REMark®

Volume 8, Issue 8 • August 1987
P/N 885-2091 Issue 91

Official magazine for users of computer equipment.

HUG Discovers New (Old) 8-Bit Software

$2.50

WILDFIRE™

An INCREDIBLE PERFORMANCE IMPROVEMENT from Software Wizardry

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

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

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

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

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

Software Wizardry
THE MAGIC TOUCH

8 Cherokee Dr.
St. Peters, MO 63376-3926
(314)447-7337

Also from Software Wizardry...

For the Z-151/152/161

RamPal

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

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

Z-100 Upgrade Package Deal

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

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

8mHz Speed Upgrade

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

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

COMMITMENT

We continue our commitment to provide only the best Amateur Radio gear with the NEW Heathkit® SB-1000 and HK-232 kits.

SB-1000 Linear Amplifier

HK-232 PackKit Multi-Mode TNC

Our commitment to Amateur Radio means the Heathkit line is always expanding to meet the demands of even the most veteran ham. Our introduction of the SB-1000 Linear Amplifier and HK-232 PackKit Multi-Mode Terminal Node Controller Kits gives you two more value-packed amateur products to build and use.

The Heathkit SB-1000 Linear Amplifier Kit continues our commitment to produce the most popular linear amplifiers in the industry. Designed to operate at a full 1000 watts PEP output on SSB, 850 watts on CW or 500 watts for 30 minutes continuous on RTTY, this amp covers all bands from 160 to 15 meters including WARC bands. The SB-1000 uses a single 3-500Z tube in a high efficiency circuit for unparalleled performance at the price. Its high silicon E-I core transformer takes up less room and runs cooler. And it features a quiet computer-style fan, a stiff full-wave power supply with computer grade capaciters, adjustable ALC, and plate and load controls with smooth vernier tuning. And the SB-1000 is yours for only **$739.95.**

Consider the Heathkit HK-232 TNC. This versatile unit works on CW, RTTY, ASCII, AMTOR, both HF and VHF Packet, and now even WeFAX. You can work Packet in both HF (300 baud) and VHF (1200 baud or up to 9600 baud with external modem). Operate Morse from 5 WPM to speeds you never dreamed of, or print Weather Facsimile pictures on an Epson compatible printer. Connects to your computer through a standard RS-232 port. Connects to both your HF and VHF radios' PPT line, microphone input and speaker output. The same connections for Packet work on all other modes. Includes bar graph display to make HF tuning a breeze. Operates on 12 VDC at 750 mA with 10% ripple or less. The HK-232 is priced at only **$279.95.**

Because you build these kits, there aren't any surprises inside. And at Heath we are just as committed to you after the sale. All Heathkit products are backed by our highly respected manuals and even our technical consultation service.

Simply call Toll-Free: 1-800-253-0570 to order your kits today.

We also have 66 Heath/Zenith Computers and Electronics stores in North America. Call 616-982-3614 for the store location nearest you.

Heathkit®
Heath
Company

KB-101R1

Heath/Zenith Users' Group

REMark®

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
Hilltop Road
St. Joseph, MI 49085
(616) 982-3463

Official magazine for users of computer equipment.
Volume 8, Issue 8 • August 1987

Index of Advertisers

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

On The Cover: *Enable for the Z-100. Check out the special things it does. See pages 45 and 51 for reviews.*

Regionals To Replace HUGCON In '88

What! You still haven't registered for the Sixth International HUG Conference? May your memory-disk develop amnesia! Com'on, you've always wanted to see what one of these conferences were like. This is your last chance. THIS YEAR WILL MARK THE LAST INTERNATIONAL HUG CONFERENCE! Future conferences will be regional.

By the way, if you didn't happen to catch the changes we made to our major article program in the May issue of REMark, it just might be worth ($$$) your while to check it out.

You may have noticed last month that we've added another contributing editor to our list of credits in the front of the magazine. Harold Bauman is the new kid on the block. Actually, Harold isn't quite new. He's been writing steadily for REMark since October of 1983, when "Cobol Corner — Part I" first appeared. In later issues, Harold's focus changed to spreadsheets with his series of "Spreadsheet Corner" articles. Now, Harold is going to continue with a series of articles on using database programs, how they inter-react with spreadsheets, and actual product reviews. I would like to issue a long overdue "Thank You" to Harold for his efforts and contributions to the HUG community over the past 4 years, and "officially" welcome him to the HUG staff.

And now from spelunking department, we have uncovered several palets full, of very dusty, but still new in the box, 8-bit software. We have obtained this software at a ridiculously low (song-and-a-dance) price, and are passing these savings on to you. I realize that there are still alot of 8-bit H-8's and H/Z-89's out there, and this software could still be of use to Huggies who own them. The quantities of these items are VERY limited, and will be sold on a first come, first serve basis. When these are gone, that's it (maybe)!

Although this software is still new in the box, with full documentation, it is not supported in any way, other than we'll say that the disk is readable. This is what we have:

OS-63-2	CP/M-86 (Runs standard CP/M software)	$25
HOS-8917-2	CP/M-80 (Hard Sector CP/M for H-8's, H/Z-89's)	$30
MS-253-1	MicroSoft Basic 80 for CP/M (MBASIC)	$25
HMS-837-4	MicroSoft Basic Compiler for CP/M	$25
HMS-837-52	MicroSoft Macro 80 Assembler for CP/M	$25

For you H/Z-100 owners, CP/M-86 will also run standard 8-bit CP/M programs. The three MicroSoft packages should run on any machine capable of running CP/M programs.

These Items can ONLY BE ORDERED FROM US HERE AT HUG by calling Margaret Bacon at (616) 982-3463, or from the HUGPBBS bulletin board (if you're registered) at (616) 982-3956. These won't last long, so don't miss out, call today!

Following this editorial is the schedule of discussion groups for this year's HUGCON. This year we have the biggest selection of speakers and topics than any other year; 27 different speakers and topics!

As you can see from this month's editorial, change is inevitable. Just as the field of personal computing has changed from the 8080 and CP/M to the 8088 and MS-DOS, so must we. I resisted MS-DOS at first. In fact, everyone knows I used to be a CP/M nut. But you know, changing is not so bad! I think the upcoming changes you'll see in us and REMark are for the better. Keep watching!

Jim

-Jim Buszkiewicz
HUG Managing Editor

1987 International HUG Conference

Discussion Group Schedule

Saturday August 22, 1987

Time	Room	Subjects	Speaker
09:00 am	Ozark AB	Software Workshop	Bill Parrott
	TWA/NWO	Writing 'Terminate-And-Stay-Resident' Programs	Pat Swayne
	Continental AB	Bionics	Steve Greelish
	Eastern AB	Enable for The H/Z-100	Mary Pelagalli
	Delta AB	Hardware 'Bull' Session For Old And New Computer Users	Bruce Denton
	Forum	What's New In Three-Two (MS-DOS 3.2)	Bill Adney
	Lufthansa	The SCSI (Small computer Systems Interface) Buss	Marc Brooks
	Alitalia	Heath Company's Product Line Manager's Presentation	
		* New Products — Weather, TV, & Amateur Radio	Wayne Wilson
		* Truth, Beauty, And The Art Of Computer Construction	Denton Bramwell
		* New Products In Education	Jim Wilson
		* New Products In Computer Based Instruments	Mark Witsaman
		* What's Coming In The Buy-Sell Product Line	Jim Bungard
10:30 am	Ozark AB	Using RS-232 Ports (Without Fear)	Bob Harris
	TWA/NWO	Programming In 'C' For Efficiency	Jack Purdum
	Continental AB	Multi-User Multi-Tasking Operating Systems: XENIX	John Guenther
	Eastern AB	The New Z-386 Hardware	Jim Kelly
	Delta AB	A 2-D Graphics Editor Using A Mouse Input Device	David Powers
	Forum	Computers For Beginners	Ron Hackney
	Alitalia	Z-LAN	Gordon Reichard
	Lufthansa	High Speed File Transfers With HyperAccess	Matt Gray
12:00 pm	Ozark AB	OS-2, MicroSoft's New Operating System	Rachel Klau
	TWA/NWO	Writing Programs for MicroSoft Windows	Bill Rothman
	Continental AB	DataBases And Their Applications In Business	Esther Czekalski
	Eastern AB	Multi-User Application Software For Local Area Network	Steve Hesterman
	Delta AB	Hi-Res Graphics Using ShowOff On The H/Z-100	Janet Hirsch
	Forum	Desk Top Publishing	John Roach
	Alitalia	Public Domain Software	Bob Todd

Discussion Group Schedule (Con't.)

01:30 pm	Ozark AB	Software Workshop	Bill Parrott
	TWA/NWO	Writing 'Terminate-And-Stay-Resident' Programs	Pat Swayne
	Continental AB	Bionics	Steve Greelish
	Eastern AB	Enable for The H/Z-100	Mary Pelagalli
	Delta AB	Hardware 'Bull' Session For Old And New Computer Users	Bruce Denton
	Forum	What's New In Three-Two (MS-DOS 3.2)	Bill Adney
	Lufthansa	The SCSI (Small computer Systems Interface) Buss	Marc Brooks
	Alitalia	Heath Company's Product Line Manager's Presentation	
		* New Products — Weather, TV, & Amateur Radio	Wayne Wilson
		* Truth, Beauty, And The Art Of Computer Construction	Denton Bramwell
		* New Products In Education	Jim Wilson
		* New Products In Computer Based Instruments	Mark Witsaman
		* What's Coming In The Buy-Sell Product Line	Jim Bungard
03:00 pm	Ozark AB	Using RS-232 Ports (Without Fear)	Bob Harris
	TWA/NWO	Programming In 'C' For Efficiency	Jack Purdum
	Continental AB	Multi-User Multi-Tasking Operating Systems: XENIX	John Guenther
	Eastern AB	The New Z-386 Hardware	Jim Kelly
	Delta AB	A 2-D Graphics Editor Using A Mouse Input Device	David Powers
	Forum	Computers For Beginners	Ron Hackney
	Alitalia	Z-LAN	Gordon Reichard
04:30 pm	Ozark AB	OS-2, MicroSoft's New Operating System	Rachel Klau
	TWA/NWO	Writing Programs for MicroSoft Windows	Bill Rothman
	Continental AB	DataBases And Their Applications In Business	Esther Czekalski
	Eastern AB	Multi-User Application Software For Local Area Network	Steve Hesterman
	Delta AB	Hi-Res Graphics Using ShowOff On The H/Z-100	Janet Hirsch
	Forum	Desk Top Publishing	John Roach
	Alitalia	Public Domain Software	Bob Todd

Sunday August 23, 1987

09:00 am	Ozark AB	Software Workshop	Bill Parrott
	TWA/NWO	Writing 'Terminate-And-Stay-Resident' Programs	Pat Swayne
	Continental AB	What's New In Three-Two (MS-DOS 3.2)	Bill Adney
	Eastern AB	Computers For Beginners	Ron Hackney
	Delta AB	Hardware 'Bull' Session For Old And New Computer Users	Bruce Denton
	Lufthansa	Heath Company's Product Line Manager's Presentation	
		* New Products — Weather, TV, & Amateur Radio	Wayne Wilson
		* Truth, Beauty, And The Art Of Computer Construction	Denton Bramwell
		* New Products In Education	Jim Wilson
		* New Products In Computer Based Instruments	Mark Witsaman
		* What's Coming In The Buy-Sell Product Line	Jim Bungard
10:30 am	Ozark AB	Using RS-232 Ports (Without Fear)	Bob Harris
	TWA/NWO	OS-2, MicroSoft's New Operating System	Rachel Klau
	Continental AB	Multi-User Multi-Tasking Operating Systems: XENIX	John Guenther
	Eastern AB	DataBases And Their Applications In Business	Esther Czekalski
	Delta AB	The New Z-386 Hardware	Jim Kelly
	Alitalia	Z-LAN	Gordon Reichard

International HUGCON87

HUG • Hilltop Road • St. Joseph, MI 49085 • (616) 982-3463

10th Anniversary Celebration!

INTERNATIONAL HEATH/ZENITH USERS' GROUP CONFERENCE

O'Hare Hyatt Regency
Rosemont, Illinois
August 21, 22, 23, 1987

Name(s): _____

_____

_____

Company: _____

Address: _____

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

Enclosed is $27.00 for each of the individuals listed above to attend the International HUG Conference being held the weekend of August 21, 22, and 23, 1987. Please send tickets along with information regarding hotel reservations and transportation.

Amt. Enclosed: _____ No. Attending: _____

For Our Information:

Which Heath/Zenith computer do you now operate? _____

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

If yes, please indicate any suggestions you may have:

_____

Special Notice To Exhibitors:

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

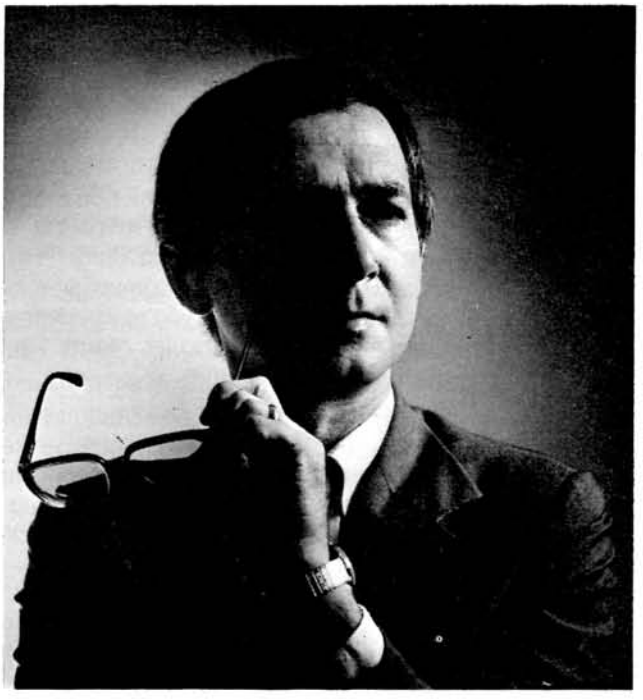
For Your Information:

The $27.00 you are paying for your reservation to the International HUG Conference entitles you to all functions of the Conference. Visitor tickets, for those of you simply attending the seminars and exhibits, are available for $12.00. Visitor tickets do not include eligibility for prizes or food while attending the Conference.

Please send your completed registration form or suitable copy to:

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

Registration(s) must be post marked no later than July 31,1987. Cancellation will not be accepted after this date.
Sorry, We cannot accept purchase orders



Mainstream Computing

Joseph Katz
103 South Edisto Avenue
Columbia, SC 29205

COMDEX in Atlanta this Spring was at a sensible time for us: June 1-4, after the University's semester ended. There were fewer major exhibitors (no Apple, Microsoft, or Ashton-Tate, for example), so all the action was confined to the Georgia World Congress Center and we could cut our stay to two days. Even so, there was a lot to see. I won't swear I saw everything, but I will affirm I left feeling I must have. You understand that it's impossible to get good demonstrations of everything, or even of everything interesting, at such an event. What I look for and depend upon is the skill of vendors in presenting immediately what their product does and why we should buy it instead of similar products — or even buy it at all.

