it to NOINPUT to avoid any conflict. Note that type will by default be set to NOINPUT, since this token is equal to zero, but it is best to explicitly initialize a variable to highlight the initialization requirement.

Note in the listing, the variables have all been declared static, and defined outside of any of the functions. The definitions must be outside of the functions, because they will be used by several functions within the input module. They could be passed as parameters, but that would add unnecessary complication to the functions.

The reason for the static declaration is at least somewhat obscure. By defining the variables outside of any functions in the very, very small print in the C manual, you will see that the static declaration does one other thing. It forces the variable to be defined only within the module. While the variable will be known throughout the input module, its name will not be passed to the linker. Without the static declaration, the name would automatically be passed to the linker as a public variable. Since these variables are used only by the input module, we don't want the linker to know about them.

Directly after the variable declarations are four function declarations. These four functions are contained in this file. These declarations serve as forward references for the C compiler. The functions have been declared static just as the variables were, and for the same reason. The static attribute prevents the function names from being passed to the linker as public names. If the function names are not predeclared in the file, they will be declared as public by default the first time they appear in a function definition. Note that these declarations are not definitions, the functions will be defined later. The `gettkn` function itself has not been declared as static, as its name must be passed to the linker.

The actual `gettkn` function is fairly simple. Only a few points need additional discus-

sion. First, the newline character returns a separate token. An immediate carriage return will return an end-of-line token. One piece of data will return two tokens; The first call to `gettkn` will return the token representing the single piece of data. The next call to `gettkn` will return an end-of-line token. Second, after the error and end-of-line tests, the function `skpwht` is called, to skip over any white space before the next data appears, so that leading spaces or tabs do not confuse the function. The newline character is not considered white space in this function.

In the `gettkn` listing, we have put off the definition of the table lookup and numeric/expression input by simply entering them in the `gettkn` function as additional functions to be executed. Before we can write these functions, we must define them adequately. First, we will consider the function `lookup`, which is used to search for a matching string in the supplied list.

The Lookup Function

Very often a function (perhaps the main program) will go off and execute some section of code based on a users input. For instance, it may execute some additional input code, some computation code, or perhaps exit to the operating system. Many programs allow selection by offering a menu to the user. I don't like menus. If I did, I'd have an Apple computer. When I decide to exit a program, I want to simply type the word 'quit'.

We already know that the lookup function is executed whenever the input begins with an alpha character. We need to further define what type of input we will accept.

For input we will accept any string, so long as it begins with an alpha character. We will not allow the function to be case sensitive. The string will end at the first delimiter, which will be defined as a space, tab, comma, or newline character. The length of the string will be limited by the length of the character buffer, which was chosen as 30 characters for other reasons. This will certainly be long enough. Now, just what do we expect `lookup` to do?

The calling function should be able to pass to `gettkn` a pointer to a list something like this:

```
list:  "help"
       "list"
       "quit"      ,etc.
```

If the user types "help", `gettkn` should return a value of one (A help token). If the user types "list", a token value of two, and three if the user types quit. The calling rou-

tine can then execute a switch statement based on the value returned by `gettkn`. If a matching entry cannot be located, `lookup` should return an error token.

But let's reconsider what we would like to mean when we say that a matching entry has been located. The obvious definition would be to require an exact, although case insensitive, match. But I'm going to go a little further. I may be too lazy to type in the entire word "help". Maybe just the letter 'h' should be sufficient.

Strings in my list will be combinations of upper and lower case characters. The first portion of each string will be upper case, and the last portion lower case. The input string must match all the upper case characters. Once we get to the lower case characters, the input characters must match only if present. With this definition we can redefine the above list as follows:

```
list:  "Help"
       "LIst"
       "LOad"
       "QUIT"      ,etc.
```

In this case, any of the strings "h", "he", "hel", or "help" will return a token value of 1. I have added the string "LOad" to illustrate the need for multiple comparisons. In this case, at least two letters must be specified for the list and load options. To quit, the user must type all four letters, not a bad idea for such a drastic action.

At this point, we need to define the format of the list referred to in the function.

The function is passed a pointer to the list. But we don't really want a pointer to a list, what we really want is a pointer to an array of pointers. Each pointer in the array will point to a separate string as defined above. As is customary in C, each string will be zero terminated. The last pointer in the list will point to a null string. It is this null string that will inform the lookup function that it has located the end of the list. An alternative here would be simply to return a null pointer as the last pointer in the array.

We next need to define the variables required by `lookup`.

First, we have the pointer to the array passed to `lookup`. We will define this variable as 'tknptr'. The various pointers in the array can be accessed by using `tknptr` as an array, and accessing them as the various elements of the array.

Next, we need pointers to scan both the input character string and each of the character strings in the supplied list. The input string will be scanned by the pointer 'i', and each of the supplied token strings will be scanned by the pointer 't'.

Finally, we must count the token strings as we search through the list. We will do this with a variable labeled 'tknctr'.

The operation of the lookup subroutine can be paraphrased as follows:

```
read the complete input string

for each string in the supplied list
{
  test for a match.
  if a match
    return its location in the list
}

return error token
/* No match was found */
```

Again, the module listing in Listing 3 should be consulted to see the actual implementation of the lookup function. And again, while writing the function, some additional problems arise which must be taken care of.

First, we must place the input string into buffer for the search operation. But there is the possibility that buffer will fill up before the end of the string is located. In this case, an ERROR token must be returned by `lookup`. The rest of the input string will be discarded by the error processing already incorporated into `gettkn`.

Next, the token may be terminated by a space, tab, comma, or newline character, or some combination. Our previous implementation of `gettkn` will skip over white space, but not commas. In fact, it cannot skip over commas, as the receipt of a comma signals no-input to the `gettkn` function. The lookup function must therefore swallow a comma if used as a delimiter. In order to do this, we first swallow any white space following the input string. If the next character is a comma, we swallow it too. Otherwise, the next character is left in `nextchar` for the next invocation of the `gettkn` function. Any white space after the comma will be devoured by the next invocation of `gettkn`.

The next portion of the code tests each entry for a match. Keeping in mind that `tknptr` is a pointer to an array of pointers, `t` is first set equal to the value of the first pointer (the first element of the array). In each additional pass through the for loop, `t` is set equal to the next element, which is the next pointer. In this manner, `t` will point to each string. It should be noted that the array contains pointers, so it is these pointers that will be adjacent in memory. The actual strings can be located anywhere in memory, in any order.

As each string in the list is scanned, each character is converted to upper case prior

to the comparison with the input string. The for loop continues so long as the two strings match, or until the terminating zero is found in the input string (&& *i). The condition for success is somewhat confusing. The !*t and !*i specify that if we have reached the end of both strings, the match is successful. The islower(*t) is the code that defines the match to be successful if the strings match throughout the length of the upper case characters, and of course, if the end of the input string has been reached.

Should a match occur, the position of the string in the list is returned, which is one greater than the value of tknctr, due to the zero based arrays used in C. Should no match occur, an error token is returned.

The getdat Function

One of the reasons for developing this input module was to partially eliminate the need to carry a calculator along on every trip to the computer. How often do we have to calculate a number just to plug it into the computer? Wouldn't it be much simpler to let the computer perform the calculation for us? I wanted my input routine to do just that, while at the same time screening out illegal input.

I see the need for an input routine to supply both integer and floating point data. While it would be possible to check any data to see if it is in fact integer, I don't see the need for that. Integers are generally used for very simple input, such as to represent line numbers. I don't believe that there is much need to use the expression feature of the input routine when integer input is required. I can usually add and subtract integers in my head. So my definition of integer input applies only to integer constants. Any time an expression is entered, the data will automatically be flagged as floating point.

The getdat function does not return data, but rather returns a token. The actual data is stored in a variable that I have chosen to call 'result'. Two additional functions will be used to obtain the numeric result from this module, one to obtain an integer result, and the other to obtain a floating point result. The token returned by getdat will be integer, floating point, or error.

Although we are looking for both floating point and integer data, there is no need to provide for both floating point and integer processing in getdat. All processing will be done using floating point math. If integer data is returned, it can be converted during the final assignment operation. All we need to do is assure ourselves that the data was in fact an integer number.

How do we define an integer? In order to be flagged as integer data, we have already decided that the data must be a numeric constant. In addition, we must define the format of an integer constant. We will use a common definition. Any string consisting of an optional plus or minus sign followed by a string of digits only is an integer constant. A decimal point or exponent will not be accepted. Any other input will be floating point, or flagged as erroneous input.

We have already stated that the input function allows the input of expressions consisting of the four operators +, -, *, and /, as well as parenthesized sub-expressions. We will use the standard operator precedence in the expression, with * and / having precedence over the + and - operators. Otherwise, we will scan from right to left. Parentheses are allowed to establish precedence in the normal manner. An additional requirement I have placed on expression input is that the expression must not contain any white space. This simplifies the scanning of the expression, and eliminates a problem.

The gettkn function was written to accept multiple inputs on a single line. This includes multiple numeric data and multiple expressions on a single line. The problem that arises is how to deal with the input line:

```
2 -1 <CR>
   where <CR> represents the carriage return.
```

Is this the two integer values two and negative one, or is it a single expression equal to 1.0? We solve the problem by defining an expression to contain no white space. If the negative sign followed the 2 directly, the input would be an expression. As entered, the data represents two integer constants.

Our definition of getdat is now complete except for the definition of a floating point constant. We will leave that for another function. Even the definition already assigned to integer constant will not be required immediately. We do, however, still have to define the variables required.

The result of the expression (or integer constant) will be placed in the floating point variable 'result'. The already defined integer variable 'type' will contain the integer token, floating point token, or error token.

The implementation of getdat is not very straight-forward. In order to simplify the description, I will present a few more definitions.

First, the lowest level must add and subtract various sub-results which we will define as terms. The value of the terms will be calculated before the additions and subtractions. This makes sense, because the addition and subtractions have lowest precedence. Next, the terms are formed by multiplications and divisions of various different sub-results that we will define as factors. Again, the value of the factors will be calculated before the multiplications and divisions are executed.

What then is a factor? It can be one of two things. It is either a numeric constant, or a sub-expression contained in matched parentheses. If it is numeric constant, we are done. If it is a subexpression, we calculate it by a recursive call to our function that calculates the value of an expression.

Consideration of the above definitions will show that our expression evaluation will in fact evaluate the expression just as we desire. In order to simplify the writing of the getdat function, we will separate it into four functions following the description above.