Believe it or not, many exhibitors don't even seem to try to do that. I've seen flossy exhibits — real dazzlers, in fact — of products that are treated like carnival prizes or celebrities instead of something functional. I am dazzled. I am excited. My antennae tingle with anticipation. But I'll be damned if I can figure out from the exhibit what the product does, and I can't find anyone at the exhibit who can tell me. Or wants to. I'm an old hand at trade shows in various fields, so it took a long time for me to realize that most computer shows were different in one essential respect from all the other trade shows I've ever attended. The difference is that no one at many companies seems to have remembered to tell the people manning the booth that the idea is selling products to or through the people who visit the booth. Personality, dazzle, or arrogance don't cut it. I am not alone in drawing this conclusion, by the way.

Well, on to other things.

A few weeks before COMDEX, I got the new version of the *Mirror* communications program, *Mirror II*. The program itself is grand and the story of how it came to be is kind of interesting, with a couple of humorous elements. At COMDEX, I saw some products displayed by companies who have nice things to sell and are trying to sell them to us, and two of those products arrived shortly after my return. I got right onto them. *Bookmark* and *Referee* don't require study or demand lengthy exploration. You need to know what they do and how well they do it, then you can decide if it's what you need done. I've also got a Handy Household Hint for you, and something free. Free? Free!

Then I have to get outside and mow the lawn. I used *Miracle-Gro* on it and the green stuff is threatening to insinuate past the doors and windows. "Nephew," my Uncle Abe said to me many years ago, "you will reach a stage in life when people will gauge your worth by the grass around your house. They will expect you to spend part of your time feeding and watering it so that it will grow tall. Then they will expect you to spend more of your time cutting the grass that does grow tall. Do not fall into this craziness. It is not a sensible way for an adult male to behave. Emulate your father and brothers. Grass should be brown." Ever since then I followed my Uncle Abe's advice and had beautiful brown grass that was the pride of my family. Then this spring I fell from grace and fed the stuff. It's all turned green on me. I wonder if there's a cure. Why did I ignore my Uncle Abe's advice?

Mirror II After Judgment Day

Mirror II tickles me. In December 1985, SoftKlone Distributing Corporation began marketing *Mirror*, a lookalike and workalike of Microstuff's widely-used *Crosstalk XVI* communications program. *Crosstalk XVI* is an excellent program. *Mirror*, to my mind, was even better because it kept all of *Crosstalk*'s features and commands, added truly valuable new features and commands, and fixed some *Crosstalk* bugs that had bitten me a few times. In other words, *Mirror* was a *Crosstalk XVI* clone that really was better than the original. One of *Mirror*'s winning features was that it also cost much less than *Crosstalk XVI*: $50 instead of $195. I like telling you about ways to save money without sacrificing functionality.

Many other people apparently preferred *Mirror* to *Crosstalk XVI*, because Microstuff

soon sued SoftKlone and its parent, ForeTec Development Corporation, for copyright infringement. The case interested me because *Mirror* obviously was an "original" program, in the sense that its code was original, that just as obviously copied the "look and feel" of *Crosstalk XVI*. SoftKlone built its marketing plan on the clonish relationship and advertised it openly. Hey, look at the names of the product and distributor. There was nothing furtive, sneaky, or underhanded going on. ForeTec and SoftKlone obviously were confronting, head on, an issue that we've all been hearing a great deal about for a few years.

Then I heard that the court's decision on May 31, 1987, went against ForeTec and SoftKlone on the sole grounds that *Mirror* violated copyright on *Crosstalk XVI*'s distinctive status screen — the display of the program's current settings. (I'm grateful to Keith Ackerman of SoftKlone who volunteered a copy of the court decision.) I hoped the verdict would not be the end of *Mirror*, because the program was good and not just a cheap knockoff of something better. So when SoftKlone began advertising *Mirror II*, I wanted to take a look.

What I saw is what tickles me about *Mirror II*, and ForeTec and SoftKlone, too. The bottom line is that this *Mirror II* is even better than the former *Mirror*, just as the former *Mirror* was better than *Crosstalk XVI*, and its major advance is the status screen ForeTec designed to avoid future violations of the *Crosstalk* copyright. Right. It's not only a better screen, but it's also better in ways that actually improve the program a great deal. The status screen in *Mirror II* is more informative, clearer, better designed, and much easier to use than the *Crosstalk XVI/Mirror* status screen.

So you can follow what I'm saying even if you don't own any of these programs, let's do a brief pictorial tour through history. Figure 1 is the status screen from *Crosstalk XVI*. Compare it with Figure 2, the status screen in *Mirror* that annoyed Microstuff. Compare both of those with Figure 3, the status screen from *Mirror II* that I think much, much clearer than either of its predecessors. An important result of the copyright suit, therefore, was that ForeTec and SoftKlone improved their program in an important way. I present Figure 4 with no comment other than some chuckles I can't seem to repress. You may add your own caption, because everything I've tried is bound to get me in trouble.

Now *Mirror II* still is a *Crosstalk XVI* clone in the sense that it retains all major *Crosstalk* features and commands, all of which behave the same in both programs, except of course, in those instances where *Mirror II* has exterminated a *Crosstalk XVI* bug. My *Mirror* scripts, most of which began life as my *Crosstalk XVI* scripts, work just as they had before.

There also are many enhancements in *Mirror II*. Some seem cute but essentially frivolous, then turn out to be curiously useful: with eight ALarm tones now available, for example, I'm able to build distinctive signals into my scripts so I can work across my office until *Mirror II* summons me back to the keyboard. The horsetrack bugle call that announces the start of a race — ALarm 4 — announces that *Mirror II* has successfully logged onto CompuServe in my CSERVE.XTK. In case you're wondering why I chose that particular ALarm, you ought to clock my sprints.

Among the many enhancements in *Mirror II* are several that dramatically increase the program's power. Some of these concern communications protocols: *Mirror II* supports single file XMODEM (the canonical XMODEM protocol used on bulletin board systems), multifile XMODEM (useful in transferring batches of files between two computers), YMODEM (used on some bulletin board systems), Hayes (compatible with the Hayes Smartcom program), *Crosstalk* (of course), and *Kermit* (especially useful when transferring files with minicomputers and main frames). Other important enhancements are additional terminal emulations: ADDS Viewpoint A1, Lear Siegler ADM-3A, ANSI (as used with IBM

```
                    CROSSTALK - XVI Status Screen                On line

NAme   CROSSTALK defaults (Hayes 1200)          LOaded   A:STD.XTK
NUmber                                          CApture  Off

Communications parameters                       Filter settings
SPeed 1200    PArity None    DUplex  Full       DEbug    Off   LFauto  Off
DAta  8       STop   1       EMulate None       TAbex    Off   BLankex Off
POrt  1                      Mode    Call       INfilter On    OUtfiltr On

Key settings                                    SEnd control settings
ATten  Esc                   COmmand ETX (^C)   CWait    None
SWitch Home                  BReak   End        LWait    None

Command?
```

Figure 1. Crosstalk XVI status screen.

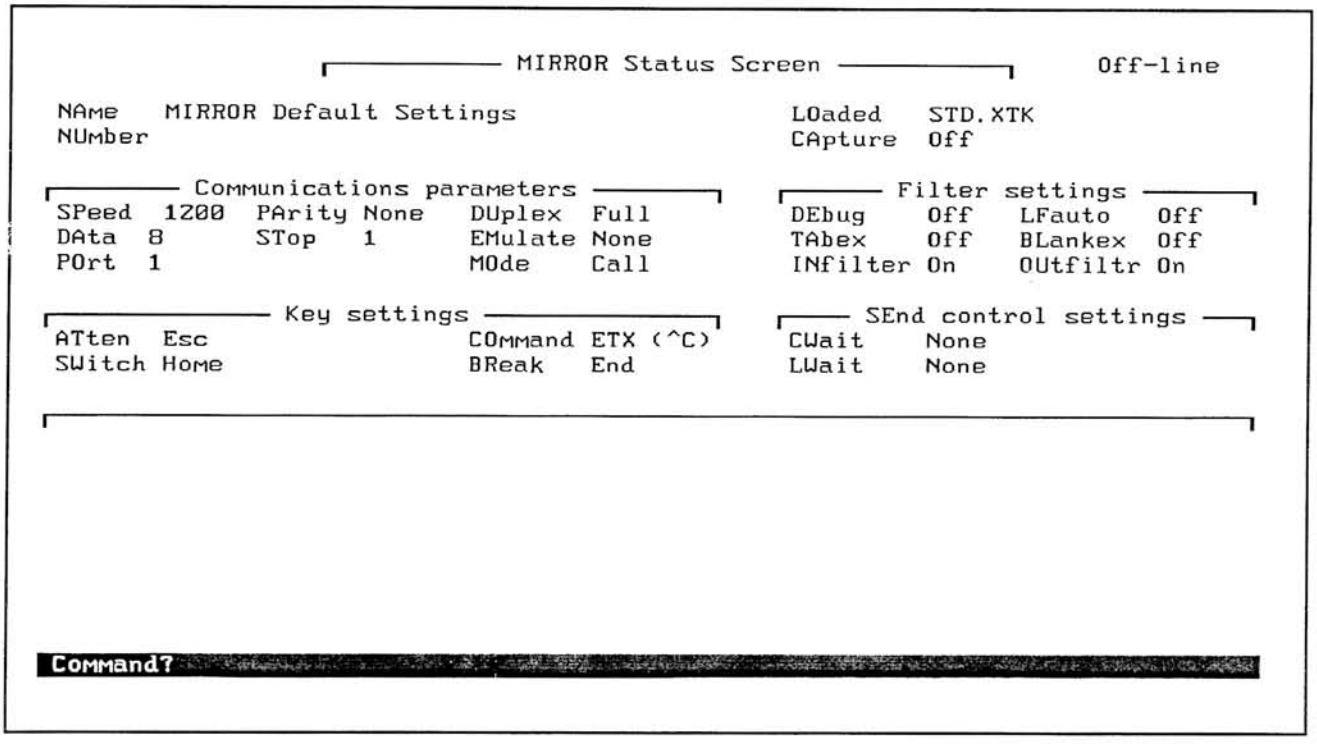


Figure 2. Mirror's status screen, which resembles Crosstalk XVI's.

bulletin board systems that allow ANSI graphics), Data General D210, IBM 3101, Texas Instruments 940, Televideo 912C/920C and 925, DEC VT100 and DT52. Those things, of course, follow the paths of growth usual in mature communications programs.

What I particularly like are conveniences I use often. Use the command "MIRROR / CSERVE" (note the solidus surrounded by spaces) and you bypass the screen in Figure 4, which makes things move a bit more quickly and cleanly. (You can bypass that screen also with the command "MIRROR /" if you don't want to invoke a script from the command line.) The "IF" command now allows a script to branch depending on the existence of a disk file, a useful feature because a new "ReTurn" command makes possible the nesting of script files. A nice example of the capa-

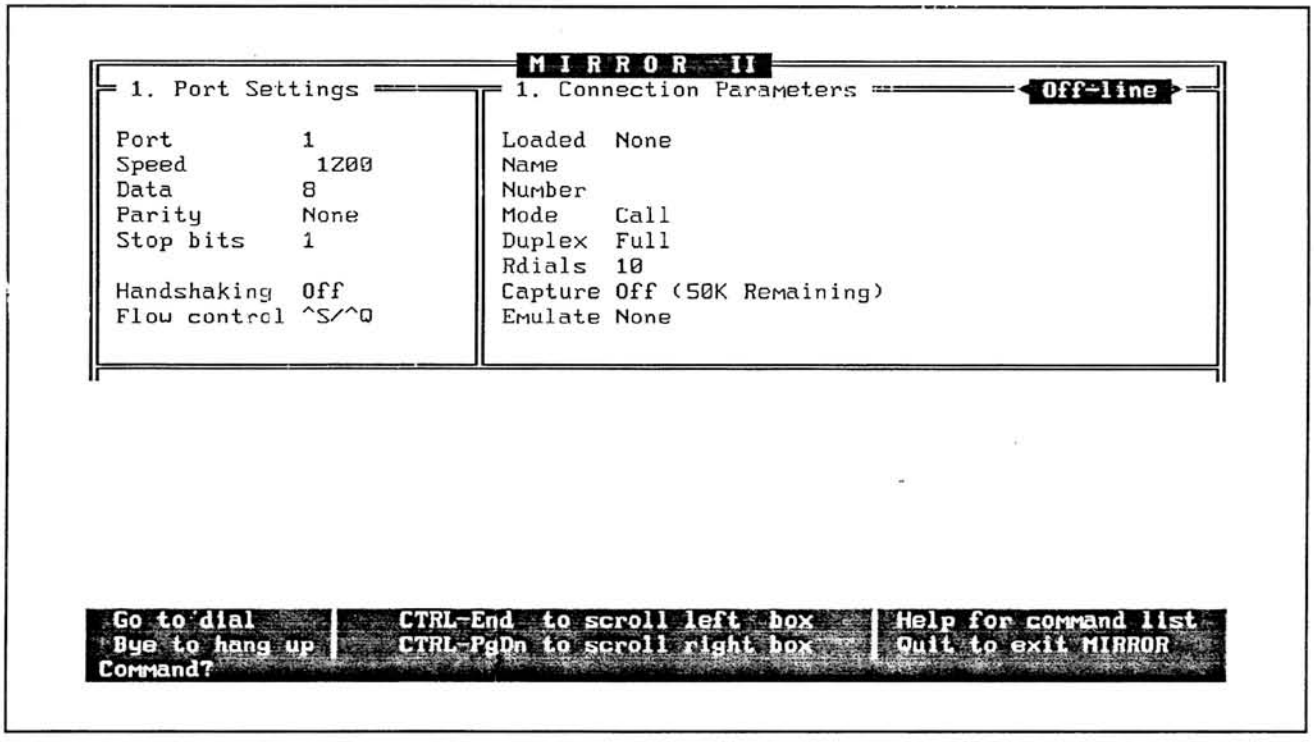


Figure 3. Mirror II's status screen: lovely to look at, delightful to use.