The first of these functions is getdat. This function must contain any initialization and termination code, and will obtain the expression value by calling another function that we will name (appropriately) 'getexp'. This function will in turn call the function 'getterm', which will call the function 'getfact'. The getfact function will call a function yet to be defined which we will name 'getnum', and will also make a recursive call to the function 'getexp'. It is this recursive call that requires us to put our initialization and termination code in the separate function 'getdat'. The operation of the getdat function can be paraphrased as follows:

```
flag integer data;
/* Assumed integer */
get the value of the expression;

if an error has occurred
    return error token;
return type token;
/* may be integer or float */
```

The getexp function can be paraphrased as follows:

```
get a term;

so long as we find a + or - operator
{
    if we find a + operator
        add the next term to the total;
    else if we find a - operator
        subtract the next term from
        the total;
}
return the total;
```

The getterm function is a twin:

Listing 2

```

/* Listing #2, parallel resistance calculation */
#define C80 /* Delete if not compiled using C80 */
#ifdef C80
#include "fprintf.h"
#endif

/* Definitions */
#define NOINPUT 0 /* Define token values */
#define EOF -1
#define EOL -2
#define ERROR -3
#define FLTVAL -4
#define INTVAL -5

#define MINVAL 0.1 /* Min and max allowable resistance values */
#define MAXVAL 1e7

#define INFINITY 1e30 /* Used to represent infinite resistance */

/* Forward declarations */
static disval(); /* Display value subroutine */
static diserr(); /* Display error message subroutine */

/* Variables */
static int token; /* Token value returned by input routine */
static float newval; /* Storage for newly inputted values */
static float newresist; /* Storage for newly inputted result */
static float resist = INFINITY; /* Value of parallel resistance */

static char *list[] = /* Our token list */
{ "Reset", /* #1 */
  "QUIT", /* #2 */
  ""
};

main() /* Parallel resistance calculators */
{
    printf("Parallel resistance calculator\n\n");
    for(;;) /* Infinite loop */
    {
        printf("OK: ");
        token = gettkn(list);
        switch(token) /* Select appropriate action */
        {

```

```

case 1: /* Reset the Calculator */
    if((token = gettkn("")) == EOL)
    { printf("Resetting Calculator\n\n");
      resist = INFINITY; /* Reset the Resistance */
      break;
    }
    diserr("Illegal format"); /* Must be followed by newline */
    break;

case 2: /* Quit */
    if ((token = gettkn("")) == EOL)
        goto quit;
    diserr("Illegal format"); /* Must be followed by newline */
    break;

case ERROR: /* Error Detected */
    diserr("Illegal input");
    break;

case INTVAL: /* Numeric data read */
case FLTVAL: /* Get first value */

    token = getreal(&newval, MINVAL, MAXVAL);
    if (token == FLTVAL) /* Data may be out of range */
        newresist = newval; /* Continue reading values */

    while(token == FLTVAL || token == NOINPUT)
    { token = getreal(&newval, MINVAL, MAXVAL);
      if (token == FLTVAL)
          newresist = newresist * newval
            /(newresist + newval);
    }

    if (token != EOL) /* Illegal input */
        diserr("Illegal Numeric Input");

    else /* Calculate new value */
    { if (resist == INFINITY) /* First input */
      resist = newresist;
      else /* Update result */
        resist = resist * newresist /(resist + newresist);
      disval(resist);
    }
    break;

```

```

        case EOF: /* End of file detected */
            goto end_file;

        } /* End of switch */
    } /* End of infinite for loop */

end_file: /* Indicate end-of-file was detected */
    printf("\n\nEnd of File detected\n\n");

quit: /* Terminate execution */
    printf("\n\nExecution Completed\n\n");
    exit();

} /* End of main */

static diserr(strng) /* Display error message */
char *strng;
{
    printf("%s, Please re-enter.\n\n",strng);
}

static disval(data) /* Display present resistance value */
float data;
{
    printf("Parallel resistance is: ");
    printf("%10.2f\n\n",data);
}

```

```

get a factor;

so long as we find a * or / operator
{ if we find a * operator
    multiply total by the next
    factor;
  else if we find a / operator
    divide total by the next
    factor;
}

return the total;

```

And finally, the getfact function can be paraphrased:

```

if we find a '('
    return the value of an expression;
else return the value of a number;

```

While each of these functions is executing, we must be watching for errors. We must also be watching for any circumstance that would require that we flag the data as floating point. In order to fit the requirement for integer data, the expression must consist of only one term which consists of a single factor which is an integer constant. Any additional evaluation, or the presence of a floating point constant, will result in flagging the data as floating point.

The error checking consists of looking for an invalid character during the function executions. If getnum finds an invalid character during the numeric conversion, it will flag an error. The only error that can occur in the function getfact is unmatched parentheses. If a left parenthesis is located, the function getexp is called. Upon return, nextchar must contain a right parenthesis. If so, it is swallowed.

The function getterm is always looking for a * or / operator. If one is not found, we return to getexp with the next character still in nextchar. An error must be flagged, however, if we attempt a division by zero.

The function getexp is always looking for a + or - operator. If anything else is found, it must return to the function that called it, either getfact or getdat. Once we have arrived back at getdat, nextchar must contain a valid delimiter to flag the end of the expression. If it contains anything else, we have an error condition.

This has been an awfully wordy and confusing description of what we want to do, but then the process of scanning an expres-

sion is not simple. If we don't pick it apart like this, we stand very little chance of actually getting the function working.

At this time, you should consult the module listing in order to see the actual implementation of these functions. Again, in writing the function additional problems arise, and solutions must be found.

We have had to include code in getdat to swallow a comma, just as we required code in lookup to swallow a comma. The C80 specific code is used to determine if an error has occurred during the expression calculation. It will be described in more detail later.

As mentioned earlier, the getexp and getterm functions are twins, although not identical. In each function, we have had to define two variables, 'op' and 'data', for temporary storage. The op variable is required to hold the operator, which is lost from nextchar by the function call which returns the second operand, and so must be saved. The 'data' variable is required to hold the running total. Because of the recursion, they must be an auto variable (and are by default). If ever an operator is found, in either function, floating point data is flagged. The only error detection is in getterm which flags division by zero. Note that the division is not attempted if the second operand is zero. Some run-time modules will kick us back into the operating system if a division by zero is attempted. We want to avoid that. Note that the coding is such that any time an error condition has been flagged the indefinite loop is immediately terminated. There is no need to continue scanning the expression after an error has been located.

The function getfact is just as described. The only error checking is to assure that a left parenthesis is matched with a right one.

What about the C80 specific code? It is almost impossible to assure that the user has not typed in some combination that will result in an error condition during the numeric calculations. The most likely problem would be an overflow condition, or underflow if we define it as an error. Assuming that we don't get kicked back into the operating system, and we won't in C80, we would like to know that an error has occurred.

An undocumented feature of the C80 floating point run-time package is the existence of the public variable 'errcod'. (You find these undocumented features by looking through the supplied source

code.) Errcod is initialized to zero. If an overflow or divide by zero error occurs during an operation, errcod will be set to a non-zero value to flag the error. The divide by zero error cannot happen, because we have screened it out, but the overflow condition can still occur. After the expression has been fully scanned, we check errcod to see if it is non-zero. If so, we reset it to zero, and flag an error. This is non-standard, but an absolute must for our input routine. Otherwise, we can be reading bad data and not even know it. I have modified my runtime code to also flag underflow, while still return a value of zero. If underflow occurs during the evaluation, it too will be flagged as an error.

Finally, we come to the getnum function, which will return the value of a numeric constant. Its definition is simple, and obvious from the listing:

```
look for an optional + or - sign;
look for integer digits;
look for a decimal point: ;
    look for fractional digits;
look for an 'E' or 'e';
    look for an optional + or - sign;
    look for an exponent;
```

where the indented tasks are performed only if the decimal point or E/e characters are in fact found. If either of these are located, the data must be flagged as floating point.

The C language provides a standard function to convert a string in a valid format to a floating point value. This function is ftoa. Rather than write our own routine for the conversion, we will simply use the ftoa function. The getnum function must then only move the characters into our buffer, and test for valid format.

At this time, we need to further define a valid floating point constant. We already know it must meet the above definition. But it must also lie within the representable limits of the computer. We must also decide if we want to limit the number of digits of input. In order to track all this, we need to count the number of integer digits, the total number of mantissa digits, and the number of exponent digits.

This is a good time to begin writing the C function getnum. We have a pretty good idea of what it needs to do, and what errors we need to look out for. The actual implementation will be hard to paraphrase, and is best written directly in C. Again, Listing 3 should be consulted.

In addition to the variables 'idigs', 'mdigs', and 'edigs' to count the number of integers, mantissa, and exponent digits, we have also provided a variable to hold the

result. The pointer to this variable must be passed to the function atof.

The layout of the function is quite similar to the layout described above. A handful of additional functions are called to perform the scanning and movement of the input character string.

The function getsign will pass along a negative sign to the buffer, while it will swallow a plus sign. Next getdigs moves the integer digits into the buffer, while returning a count of these digits to the variable idigs. If a decimal point is found, it too is moved, followed by the fractional digits while setting mdigs to the total number of digits found. Also, the data must now be flagged as floating point. If no decimal point is located, the total number of digits equals the number of integer digits.

Now we look at the total number of mantissa digits input. We must have at least one digit, and flag an error if more than eight digits were received, as we don't have enough precision for so many digits. This prevents the user from entering large or small data in floating point notation, as the non-significant zeros count as digits.

Finally, we look for an exponent. If 'e' or 'E' is located, we have found one, and the data is flagged as floating point. We move the e/E and any trailing digits into the buffer, while setting edigs to the number of digits moved. We need at least one digit, by no more than two for a representable number. Otherwise, an error condition is flagged.

Assuming we have made it this far, the only problem left is that the value of the data may still be out of range. If only one exponent digit was entered, we assume that the data is in range. If not, we will get an error from too many digits before we can exceed the representable range. If two exponent digits are received, we must test for a too large condition. This is accomplished by converting the two digit exponent to binary, and adding the number of integer digits. If the result exceeds 37, we flag an error.

I have taken several short cuts here. Again, the user cannot enter large or small numbers in floating point notation. For example, the following inputs would result in an error:

```
123456700      nine digits
0.00000123    nine digits
000000001     nine digits
```

In order to allow this type of input, we would also have to count zeros, but only leading or trailing zeros. Another solution

would be to eliminate the test for significant digits, and allow any number of digits.

In the case of a two digit exponent, I only test the exponent value if the exponent is positive. Any two digit exponent will slip through the test if it is negative. While this is undesirable, the atof function is kind enough to set underflowed data to zero. This was determined by testing the input for many different values.

The next five subroutines are used to move characters from the input into the character buffer. The actual character movement is accomplished with the movchar function, which must move the character, and test for a full buffer condition. If the buffer is full, an error must be flagged, and no additional characters moved. The movchar function must continue to swallow characters however, or we will never complete the getdat function.

I hope this lengthy discussion has left you with an appreciation for the importance of an adequate function definition before any attempt has been made to write the function. The basic layout of the function is dictated by the definition of what the function should do. If the function is not adequately defined initially, the basic layout may be wrong. As problems arise, they will be solved by patches to the incorrect layout. The result will be a non-optimum coding, which is most likely also not a clean coding, and which is difficult to fix or modify.

The getreal And getint Functions

Finally, we have arrived at the two numeric input functions, getreal and getint. These functions are twins, with getreal supplying a floating point value, and getint supplying an integer value. Again, we must first define just exactly what these functions should do.

These two functions will be required to serve two purposes. First of all, if the gettkn function locates a numeric input, we need to call either getreal or getint to retrieve this value. But I also want to be able to call the functions directly, if I am looking for an integer or floating point datum. In this case, I still need to obtain a token to inform me of an error condition at the very least, and preferable also to inform me of other conditions, such as no-input, end-of-line, and so on. I also don't want to return data if the input data was bad or non-existent. For these reasons, I have chosen to let getreal and getint return token values, just as gettkn does. The only difference is that getreal will not return an integer token, and

getint will not return a float token. The getreal function will return a float token for either type of data, while getint will return an error token if the input was flagged as floating point.

The actual variable will be set to the input value within the functions by passing a pointer to the variable as a parameter. If no valid data is received, we will not modify the variable.

Another implication of the dual use of each function is that we must remember if gettkn has just returned an integer or float token. If so, we return its value. If not, we must call gettkn to see if the next input is a numeric input. Any time getint or getreal is called, we must reset 'type' to assure that the next call to getint or getreal will not simply return the same input that has already been returned.

Finally, I want each function to also test the input value to assure that it is within a valid range. The range will be specified by providing two additional parameters delimiting the allowable range. If outside of this range, an error token will be returned.

The variables passed to the getreal function are as follows:

data: This is the pointer to the floating point/integer variable where the input data is to be stored. It has to be a pointer so that we can store the data.

min: This is a floating point variable that represents the minimum allowable input data.

max: This is a floating point variable that represents the maximum allowable input data.

The getreal function can be paraphrased as follows:

- if last input was not integer or floating point data, attempt to read numeric data;
- if the data is still not integer or floating point, return the token describing the input;
- if the data is not within the allowable range, return an error token;
- store the data;
- flag this input as no-input; /* So we only read it once */
- return a floating point token;

The getreal function listing should now be consulted for the implementation details. The only point that needs to be mentioned is the call to gettkn. A null list is supplied in

this call to assure that no matching string can be found, since we are looking for numeric data. If a string is input starting with an alpha character, an error token will be returned.

The function getint is very similar except the data must be flagged as integer for the function to return valid data. If the data is flagged as floating point, an error token is returned. The data is converted to integer in the assignment to the parameter 'data'.

Compiling The Functions

The complete module listing is included at the end of this article. Assuming you are using the C80 compiler with the Macro-80 assembler, you must first compile the module with the m switch (assuming the compiler has not been configured for Macro-80 as the default assembler):

```
C -M INPUT
```

Then assemble the input file using the Macro-80 assembler:

```
M80 INPUT=INPUT
```

At this point, we have a relocatable module to be linked to our programs. We also must make sure that the relocatable versions of the main run-time library (CLIB.REL), and floating point run-time library (FLIBRARY.REL) are on the disk.

The C80 mathpak also provides us with the C code for the formatted output function printf, called FPRINTF.C on the mathpak disk. This must be compiled into Macro-80 format, and assembled to produce a relocatable version. If we are going to use printf to output floating point data, we must be sure to use this version supplied on the mathpak disk.

Similarly, if we wish to use scanf for floating point input, we must be sure to use a version written for floating point input. The file SCANF.C supplied with C80 can be configured for floating point input by making sure that the #define flong statement is not commented out, although as I mentioned, I have had problems using scanf for floating point input. Why not just use the input procedure described here?

When all this has been accomplished, the main function can be linked with the relocatable modules to produce a com file:

```
L80 MAIN,FPRINT,INPUT,FLIBRARY/S,  
CLIBRARY/S,MAIN/N/E
```

where the /S switch forces the two libraries to be searched as libraries, loading only the modules required.

Ultimately, you will want to combine the relocatable files into a single relocatable

library to simplify the linking process. The required details are in the LIB-80 manual.

During the linking process, the modules must be listed in the above order, with the exception that the FPRINT and INPUT can be reversed. This is to assure that each module linked will contain external references only to modules on the right, which have not been linked yet. If the modules are combined into a single library, the same ordering restriction applies.

Testing The Input Module

During the development of the input module, we will have to test it often to assure that it is doing what we think it should be doing. After completion, we need to test it even more to assure that we are finished. Listing 1 is a program written specifically for testing the input module. We should be able to exercise every capability of the input functions with this routine, to confirm that they do perform correctly.

Note in Listing 1 how the list of command strings is defined. As mentioned earlier, the variable passed to the gettkn function is a pointer to an array of pointers to the strings. The definition here specifies a static (we don't want the list to be globally defined) array whose dimension will be defined by the initialization that follows ([] contains no dimension). The array name is "list", but the asterisk specifies that the elements of 'list' are pointers, and we declare that the pointers point to character data (strings, in this case). The value of the variable, list, is in fact a pointer to the array list, and it is this pointer that is passed to the gettkn function.

The initialization occurs inside the curly brackets, and defines the list that will be passed to the gettkn function. The last string entered into the list must be a null string, which in C will be represented as the single terminating character usually defined as '\0'. Note that the strings are not variables, but string constants located somewhere in memory. It is ok to reassign the pointer elements in the array, although this would of course change the list, not a desirable thing to do, but legal. But no attempt should ever be made to change the strings pointed to by these pointers. These strings are constants, not meant to be changed. If two arrays are defined, with duplicate strings, both arrays will have pointers pointing to the same string constant. Changing the string would affect not one, but both of the arrays.

An Application

After developing the input functions, let's look at a simple application for them. We'll write a simple function to compute the resistance of an undefined number of parallel resistors. First, let's define just what the function should do.

The function will accept resistor values as the input, one or more values per line. After accepting a full line of data, the resultant resistance should be displayed. Additional resistors can then be entered in parallel with the present resistance value. A null or no-input entry is allowable, and should not affect the calculation. Any erroneous entry should result in the entire line being discarded, so that we don't have to figure out which values were accepted.

The parallel resistance value will be computed from the formula:

$$R_p = R_1 * R_2 / (R_1 + R_2)$$

The calculation can be restarted with the string 'reset', but only the 'r' will be required. The function is terminated with the string 'quit', with all four letters required. In either case, we will require the string to be the only input data on a line. If additional data appears on the line, an error will be flagged.

What should we do with the end-of-file token. Well, the end-of-file token should only occur if the input is obtained from a file. In that case, the file is completed and no additional input is possible. An exit would be in order here.

What are reasonable resistor values? We won't allow negative resistances, and will limit the values to 0.1 ohm at the low end, and 10e7, or 10 megohms at the high end. This should pretty well cover the range of reasonable values.

Listing 2 is the parallel resistance demo routine. Again variables and functions are defined as static to prevent them from being passed to the linker. The only options present in the list are reset and quit. One point worth noticing is the use of the definition for the string 'INFINITY'. I have chosen a value outside of the possible range of resistance values to represent infinite data. This same technique can be used to flag data as undefined or illegal, and is another good reason to limit the input range for data.

While the numeric computations in this demo are quite simple, they do illustrate another point in favor of limited range input. It is almost impossible to obtain any type of numeric error during execution of

this routine. The only possible error is underflow, which would require an enormous number of entries.

The function can be paraphrased as follows:

```
forever
{ get a token;

  switch(token)
  {
    case 'reset':
      set resistance to infinity;
      break;

    case 'quit':
      exit to operating system;

    case ERROR:
      display error message;
      break;

    case integer or float:
      input new value(s);

      if error
      { display error message
        discard input
      }
      else
      { update resistance;
        display new value;
      }

      break;

    case EOF:
      exit to operating system;

  } /* end switch */
} /* end forever */
```

The function resist is listed in Listing 2. Resist consists of an infinite for loop, containing a switch statement for each of the input options. The switch case is determined by the first input token of each line.

During the C implementation of 'resist', error coding was added as required to complete the function, and additional sub-functions were defined.

If the string 'reset' is the only entry on a line, a message will be displayed, and resist will be reset to infinity. If other data appears on the line, an error message will be displayed.

The string 'quit', which again must be the only entry on the line will execute a goto to the quit code. This is a legitimate use of the goto statement, as it does not create difficult to follow code.

An error token will simply display an error message.

Skipping ahead to the end-of-file case, we execute a goto to the end-of-file code. The end-of-file code and quit code both fall outside of the infinite for loop, as the

code will terminate by exiting to the operating system. The end-of-file code, which simply displays a message, falls through to the quit code, which displays another message and exits. This falling through is a questionable coding technique, but accomplished the purpose.

If either a FLTVAL token or INTVAL token is received, we want to begin the calculation process. The first value, which we have already received, should be placed into the variable newresist, which represents a running 'total' of the resistance value for the data contained in this line only. Thereafter, new values are incorporated into the value of newresist. This continues so long as getreal returns a FLTVAL or NOINPUT token. Any other token will terminate the calculation. NOINPUT tokens are simply skipped, as they do not affect the calculation.

Assuming the last token received was end-of-line, the value of newresist is either placed into the variable resist, if resist was infinite, or it is combined with the non-infinite value of resist. If any other token is received, we need to display an error message. If this happens, the value of newresist is never combined with resist, in effect discarding the entire line of input.

The function diserr is used to simply display the error string passed to it, along with an additional request to re-enter the input.

The disval function simply displays the value of the parallel resistance. Because disval cannot be reached until resist has a non-infinite value, we do not have to worry about displaying the value infinity. Were such a situation possible, the following coding would be required:

```
if (resist == INFINITY)
  printf("infinite\n\n");
else
  printf("%10.2f\n\n",data);
```

Modular Compilation

We have already discussed some of the benefits of modular compilation, as well as some of the techniques involved. I would like to mention some of these again, and further discuss the proper format of a module.

While the most often quoted reasons for separate compilation are time and disk space, I feel that the requirement for a clean interface with the called functions is far more important. Hopefully, we will be able to create a module in which the only publicly defined names are those of the functions that we want access to. In the

case of our input module, this would be the three functions `gettkn`, `getreal`, and `getint`. There will be occasions where we will want to define some public variables, but this should be avoided whenever possible. Public variables tend to confuse matters, and are rarely required.

The module should consist of any functions that are closely tied together. For example, the `sin` and `cosine` functions are usually contained in a single module. Any functions called by the public functions should be contained in the module, unless they are general functions that we also would like to have public. In that case, they should be placed in their own private module for use by other functions. For example, the `sin` and `cosine` functions usually call a polynomial evaluation function, which is a general purpose function used by other mathematical functions. The polynomial function should be contained in its own module.

Functions that must share variables should be contained in a common module to prevent the use of publicly defined variables. When we start defining variables as public which are used only in support functions, we are going to end up with conflicts as we link with several support modules containing public variables that somehow got the same name. Avoid the problem before it happens.

Functions that are independent should be in separate modules to simplify their implementation, and to reduce the code size. When linking to a library, you will only link to modules required by the program. But the entire module will be loaded, so keep them small by placing independent functions in separate modules.

As pointed out earlier, any variables or functions not meant to be public should be declared as static, to prevent the names from being passed to the linker. Any functions meant to be non-public need to be declared static before they are listed as part of another function definition. Variables defined within a function will automatically be local.

And a comment about comments. I have noticed that assembly language programmers tend to over-comment their programs, while all other programmers tend to under-comment their programs. Never count on the source code to act as a description of the function. We should never have to read the C statements to decide what the function is doing. We should always be able to determine that from the comments. Consult the source code only

to determine the specifics of how we are doing it. I find that comments are in order for every little task, usually anywhere from a couple of lines to maybe a half dozen lines. Any more lines than that between comments forces some real concentration to see just what's going on.

Any operating system or compiler specific code should be avoided if possible, and well commented if not. Where I have used C80 specific code in these listings, I have provided a `define` statement to enable them. If removed, the listing should become generic, along with the limitations involved with removing the C80 specific code. Operating system specific or computer specific coding is best left in separate functions written to serve as a non-portable function. Graphics routines fall into this category.

Enhancements

The described functions are by no means perfect. I have already mentioned some known problems with the implementation, especially in the area of numeric input. But they are certainly much better than the supplied input functions. There are also some enhancements that could be added to the module.