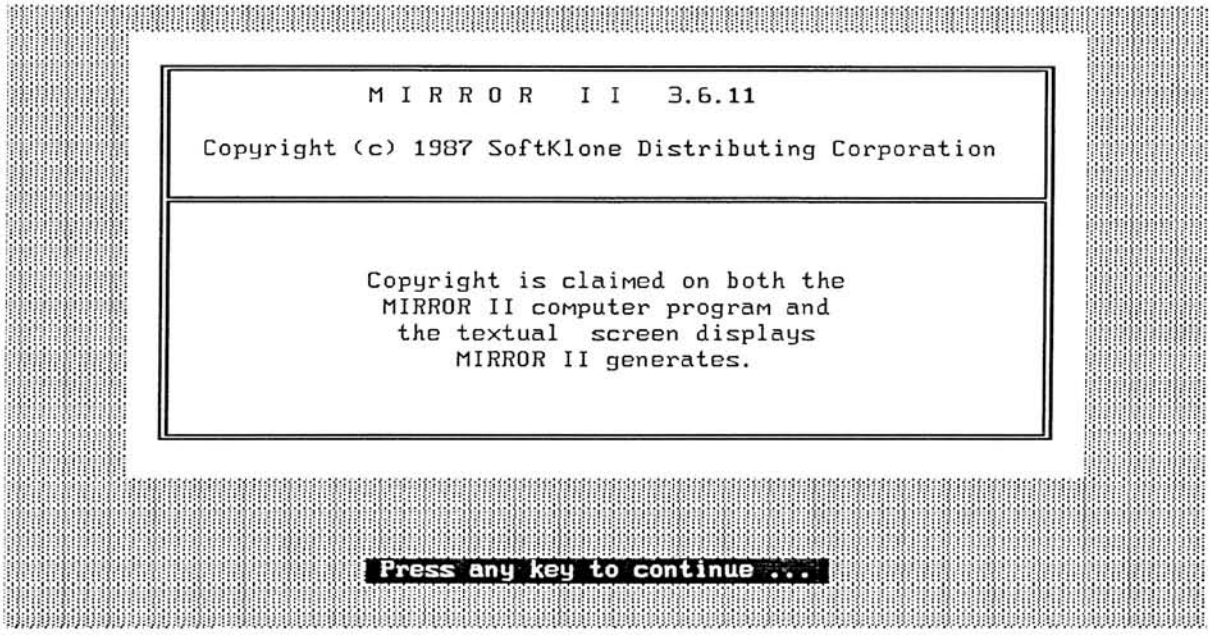


Figure 4. Write your own caption here: _____

bilities brought to *Mirror II* by this expansion of its command language is Curt Timmerman's electronic mail scripts, available as MAIL.ARC on SoftKlone's bulletin board system (904/878-9884). Look there for more of just about everything related to *Mirror II*, including more emulations and more scripts. Especially nice in *Mirror II* is the LeaRn command, which allows the program to record whatever it takes to get onto a new system. In the works is an expanded script language.

Mirror II is a developing program. Maybe its development actually was spurred by the Microstuff suit. Certainly, *Mirror II* is an improvement over *Mirror*, as well as over *Crosstalk XVI*. The price, by the way, has developed too: it's now $70. The suit must have been expensive.

Intellisoft's *Bookmark*

So you're banging away at an important report and you're nearly finished and the computer crashes and you were in such a hurry that you forgot to save your work and everything is lost and you can't face starting over again and you're tired of living and scared of dying. So what next?

If you've installed Intellisoft International's *Bookmark* utility, what you do is reboot the computer. You get a warm cuddly "Resume, y/n?" from *Bookmark*. You reply with an affirmative, and within a few seconds *Bookmark* brings you back to approximately where you were when you thought your life had ended. What *Bookmark* does is automatically save your place in your work from time to time, so the point to which you return after a disaster depends on how close you were to a saving time when the disaster struck. With the press of a hotkey you can force saves as often as you'd like, which ought to be fairly often if your life really does depend on the work in progress. If you're a bit of a daredevil and maybe a little addlepated or self-destructive, you can toggle *Bookmark* off with another hotkey.

Of course, there's no such thing as a free lunch, not even with *Bookmark*. It needs a lot of disk space to work its magic: *Bookmark* makes its notes in a hidden file, which was nearly 1MB when I hunted it down. *Bookmark* takes control of the computer during every save period: you get a message telling you to wait while it happens, and wait you must. Since *Bookmark* is a TSR ("Terminate and Stay Resident") program, the more other TSRs you use, the more possibility there is of a conflict between one of them and *Bookmark*. Because *Bookmark* writes a file automatically, make sure you deinstall it before using DOG or any other disk optimizing program, or else you risk sending your hard disk to Tijuana.

Nevertheless, I do think you ought to consider *Bookmark* whenever those disadvantages are outweighed by its considerable advantages, as — for example — in any office that harbors the neighborhood computer klutz. If your office has one of those guys who powers down the computer before saving his work, *Bookmark* him. I risked one of my hard disks in a few death-defying tests, and *Bookmark* passed them all. My own experiences with computer klutzes have given me sufficient awe for their destructive abilities so that I won't guarantee anything as failsafe, but *Bookmark* does look good. It used to be copy protected, but it isn't anymore.

Persoft's *Referee*

You may know Persoft, Inc., the way I do, for its nice *SmarTerm* terminal emulation programs. I have occasion to use an oldish copy of Persoft's *SmarTerm 100* to communicate with an oldish Ultimate minicomputer from time to time, and the *SmarTerm* thing is a little marvel. *Referee* is a little marvel of a completely different kind: it's a manager for Terminate and Stay Resident programs.

Both the idea and the execution are nifty. If you're a TSR junky, as I am, you have suites of programs. When I'm programming in C, for example, I often have a couple of debugging tools resident. But those things are completely useless when I start writing and need entirely different kinds of tools. My problem is that I alternate among programming, writing, and a number of other tasks

too, and then the mix of resident programs shakes up into something like a Molotov Cocktail. Question: "What do you get when you activate *XRAY* while running *XyWrite III*?" Answer: "A locked computer." So you don't get the joke? Well, neither do I. But my computer springs it on me from time to time and expects me to laugh. I can't. That's the reason I add a reset button to each of my computers.

Another thing I've been doing until I got *Referee* is use TurboPower Software's TSRCOM package to aid in unloading one set of TSRs before loading the next set. I've talked about TSRCOM in an earlier column, so I'll give you the nickel tour now. You run a TSR program called "*Mark*" to establish a placemarker in RAM before you run any other TSRs. Then you run those other TSRs, the ones you really wanted to use with your application program. When you've finished and want to use another set of TSRs, you run a program called "*Release*" to unload all the TSRs after the placemarker set with *Mark*. Then you run *Mark* again to set the placemarker, run your next set of other TSRs, and go about your business. It's not really such a tedious process in practice if you do as I do and automate everything with batch files. The idea behind *Referee* is quite different and much more sophisticated: *Referee* is a TSR manager with five major components that work in combination.

Refwatch is a TSR program that monitors a slew of major interrupts (I counted 42) involved in the operation of other TSR programs. You run *Refwatch* just before the first time you plan on using any other TSRs: it's, therefore, a candidate for your AUTOEXEC.BAT file so it will be run whenever you boot the computer.

Referee itself is the TSR manager and the main program to concern you. When you run *Referee* you get a menu with four options (see Figure 5). The very first is one you don't use until later in the game, after the *Referee* package and the TSRs are running. Then you can activate or deactivate any TSR that has already been run and made RAM resident (see Figure 6): deactivation simply disables its trigger, activation reenables that trigger, and the programs remain in RAM unless you choose to unload them. More about that in a moment.

The second and third menu options let you build the rosters from which you list the members of each RAM team: the association between TSRs and transient application programs. You use the second option (Figure 7) to list your TSRs. Be generous. All you're doing here is recording the names of any TSRs you might use even occasionally, not just those you use most frequently. *Referee* isn't bothered when it doesn't encounter a TSR on this list. Then you go on to the third menu option (Figure 8) and build a list of the transient programs you actually use, giving the executable file name (as "EDITOR.EXE") and the name by which you know the program (as "*XyWrite III*"). When that list is complete, you cross reference each transient program with the list of TSRs you entered earlier. You note how you want *Referee* to treat any TSRs resident at the time the transient application is run, establishing which TSRs should be active, which inactive, and which really don't matter either way. This function is the heart of the *Referee* package because it builds a control file called "RAMTEAMS .REF," another major component in the package, that determines working combinations when you run the applications. When I run *XyWrite III*, for example, all I do is run *XyWrite*'s EDITOR.EXE as usual. *Referee* automatically deactivates the TSRs that I know won't work with it. When I switch to *dBASE III Plus*, any of those TSRs I want activated for use with it are magically made available. I don't have to think, and the promise I could suspend thinking is what made me buy a computer in the first place.

The last item on the menu lets you exit to DOS. When you choose that option, you are given the opportunity to save any changes you made. RAMTEAMS.REF is updated if you do. Of course, you don't have to update RAMTEAMS.REF unless you change the RAM teams — the assignments of TSRs to applications. You also can un-

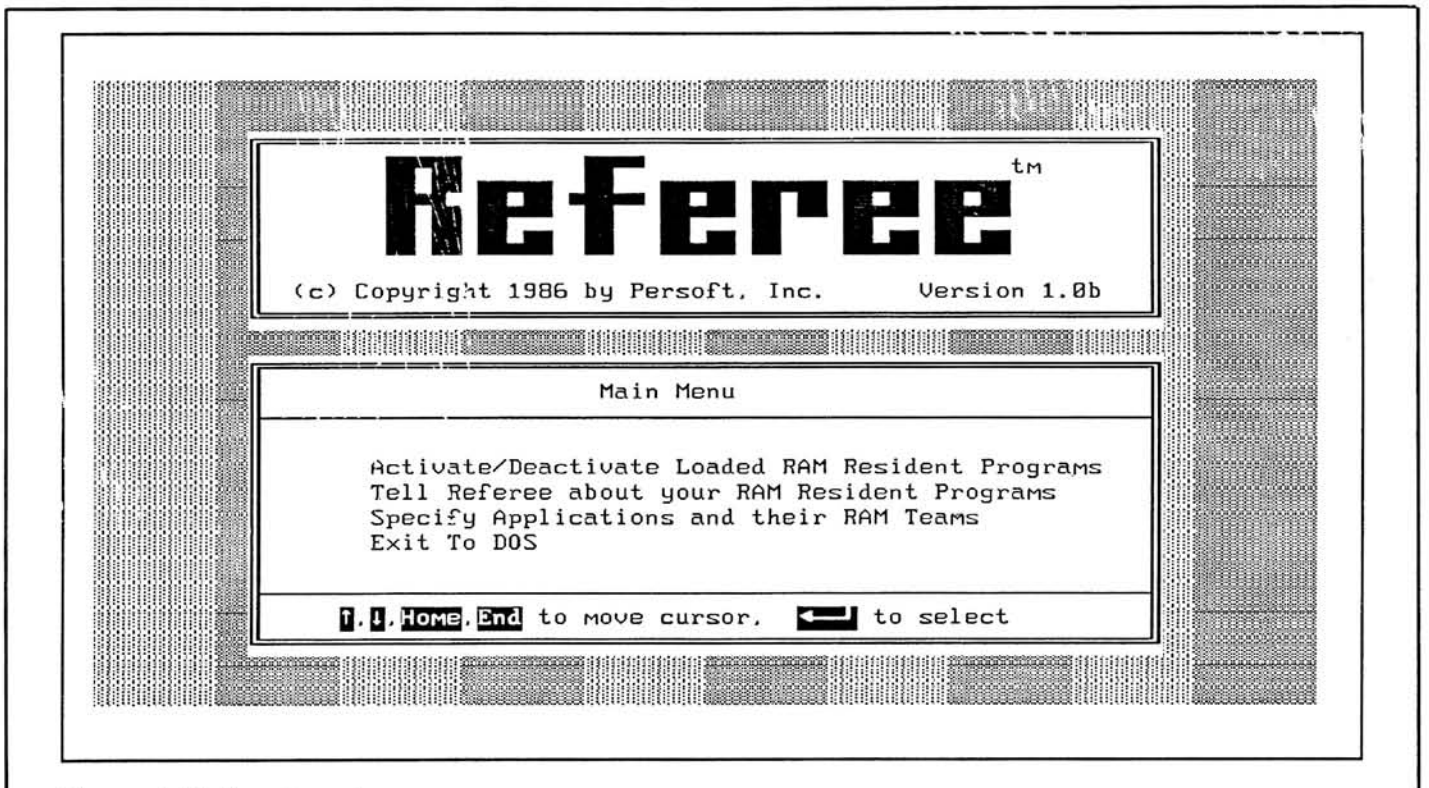


Figure 5. Referee's main menu.

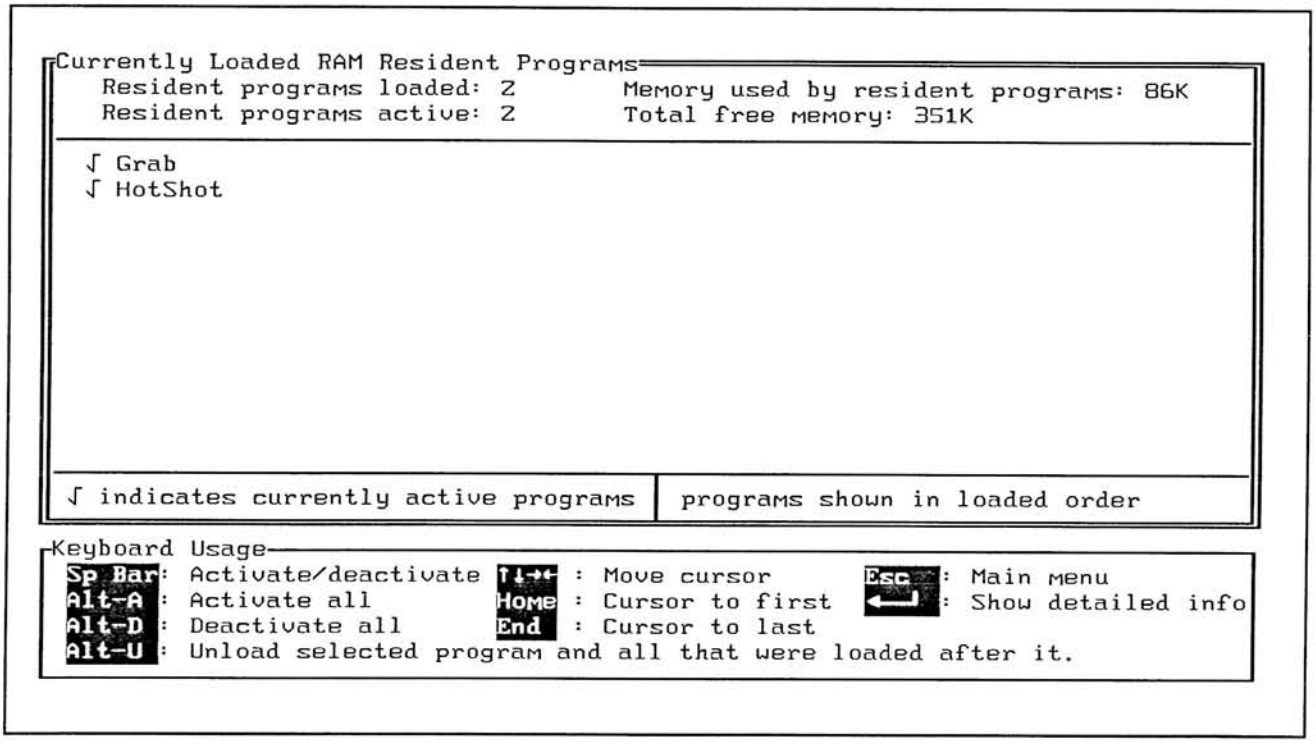


Figure 6. Referee lets you toggle TSRs.

load TSRs — remove them from RAM — with *Referee*. For safety's sake, so you don't wind up with a potentially disastrous "hole" in RAM, *Referee* unloads not only the TSR you designated, but also all those loaded after it. That procedure is first-rate programming practice.

Sideline, the last major component in the package, is a TSR program that lets you activate and deactivate other TSRs from within other programs. *Sideline* is useful because *Referee* itself is a transient program that can be called only from the DOS command line.