Additional tokens could be added for specific purposes. For instance, a question mark input could return a token defined as a help token requesting information.

The expression routine could be improved. We could add the ability to calculate certain mathematical functions. The ability to search a list of mathematical function names is already present in `gettkn`. I chose not to do this for reasons of simplicity, and to prevent the requirement to link to a bunch of other modules from the math library that may never be used. We could also add the ability to refer to certain data by name, such as `pi`, and maybe even provide temporary named storage for user defined variables.

At present, the input routines should provide some real improvement over the standard `scanf` input routine. This module is to be considered public domain. I encourage you to use it, modify it, and pass it along. I hope it provides a useful addition to your C tools, and I hope the discussion provided some additional insight into the development of separately compiled function modules.

Listing 3

```

/*          INPUT: Input Functions          1/30/87

      gettkn: Get Input Token

This function is used to determine the type of input
token input from the console. A token list is supplied
as a parameter to the function which is searched for the
token type. Should the input match a token in the list,
its location in the list is returned as the token type.
Other possible token types are:

    1. No Input
    2. Integer Data
    3. Real Data or Expression
    4. Error

      getreal: Get Real Data

This function is used to obtain the value of real data
or expression entered from the system console. It also
returns a token type as define above, but no token list
is provided.

      getint: Get Integer Data

This function is used to obtain the value of the integer
data entered from the system console.

*/

/*          DEFINITIONS          */
#define C80          /* Delete if not compiled with C80 compiler */
#define TAB          9          /* ASCII Definitions */
#define SPACE        32

```

```

#define COMMA 44
#define BUFLen 30 /* Input Buffer Length */
/* Token Definitions */
#define NOINPUT 0 /* No Input Token */
#define EOF -1 /* End of File */
#define EOL -2 /* End of Line */
#define ERROR -3 /* Error Token */
#define FLTVAL -4 /* Real Data or Expression Token */
#define INTVAL -5 /* Integer Data Token */

/* V A R I A B L E S */
static int type = NOINPUT; /* Contains token type */
static int nextchar = EOL; /* Contains Next Character */
static char buffer[BUFLen+1]; /* Temp Storage for ASCII Packet */
static char *i; /* Buffer pointer */
static float result; /* Result of Expression */

/* Forward declarations */
static lookup();
static getdat();
static skipwh();
static swlcom();

gettkn(list) /* Returns token value as defined above */
char *list; /* Pointer to token list */
{
    if (type == ERROR) /* Discard rest of line */
    {
        for (; nextchar != '\n'; nextchar = getchar());
        nextchar = EOL;
    }
    if (nextchar == EOL) /* Get a new line */
        nextchar = getchar();
    skipwh(); /* Skip over white space */
    switch(nextchar)
    {
        case COMMA: /* No Data */
            nextchar = getdat();
            return(type = NOINPUT);
        case '\n': /* Return End-of-Line token */
            nextchar = EOL;
            return(type = EOL);
        case EOF: /* Return End-of-File token */
            type = ERROR;
            return(EOF);
    }
}

}
if ((nextchar=toupper(nextchar)) >= 'A' && nextchar <= 'Z')
    return(lookup(list));
else
    return(getdat());
}

static lookup(tknptr) /* Look for Token String in List */
/* This function will search the supplied token list for a match
with the token input from the console. If a match is found,
its position in the list will be returned. If no match is
found, an error value will be returned. */
int *tknptr; /* Pointer to Token String List */
{
    char *i; /* Scans Input String */
    char *t; /* Scans Token String */
    int tknctr; /* Counts Token Number */
    /* Extract the input string */
    i = buffer; /* Move it into buffer */
    do
    {
        *i++ = nextchar;
        if ((i - buffer) == BUFLen) /* Buffer is full */
            return(type = ERROR);
        nextchar = toupper(getchar());
    }
    while (nextchar != TAB && nextchar != SPACE && nextchar != COMMA
        && nextchar != '\n');
    *i = '\0'; /* Flag end of string */
    skipwh(); /* Skip over white space */
    swlcom(); /* And swallow a comma */
    /* For each token in the supplied list */
    for (t = tknptr[tknctr = 0]; *t; t = tknptr[++tknctr])
        /* Look for a match */
        {
            for (i = buffer; *i == toupper(*t) && *i; i++, t++);
            if ((i*t || islower(*t)) && !*i)
                return(type = tknctr+1); /* A match was found */
        }
    /* No match was found */
}

```

```

return(type = ERROR); /* Return an error value */

/* Forward Declarations */
static float getexp();
static float getterm();
static float getfact();
static float getnum();

static getdat() /* Get floating point or integer data */
/* This function will evaluate an expression, and return a type */
{
#ifdef C80 /* C80 Specific */
extern char errcod; /* Non-Zero if math error occurs */
#endif
type = INTVAL; /* Assume integer data */
result = getexp(); /* Evaluate Expression */
#ifdef C80 /* C80 Specific */
/* Error During Expression Evaluation */
if (errcod != 0)
{ errcod = 0; /* Reset the Error Flag */
type = ERROR;
}
#endif
}

/* Skip through rest of field */
skpwht();
swlcom();

return(type);

static float getexp() /* Return value of an expression */
/* This function will evaluate an expression, and return a token
value for integer or floating point data. Integer data must
be input as a literal with no decimal point or exponent.
Any other form of input, including an expression, will result
in a floating point token value being returned. This function
may also return an error value.
*/
{
int op; /* Storage for + or - operator */
float data; /* Temp storage for result */
}

```

```

/* This function forms the sum/difference of an arbitrary number
of terms.
*/
data = getterm(); /* Get first term */
/* Continue with additional terms */
while ((nextchar == '+' || nextchar == '-') && type != ERROR)
{
type = FLTVAL; /* An expression. Flag data as float */
op = nextchar; /* Save operator */
nextchar = getchar();
if (op == '+') /* Add/Subtract next term */
data += getterm();
else
data -= getterm();
}
return(data); /* Return result */

static float getterm() /* Return value of a term */
{
int op; /* Storage for * or / operator */
float data; /* Temp storage for result */
float divisor; /* Temp storage for divisor */

/* This function forms the product, quotient of an arbitrary
number of factors */
data = getfact(); /* Get the first factor */
/* Get additional factors */
while ((nextchar == '*' || nextchar == '/') && type != ERROR)
{
type = FLTVAL; /* An expression. Flag data as float */
op = nextchar; /* Save the operator */
nextchar = getchar();
if (op == '*') /* Multiply/Divide next factor */
data *= getfact();
else
{ divisor = getfact(); /* Divide by zero error */
if (divisor == 0)
type = ERROR;
else
data /= divisor;
}
}
}

```

```

}
return(data); /* Return the result */
}

static float getfact() /* Return value of a factor */
{
float data; /* Temp storage for result */
if (nextchar == '(') /* Evaluate a sub-expression */
{
nextchar = getchar();
data = getexp();
if (nextchar == ')') /* Swallow ')' */
nextchar = getchar();
else /* Expected ')' */
type = ERROR;
return(data); /* Return result */
}
return(getnum()); /* Return a number */
}

static getdigs(); /* Forward Declarations */
static getsign();
static movchar();

static float getnum() /* Return the value of a number */
{
float data; /* Temp storage for result of conversion */
int idigs; /* Number of integer digits received */
int mdigs; /* Total number of mantissa digits received */
int edigs; /* Number of exponent digits received */

float atof();
i = buffer; /* Set up Buffer Pointer */
getsign(); /* Get/Swallow sign */
idigs = getdigs(); /* Get integer digits */
if (nextchar == '.') /* Get fractional digits */
{
type = FLTVAL; /* Data is floating point */
movchar(); /* Get decimal point */
mdigs = idigs + getdigs(); /* And total mantissa digits */
}
else
mdigs = idigs; /* Set total digits */
if (mdigs == 0 || mdigs > 8) /* Require 1 to 8 digits */
}

}
type = ERROR;
return;

if (toupper(nextchar) == 'E') /* Get Exponent */
{
type = FLTVAL; /* Data is floating point */
movchar(); /* Get 'E' or 'e' */
getsign(); /* Get/Swallow exponent sign */
edigs = getdigs(); /* Get exponent digits */
}
if (edigs == 0 || edigs > 2) /* Too many or few exponent digits */
{
type = ERROR;
return;
}
if (edigs == 2) /* Calculate the exponent value */
if (*(i-3) != '-' && 10 * (*(i-2)-'0') + *(i-1)-'0' + idigs > 37)
{
type = ERROR; /* Exponent is too big */
return;
}
}
*i = '\0'; /* Terminate string and convert */
data = atof(buffer);
return(data); /* Return result */
}

static movchar() /* Move a character to the buffer */
{
if ((i - buffer) == BUFLen) /* Buffer is full */
{
type = ERROR;
nextchar = getchar(); /* Swallow Characters */
}
else /* Move a character into the buffer */
{
*i++ = nextchar;
nextchar = getchar();
}
}

static getdigs() /* Move digits into the buffer */
{
int digs; /* Number of digits moved */
for (digs = 0; nextchar >= '0' && nextchar <= '9'; digs++)
movchar();
return(digs);
}

static getsign() /* Move/Swallow sign character */
{
}

```

```

if (nextchar == '+') /* Swallow a plus sign */
    nextchar = getch();
else if (nextchar == '-') /* Move a minus sign */
    movchar();
}

```

```

static skipwh() /* Skip over white space */
{
    while (nextchar == TAB || nextchar == SPACE)
        nextchar = getch();
}

```

```

static swlcom() /* Swallow a comma */
{
    if (nextchar == COMMA)
        nextchar = getch();
}

```

```

getreal(data, min, max) /* Get floating point data */
float *data; /* Pointer to result storage */
float max; /* Maximum allowable value */
float min; /* Minimum allowable value */

```

/* This function will accept the value of a floating point number or expression and return it at the location data. A token value is returned by the function, and may be any of the values returned by getch except INTVAL. Should getch return INTVAL, float data will be returned, and getreal will return a FLTVAL token value. If the data does not lie within the range defined by min and max, getreal will return an ERROR value, and the value of the variable will not be changed.

If getch has just been called and returned a INTVAL or FLTVAL token, getreal will return the value of that data. Otherwise, it will call getch to obtain a value.

```

{
    if (type != FLTVAL && type != INTVAL) /* Data is not ready */
        type = getch("");
    if (type != FLTVAL && type != INTVAL) /* Data was not received */
        return(type);
    if (result < min || result > max) /* Data is out of range */
        return(type = ERROR);
    /* Good conversion */
    *data = result;
    type = NOINPUT;
    return(FLTVAL);
}

```

```

getint(data, min, max)
int *data; /* Pointer to result storage */
int max; /* Maximum allowable value */
int min; /* Minimum allowable value */

```

/* This function will accept the value of an integer number and return it at the location data. A token value is returned by the function, and may be any of the values returned by getch except FLTVAL. Should getch return FLTVAL, an ERROR value will be returned by getch. If the data does not lie within the range defined by min and max, getint will return an ERROR value. If an ERROR value is returned, the variable will not be changed.

If getch has just been called and returned a INTVAL token, getint will return the value of that data. Otherwise, it will call getch to obtain a value.

```

{
    if (type != INTVAL) /* Data is not ready */
        type = getch("");
    if (type == FLTVAL) /* Float data is not acceptable */
        return(type = ERROR);
    if (type != INTVAL) /* Integer data was not received */
        return(type);
    if (result < min || result > max) /* Data is out of range */
        return(type = ERROR);
}

```

```

/* Good Conversion */
*data = result;
type = NOINPUT;
return(INTVAL);
/* Store the result */
/* Force a new conversion next call */
/* Return a INTVAL token */
}

```

*

**EXPLORE
NEW WORLDS
WITH
HUG
GAME
SOFTWARE**



C__Power — Part 8

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

The last C__Power article presented the reader with a utility for use under the CP/M operating system, a rather useful function that enabled memory examination. Although I did not realize it at the time of writing, that article marked the end of my involvement with CP/M except as may be required as a follow up on preceding articles. One reason for my continued preoccupation with CP/M was a dissatisfaction with MS-DOS, or should I say a blase' attitude toward the programming tools I had available to use under MS-DOS.

When I converted the data base program which has been the nucleus for this series to run under MS-DOS using Brand X compiler, I found that the resulting program was slower, larger and required quite a few changes just to get it to run at all. Hardly a serious contender for program of the year. After spending many hours in constructing the screen management routines, rewriting the string handling functions and finally getting the disk I/O to work, I discovered Turbo Pascal.

Eureka, the answer to many of my problems, or so I thought. I soon converted several programs to compile under Borland's Pascal. They proved to be fast in execution but I missed being able to utilize some of the routines I had developed earlier. Not to mention the inherent power of "C". Then I read about a new compiler: Turbo C. Had I finally stumbled on the answer? Would Borland's new "C" compiler provide me with the kind of performance I was looking for? Was this the MS-DOS development tool many people were searching for? These questions would be answered in a very emphatic way soon after the release of the new compiler, but . . . many who were eagerly awaiting the arrival of Turbo C would simply have to wait a little longer.

I purchased my copy of Borland's (then) latest release near the end of July after many queries to my dealer on the subject of Turbo C availability (I think he was getting tired of hearing me whine). I rushed home and proceeded to transfer the files to my hard drive after creating the subdirectories needed. The next job was to get the screen handling routines and string functions working in their new environment. Easier said than done! I was beginning to wonder if frustration was making my hair fall out or just old age. After adding some very profound words to my vocabulary, my patience (?) was rewarded. I could clear the screen and position the cursor, even `gofor(string,numchar)` was working properly. More on this later.

I was sure now that I would have the data base program working in a matter of hours. I guess that shows how naive I still am. It seems that the disk I/O functions were not working at all the way they were written for my brand X compiler. Borland's documentation on Turbo C seems to be remarkably complete and very helpful in most areas, but either I'm ready for a chapter eleven in mental competency or the material covering disk I/O is a little vague. Perseverance finally paid off and I was able to convert the program to compile and run under Turbo C.

The big question foremost in my mind at this point - was it worth the wait, the frustration, the effort? The answer: a resounding YES. Every facet of the program's performance was improved. The speed of execution in the search mode, already fast, was now phenomenal. The sort, which did admittedly undergo some changes, was now very fast. I was able to read one hundred records from disk, sort them in memory and write the sorted file back to disk in eighteen seconds (using

Ramdrive). That's quick!

Now let's take a good look at this compiler. How easy is it to use? Is the function library extensive, is it compatible with the standards set forth by K & R? What computers is it available for? How many of my hard earned dollars will I have to part with to purchase this latest Borland product? Will I have to purchase other utilities to expand its capabilities?

Let's answer the last question first. The only utility that you might need, if you are contemplating the inclusion of inline assembly routines, is Microsoft's MASM. Everything else is included in the package from Borland, right down to an 8087 emulator. The good news here is: yes Virginia, you can include inline assembly code.

Those people who have been wanting to learn "C" but were intimidated by its complexities and the many operations required to produce an executable file can now break into their piggy banks and buy Turbo C. It is much easier to use than other implementations, one reason being the integrated environment provided by Borland. You can invoke the compiler, use the integrated editor to write the source code, compile the result and test your pride and joy without having to exit the compiler. Another nice touch: any errors encountered by the compiler are listed in order and the cursor is placed at the location of the first mistake, ala Turbo Pascal. You can even utilize the DOS shell and invoke DOS commands from within this environment. Borland's own linker (included) makes a large contribution to the ease with which this package can produce a useable program. Also provided: a separate compiler for those who prefer the more conventional method of source code compilation.

All of this ease of use is of little import if the compiler suffers from an abbreviated library. This is not the case with Turbo C. The functions available are extensive and they are Ansi "C" compatible. For those of you who have deep pockets, the source code is available for the run time library routines. Being somewhat parsimonious, I have not availed myself of this feature.

Turbo C, to my knowledge, is available only for IBM PC's and compatibles. The user will need two floppy drives or one floppy and a Winchester. The minimum memory and DOS requirements are: 384K of ram and MS-DOS version 2.0 or later. Probably one drive would suffice if all of the needed files were transferred to the working disk, however I think this scenario would tax my patience past the breaking point.

As for putting the Piggy Bank at risk, the cost of Turbo C is quite nominal, in fact very reasonable, considering the power and versatility you are buying. The retail list price is \$99.95 and it is my belief that you can't buy a better "C" compiler at any price.

If you have already become a "C" user, you have found that the clear screen commands and cursor location routines are missing from your compiler. Here is an area, just as you are getting started in writing for your new compiler, where you get the opportunity to employ some C__Power. You will need to develop some routines to accomplish these tasks on your computer. This is no small job for the uninitiated and if you go to your local bookstore in search of some help on this score you will find a dearth of material on the subject. I have not discovered why this is the case but I should be an expert on the subject since I've written these routines for three different compilers and three different computers. For some reason (I think it is called obtuseness), I had nearly the same degree of difficulty the third time as I did the second. Fortunately what I lack in mental acuity I make up in perseverance and I was finally rewarded with routines that clear the screen, locate the cursor and even perform some unique string functions.

If you have access to either the IBM PC manuals or the Microsoft "C" documentation, this information is available to you. Since I had neither and furthermore was unaware of these

sources, I had to develop these routines the hard way (that seems to be my modus operandi). These routines are included in this article for your convenience. They will also be used in the upcoming data base program revision for the IBM PC. The source code for this update will be written for Turbo C (of course).

I'm very pleased with this program in its MS-DOS (read Turbo C) version since it out performs the C/PM version in every aspect. The only areas of revision that involved more than minor syntax changes were in the file I/O and the writerec(parm) function. These last changes were in an effort to reduce the time required for writing the sorted record back to disk. I was more successful than I anticipated, so those of you who are currently using this program under CP/M may want to incorporate these changes in your program. The next C__Power article will include a listing of the revised source code for use with Turbo C. I hope that you will find it useful both as a reference in developing your own programs and as a stand alone data base.

For those of you who tuned in late, the upcoming article will be rather brief in its coverage of the code listing since this was rather thoroughly documented in preceding articles and we will be getting on to other "fascinating facets of C" in succeeding articles of this series. The intended direction for future C__Power articles will be toward pointers and their use in linked lists. The eventual objective being a basic text editor. I would appreciate any reader feedback relating to this or any other topic having to do with "C" programming.

O.K., let's look at the listing. You will notice that I have named it "PCLIB.C". These functions are peculiar to the PC and will replace the file "FUNCLIB.C" used in the articles dealing with CP/M. The included functions include: locate(row,col) (locates the cursor at row,col), cls() (clears the entire screen), Clr_dn(row1,row2) (clears the portion of the screen between and including row1 to row2), and our old friend gofor(string,numchar) (returns a string of maxnumchar characters).

Those of you who have followed this series from its inception will notice a difference in the way these functions are implemented. Instead of escape sequences we are using a MS-DOS interrupt. This is one of the facets of "C"

that give it its power. We can change the library a little bit, make some changes to the disk I/O and use an existing program written for an eight bit compiler on a larger, more modern machine. Portable code is the term for this phenomenon and this is another area where "C" shows its muscle.

We could use escape sequences on our CP/M compiler to perform all the screen handling functions listed above, however we had to resort to inline assembly for our gofor(string,numchar) routine. I guess you could call it a trade off in ease of implementation. The locate cursor and clear screen functions in our new library all use a union declaration which in turn allows a call to DOS.h (one of the include files) where the registers are further defined within a struct(ure). Take the time to examine DOS.h. Look at the way the registers are declared. You will notice that a pair of registers (ax or bx, etc.) are declared of type int(eger) and a single register (ah or al etc.) is declared of type byte. These registers are loaded by the routine and an interrupt call is made where MS-DOS takes control. A low level (DOS) function is performed based on the value placed in the ah register and then we are returned to our calling program. Very neat and tidy.

Our friend gofor(string,numchar) was somewhat easier to implement, requiring no inline assembly or interrupt calls. In fact we were able to build this function using only a library function and a "for" loop. This was much easier than the implementation under CP/M where we had to resort to assembly language to create our string retriever. Many of you might be wondering why we need this (gofor) function at all. If you are wondering, then you tuned in late. We discussed this facet of "C" peculiarity in a previous article but I will reiterate since I feel that it is quite important to understand this personality aberration. When a character string is declared in "C", the room needed for storage is declared at the same time as in: char string[25];. Twenty five bytes of memory have been allocated for storage of this string. If, in the body of the program, we use "scanf" or "gets" to input string[25] and the user inputs more than twenty five characters, then somewhere within the deep dark recesses of your computer, a part of the memory will be overwritten. Sometimes with catastrophic results leading to a program crash at a most embarrassing moment. The use of gofor(string,numchar) can eliminate this

problem and make a large contribution towards making your programs "bullet proof".

I'm glad you joined us for C_Power, part eight. I hope you will be here for the next article when we will present the revised listing of the database program presented in previous issues, rewritten for compilation under Turbo C and altered to run on your PC (or compatible). "C" you soon.

Turbo C is a product of:
 Borland International
 4585 Scotts Valley Drive
 Scotts Valley, CA 95066

```

}
puts("");c[j-1]='\0';
}

main()
{
char string[31]; /* provide space for string + terminator */
cls();locate(4,4);printf("<- coordinates: row4, col4");
locate(10,6);printf("<- coordinates: row10, col6");
locate(16,10);printf("<- coordinates: row16, col10");
locate(18,4);printf("The next operation will clear from row 10 to row 16");
locate(20,4);printf("Enter \\"This line will appear at row 12*\\" ");
gofor(string,31);clr_dn(10,16);locate(12,2);printf("%s",string);
/* Delete "main" after testing "pclib" to use with "Turbo C" source code */
}

```



Listing

/* PCLIB.C screen and string function library for the PC. Written */
 /* to compile and run under Turbo C., using MS-DOS 2.0 or later */
 /* ***** by John P. Lewis ***** */

```

#include <dos.h>
#include <stdio.h>
#define VIDEO 0x10

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

void cls()
{
union REGS regs;
regs.h.ah=7; /* clear screen function */
regs.x.cx=(00 << 8) + 00; /* top left of screen */
regs.x.dx=(23 << 8) + 79; /* bottom right */
regs.x.bx=(15 << 8) + 00;
regs.h.al=0;
int86(VIDEO, &regs, &regs);
}

void clr_dn(row1,row2)
int row1, row2;
{
union REGS regs;
row1-=1;row2-=1; /* enable usage first row = 1 */
regs.h.ah=7; /* clears the screen between and including */
regs.x.cx=(row1 << 8) + 0; /* row one, and row two */
regs.x.dx=(row2 << 8) + 79;
regs.x.bx=(15 << 8) + 00;
regs.h.al=0;
int86(VIDEO, &regs, &regs);
}

gofor(c,i)
int i; /* retrieves a string of maxnumchars */
char c[];
{
int j;
for(j=0; j <= i && c[j-1] != 13 ; ++j)
{
c[j]=getch();printf("%c",c[j]);
if ( c[j]==8)
{
printf(" %c",8);j-=2;
}
}
}

```


**HARDWARE
HEATH/ZENITH**

SURPLUS - SALVAGE - ETC.