I would have predicted the *Referee* system vulnerable itself to conflicts with TSRs, because *Refwatch* and *Sideline* are TSRs themselves. But so far *Referee* has resisted all the battering I've given it, and I have tried hard. After a little thought, I realized it would take a pretty sneaky program to zap

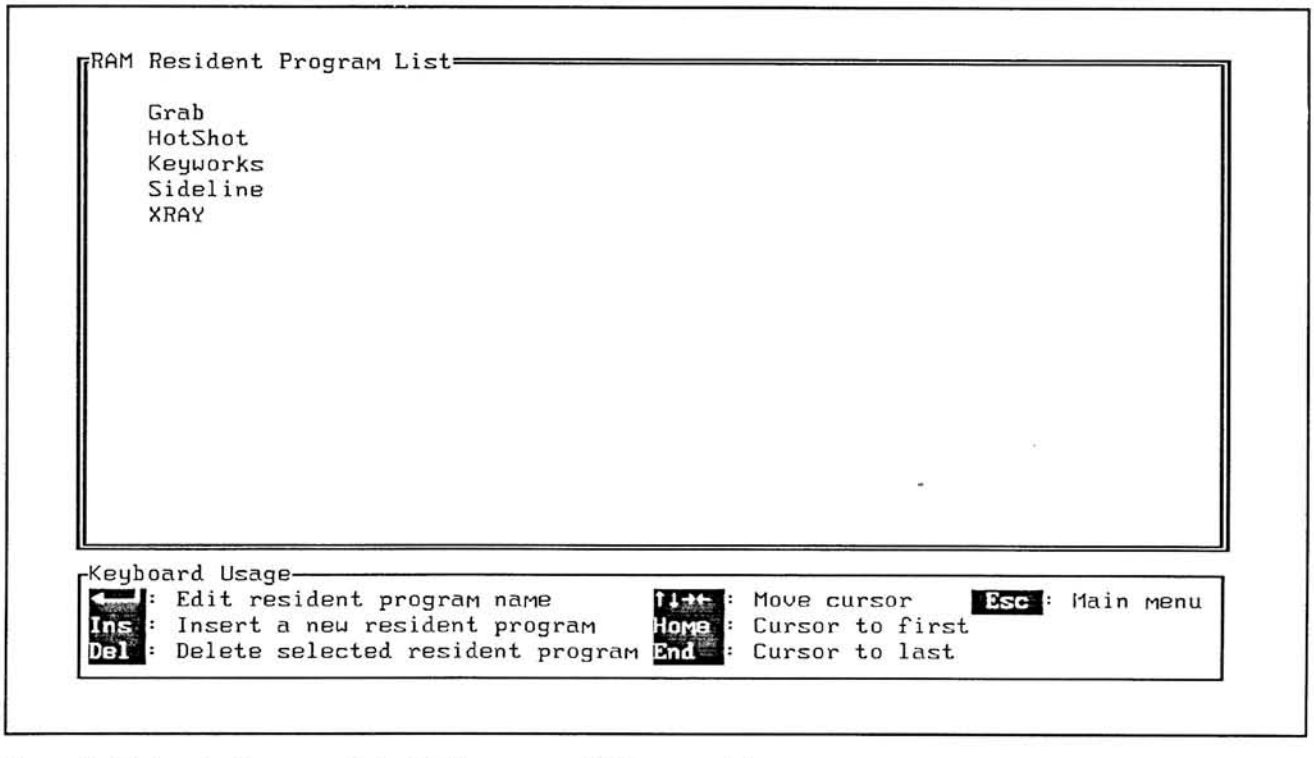


Figure 7. Listing the first part of the RAM team, any TSRs you might use.

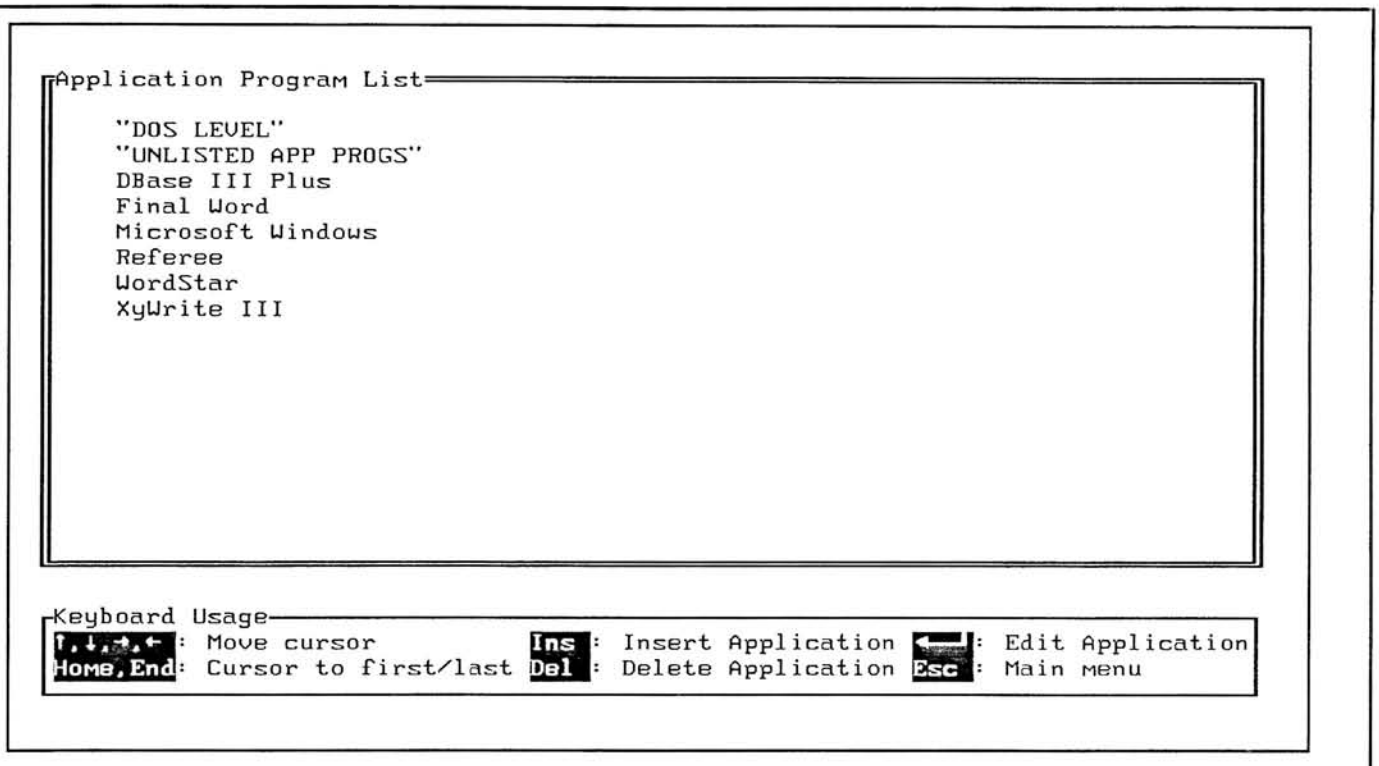


Figure 8. Listing the second part of the RAM team, the transients, and their associations with the TSRs.

Refwatch because of the sneaky way it manages the interrupts. It amused me to figure out that the traffic controllers in *The LaserWriter Connection*, my own package for HUG Software, do some of the same sneakiness as *Refwatch*. The thing to do is install my traffic controller (*LaserOne* or *LaserTwo*) before you install *Refwatch*. Then, everything slides as smoothly as frozen grits down an icy slope. Anytime you have a TSR that should be run all the time, or that turns out to conflict with *Referee*, run it before you run *Refwatch*. The manual says so.

There's one important point to keep in mind when installing the *Referee* package. Make sure to install all the *Referee* programs in the same directory, then make sure to run *Refwatch* from that directory. Otherwise, it won't find the RAMTEAMS .REF file it needs and you will go nuts wondering what's wrong. Persoft advises you to install the programs in the root directory of your hard disk. I didn't like that idea, but experience makes me agree. All I'll add to such good advice is that you use the root directory of the hard disk from which your computer boots, and that you run *Refwatch* from your AUTOEXEC.BAT file so you don't accidentally run it from some other directory and mess yourself up.

Of course, before you buy *Referee* you want to consider your available RAM, the appetites of all those TSRs you dream of stacking in it, and the requirements of any application programs you think of using with that stack. *Refwatch* consumes 25KB of RAM. *Sideline* (which is optional and which you might not often use) takes another 15KB. *Referee* requires 64KB, but since it's a transient program run from the DOS command line who cares? Where you have to be especially careful is if you use a big program like Microsoft *Windows*. It needs a minimum of 512KB of RAM, which means you can't expect to do a quicky in a *Windows* application after *Referee* merely deactivates your TSRs. Even though they're all useless in *Windows*, they continue to occupy RAM. You'll therefore have to unload those TSRs and, probably, *Refwatch* too before you run *Windows*. Sigh.

But that's not *Referee*'s fault, or even *Windows*' fault. It's a Circumstance of Life. In most situations, *Referee* is such a useful charmer that I am coming to believe it's almost an imperative for people who use many TSRs.

Incidentally, I admire the *Referee* manual. It's handsomely produced, nicely indexed, and extremely well written. Nice job, Persoft.

Dr. Katz's Handy Household Hint

My friend, Tom Watts, didn't believe me when I remarked some years ago that I might be the laziest man in the world. I had forgotten I said such a foolish thing until a few weeks ago, when we were discussing keyboard enhancers on the order of Alpha Software's *Keyworks*. Tom asked how I remembered all the macros I wrote on the fly. I said I really didn't write many. "Huh?" he replied, "I thought you used them all the time."

Well of course I do. An article about Stephen Crane, for example, will be replete with references singular, possessive, and plural: "Crane," "Crane's," and "Cranes." But I use only one macro key for all three because I only store "Crane" in it. I don't put a space after the "e." When I need to type the singular, it's easy enough to hit the macro trigger followed by a flick of the spacebar with my thumb or whatever part of my hands happens to be in the vicinity at that moment. When it's the possessive I want, I hit the macro trigger followed by the apostrophe and the "s." For the plural it's the macro trigger followed by the "s" alone. The secret is boiling a frequently-used word down to its most-frequently-used root and omitting the space after it. That saves having to define or remember several different macros. Bit by bit all these little savings accumulate so I have lots of spare time.

There was a moment of silence when I finished. Then Tom said, reverently, I think, "Lordy. You are the laziest man in the world."

A Free *PostScript* Utility Program

While I was at IBM's COMDEX booth, I explored IBM's new desktop publishing stuff. It still strikes me as a lot more expensive and a little less powerful than the combination of the Heath H-248, Aldus *PageMaker*, and Apple *LaserWriter* I've been using for months. You understand that sometimes it takes me longer than everyone else to get the point of a joke. While I was there, though, I got to thinking.

All *PostScript* printers use VM ("Virtual Memory") to process data and programs during each printing job. VM is a precious resource, so big programs usually are preceded by a header that defines time- and space-saving macros: "/TB" for "Times-Bold," for example. You don't have to know about any of this to use a *PostScript* printer, of course, because it's the province of programmers. But some programs deposit headers that remain after the job is finished and you've gone on to something else. It's sloppy programming practice to not clean up after oneself, I think. Use enough of such programs and eventually the discarded headers stack up in VM, with the result that the printer doesn't have the space it needs to function efficiently.

The solution is simple: reset the printer to clear the VM. It used to be that you didn't know when to reset because there wasn't a way to check on the VM status. Then I wrote the Stat program included in HUG Software's *The LaserWriter Connection*. Now all you have to do is get Stat to report on the printer's internals, including its VM status. Takes a second.

But what hit me while I was standing at IBM's COMDEX booth is the waste of energy involved in leaning over to turn the printer off and on again, which is the only way to reset it. It's that kind of waste that starts my juices flowing towards a solution. I devised one. As you might have guessed, it's a program called "Reset," which resets the *LaserWriter* with one command. It should work with any *PostScript* printer connected to any MS-DOS computer.

If you are as concerned as I am about hoarding precious bodily energy, you can download *RESET.ARC* (there's a short doc file in it saying at greater length what I've said here) from CompuServe's HUG SIG (Go PCS48). Or talk someone else into doing it for you.

See you later.

Products

Mirror II Version 3.6.11 — $69.95
SoftKlone Distributing Corporation
336 Office Plaza Drive
Tallahassee, FL 32301
(904) 878-8564

Bookmark Version 1.1 — $95.00
Intellisoft International
70 Digital Drive
P.O. Box 5055
Novato, CA 94948
(415) 883-1188

Referee Version 1.06 — $69.95
Persoft Inc.
465 Science Drive
Madison, WI 53711
(608) 273-6000

*

C__Power

Part 6

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

Hi Huggies, my thanks to those of you who have followed this series from Part One to the present. Your patience is about to be rewarded. Those of you who have joined us more recently will probably want to obtain the back issues of REMark that include the preceding articles on "C__ Power", since this series builds a program, one step at a time.

This issue includes the code to complete option five from the program menu, "5. Sort list (on specified field)". I will explain the program in a top down manner as done previously, but before we get into that, I would like to explain how the sort works and also discuss various methods of sorting data.

There is no shortage of algorithms for sorting. Perhaps the biggest problem facing the programmer who is contemplating writing a sort routine to include in a data storage program, is which one to use and how to implement it. My first effort in this direction consisted of some experiments with first, the "bubble" sort and then the "shell" sort. After some testing and more reading, I decided that the bubble sort would work fine with a small number of records, but when dealing with a large database, more sophistication was desired. Another point to be remembered here is the way in which records are added to the database. The new entries are appended to the existing file, thereby creating a lot of work for the bubble sort in most cases, the exception being the unlikely event that the record being added is already in order. Of course, all this becomes academic when we might choose to sort alphabetically one time and then by zip code the next. The shell sort first compares items near the low end of the base with those near the middle, moving toward the high end with each iteration, halving the gap with each pass. This means that most items are nearly in order after only two or three passes, thereby greatly reducing the number of passes required to unjumble the data.

All this theory is great mental exercise, but when we put some of these ideas to work, we find another problem to be dealt with, time of execution. Our data is on disk and cannot be successfully sorted in its present format. We need to read it into memory and segment the record into the fields we wish to compare before it can be sorted and then put it back on the disk. One way to accomplish this would be to use the sort algorithm to generate two integers which are converted to the record offset using: offset=i*128. Eureka! Using the other integer variable (j) and this same algorithm will provide us with two records to compare and swap, if needed. We could employ this method to sort the entire file, two records at a time. You could also grow old waiting for this job to be accomplished if the file is very large. You might decide to take up another hobby to occupy your time, such as watching the grass grow. How can we speed this job up? I thought you would never ask!

The big time consuming factor here is the number of disk access operations (reading and writing). OK, we'll just read the records into memory, sort them and then write them back to the disk. Problem solved; or is it? This solution will only work with a relatively small number of records, since we have a severe memory limitation imposed by our hardware (64k max, minus overhead, minus space occupied by our program code). Alright, I got you into this dilemma and I'm going to get you out. You didn't think I would abandon you now did you?

My solution to this problem is really quite simple. The memory requirements can be reduced drastically by parsing the record for the field to be sorted, then reading only this record segment along with a location reference into memory, appending each in turn; ultimately constructing a long string with a comparison field and location reference in each segment. When this is done (less than 10k of memory needed for 1000 records), we can compare string segments, swap if necessary (along with our location reference) and construct a model for records retrieval. Have I got you confused? Stand by, it's really quite easy after you get the picture.