Most Circuit Boards for the Heath/Zenith Line, Z-241 and Z-248 CPU/MEM and I/O Boards \$75.00 Ea. Z-158 (NEW) Winchester Cont. Bb. \$75.00 Ea. CPU/MEM and Floppy Disc Cont. \$75.00 Ea. Z-151 CPU - Video - MEM \$35.00 Ea. Floppy Disc \$45.00 Ea. Z-100 8 MHz Mother Bd. \$100.00 Ea. 5 MHz Mother Bd. \$25.00 Ea. Video Bd.(NEW) \$35.00 Ea. Z-171 Main Bd. \$50.00 Ea. NEW Modem \$100.00 Ea. Thousands of Cables and Miscellaneous Items for Z-171. Z-148 CPU \$50.00 Ea. I/O Bd. \$50.00 Ea. H-89 CPU \$35.00 Ea. Terminal Logic \$25.00. Call or Write for a Price List - Continually Getting More Stock Each Month - Shipped U.P.S. - C.O.D.

Self Addressed Stamped Envelope to:
 Al Davis
 4894 Lake Chapin Road
 Berrien Springs, MI 49103
 Phone: (616) 471-1792

SAVE THIS AD!



"Why Haven't You Called?"

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

M&C

SPREADSHEET/DATABASE

Corner

Review #3

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



The readers of this series of articles will find that I have changed the title of the sequence of subjects. The new name explains what I expect to cover. Spreadsheets and Database programs are closely related and they in turn are related to word processors, windows, graphs, and nearly any other type of software. The articles will cover all of the hardware peripherals; such as, EEMS/EMS memory boards, EGA/EGA+ graphics/color boards and enhanced color monitors.

I have been using hard disks for over four years and I have had two serious crashes. The last one was a few weeks ago and lost two articles that I had prepared. With the Norton utilities and PC Tools utilities I could not recover the lost files. I was performing a hard disk backup when this took place. My first crash was from a disk platter that had flaking media. The last crash was caused by a contractor building houses above me striking an underground electric cable resulting in the electric power going on and off. The result was error messages saying no FAT tracks and no Directory track.

I decided that it was time to look into what was new in software and hardware that would prevent or repair this type of damage. I also wanted to replace my damaged disks with larger capacity disks.

It costs almost as much to have a hard disk repaired as to replace it. Repair time may take weeks and all files on the disks are lost. I found a battery type power backup in the surplus market which should solve power interruptions long enough to get the system shut down for a little over \$300. I wanted to replace the hard disk with one with 40 megabytes of formatted storage. This means that the "magic" 32 megabyte barrier had to be passed. I knew that software existed to do this. Anyway, this leads to the reason for this article.

PRIME SOLUTIONS INC.--Disk Technician Ver. 1.0 Rev. 3.97

My tests show that Disk Technician WILL repair nearly every hard disk up to 32 megabyte capacity (version 2.0 will be out this fall and will work over 32MB). I first tried DT on the crashed CMI CM-6426 with 20 megabyte with 1107 faults. The DT manual suggests that if the disk has crashed and DT cannot run the drive, do a soft format with an interleave of 2 for an AT type computer and 4 for an XT type. Then run the full monthly test 6 times, followed by alternately running the weekly and monthly tests for the equivalent test period of 4 weeks. Finally end up with another full automatic monthly test. Disk Technician showed that the disk had 1107 soft errors and 79

hard errors. Ordinarily this disk would be sent out for repairs! At the end of the above testing however, the CMI drive had over 20 megabytes of usable storage with 7 errors that were not repaired. As the drive was used and the daily, weekly and monthly automatic tests were run, these errors were repaired or blocked out. To say the least, I was very surprised! I am using that hard drive right now as I write this article.

The manual explains that fault as caused by three types of errors:

- 1--Out of alignment.
- 2--Media and distance.
- 3--Power.

Out of alignment involves head positioning. As the drives change temperature, the head positioning mechanism expands or shrinks, causing the heads to read and write in a different track area. Add to this friction changes, wear-and-tear, play, repeatability, vibration, irregular platters and surfaces, etc. This is an ongoing process that gradually ends up in a continual, serious error. Valuable data may be written so far off its track that the head cannot read it, resulting in lost data and ERRORS. To overcome this problem, Disk Technician must be used daily to keep monitoring these changes so that it

can keep everything in alignment. If DT finds the slightest change, it can safely remove the data and store it in RAM while it restores the alignment by doing a low level, real factory format of the track. It next retests the new track and only if it finds the track perfectly repaired, it then records the data from RAM to the track. If the retest reveals that any spot on the track is not perfect, it relocates the data to a NEW track that it has tested and safely blocks out the bad track for DOS use. This is all a totally automatic, unattended process! The typical distance between the heads and the media is about 25 millionth of an inch. For comparison, a smoke particle is about 10 times larger and a human hair is over 100 times larger. Again the expansion and contraction caused by normal heating and cooling, wear-and-tear, platter wobble, etc., all vary this spacing.

The disk read and write quality varies directly with the head to media spacing. Even a new drive has variations in coating thickness, smoothness, and magnetic qualities. These track conditions can change overnight. Disk Technician reads, writes, and tests every single sector, occupied or not, using special proprietary tests and repair algorithms. DOS and other processes can only tell about problems when 10 to 30 errors occur. Disk Technician reveals a possible fault when 1 or 2 errors are found. With regular use, these errors are continually monitored and DT will repair them as previously described!

Electric power on and off surges, brownouts, spikes, etc. can all cause "garbage" to be written by the heads into tracks wherever the heads happen to be. Computers are being built with better line filtering devices and have minimized these problems, but the problems can still occur if the conditions are bad enough. They can cause major crashes such as my last crash! Disk Technician has a new approach to this type of problem. It automatically installs "SAFE PARK" as a memory resident software program and creates a "safe zone" on the hard disk. Whenever you boot from the hard disk, Safe Park becomes memory resident and operates transparently with all the programs you run all of the TIME! When there is no disk activity for 7 seconds (or whatever time you choose between 1 to 15 seconds) it will move the heads to the safe zone. Once the heads are moved, any possible damage would be limited to the safe zone and the data protected.

Therefore, if you use Disk Technician for the first time and it runs the Monthly Automatic Test and the regular weekly and daily automatic scheduled tests, you will be fully protected. It takes less than 3 minutes for the daily test, less than 60 minutes for a weekly test, and 3 to 5 hours for a monthly test. This breakthrough artificial intelligence system will automatically PREVENT the majority of disk failures if not all! Best of all, it does this without losing or damaging the data on the disk. I do not know about some of the copy protection methods. I uninstall these before the tests and I still backup my more important files. However, I believe I will always live dangerously. Disk Technician should work with all files because it simply reads, writes every single byte on the disk, occupied or not, and puts it back on only error free tracks. I really like this program and I believe every hard disk user needs such a program.

Disk Technician will work on any single system PC, XT, or AT type computers. It will work on drives or partitions up to 32 megabytes. The user can now find good buys on used hard disks. With a little looking I found ATASI-3046 with 39MB, 30ms, full height refurbished drives that sold originally for list \$1600 and now for as low as \$275 to \$449. I also found CMI-3426 20MB for \$229 and CMI-6426 39MB drives for \$275. I found LAPINE-Titan 3.5 inch, 20MB in a 5.25 inch frame for \$199. I purchased the ATASI- 3046 for \$275 and the CMI-6426 for \$275. Both of these drives are fully operational after running them with Disk Technician several days in a row. Therefore the Disk Technician cost of \$99.95 fully paid for itself if you need a hard disk drive!

I obtained a copy of the Computer Hot Line Weekly from a friend; however, you can get information from:

Hot Line, Inc.
Computer Hot Line Weekly
Box 1373
Fort Dodge, Iowa 50501
800-247-2000

Who knows, you may find other items of interest as well! This is where I found the sources for the disks that I purchased.

STORAGE DIMENSIONS--SpeedStor V. 5.00b & SpeedCache V. 1.02

SpeedStor is a PC Hard Disk Utility Software for integration, partitioning, and diagnostics. SpeedCache is a performance enhancement program for MS-DOS. With

SpeedStor the user can:

- 1--Integrate virtually any drive with ST506/412, ESDI, SCSI or RRL interface (you may not know of some of these now but they are coming).
- 2--Add 10mb to 320mb (and more) of disk storage to the XT or AT type computer.
- 3--Overcome the 32mb DOS limitation and install partitions up to 1024mb.
- 4--Boot from the hard disk.
- 5--Improve data throughput as much as 30%.

SpeedStor consists of four major modules:

- 1--HardPrep-an executable program for initialization and diagnostics.
- 2--PartEd-an executable program to create and format DOS partitions.
- 3--HarDrive.SYS-an installable device driver that allows DOS to use high capacity drives.
- 4--Automatic installation of several files on the SpeedStor diskette that use automatic Batch routines.

SpeedStor is the easiest software for a beginner to use. The INSTALL batch file will do the complete hard disk installation with the user adding simple information. The experienced system integrators can execute HardPrep and PartEd and operate the programs manually to use the advanced features that are available to improve system performance, storage efficiency, or system flexibility. Here is a brief list of features:

HardPrep

- 1--Handle flexible initialization options.
- 2--Use comprehensive diagnostics for quick and easy fault solutions.
- 3--Select drive type from a large internal disk type table.
- 4--Lock out bad tracks without reformatting drive.
- 5--Park drive heads before moving computer or drive.
- 6--Use media analysis to detect additional media defects.
- 7--Use bounded initialization for different interleaves in each partition.
- 8--Change interleave without destroying disk data.

ParteD

- 1--Create and format up to eight fully dos compatible partitions.
- 2--Create partitions as large as the drive capacity.

- 3--Create read-only partitions. Prevents valuable data from accidental erasure or illegal access.
- 4--Span two or more drives up to eight with a single partition.

HardDrive.SYS

- 1--Makes SpeedStor transparent to the user or programs.
- 2--Gives DOS the power to handle high capacity drives.
- 3--Improves system performance by as much as 30%.
- 4--Supports read-only partitions.
- 5--Boots up with DOS.

To sum up, SpeedStor with its batch file is the easiest way to handle large capacity drives. It will not repair a really bad hard disk drive.

Golden Bow Systems--Vfeature & Vfeature Deluxe

Vfeature Deluxe is the newest version of Vfeature and can be used with drives exceeding 32 megabytes. I am going to write about the Vfeature Deluxe. It will allow the computer to work with hard drive capacity up to 1 gigabyte in size. How many have this? DOS Version 3.1 or up is required. The user can choose a single large partition or subdivide into multiple partitions. Either way, the full use of DOS will be available. It provides true disk BIOS calls compatibility, allowing the use of multitasking, networking, or other enhancement products. Golden Bow Systems has gone to great lengths to provide security features. This will come out as we proceed. I will be reviewing some more advanced security methods in future articles. Vfeature security provides optional features such as password control of system access, control of diskette drive usage, and user access and copy protection for one or more partitions. This is THE system for breaking the DOS 32mb limitation!