Imagine, if you will, that you can see this string residing in the computer's memory and that all these record segments have been placed there by our program. Each record segment occupies ten bytes of memory, formatted so that the name (or zip code) occupies the first seven bytes (padded with spaces in the event the name is less than seven characters). The location reference is the disk offset divided by 128 (allowing up to 1000 records in this program implementation), and occupies the last three bytes of each string segment. The first ten bytes of our very long string might look like this: Lewis(space)(space) 0(space)(space). Notice that the name Lewis is padded with two spaces and the record number (zero) is also padded with two to make a total of ten characters. The next ten bytes of our string might look like this: Asner(space)(space)1(space)(space). This example reflects the string before sorting. After sorting, these two string segments would exchange locations in memory, but would retain their disk offset references since they are an integral part of the individual segment. Now, we parse this sorted string looking only for the offset reference in each field. We retrieve each record by its reference number and write it back to a temporary file on disk in the new sequence (1, 0, etc.), thus achieving a sorted file. After completion of the write operation, we destroy (erase) the old data file and rename the temporary file to provide a replacement for the original. The disk must have enough space for two data files simultaneously, in order for this last operation to succeed. The entire operation takes about forty seconds on my H-120 utilizing a 5 mhz clock and thirty-five records, which is admittedly a very small database. The educational part of this (for me) was finding that the memory sort was completed in under seven seconds, the rest of the time (33 seconds) was used in the disk operations.

Now that you have an idea of what is happening, let's look at the code for this routine.

I discovered a slight discrepancy in the C/80 STDLIB.C file while I was working on this sort routine, and perhaps we'd better fix the problem before proceeding. We are dealing with numbers (record number and offset) when writing our sort fields and location references to the disk and the numbers must be converted to ASCII characters before the write operation is invoked. Likewise, they must be converted back to numbers after reading this data from the disk and before attempting to do any math with the data. Two library functions are provided by the C/80 STDLIB.C file, which provide functions to bring this about. To convert integers to ASCII characters, we invoke itoa(n,s), passing the two arguments to the function within the parentheses. The first argument n represents the integer we are converting, and the second is actually a dummy argument (has no value when passed) that is returned with our ASCII representation. My copy (circa July 1985) of C/80 included an invalid code segment in the STDLIB.C file. In place of the itoa(n,s) code fragment was this line: char *itoa(n,s). I replaced it with: itoa(n,s), which stopped the error messages that were plaguing me when I attempted to compile the new program. This would be a good time to fire up your computer, get your word processor, the STDLIB.C file and make this correction (if needed, your copy may be OK).

Back already? You work fast! Before we get into the actual listing, we'll need to review the "FPRINTF" function in order to emphasize one point. For instance: the command fprintf (fd,"%-25s",name); will cause the computer to write the variable "name" to the disk in a field that is 25 characters wide, left justified. Assume, for the moment, that the "name" is thirteen characters in length. This would dictate that the FPRINTF function pads the name field with twelve spaces (for a total of 25 characters). The success of the parsing routine (found below in function preparec) depends on this format for its execution, so be sure you understand this concept before proceeding.

Now let's look at the program listing and see how this sort routine works. You will notice that four new define statements have been added and the comments should explain them adequately with the exception of the ZIP 115. We'll elaborate on this statement when we discuss it's use, later in the text.

Moving down, we find a new function, preparec(gap). This bit of code, surprisingly enough, prepares the disk record for sorting. Notice that the value "gap" is passed to the function from the calling routine. This variable represents the (total) number of records in the data file and the value will be used for a reference in the "while" loop that is retrieving our records. Before entering this loop look at the variable declarations, in particular, the "char recnum[7];. Here, we're providing a storage area for the ASCII representation of our record number. The integer declarations are mostly used for counters excepting rnum, len and gp. The record offset is used to derive rnum, much the same as when obtaining "loc", used in the search function. We will cover these in greater detail when they are encountered within the body of the routine. Let's get into the retrieval loop and see what is happening.

First, after the "while" statement which controls our exit, we see the familiar (by now) "seek" statement and two variable initializers, primarily used to reset these values each time the loop is executed. Notice the variable "pad" within the "for" loop. This is a global variable, set by the user within the calling routine, causing an exit from the "for" loop when (pad) characters have been retrieved. The variable "pad" will have a value of 49 (LEN1+LEN2-1) or 115 (ZIP) depending on whether the user wishes to sort this file on the name field or the ZIP code. Glance down at the code which sets these values under case '5':. The user responds to the prompt for a field to be used in the sort function and thereby determines the value of "pad". We utilize a new (for this series) feature of the switch-case decision making operation, that of "default". We assign a value of 115 (ZIP) to the global variable pad if the user presses "return" without first pressing "N" or "Z". You, the programmer, may wish to change this default value and I have made it easy to do so as is self evident.

Meanwhile, back at the function, we can see that our record segment has been stored in c[i] and our subscript (i) is equal to one less than pad. The object of this little exercise is to ensure the retrieval of the characters comprising the desired field. Next, we will parse this same string from back to front, looking first for space characters and then for the string containing the last name or zip code. Please note that the "for" loop used for this job (for (; c[i] == SPACE ; --i)) does not initialize i, but uses the value derived from the previous loop. The program falls out of this loop on the first non space character and enters a very similar loop, which is looking for non space characters and will exit on the first space encountered, all the while counting the non space characters (++len). Notice that we are using the value of i derived from the previous loop in each case and decrementing its value (--i) as we parse the string c[i]. Upon the exit from this loop (for (; c[i] != SPACE ; --i,++len)), we bump our subscript back up (i+=1;) to coincide with the last non space character and count the loop passage (gp-=1;). Now we have the proper values stored in i and len to construct a second string containing the field we will use for our sort. The character contained within c[i] is now the first character

to be used in our new string. A new string called ptr is constructed from the string segment c[i to i+6], yielding a seven element string, which we will append with a string segment containing the ASCII representation of the record number. Are you still with me? I know this is a bit tedious, but the whole sort routine depends on this operation for its proper functioning.

One point to all this string manipulation that is not immediately obvious, is that when referencing a string, such as "ptr", we are referring to the entire structure, starting at subscript 0 and through its entire length. This might help explain the logic of the "for" loop which creates a string called "ptr" from a string segment c[i to i+6]. Our "for" loop (j=0; j < len && j < 7 ; ++j++i) initializes j to zero, which is our ptr[j] subscript. The code just below this loop, ptr [j]=c[i];, builds a new string from c[i to i+(len-1)]. Actually, the loop is exited when j is equal to len -1 or six, whichever happens first. Assuming the length of our string "ptr" is less than seven characters, the next "for" loop will pad it with spaces until the string occupies seven characters. In any event, after leaving this loop, we have clearly delineated our string and can now append the record number. This number is arrived at through dividing the offset (disk location in bytes past the beginning of given file) by 128 (rnum=offset/RECORD) [RECORD=128]. Next, we take our repaired function "itoa(n,s)" and use it to transform our record number into an ASCII representation of same (recnum). Then, call on the STDLIB file again to tell us how long recnum is (len=strlen(recnum)). Now we are going to place these two elements into their proper homes where we will append each successive field in turn. Remember string[SIZE]? Well, it's just sitting there, waiting for us to put it to work so we'll utilize it for storing this rather long string (10*number of records). The next three "for" loops accomplish this by first reading ptr into string[k], recnum into string[k], and finally padding it with spaces to make each segment residing within string, ten characters long. Meanwhile, keeping track of the number of characters in string by incrementing k with each addition, thereby maintaining a pointer to the next segment position. This routine is repeated until gp equals zero, at which time we will fall out of the loop and call "sort".

The sort routine has the variable "gap" passed to it, which is used as a reference to provide the program with a pair of integers which are passed to the "get" function where they are multiplied by ten, and then used to generate a position within string where we will retrieve a pair of (string) segments for sorting. By looking closely at this (sort) algorithm, you will see that it is a derivation of the "shell" sort. Notice how the value of gap is halved with each pass through the loop and this value is then used to generate two additional integers which are then incremented until the largest of the two (j) is equal to the original value of gap (q) before division, thus constituting one pass. I make no claims for the efficiency of this routine, which is a ripe algorithm for modification by anyone who has a penchant for experimentation. Be my guest!

Our next new function, "get", uses the integer values passed to it by sort to create a pair of pointers into string where our (string) segments are waiting to be sorted. Since each segment is now ten characters long and lying end to end with the rest of the concatenated fields, we have only to multiply i and j by ten to provide us with the proper subscript to access a pair of fields within string. The code fragment: offset 2=10*j;offset=10*i; does this for us. Now we can retrieve these two segments and compare them. We use strng1 and strng2 (declared within this function) to store the ten character fragments derived from string through the use of a couple of "for" loops. Notice too, that we make i and j equal to offset and offset2 respectively, thus pointing to the segments to be compared. Now we call on another of the library functions contained in "STDLIB.C" (strcmp) to do the comparing for us. This function returns "1" when strng1 is greater than strng2. We will test for this event (cmp=strcmp (strng1,strng2)); if (cmp==1) swap(strng1, strng2,offset,offset2); and perform a swap if cmp is equal to 1 while setting flag equal to q (this prevents escaping the function before the sort is complete). If cmp is not equal to 1, we return to the calling function (sort) for another iteration (or exit if gap is equal to zero).

The swap function is another area ripe for modification. I left it as you see it for the sake of clarity. You might want to modify it for the sake of speed. If so, you are on your own. Here, we are using the variables passed (from the calling function) for the actual swap, as well as their locations so we will know where to put them after the swap is complete. First, strng1 is read into a temporary storage, then strng2 is read into the strng1 storage. Next, the respective string segments are read into their new locations within string using yet another "for" loop. Notice that we did not read temp into strng2, thereby eliminating an unnecessary extra step. When swap is not called during the last iteration of sort, we have completed all the string manipulations necessary for creation of a sorted file. This is the job performed by writerec, our next function.

The next operation to be performed is the actual writing of a new data file using our sorted string with the offset locations for our record retrieval reference. To wit; the string segments are in the order we specified, but the records on disk are still where they were when all this started. We are going to read the individual segments from string, discarding the actual character strings, but retaining the ASCII representation of the record numbers, convert these into the disk offset corresponding to each record, retrieve the individual records, and finally, write these records back to a temporary file in sequence, thus achieving a sorted file. This might sound a bit complicated, but is really quite simple when broken down into the respective tasks.

Before we get into the actual routine, notice that this function is called from case '5': where we informed the user that the computer was in the process of writing the sorted file to disk, implying that the memory sort was finished. The variable q is passed to writerec for a reference, indicating the number of records to be written.

The first variable to be declared within the body of our writerec function is an integer named recnum, subscripted to NUMREC. Here is the key to understanding how this routine makes order out of chaos. Within the first "while" loop (while(no > 0)), there are two "for" loops, one to discard our string segment and the next to retrieve the ASCII facsimile of our record number. Each time this "while" loop is executed, recnum[j] is set to equal the numeric value of the stored record number through the use of another STDLIB.C function; (atoi). Now all that is left for this function to do is simply the nuts and bolts of retrieving the records in the order dictated by our subscripted recnum and then writing them back, in sequence, to the new temporary file which will, in turn, become our new data file. Before we leave this routine, notice that we read four records at a time into a buffer created especially for this job and sized to hold four records, specifically, holdit[512]. This enables the program to deal with four records at a time, thus cutting down on the total number of disk access operations. After writing this new file, we delineate the last record by inserting a DEL character in the first byte and EOF in the next while printing a user message to indicate the last

record. We close the new file as before and return to the calling routine where two new functions are discovered; namely, unlink and rename. The first of these (unlink) merely alters the disk directory to make it appear that the data file no longer exists, and then the temporary file is renamed.

Now your patience has been rewarded and I hope this series has contributed to your understanding of random records and the "C" programming language. The next article in this series will "recap" some of the important concepts and furnish a rather valuable troubleshooting tool, described in "C_Power — Part 2". We will provide the code for: "HEXOUT", a modified version of "HEXDUMP" found in "Soul of CP/M", by Mitchell Waite and Robert Lafore, published by Howard W. Sams & Co., Inc. They have been kind enough to allow me to furnish this program. I will show you how to add this code, which we will change to a function, to your library of "C" functions. Then, you will be able to "include" it in any of your "C" programs and "call" it from any location within the program that you wish, thus providing a "window" into the computer's memory.

This issue of REMark is the last to provide a listing of our data storage program. If you are having problems with program "bugs" that you cannot resolve, or simply lack the time to enter this rather extensive program from the keyboard, I will furnish the source code, as well as the executable COM file for this program to any "Huggies" that send me a blank, formatted disk (hard sector for the H-89 or soft sector, double-sided for the H/Z-100). Please include five dollars for postage and handling.

"C" you next month.

Listing 1

```
 /* --- "CPOWER" by John P. Lewis --- */
 /*  source code for part six,   11/12/86  Ver. 112086 */
#include "printf.c"
#include "funclib.c" /* routines cls, locate, fetchc and gofor
                        from "CPOWER" */
#include "seek.c"  /* one of random access library files */
/* #include "jackslib.c" */  /* remove or modify for HUG */
/* #include "hexlib.c"*/  /* hexout library file */
#define DEL 127  /* this char used for an "end of records" delimiter */
#define SIZE 2048  /* size of future "search" area */
#define DELAY 10000  /* number of iterations in delay loop */
#define LEN1 25     /* length of field for compny & name fields */
#define LEN2 25
#define RECORD 128   /* record size  */
#define SPACE 32   /* space char used for fill */
#define LEN 7      /* length of comparison string */
#define NUMREC 200  /* number of records */
#define ZIP 115

unsigned fd, offset, loc, q, f2;  /*global variables to be used
                                    by functions and main */
int init, pad, flag;
char ptr[LEN1], string[SIZE], c[RECORD], dummy[2], men[2], holdit[512];
                          /* holdit = buffer for reading (4)records */
search ()
{
int i, k;k=0;
opnforrd ();seek(fd,offset,0);
if (init != 1)
  {
  printf("Please stand by, I'm initializing \n");
   while ( c[0] != DEL )
    {
  seek(fd,offset,0);
  for ( i=0; i < LEN1+LEN2 ; ++i,++k)
     {
     c[i] = string[k] = getc(fd);
     }
    offset+=RECORD;
    }
init =1;q=k-(LEN1+LEN2);offset=0;c[0]='\0';   /* purge c[0] fix */
  }
}
srch2 ()
{
printf("Please enter the search 'Key' ");
gofor(ptr,LEN1);
loc=jindex(string,ptr,offset);offset = (loc/(LEN1+LEN2))*RECORD;
}
jindex (s,t,p)
char s[],t[];
unsigned p;
{
int i, j, k;
for ( i=p; i <= q ; i++)
   {
   for ( j=i, k=0; t[k] != '\0' && s[j] == t[k] ; j++, k++);
   if (t[k] =='\0' )
     return (i); /* i=pointer if match is found */
   }
return (-1);   /* no match */
}

opnforrd ()
   {
   fd=fopen("data","r");  /* open "data" file in read mode */
   if ( fd == 0 )
     {
     fd=fopen("data","w");seek(fd,0,0); /* if file does not exist, create */
     fprintf(fd,"%c%c",DEL,EOF);seek(fd,0,2);fclose(fd);/* one & insert */
     fd=fopen("data","r");  /* delimiter */
     }
   }

o_pen()
   {
```

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

open_t()
    {
    f2=fopen("temp","u");  /* open destination sort file */
    }

open_p()
    {
    f2 = fopen("lst:","w");   /* "open" printer */
    }

lprint ()
  {
  int i;
  char ch;
/*  putc(27,f2);putc(21,f2);*/   /* Tandy printer line feed defeat */
  seek(fd,offset,0);
  for ( i=0; i <= 116; i++)
    c[i]=getc(fd);
  putc('\n',f2);
  for ( i=0; i <= 24; i++)
  {
  putc(c[i],f2);
  }
  putc('\n',f2);
  for (i=25; i <=49; i++)
     {
     putc(c[i],f2);
     }
  putc('\n',f2);
  for ( i=50; i <=84; i++)      /* output to printer */
     {
     putc(c[i],f2);
     }
  putc('\n',f2);
  for ( i=85; i <=116; i++)
     {
     putc(c[i],f2);
     }
  putc('\n',f2);putc('\n',f2);
   if (men[0] != '4')
     {
     printf("\n\tCopy (y/n) ");
     gofor(dummy,1);ch=dummy[0];
     ch = toupper(ch);
     if (ch == 'Y')
     lprint ();
     }
  }

preparec(gap)
   int gap;      /* this function retrieves a record & parses  */
   {            /* the field to be compared, stores last string(in field) */
   char recnum[7]; /* & record number in memory for future sort */
   int i, k, j, rnum, len, gp;gp=gap;k=0;offset=0;opnforrd();
   while ( gp > 0 )
     {
      seek(fd,offset,0);j=0;len=0;
        for( i=0 ; i < pad; ++i)
        c[i]=getc(fd); /* retrieve name field */
        i-=1;      /* bump past end */
        for( ; c[i] == SPACE ; --i );   /*  parse for spaces */
        for( ; c[i] != SPACE ; --i,++len ); /* look for non space char */
        i+=1;gp-=1;
        for( j=0; j < len && j < 7 ; ++j,++i )/*new string from last name */
        ptr[j]=c[i];                          /* or zip */
        for( ; j < 7 ; ++j )
        ptr[j]=SPACE;     /* pad ptr with spaces if less than 7 char */
        rnum=offset/RECORD;  /* derive record number */
        itoa(rnum,recnum);len=strlen(recnum);  /* convert record number */
        for( i=0 ; i < LEN ; ++i, ++k )  /* to ascii & get length */
        string[k]=ptr[i];
        for( i=0 ; i < len ; ++i, ++k )  /* store appended string in mem */
        string[k]=recnum[i];          /* at string[k]  */
        for( i=7+len ; i < 10 ; ++i,++k )
        string[k]=SPACE;    /* pad string to length = 10 */
        offset+=128; /* continue until all records have been read into mem */
        }
     offset=0;sort(gap);    /* call sort function */
 }

sort (gap)
   int gap;  /* gap = num of records  */
   {
   unsigned i, j;q=gap;
   while ( gap >0 )
        {
        gap=(gap/2);        /* this is a modified shell sort */
        if( gap == 0 )
         break;
        for( j=gap; j < q; ++j,--flag )
        {
        i=j-gap;    /* i & j are incremented each iteration */
        get(i,j);
        }
       if ( flag > 0 && gap < 2 )
        {
        gap+=1;continue;  /* if swap was performed during last iteration */
        }                 /* increment gap and do it again */
       }
   }

get(i,j)
    int i, j;
    {
    unsigned offset2, k, temp, cmp;
    char strngl[11], strng2[11], fieldl[LEN+1], field2[LEN+1];
    offset2=10*j;offset=10*i;  /* i & j become locations in string[i] */
    i=offset;j=offset2;        /* for conversion back into strings for */
    for( k=0; k < 10 ; ++k,++i ) /* comparison */
    strngl[k]=string[i];
    for( k=0; k < 10 ; ++k,++j)
    strng2[k]=string[j];
    for( i=0; i < LEN ;++i)  /* shorten comparison string by leaving off */
        {                                    /* record number */
        fieldl[i]=strngl[i];field2[i]=strng2[i];
```

```
        }
    cmp=strcmp(field1,field2);
    if ( cmp == 1 )
        {
      swap(strng1,strng2,offset,offset2);flag=q;  /* swap if field 1 is > */
        }                              /* than field 2, reflect swap in flag */
    }

swap(strng1,strng2,offset,offset2)
   char strng1[], strng2[];  /* this function swaps strings in memory only */
   int offset,offset2;     /* to be used for computation of new offset */
    {
    char temp[11];
    int i, j;
    for ( i=0; i < 10 ; ++i)  /* strng one is read into strng temp */
    temp[i]=strng1[i];
    for( i=0; i < 10 ; ++i)
    strng1[i]=strng2[i];     /* strng two is read into strng one */
    for( i=offset; i < offset+10; ++i,++j)
    string[i]=strng1[j];j=0;    /* i = offset within string[ ], read strng1*/
    for( i=offset2; i < offset2+10; ++i,++j) /* into new location, likewise */
    string[i]=temp[j];   /* strng2, swap is now completed */
    }

writerec(no)
    int no;
    {
    int recnum[NUMREC], i, j, k, offset2, temp, last;
    char  retrieve[3], dump[7];
    last=temp=no;j=0;k=0;offset2=0;f2=fopen("temp","w");
    fclose(f2);open_t();fclose(fd); /* open temp file, close data */
    while ( no > 0 )
      {
      for ( i=0; i < 7 ; ++i, ++k ) /* retrieve & dump string[k] */
      dump[i]=string[k];
      for ( i=0; i < 3 ; ++i,++k ) /* retrieve ascii representation of */
      retrieve[i]=string[k];--no;  /* record number */
      recnum[j]=atoi(retrieve);++j;  /* convert it to numeric val */
      }                            /* store in subscripted recnum */
      no=temp;j=0;i=temp=0;offset2=0;
      while ( no > 0 )
        {
        opnforrd();k=0;
        for( i=temp; i <= temp+3 && no > 0 ; ++i,--no)
           {
           offset=recnum[i]*128;   /* convert recnum[i] into offset */
           seek(fd,offset,0);     /* retrieve records in sorted order */
           for ( j=0; j <= 127 ; ++j,++k)  /* store 4 in holdit buffer */
             holdit[k]=getc(fd);
           }
           k=0;temp+=4;fclose(fd); /* write sorted records to temp file */
           for ( i=0 ; i <= 3 && last > 0; ++i,--last )
            {
            for ( j=0 ; j <= 127 ; ++j,++k )
            string[j]=holdit[k];seek(f2,offset2,0);
            fprintf(f2,"%-128s",string);
            offset2+=128;
          }
        }
      seek(f2,offset2,0);fprintf(f2,"%c%cLast Record            ",DEL,EOF);
      seek(f2,0,2);fclose(f2);k=0;
    }

main ()
  {
    int i, k, on, j;
    char  compny[LEN1], name[LEN2], street[35], city[33];
    char phone[11], ch; /* char men[2] deleted & moved to global */

    do
    {
    cls ();putchar(27);putchar(70);locate (4,5);  /* enter graphics mode */
    printf("pppppppppppppppppppppppppppppppppppppp");
    printf("pppppppppppppppppppppppppppppppppppppp");
    putchar(27);putchar(71);  /* exit graphics mode */
    printf("\n\n\t         Please ....");
    printf("\n\n\t              1. Search for record.");
    printf("\n\t              2. Review existing record names.");
    printf("\n\t              3. Create new record(s).");
    printf("\n\t              4. Print entire mailing list (to printer).");
    printf("\n\t              5. Sort list (on specified field).");
    printf("\n\t              6. Exit to operating system.");
    printf("\n\n\n\t   Enter the number corresponding to your choice ");
    gofor(men,1);
    switch (men[0])
     {
  case '1':
     cls();locate (4,10);offset=0;
     if (init != 1)
       search ();
       locate (8,8);
       srch2 ();on=0;
       label:
       if (loc == -1 )
        {
        cls ();locate (6,4);
        printf("No match found, press return for menu ");
        gofor(men,1);break; /* force return to menu */
        }
       seek(fd,offset,0);
         for ( i=0; i < RECORD ; ++i) /* get a record, a char at a time */
            c[i] = getc(fd);
       cls();locate (2,6);printf("Data for record no. %d",offset/128+1);
       locate (4,8);
       printf("Company: ");locate (6,8);  /* print headings to screen */
       printf("Name: ");
       locate(8,8);printf("Street: ");
       locate(10,8);printf("City, State, Zip: ");
       locate (12,8);printf("Phone: ");
       locate (4,18);
       for (i=0; i <=24; ++i)  /* insert data under correct heading */
          putchar(c[i]);
       locate (6,15);
       for ( i=25; i <= 49; ++i)
          putchar(c[i]);
        locate (8,17);
       for ( i=50; i <= 84; ++i)
          putchar(c[i]);
```

```
        locate (10,27);
        for ( i=85; i <= 116; ++i)
          putchar(c[i]);
        locate (12,16);
        for ( i=117; i <= 127; ++i)
        putchar(c[i]);
        do
        {
        locate(16,1);putchar(27);putchar(74);locate(16,4);
        printf("You may (E)dit, (R)eplace, (P)rint, (C)ontinue\
 or (M)enu ");
        gofor(men,1);ch=men[0];ch=toupper(ch); /* allow for either */
        switch(ch)                    /*  upper or lower case entry */
        {
        case 'P':
        open_p();lprint (); /* open printer, print one record */
        fclose(f2); /* close printer */
        break;
        case 'C':
        loc /= 50;loc +=1;loc *= 50;loc = jindex(string,ptr,loc);
        offset=loc/(LEN1+LEN2)*RECORD;   /* bump pointer */
        goto label;                      /* and establish new offset */
        break;
        case 'R':
        goto replace;
        break;
        case 'E': init=0;
        locate(14,1);putchar(27);putchar(74);fclose(fd);fd=fopen("data","u");
        locate(16,4);printf("Please indicate the field you wish to edit\n");
printf("\n(1) Company, (2) Name, (3) Street, (4) City-State-Zip, (5) Phone ");
        gofor(men,1);
          switch(men[0])
          {
          case '1':
          locate(20,4);printf("Company: _____________________________¦");
          locate(20,13);gofor(compny,LEN1-1);seek(fd,offset,0);
          fprintf(fd,"%-25s",compny);
          goto done;
          case '2':
          locate(20,4);printf("Name: _____________________________¦");
          locate(20,10);gofor(name,LEN2-1);seek(fd,offset+25,0);
          fprintf(fd,"%-25s",name);
          goto done;
          case '3':
          locate(20,4);printf("Street: ___________________________________¦")
          locate(20,12);gofor(street,34);seek(fd,offset+50,0);
          fprintf(fd,"%-35s",street);
          goto done;
          case '4':
          locate(20,4);
          printf("City, State, Zip: _________________________________¦");
          locate(20,22);gofor(city,32);seek(fd,offset+85,0);
          fprintf(fd,"%-33s",city);  /* fix */
          goto done;
          case '5':
          locate(20,4);printf("Phone: __________¦");
          locate(20,11);gofor(phone,10);seek(fd,offset+118,0); /* fix */
          fprintf(fd,"%-10s",phone);
          done: seek(fd,0,2);fclose(fd);break;
          } /* end of edit switch */
        } /* end of menu switch */
       } /* end of do-while switch */
       while (ch != 'M');
       break;

 case '2':
    cls ();locate (4,1);j=1;
    opnforrd ();offset=0;
  while ( c[0] != DEL )
     {
     seek (fd,offset,0);
    for ( i=0; i<=LEN1+LEN2-1 ; ++i)
        c[i] = getc(fd);
      if ( c[0] > ' ' )
      {
      for ( i=0 ; i <= LEN1-1 ; ++i, ++j )
        putchar(c[i]);
      }
      else
      {
      for ( i=LEN1 ; i <= LEN1+LEN2-1 ; ++i,++j )
      putchar(c[i]);
      }
        if (j == 76 )
           {
          printf("\n");j=1;
           }
      offset += RECORD;
      if (j > 24 && c[0] != DEL )
      {
      putchar(',');putchar(' ');
      }
     }  /* end of while loop */
     printf("\n\tEnd of records\n");
    printf("\n\n\tPress <RETURN> to return to menu ");
    gofor(dummy,1);
    fclose(fd);offset=0;c[0]='\0'; /* reset offset, purge c[0] */
    break;
case '3':
      fclose(fd);  /* close file if open */
      cls();locate(6,6);printf("Please be patient, I'm looking for the");
      locate(8,6);printf("the end of this file ");init=0;
      for( i=0; i <= DELAY ; ++i );
   opnforrd();offset=0;
   seek(fd,offset,0);
   while ( string[0] != DEL )
      {
      for (i=0; i <=127; ++i)
         {
       string[i]=getc(fd);
         }
      offset +=128;seek(fd,offset,0);
      }
   offset -= 128;fclose(fd);string[0]='0'; /* Purge string[0] */
/**/ replace:
  for ( ; ; )
     {
       cls ();locate (4,10);printf("Mail List\n");o_pen ();locate(6,6);
```

```
      printf("Enter Company (if applicable) _____¦");
      locate(8,6);
      printf("Enter name _____¦");
      locate(10,6);
      printf("Enter Street _____¦");
      locate(12,6);
      printf("Enter City, State, Zip _____¦");
      locate(14,6);
      printf("Enter Phone _____¦");
      locate (6,36);gofor(compny,LEN1-1);locate(8,17);gofor(name,LEN2-1);
      locate(10,19);gofor(street,34);locate(12,29);gofor(city,32);
      locate(14,18);gofor(phone,10);
      seek(fd,offset,0);
      fprintf(fd,"%-25s%-25s%-35s%-33s%-10s",compny,name,street,city,phone);
      if ( ch == 'R' )
        goto skip;
      seek(fd,offset+128,0);
      fprintf(fd,"%c%cLast Record              ",DEL,EOF);  /* fix */
                      /* insert delimiter & end of file marker */
      skip:init=0;     /* jump here after replace function, force init */
      seek(fd,0,2);fclose(fd);locate (16,4);
      printf("Enter 0 to exit, any other character to continue ");
      gofor(men,1);
      offset +=128;
      if ( men[0] == '0') /* test for more input activity */
      break;
    }      /* end of for loop */
   break;   /* two "break" statements necessary here, one for "for" loop
            and one to delimit the "case" */
  case '4':
      opnforrd();open_p();
      offset=0;
      while ( c[0] != DEL )
      {
      seek(fd,offset,0);
      lprint ();offset+=RECORD;
      }
      fclose(f2);c[0]='\0'; /* close file, purge c[0] */
      break;

case '5':                 '
      opnforrd();init=0;
      offset=0;seek(fd,offset,0);q=0;
      while ( c[0] != DEL)
      {
      for ( i=0; i <= 127; ++i,++q)
      c[i]=getc(fd);
      offset+=128;seek(fd,offset,0);
      }
      q/=128;q-=1;c[0]='\0';   /* purge c[0] */
      cls();locate(6,1);fclose(fd);
      printf("Please indicate the field to be used for sort function\n");
      printf("N(ame) or Z(ip) ");gofor(dummy,1);dummy[0]=toupper(dummy[0]);
      switch (dummy[0])
       {
      case 'N':
      pad=LEN1+LEN2-1;break;
      case 'Z':
      pad=ZIP;break;
      default: pad=ZIP;break;
       }
      preparec(q);
      locate(10,1);printf("Stand by, I'm writing the sorted file to disk");
      writerec(q);
      unlink("data");rename("temp","data"); /* erase data file, rename */
      break;                              /* temp file to data */
       }  /* end of switch loop */
   } /* end of menu do - while loop */
   while ( men[0] != '6');
}   /* end of main */
#include "stdlib.c"
```

Listing 2