I am going to do a brief step-through covering this software usage and features:

- 1--Install Vfeature to lock the Master Diskette to the password the user chooses. Use this disk to create a working disk. This working disk will have the Vfeature control program tailored to your system with the DOS version on it to transfer it to the hard disk.
- 2--The disk program will identify the controller board, requesting your input if it needs it. Then it will ask for a description of the drive, number of heads, and

number of cylinders. This information is from the documentation that comes with the drive or from the ROM if it is a standard drive.

- 3--Vfeature can be used to physically format the drive or partition. The "low level format" scans the disk surface to identify and flag any media defects. The additional, bad tracks, if any, can be added to the manufacturers defect list.
- 4--Now the partitions are described, up to twelve DOS compatible. Or, if desired, the user can "SPAN" two drives to hold larger files than one drive can handle. The display will tell the maximum megabytes the user can allocate. At this time the user can select the optimum cluster size (512K to 32K) to maximize access time depending on the application.
- 5--Now each partition must be DOS formatted so that they can be recognized by DOS.
- 6--Once the partitions are defined, the access to each partition can be controlled by the password the user has selected. This is very useful for sharing the computer with other users or to keep the partitioned data private. Any partition can be write protected, so a slip of the finger will not wipe out the data. How many times have you done this?
- 7--The final step is for the program to write the Vfeature installation and DOS on the hard disk. After that, it is all automatic. Whenever you boot, the Vfeature controls, along with DOS and your other programs.
- 8--The original Vfeature (not Deluxe) will not perform with disks over 32mb operations.

Vfeature provides a maximum way to create and partition any hard drives. It also provides the security system described above using a password protection to prevent data access and/or inhibit downloading to prevent data theft! Vfeature will not repair the disk errors such as Disk Technician; but, Disk Technician will not provide the hard disk control features. The regular hard disk user needs both programs.

Golden Bow Systems--Vopt

Every hard disk user will find that after a few read and write operations that the files become fragmented. This is caused when the controller board logic finds empty sectors to put part of the file into. This patching here and there to many

locations causes the files to become cluttered and disorganized. When the computer reads this file it must look up where each portion is located and read it, then look up the next part and read it, etc. This means that the hard drive heads are making many moves between tracks. This results in longer read times. The same is true for write operations. I keep reading how the readers are spending time and money to make their computer run faster. Here is a simple way to speed the throughput of the computer! This program will speed up the computer by eliminating file fragmentation. It also solves a couple of other problems. It reduces wear-and-tear and if the file is contiguous, it will be a lot easier to repair a lost file.

Thus, for faster, time saving computer operation and increased reliability, we must keep the hard disk files organized. Vopt WILL do this for you. It will eliminate the fragmented files that are slowing and wearing out the hard disk drive. Now if we set the computer to run Vopt on a regular basis, the files will always be all in contiguous order and be easy to read and write.

Vopt includes the following:

- 1--Vmap-provides a graphic display of the hard disk storage efficiency.
- 2--Vseek-provides a graphic display of the hard disk seek time.
- 3--Vmarkbad-scans the hard disk surface to flag bad clusters.
- 4--Vbench-provides a benchmark test of computer operation.
- 5--Vspeed-plots diskette drive speed and sector spacing.
- 6--Vrd-checks for and displays diskette errors.
- 7--Vprtsc-resident program to speed up Prtsc operations.
- 8--Vtscr-displays resident program status.

The key programs are the Vopt and Vmap. The other utilities are very handy. There is also one other utility that I liked. It is an upgraded CHKDSK that replaces the DOS chkdsk.

Vopt is really fast. Except for the initial organization, which takes five to ten minutes depending on the disk capacity, the test takes about one minute. I have used it every day before starting my work. It can be included in the Autoexec.bat file. Vopt uses a "best fit" algorithm with a graphic display showing what is going on with the files. It will not move a file unless it improves the disk organization.

This is a MUST software for hard disk users and a lot cheaper than a math co-processor which most computer users do not need!

Golden Bow Systems-SuperDuper

How does that name grab you? This was a utility that I have been looking for since I have been using H/Z computers. I wanted to use a single 360K floppy disk with a large capacity hard disk; BUT, one floppy drive makes DOS diskcopy useless. This program solves that problem better than diskcopy.

SuperDuper will do the following:

- 1--It makes a disk copy with one disk read. It retains this exact image of the source disk in the computer memory.
- 2--It will use the existing format on the target disk if there. If not on the disk or bad, it will reformat the disk automatically.
- 3--It will perform DOS verify or data compared copy of the source diskette with a byte-by-byte comparison to the target diskette from the source image in memory. If you wish, it will make a copy superfast (about 20 seconds per disk) that is as reliable as if DOS COPY command were used.
- 4--Multiple copies can be made from the original read because the image is still in memory.
- 5--Protection from diskette errors is provided because SuperDuper checks the target diskette for media damage.
- 6--It will work on 160/180 or 320/360kb DOS diskette. It will not work on 720kb or 1.2mb diskettes or copy protected diskettes.

I would think that many readers would like this software. It provides a full height mounting space for the low cost, fast, large capacity hard disk that I discussed under Disk Technician.

Golden Bow Systems-Vcache

As I covered above, I will be writing about Cache usage in a future article.

Golden Bow Systems-Vlock

Vlock is a software plus a hardware board with a key lock for a complete Computer Security System. It provides password protected home menus for up to 32 users. The Vlock PC board mounts in any slot that is available, long or short, 8 or 16 bit, and provides a keylock protection. Again, I will cover security systems in a future article. The security systems available with this software really impressed me and it

rates a detailed explanation. If you need such security before I can get it written, contact Golden Bow Systems.

Central Point Software-PC Tools V. 3.0 to 3.2

As always, PC Tools provides all the important DOS Utilities, only better. It has one draw back, it requires more RAM than previous versions. In this day and age, memory is less expensive and we have boards and software that permits memory to break the DOS 640K barrier. I will be discussing these in a future article as well. This software can make use of this added memory. In this article, I am going to limit my discussion to the new hard drive utilities that have been added. These are as follows:

- 1--COMPRESS.EXE is a program that will analyze a hard disk (floppies also) for file fragmentation and at your option correct it. The options are to examine the disk organization and display the results found. Another option surface scans the disk for problem areas. It takes 10 to 20 minutes. The main option performs a complex procedure, taking about 30 minutes, depending on the disk capacity, and compresses the data into contiguous files. This increases the disk storage and increases the in/out processing speed. Also, if the file is fragmented, the undelete utility may have problems rebuilding the file that might be damaged or lost. COMPRESS will not effect Hidden files, otherwise, the files will be relocated by date (newest file is last). If you have any deleted files you might want to recover, THIS MUST be done before using COMPRESS!
- 2--PCBACKUP.EXE will make floppy diskettes archival copies (backups) of the hard disk files.
- 3--PCRESTOR.EXE will restore the files back on the hard disk NOTE:-(2) & (3) features:
 - a--Formats the diskette automatically during the backup.
 - b--Backup all files, or selected files, or only altered files since the last backup.
 - c--Restore all files or selected files.
 - d--Verify that the backups are identical to the hard disk files.
 - e--Works with 3.5 inch 720K or 1.44mb drives and 5.25 inch 360K or 1.2mb drives.
- 4--MIRROR.COM backs up the current FAT and directory tracks each time it is run. It should be done daily. It can be automatic by including the command

in the Autoexec.bat file.

5--REBUILD.COM combined with MIRROR.COM provides an improved degree of protection against accidental ERASE, RECOVER, or FORMAT of the hard disk. It does this by keeping a backup copy of the FAT and Directory in a special hidden file on the hard disk. This should provide help for the REBUILD utility to work on lost files.

For a very low cost, every hard disk user should have this PC Tools version. Disk Technician CANNOT run with Mirror on the same disk! I always worry about utilities that move files around on the hard disk. The user should always keep the important files on the hard disk backed up anyway. I am one of the lax users and I have paid my dues twice!

SPECIAL!---VuTek Systems EGA Deluxe Upgrade Kit

This item has nothing to do with this article's subject--Hard Disk! BUT, I believe that it is too important to KEEP! I know that many readers are using the H/Z-200 series with an EGA board with a resolution of 640 x 350 with 25 lines and 80 column text. This low cost kit comes with software drivers to work with Lotus 1-2-3 and Symphony along with AutoCad and MS-WINDOWS that will provide the latest resolution of 640 x 480 with 120 column text, 752 x 410 or 896 x 350 graphics that the new Enhanced EGA boards are doing. This saves the cost of a new board. You do need to purchase a high resolution multiscan type monitor. This board will also provide 35, 43, & 50 lines of text. The kit comes with a small printed circuit board that plugs into the "feature female plug" on the older EGA boards. I bet a lot of the readers wondered what they would use that for! It is an easy installation and it can be purchased for \$69.95, plus shipping and handling direct from:

VuTek Systems, Inc.
10855 Sorrento Valley Road
San Diego, CA 92121
(619) 587-2800

I will be writing more about the enhanced EGA boards and Monitors in another article telling who needs these and what they will do for the user.

CONCLUSION

- 1--Disk Technician Version 1.0 Revision 3.97
Prime Solutions Inc.

1940 Garnet Avenue
 San Diego, CA 92109
 (619) 274-5000
 Price-\$99.95

2--SpeedStor & SpeedCache
 Storage Dimensions
 981 University Avenue
 Los Gatos, CA 95032(408) 395-2088--
 Mark Jones, Sales Representative
 Price-\$79.00 for both

3--Golden Bow Systems
 2870 5th Avenue-Suite 201
 San Diego, CA 92103
 (619) 298-9349
 Vfeature & Vfeature Deluxe-\$80.00 &
 \$120.00
 Vopt-----\$49.95
 Vcache-----\$49.95
 Vlock-----Not Priced as Yet
 SuperDuper-----\$49.95

4--PC Tools
 Central Point Software
 9700 SW Capitol Highway, #100
 Portland, OR 97219
 (503) 244-5782
 Price-\$39.95

*

ATTENTION!

Z-100 FANS

If you own a Heath/Zenith Z-100 Computer and plan to continue using it for a while, we would like to have you on our mailing list. We are developers of innovative, quality software for the Z-100. You're familiar with these popular software products...

DOODLER-V Graphics Package
MOUSE PACK Mouse Driver
ScreenPro Memory Resident Utility

We are continuing to develop new software and offer new accessories for the Z-100. Write to us or return the reader service card today. We will send you our **FREE CATALOG** and make sure you are the first to know about new Z-100 products.



Software Graphics Tools
3620 Amazon Drive
New Port Richey, FL 34655
Phone 813-376-5457

Continued from Page 56

There are times when a FAST or SLOW program is definitely needed. Specifically, when you want to slow a program down and ensure that (iii) doesn't speed it up again. One such case is FASTBACK(tm), others include games which use the joystick (very timing dependent). Such programs may use a diskette and thereby have (iii) restore things. One option is to throw the toggle switch, the other is to use my SLOW and FAST programs which actually call this resident program via INT 13 (ah=87) to order a speed change.

Enjoy! All I ask in return for making this available is that SPEEDUP.ARC is always circulated all together (so everyone is assured of getting all the right pieces). And, if you modify anything, please circulate it under a DIFFERENT name (SPEEDUP2, etc.)

- James R. Reinders

```

code segment
assume cs:code, ds:code
fast equ 1Fh ; byte to select fast speed
slow equ 17h ; byte to select slow speed
settled equ 0C0h ; I/O address of CPU LEDs
sphigh equ 78 ; keypad +
splow equ 74 ; keypad -
prtsc equ 55 ; prtsc
ff equ 12
keyint equ 16h
prnint equ 17h
dos equ 21h
tsr equ 27h
;
;
;
dataA equ $
off9 dw ? ; Old INT 09h address
seg9 dw ?
off13 dw ? ; Old INT 13h address
seg13 dw ?
retoff dw offset ret13
speed db fast ; speed = fast or slow
dataB equ $
;
main org 0100h
proc far
jmp place
;
keyin org 0100h + dataB - dataA
proc far ; This is the new INT 09h
sti
push ax
in al,60h

```

```

cmp    al,prtsc      ; check for PrtSc
jnz    chkspd

mov    ah,2
int    keyint
test   al,8
jz     done1

push  dx
xor    dx,dx
mov    ah,2
int    prnint
test   ah,80h
jz     done2

mov    al,ff
xor    ah,ah
int    prnint

done2: pop    dx
       jmp    short done1

-----
chkspd: cmp    al,sphigh      ; Check for high speed key
       jz     altchk
       cmp    al,slow
       jnz    done1

altchk: push  ax
       mov    ah,2
       int    keyint
       test   al,8
       pop    ax
       jz     done1
       cmp    al,sphigh

       mov    al,fast
       jz     speedy
       mov    al,slow
       ; High speed
       ; Low speed

speedy: out    settled,al
       mov    cs:speed,al

done1:  pop    ax
       jmp    dword ptr cs:off9      ; finish up with old INT 09h

keyin  endp

-----
dskio  proc far
       sti
       test  dl,0feh
       jnz  nodwn
       ; will trap only drives 0 and 1
       ; anything else passes untouched

       cmp    ah,3
       jz     dwn
       ; write

```

```

; verify
ah,4
dwn
; set speed (special for SLOW.COM
; and FAST.COM)
; format
nodwn

dwn:  push  ax
       mov   ax,slow
       out  settled,al
       pop  ax

       push ax
       ; flags
       push cs
       ; retl3 seg
       push cs:retloff
       ; retl3 off

nodwn: jmp  dword ptr cs:offl3

speeds: test dl,1
       mov  al,slow
       ; ah=87: set speed
       ; dl=0: slow speed
       ; dl=1: fast speed
       jz  setspd
       mov ax,fast

setspd: out  settled,al
       mov  cs:speed,al
       ret  2

retl3:  push ax
       mov  al,cs:speed
       ; returned to from old INT 13h
       out settled,al
       ; if speed was programmed to
       ; slow for some diskette access.
       pop  ax
       ret  2

dskio  endp
;
place: mov  ax,3513h
       int  dos
       ; get INT 13h

       mov  offl3,bx
       mov  segl3,es
       ; save it

       mov  dx,offset dskio
       mov  ax,2513h
       int  dos
       ; install our INT 13h

       mov  ax,3509h
       int  dos
       ; get INT 09h

       mov  off9,bx
       mov  seg9,es
       ; save it

       mov  dx,offset keyin
       mov  ax,2509h
       int  dos
       ; install our INT 09h

       mov  dx,offset place
       int  tsr
       ; terminate and stay
       ; resident

; main
code   endp
end    ends
end    main

```

*



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.

Jim Buszkiewicz
HUG Managing Editor

BV Engineering, author of the original "Plotter Driver Program", has just announced their release of a new product called "LCFIL". LCFIL is a stand-alone CAD program for IBM compatible PC's that provides design solutions for lowpass, highpass, and bandpass filters having up to 21 poles. You can specify input impedance, inductor Q's, and other parameters. Arbitrary component values may be entered into a general purpose building block enabling you to analyze many other L-C filter topologies.

LCFIL computes filter magnitude, phase, and delay characteristics as well as providing normalized and actual component values. Both linear and logarithmic frequency steps are selectable. Optional modules available from BV Engineering provide for CGA, EGA, and Hercules compatible graphical representation and drive 30 different popular pen plotters. An optional signal processing module works with LCFIL to provide transient analysis capability. Inputs are free format with liberal error trapping. The price is \$95, and for more information, contact Mark W. Hart, Marketing Representative, BV Engineering, 2200 Business Way, Suite 207, Riverside, CA 92501, or call (714) 781-0252.

Recently, MicroPro International released a new WordStar update for CP/M users. The new WordStar, CP/M Edition, Release 4 is loaded with over 120 enhancements, making it the best CP/M word processor available. This new release still retains its classic look and commands and reads files created with all previous versions of WordStar for CP/M. WordStar, Release 4 for CP/M is currently available on generic 8" format disks. Heath/Zenith users who do not have 8" drives, would have to have the files transferred to their specific format. For more information, contact

MicroPro International Corporation, 33 San Pablo Avenue, San Rafael, CA 94903, or call (415) 499-1200.

BV Engineering is announcing the release of a digital Logic Simulator Program called LSP. LSP will compute the resulting binary output signals of a digital circuit given a description of that circuit, device delays, and its inputs. LSP contains built-in models for combinatorial gates such as AND, OR, NAND, etc., sequential devices such as D, JK, and toggle FLIP-FLOPS, as well as TRI-STATE devices. Signals, inputs, and output nodes may be defined by common and easily-remembered names. Each signal can be assigned a delay time ranging from 1 to 255 user-defined time units. LSP handles all time scheduling and accurately propagates the input and computed output results through the design regardless of the complexity or nesting of feedback loops. The output of LSP is a timing diagram, or truth table, showing the binary states of each selected signal as a function of time. Input and output data may be saved to data files for future use. LSP retails for \$95.00 and for more information, contact BV Engineering Professional Software, 2200 Business Way, Suite 207, Riverside CA, 92501, or call (714) 781-0252.

The Hogware Company has introduced two new software utilities for the H/Z-100; CGAGRAB and CONVERT for use with ShowOff, the high resolution graphic editor for the H/Z-100.

CGAGRAB is a screen capture program that will capture a graphic screen from a PC-compatible or PC-emulator board. CGAGRAB will capture both 640 x 200 (2-color) and 320 x 200 (4-color) graphics so they can be loaded into ShowOff and enhanced with high resolution text, color and graphics.

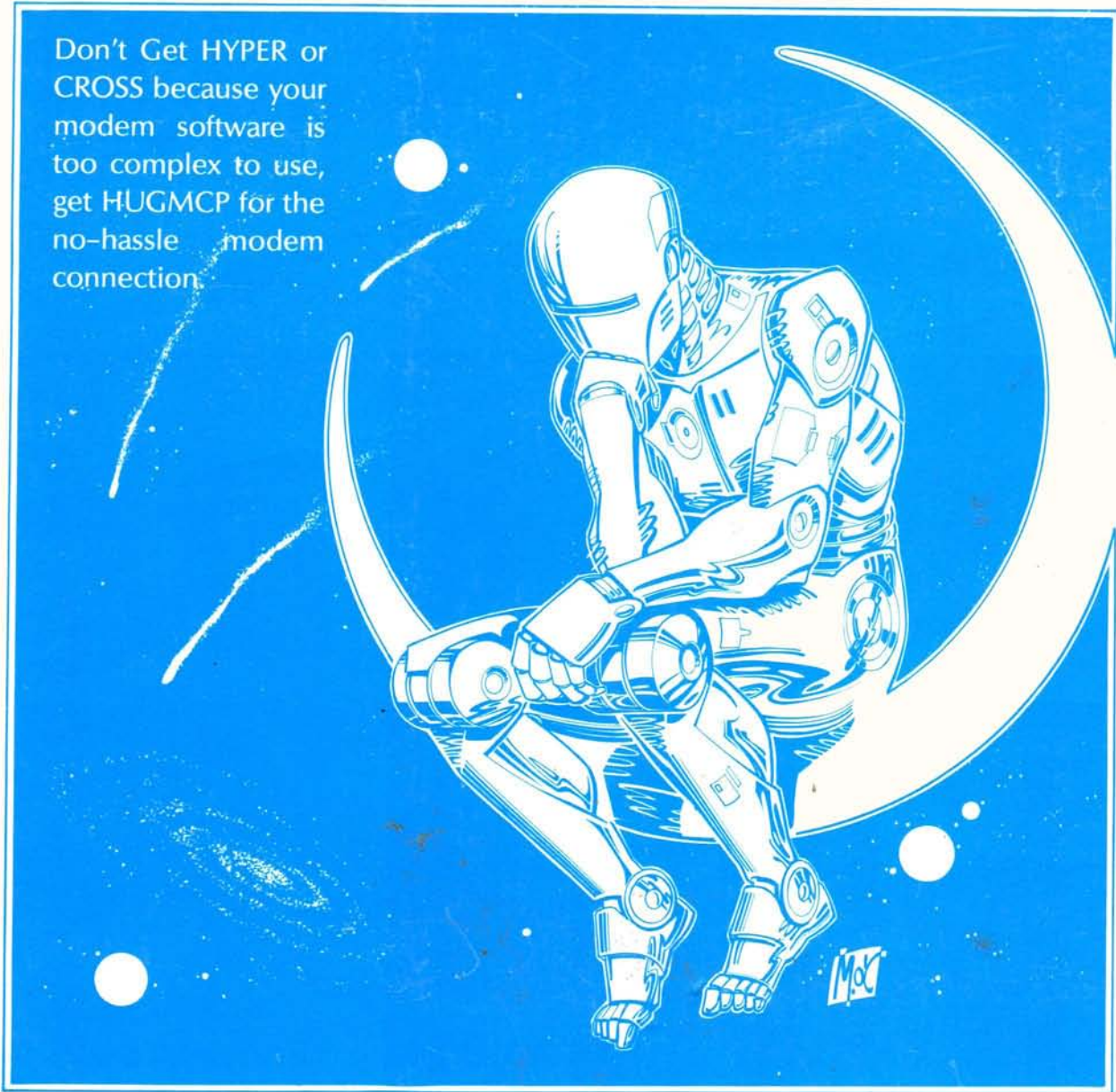
CONVERT will take a high resolution ShowOff picture and convert it into the MacIntosh format.

These utilities are on the new ShowOff Art/Utility disk which also includes printer drivers for Epson and Hewlett-Packard laser printers. This disk retails for \$15 and is available from Hogware Co., 470 Belleview, St. Louis, MO 63119, or call (314) 962-7833 for more information.



**EXPLORE
NEW WORLDS
WITH
HUG
GAME
SOFTWARE**

Don't Get HYPER or
CROSS because your
modem software is
too complex to use,
get HUGMCP for the
no-hassle modem
connection.



Heath/*ZENITH*
Users'
Group

Hilltop Road
Saint Joseph, Michigan 49085

BULK RATE
U.S. Postage
PAID
Heath Users' Group

POSTMASTER: If undeliverable,
please do not return.

P/N 885-2094