```
 /* --- "CPOWER" by John P. Lewis --- */
 /*  source code for part six,   11/20/86  */
/* *** NOTE!! This is NOT a stand alone program ***
   This code is for insertion into existing program, CPOWER
   Any redundant code is included for clarity only.
   The code found below "case '5':" is to replace existing
   "case '5':" code (within main). The functions are to be
   placed (inserted) above main. **** */

#define SPACE 32   /* space char used for fill */
#define LEN 7      /* length of comparison string */
#define NUMREC 200  /* number of records */
#define ZIP 115   /* reference value used to extract zip from record */

int init, pad, flag;
char ptr[LEN1], string[SIZE], c[RECORD], dummy[2], men[2], holdit[512];
                        /* holdit = buffer for reading (4)records */

preparec(gap)
   int gap;      /* this function retrieves a record & parses  */
   {            /* the field to be compared, stores last string(in field) */
   char recnum[7]; /* & record number in memory for future sort */
   int i, k, j, rnum, len, gp;gp=gap;k=0;offset=0;opnforrd();
   while ( gp > 0 )
     {
     seek(fd,offset,0);j=0;len=0;
       for( i=0 ; i < pad; ++i)
       c[i]=getc(fd); /* retrieve name field */
       i-=1;     /* bump past end */
       for( ; c[i] == SPACE ; --i );   /*  parse for spaces */
       for( ; c[i] != SPACE ; --i,++len ); /* look for non space char */
       i+=1;gp-=1;
       for( j=0; j < len && j < 7 ; ++j,++i )/*new string from last name */
       ptr[j]=c[i];                          /* or zip */
       for( ; j < 7 ; ++j )
       ptr[j]=SPACE;     /* pad ptr with spaces if less than 7 char */
       rnum=offset/RECORD;  /* derive record number */
       itoa(rnum,recnum);len=strlen(recnum);  /* convert record number */
       for( i=0 ; i < LEN ; ++i, ++k )  /* to ascii & get length */
       string[k]=ptr[i];
       for( i=0 ; i < len ; ++i, ++k )  /* store appended string in mem */
       string[k]=recnum[i];          /* at string[k]  */
       for( i=7+len ; i < 10 ; ++i,++k )
       string[k]=SPACE;    /* pad string to length = 10 */
```

```
      offset+=128; /* continue until all records have been read into mem */
      }
   offset=0;sort(gap);    /* call sort function */
}

sort (gap)
   int gap;  /* gap = num of records  */
   {
   unsigned i, j;q=gap;
   while ( gap >0 )
        {
        gap=(gap/2);        /* this is a modified shell sort */
        if( gap == 0 )
         break;
        for( j=gap; j < q; ++j,--flag )
        {
        i=j-gap;    /* i & j are incremented each iteration */
        get(i,j);
        }
       if ( flag > 0 && gap < 2 )
        {
        gap+=1;continue;  /* if swap was performed during last iteration */
        }                 /* increment gap and do it again */
       }
   }

get(i,j)
    int i, j;
    {
    unsigned offset2, k, temp, cmp;
    char strngl[11], strng2[11], fieldl[LEN+1], field2[LEN+1];
    offset2=10*j;offset=10*i;  /* i & j become locations in string[i] */
    i=offset;j=offset2;       /* for conversion back into strings for */
    for( k=0; k < 10 ; ++k,++i ) /* comparison */
    strngl[k]=string[i];
    for( k=0; k < 10 ; ++k,++j)
    strng2[k]=string[j];
    for( i=0; i < LEN ;++i)  /* shorten comparison string by leaving off */
        {                                          /* record number */
        fieldl[i]=strngl[i];field2[i]=strng2[i];
        }
    cmp=strcmp(fieldl,field2);
    if ( cmp == 1 )
        {
      swap(strngl,strng2,offset,offset2);flag=q;  /* swap if field 1 is > */
        }                            /* than field 2, reflect swap in flag */
    }

swap(strngl,strng2,offset,offset2)
   char strngl[], strng2[];  /* this function swaps strings in memory only */
   int offset,offset2;     /* to be used for computation of new offset */
    {
    char temp[11];
    int i, j;
    for ( i=0; i < 10 ; ++i)  /* strng one is read into temp */
    temp[i]=strngl[i];
    for( i=0; i < 10 ; ++i)
    strngl[i]=strng2[i];     /* strng two is read into strng one */
    for( i=offset; i < offset+10; ++i,++j)
    string[i]=strngl[j];j=0;    /* i = offset within string[ ], read strngl*/
    for( i=offset2; i < offset2+10; ++i,++j) /* into new location, likewise */
    string[i]=temp[j];   /* strng2, swap is now completed */
    }

writerec(no)
    int no;
    {
    int recnum[NUMREC], i, j, k, offset2, temp, last;
    char  retrieve[3], dump[7];
    last=temp=no;j=0;k=0;offset2=0;f2=fopen("temp","w");
    fclose(f2);open_t();fclose(fd); /* open temp file, close data */
    while ( no > 0 )
      {
      for ( i=0; i < 7 ; ++i, ++k ) /* retrieve & dump string[k] */
      dump[i]=string[k];
      for ( i=0; i < 3 ; ++i,++k ) /* retrieve ascii representation of */
      retrieve[i]=string[k];--no;  /* record number */
      recnum[j]=atoi(retrieve);++j;  /* convert it to numeric val */
      }                              /* store in subscripted recnum */
      no=temp;j=0;i=temp=0;offset2=0;
      while ( no > 0 )
        {
        opnforrd();k=0;
        for( i=temp; i <= temp+3 && no > 0 ; ++i,--no)
           {
           offset=recnum[i]*128;   /* convert recnum[i] into offset */
           seek(fd,offset,0);     /* retrieve records in sorted order */
           for ( j=0; j <= 127 ; ++j,++k)  /* store 4 in holdit buffer */
             holdit[k]=getc(fd);
           }
           k=0;temp+=4;fclose(fd); /* write sorted records to temp file */
           for ( i=0 ; i <= 3 && last > 0; ++i,--last )
            {
            for ( j=0 ; j <= 127 ; ++j,++k )
            string[j]=holdit[k];seek(f2,offset2,0);
            fprintf(f2,"%-128s",string);
            offset2+=128;
          }
        }
      seek(f2,offset2,0);fprintf(f2,"%c%cLast Record            ",DEL,EOF);
      seek(f2,0,2);fclose(f2);k=0;
    }

open_t()
   {
   f2=fopen("temp","u");  /* open destination sort file */
   }

  case '5':
        opnforrd();init=0;
        offset=0;seek(fd,offset,0);q=0;
        while ( c[0] != DEL)
        {
        for ( i=0; i <= 127; ++i,++q)
        c[i]=getc(fd);
        offset+=128;seek(fd,offset,0);
        }
        q/=128;q-=1;c[0]='\0';   /* purge c[0] */
```

```
        cls();locate(6,1);fclose(fd);
        printf("Please indicate the field to be used for sort function\n");
        printf("N(ame) or Z(ip) ");gofor(dummy,1);dummy[0]=toupper(dummy[0]);
        switch (dummy[0])
         {
        case 'N':
        pad=LEN1+LEN2-1;break;
        case 'Z':
        pad=ZIP;break;
        default: pad=ZIP;break;
         }
        preparec(q);
        locate(10,1);printf("Stand by, I'm writing the sorted file to disk");
        writerec(q);
        unlink("data");rename("temp","data"); /* erase data file, rename */
        break;                                /* temp file to data */
         }  /* end of switch loop */
   } /* end of menu do - while loop */
   while ( men[0] != '6');
}   /* end of main */
#include "stdlib.c"
```

*

SPREADSHEET Corner

Review #2

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

This article will continue the review series. I will discuss the Lotus 1-2-3 Release 2.0, the 8087 Co-Processor, and complete the discussion about the AST Research RAMpage! EMS/EEMS Board and the software furnished with the board, except for DESQview which will come in a later article.

Since Lotus Development introduced 1-2-3 with the biggest fan-fare I have ever seen in the computing industry, it received a tremendous momentum (over 1.5 million copies sold). This lavish display, plus an industry that was ready for a change from CP/M with 64K RAM and VisiCalc to the birth of THE Personal Computer and the Microsoft DOS operating system series! The Lotus 1-2-3 package was an immediate success with one setback requiring a revision to the 1A release which Lotus handled in a first class way. Over these past few years, the marketing of this 1-2-3 program greatly benefited the growth of the micro-computer! It gave rise to everything from multi-function boards that provided the features that the PC computers had left off (mainly up to 640K RAM) to a highly lucrative book, magazine and training sub-industry. In-fact, the 1-2-3 software so dominated the PC compatible business computer that the "MICRO" became the "DESKTOP" computer.

1-2-3 was so named because it performed three major functions. The spreadsheet (MOST important), allowed numbers, text and formulas to be entered so that they could be used for calculations, financial projections, etc.; second, the database provided sort and retrieved lists of data information (It did not put the other major software package, dBASE by Ashton-Tate back, but really reinforced dBASE); third, the Graphics feature provided a way to display data from the spreadsheet and database using line, bar, stacked-bar, pie, and line graphs (the old story that a picture is worth a hundred words).

When 1-2-3 1A first came out, most users did not know it's POWER! New uses appeared on a regular basis. Also, many add-on programs supplemented the package. Today, Lotus is still the KING of the pack! This does not say it is the BEST, but it is the pseudo-standard! It did not achieve this record by any pioneering improvements to the spreadsheet concept, but it was just the fastest and easiest program to manipulate. Its screen presentation of menus of commands was simpler and better than any other program around at that time! It had the largest work sheet; however, this feature quickly lost its glitter when the users realized that the normal memory limit of the early PC computer prevented use of but a small fraction of this worksheet. This caused a wave of methods of adding memory to the computer itself or by means of memory add-on boards. Soon, the 640K RAM memory limit was reached caused by the 8088 CPU ability to address only one megabyte of RAM and the DOS design itself.

To answer this limitation, Lotus, Intel and Microsoft worked together to devise an expanded memory system (EMS) board using "bank-switching" technics! This bank-switching has been around for some time. In fact, some of the readers might have employed this method to get past the 64K RAM limit of their H-8's!

With the new Expanded Memory Systems boards, 1-2-3 Release 2 can address up to 2 megabytes (8 megabytes possible) of RAM! The user will find that one megabyte is a practical limit when debugging and recalculation time is considered. Release 2 makes more efficient use of working memory using the "sparse matrix memory management" method. The 1-2-3 (1A) required careful memory management. It stored data in active cells of the worksheet; BUT, also used RAM for all the non-active cells within the rectangular boundary of the worksheet (between cell A1 and the last, lowest row/column cell). This was a waste of RAM memory! Release 2 corrects this problem using the sparse memory method; however, there is a price to pay for this. The new 1-2-3 program now takes up more memory when loaded into RAM. There is still a small memory net gain however!

In my mind, Lotus has hurt the valuable customer image that they had with the Release 2 Update for the following reasons:

1. The Update costs the user $150, plus the return of the Release 1A Systems Disk. High volume users (100 to 1000s) pay $100 to $125 per copy.

2. The Upgrade consists of a "cardboard box" (no slip-case) with "paperback" manuals! Have you ever tried to work with paperback books that will not lay flat and stay open to the page being used? No ringed, looseleaf or spiral bound manuals are supplied! This makes the product look "CHEAP"! I cannot understand why Lotus would want other users to see what has happened to their product when it lays on the user's desk. Also, how can a company that spent the kind of money that Lotus did making the big splash when they first announced 1-2-3 supplied with an impressive package, stoop so low on this upgrade? It is not like they are giving the upgrade away or losing money on it. Lotus IS losing money that they spent building that image!

3. Lotus took good care of the users with the Release 1A; however, when they announced Release 2, Lotus made a hasty withdrawal of the 1A even though almost 80% of the users did not need or want Release 2! This is a FORCED way to MAKE users upgrade and Lotus makes a big profit at $150 per upgrade with a cardboard box and paperback manuals! What do the readers think?

4. Release 2 worksheets are NOT compatible with the 1A or Symphony worksheets and this will require users to rework all existing worksheets at an additional cost!

5. The user can expect a Release 2.1 before long which hopefully will solve compatibility problems! Lotus has not announced this as yet or are they telling the users whether they will be charged for this upgrade made necessary by their errors? I have heard that the changes have been made and that this time they are carefully checking the changes before this upgrade is released! I hope so.

I will now list some of the compatibility problems that I have found with Release 2:

1. Release 1A treats cells with labels as zeros when referenced in formulas causing the upgrade to return an error message.

2. Release 2 has memory problems with some resident loaded programs, like SideKick, when used with EMS boards.

3. Release 2 takes 2 to 4 times longer to load a worksheet as compared to 1A.

4. The Translate facility that converts one release file to the other is very time consuming with some files depending on what commands are used. Yet the user will have to change all their files.

My conclusion about the Release 2 Upgrade would be to wait until the upgrade period is nearly ended before upgrading, unless the user needs the new features now. Hopefully, this will save the user new problems if the new Release 2.1 comes out before the end of this period!

Now, I had better start telling what the new features are. At first glance, the Release 2 looks and works like the older version and the retraining will be limited to how many of the new features will be used. Not all users will need or want to use them.

1. I have already discussed the memory management changes. If you are one of the users that have run up against the 640K RAM limit, this update is for you! Remember that an EMS board will be required, another extra cost!

2. This upgrade allows the user to reach DOS to run another program or DOS function and return to 1-2-3 by typing "EXIT" and be back to the worksheet exactly where you left off.

3. The Copy Protection is still there, (unhappy as it is) but if the user has a hard disk it will not be necessary to put the Systems disk into the floppy drive A every time the 1-2-3 program is run. This has complicated the backup task. So, which is the worst? That is a subject to be discussed in some later article.

4. The new release can be used on Local Area Networks where several PC compatible computers are linked together. This is a specialized project that is not in the scope of this article.

5. More Symphony commands have been added and the large Symphony (8191 Rows) worksheet has been provided. Again, Release 2 is not compatible with Symphony 1.1, but 1-2-3 users will find it easier to change over to Symphony if that is desired. Here again, Symphony 1.1 will soon have another upgrade! Who knows what that will do? I will be discussing Symphony 1.1 in a following article.

6. The INTEL 8087 or 80287 Math Co-Processors are now supported. I will discuss what these chips will do for the user a little later in this article.

7. Release 2 now offers "string" functions! How many readers know what this feature will do? I will try to explain with an example using very little space. If the worksheet was set up to figure the total cost of purchasing a home and a ceiling of $125,000 was set to determine whether the buyer could buy; we could do the following. We would use cell E20 for the total and a message could be in cell F20. The formula in cell F20 could be like this:

```
@IF(E20>120000,"OVER BUDGET","OK! BUY")
```

   This is a string function which may provide better readability. Do you think so? I do!

8. There are about 40 new, advanced MACRO commands that have been added. This is GOOD or BAD! This is one big source of incompatibility depending on what MACRO commands have been used in the prior project.

9. The Upgrade provides Password Protection to a worksheet to prevent unauthorized use of say a Payroll!

10. Many new statistical and financial functions have been added, including depreciation calculations as an example.

11. The new software can perform regression analysis, which shows the relationship between two or more sets of data.

12. Exploded PIE graphics have been added (my favorite).

13. Database uses are upgraded to 8191 records and a Data Query Find command is now available.

14. Have you ever had a "circular reference"? Did you have trouble finding where it originated? Release 2 will now tell you which cell to start debugging.

Even though I haven't discussed all the problem areas, I like the new Update and many of the features will be useful. I like the new MACRO commands. They allow the user to design a project that any novice can use without getting into problems (truly "user friendly")! The new financial functions will open up new applications, like depreciation, which was not easy to do. Using 1-2-3 with the EMS boards gives the user a nice feeling that memory will not be

a problem! One of the BIG problems with 1-2-3 and still not solved is the continuing absence of a practical way to create linked worksheets! Example, if the user would want to link sub-worksheets to one consolidated re-cap worksheet. This is necessary in many business applications (Budgets by quarters and by the year; P & L statements by month, quarter and year; cash-flow by departments and for the complete company, etc.)! The EMS board provides the memory so that each sub-worksheet could be on one big large worksheet and then brought together into one by a Macro and recalculation. However, I find that if over one megabyte of memory is used, the recalculation time seems like it will never finish! This is where the user must consider the faster AT compatible computers. I am getting tempted.

Another handy new feature that many users might find a need for takes another step over the Password Protection. It is the Hidden cells/columns that can be used for security. Example, pay rates could be hidden! Best of all, I find that the newly added MACRO commands provide 1-2-3 with a TRUE programming language so that the user CAN BUILD UNIQUE WORKSHEETS!

In conclusion, I would state that this costly, cheaply packaged upgrade Release 2 is worthwhile to me. I will find many ways that it will be helpful, but this is not the conclusion everyone would find. It boils down to whether the Release 1A meets your needs. In the next article, I will offer an alternative that the readers may want to consider. It is the SuperCalc3 Release 2.1! I will discuss it's advantages and disadvantages. For now, I will say its cost would only be slightly more than the cost of the 1-2-3 Release 2 update and it uses the EMS boards, the 8087 co-processor, and the Graphics are better than 1-2-3! You might want to wait until the next article.

Note! I have just found out that a new SuperCalc4 version will be out about the end of June. I will obtain the new upgrade as soon as possible and I will learn about the new changes and report them to the readers in the same article. So, I will hold up this article and write about Symphony next. In the meantime, possible users of SuperCalc4 may wish to wait for this later article! It has a fine new package with updated manuals. It will be similar to 1-2-3 in many ways. The only bad thing is that it will sell for more money. I will report on that as well. It is not Copy Protected!

All the new spreadsheet software will now work with the 8087 or 80287 co-processors. When do you need this expensive chip? Do not add a co-processor simply because the software will work with the chip! If most of the spreadsheets that you use require the basic four math functions — addition, subtraction, division, and multiplication — the recalculation speed will gain about 10%. If the spreadsheet uses transcendental functions with logic operations (if-then analysis), the speed-up will be beneficial. The transcendental functions include exponential, logarithm, statistics, and trigonometric. The co-processors will help a small amount with division and multiplication. They really speed-up the more complex math, like exponentiation (process of raising a value to a power) in statistical analysis. This is where you will find a gain of a factor of 10 to 20. Therefore, most users can better spend the $125 to $295 expense for a better purpose!

Now, here is some more information about RAMpage! Since I wrote the last article, AST Research Inc. has received the Lotus Development Corporation's "CERTIFICATION" for operation with 1-2-3 Release 2 and Symphony 1.1. RAMpage already had the support and approval for Ashton-Tate's Framework II, Digital Research's Concurrent DOS operating system, and Quarterdeck Office System's DESQview, as well as all the other applications written to meet the Lotus/Intel/Microsoft expanded memory specification (EMS).

RAMpage! comes with an extensive "SuperPak" software package that includes the following:

1. Expanded Memory Manager (REMM).
2. Extended Memory Emulator (REX).
3. SuperDrive (A RAM diskette drive simulator).
4. fASTdisk (A RAM simulated hard (virtual) disk).
5. DESQview 1.1 (A multi-tasking/windowing program).

The REMM software driver (part of the CONFIG.SYS file) swaps memory between the RAMpage board and the computer's RAM by creating pointers, loading the registers, and mapping the computer's windows to RAMpage Extended Memory. REMM also makes a test on the RAM memory when the computer is first turned on and prevents any non-working memory from being used by RAMpage! The Expanded memory application programs or operating environment software must keep track of what Page of RAMpage memory is holding certain data or program code so that it can be retrieved when requested. The parameters supplied by the Expanded Memory Manager (all EMS

boards have this file with similar names) links windows in the computer's logical memory to Pages of RAMpage physical memory by means of 64 mapping registers, the map control register, and the Page register.

REMM also allocates RAMpage memory to several Process IDs, also known as Expanded Memory Handles. One or more Process IDs are allocated to a particular applications program, and each Process ID has certain Pages assigned to it. The Process IDs aid in multi-tasking, such as used by DESQview (discussed in a coming article).

Before the memory mapping is enabled, REMM automatically maps the 16KB Pages of RAMpage memory that may be required to fill the 640KB conventional DOS RAM. The Addressing RAMpage memory as part of the 640K RAM provides the full conventional memory that is required by many application programs or combinations of programs.

REX interfaces with the REMM program to make RAMpage Expanded memory act like the PC-AT Extended memory. REX permits the User to operate "RAM Drivers" like AST's SuperDrive, SuperSpool (print spooler), and fASTdisk from Expanded memory (fASTdisk is very much like VDISK in DOS 3.xx).

REX must follow REMM in the CONFIG.SYS file and it cannot be used without the REMM program. REX must be installed before SuperDrive, SuperSpool, and/or fASTdisk, or else they cannot be used in Expanded memory.

SuperDrive program can simulate up to four diskette drives if memory will permit. This RAMdisk capability retrieves and stores data and files within RAM much faster than using floppy disks and to a lessor extent hard disk drives. Application programs that require many disk accesses will be the best candidates. Programs using overlays are good examples.

SuperSpool program outputs data to a printer from a RAM file while the user can simultaneously perform other computer tasks. The print output is stored in a predefined area of memory and sent from there to the printer nearly invisible to the user. The CPU (microprocessor) will waste less of its valuable processing time waiting for the printer. This means that the user can execute another application program in the "foreground" while the printing is handled concurrently in the "background".

fASTdisk program simulates hard disk units within RAM. This is a virtual disk that retrieves and stores data and files at RAM speeds instead of slower hard disk drives. The number and/or size of fASTdisk are limited by the amount of RAM available and/or by the operating system. An installation command format compatible with the DOS VDISK is provided. Sample AUTOEXEC and CONFIG files are furnished so that the user can use them directly by copying them to their drive or modify them to fit their needs.

This makes RAMpage! easy to install and very flexible. It is very easy to use, as well. RAMpage! with this software and the DESQview, which I haven't found room to write about as yet, makes this EEMS/EMS board the BEST that I have tested! DESQview is a very interesting and useful program that really enriches RAMpage! RAMpage! comes in a PC/XT and a PC-AT version with about the same software.

One warning is in order at this point! Like everything else, there can be a problem. After installation of an EEMS/EMS board, the user can still encounter a "memory full" error. It does not depend on the amount of Expanded memory installed, but it depends on what other programs the user has installed in the 640K conventional memory. It also depends on the type of data in the spreadsheet worksheet. Three types of entries are stored in Expanded memory. They are:

1. Labels.
2. Formulas.
3. Floating-point numbers.

Every other cell entry is still kept in conventional memory along with DOS and the 1-2-3 or Symphony programs, drivers, add-in programs, etc. I have not run into this as yet; BUT, it could happen. Keep this in mind!

My Framework II Word Counter tells me that I am over 3000 words already. I will end this article at this point. Please let me know about any ideas, corrections, etc. If you wish a reply from me, remember to send along an SASE (business size) for my personal answer or comments. I will include any general information in future articles for all readers to see. HAPPY SPREADSHEETING! *

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

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

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

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

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

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

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

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

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

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

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

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

★★★ ZENITH SOFTWARE FOR THE ★★★ Z-100 SERIES COMPUTERS

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

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

★★★ CHIP SPECIALS ★★★

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

64K Dynamic RAM, 150 ns .....$1.35 each
256K Dynamic RAM, 150 ns .....$3.28 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

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

SmartWatch from FBE Research. Installs in ROM Socket on CPU Board in Zenith computer series Z-100/150/148/158. This tiny jewel of a product contains a ten year battery and keeps your computer informed of both time and date at each boot-up. Complete instructions and software included $44.00

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

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

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

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

★★★ ZENITH SOFTWARE FOR THE ★★★ Z-150/160 SERIES COMPUTERS

PART NUMBER	DESCRIPTION	LIST PRICE	SALE PRICE
RS-463-75	PeachText 5000	$395.00	$77.00
LEXISOFT	Spellbinder	$495.00	$292.00
LEXISOFT	Desktop Publisher	$695.00	$445.00
BP-5063-8	BPI Personal Accounting	$195.00	$55.00
SR-B410	DS BACKUP (hard disk)	$70.00	$62.50
SR-4500	Norton Utilities 4.0	$99.00	$69.00
SR-3950	dBase III plus	$695.00	$479.00
SR-4100	DAC EASY Accounting	$69.00	$62.00
SR-3090	LOTUS 123	$499.00	$389.00

★★★ UPGRADE ACCESSORIES FOR H/Z-89 ★★★ COMPUTERS

SC-837-1 SuperCalc $195.00 $40.00

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

INTERNAL DUAL DRIVE SETUPS. Includes two half height double sided disk drives and all hardware and connectors required to mount inside your H-89. Steel mounting shield/case included.
MITSUBISHI MF501 Setup, 48 TPI, 6 MS seek, 320K $262.00
MITSUBISHI M4853 Setup, 96 TPI, 3 MS seek, 640K disks ..$269.00

★★★ HALF HEIGHT FLOPPY ★★★ DISK DRIVES

MITSUBISHI 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
MITSUBISHI M4853	5.25"	96 TPI DS/DD	640K	..... $96.00

★★★ LAPINE HARD DISK DRIVES ★★★

TITAN 20 MEG Winchester Hard Disk $319.00
With Controller & Cables $395.00
TITAN 30 RLL Winchester Hard Disk $349.00
With RLL Controller & Cables $442.00

★★★ SEAGATE HARD DISK DRIVES ★★★

ST-225 20 MEG Winchester Hard Disk $319.00
With Western Digital Controller & Cables $395.00
ST-238 30 MEG, Requires RLL type controller $329.00
With RLL Controller & Cables $424.00

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.

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

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

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

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

HUG Price List

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
H8 — H/Z-89/90				
ACCOUNTING SYSTEM	885-8047-37	CPM	BUSINESS	20.00
ACTION GAMES	885-1220-[37]	CPM	GAME	20.00
ADVENTURE	885-1010	HDOS	GAME	10.00
ASCRITY	885-1238-[37]	CPM	AMATEUR RADIO	20.00
AUTOFILE (Z80 ONLY)	885-1110	HDOS	DBMS	30.00
BHBASIC SUPPORT PACKAGE	885-1119-[37]	HDOS	UTILITY	20.00
CASTLE	885-8032-[37]	HDOS	ENTERTAINMENT	20.00
CHEAPCALC	885-1131-[37]	HDOS	SPREADSHEET	20.00
CHECKOFF	885-8010	HDOS	CHECKBOOK SOFTWARE	25.00
DEVICE DRIVERS	885-1105	HDOS	UTILITY	20.00
DISK UTILITIES	885-1213-[37]	CPM	UTILITY	20.00
DUNGEONS & DRAGONS	885-1093-[37]	HDOS	GAME	20.00
FLOATING POINT PACKAGE	885-1063	HDOS	UTILITY	18.00
GALACTIC WARRIORS	885-8009-[37]	HDOS	GAME	20.00
GALACTIC WARRIORS	885-8009-[37]	CPM	GAME	20.00
GAMES 1	885-1029-[37]	HDOS	GAMES	18.00
HARD SECTOR SUPPORT PACKAGE	885-1121	HDOS	UTILITY	20.00
HDOS PROGRAMMERS HELPER	885-8017	HDOS	UTILITY	16.00
HOME FINANCE	885-1070	HDOS	BUSINESS	18.00
HUG DISK DUPLICATION UTILITIES	885-1217-[37]	CPM	UTILITY	20.00
HUG SOFTWARE CATALOG	885-4500	VARIOUS	PRODUCTS THRU 1982	9.75
HUGMAN & MOVIE ANIMATION	885-1124	HDOS	ENTERTAINMENT	20.00
INFO. SYSTEM AND TEL. & MAIL SYSTEM	885-1108-[37]	HDOS	DBMS	30.00
LOGBOOK	885-1107-[37]	HDOS	AMATEUR RADIO	30.00
MAPLE	885-8005	HDOS	COMMUNICATION	35.00
MAPLE	885-8012-[37]	CPM	COMMUNICATION	35.00
MICRONET CONNECTION	885-1122-[37]	HDOS	COMMUNICATION	20.00
MISCELLANEOUS UTILITIES	885-1089-[37]	HDOS	UTILITY	20.00
MORSE CODE TRANSCEIVER	885-8016	HDOS	AMATEUR RADIO	20.00
MORSE CODE TRANSCEIVER	885-8031-[37]	CPM	AMATEUR RADIO	20.00
PAGE EDITOR	885-1079-[37]	HDOS	UTILITY	25.00
PROGRAMS FOR PRINTERS	885-1082	HDOS	UTILITY	20.00
REMARK VOL 1 ISSUES 1-13	885-4001	N/A	1978 TO DECEMBER 1980	20.00
RUNOFF	885-1025	HDOS	TEXT PROCESSOR	35.00
SCICALC	885-8027	HDOS	UTILITY	20.00
SMALL BUSINESS PACKAGE	885-1071-[37]	HDOS	BUSINESS	75.00
SMALL-C COMPILER	885-1134	HDOS	LANGUAGE	30.00
SOFT SECTOR SUPPORT PACKAGE	885-1127-[37]	HDOS	UTILITY	20.00
STUDENT'S STATISTICS PACKAGE	885-8021	HDOS	EDUCATION	20.00
SUBMIT (Z80 ONLY)	885-8006	HDOS	UTILITY	20.00
TERM & HTOC	885-1207-[37]	CPM	COMMUNICATION & UTILITY	20.00
TINY BASIC COMPILER	885-1132-[37]	HDOS	LANGUAGE	25.00
TINY PASCAL	885-1086-[37]	HDOS	LANGUAGE	20.00
UDUMP	885-8004	HDOS	UTILITY	35.00
UTILITIES	885-1212-[37]	CPM	UTILITY	20.00
UTILITIES BY PS	885-1126	HDOS	UTILITY	20.00
VARIETY PACKAGE	885-1135-[37]	HDOS	UTILITY & GAMES	20.00
VOLUME I	885-1008	N/A	SOFTWARE LISTINGS	9.00
VOLUME II	885-1013	N/A	SOFTWARE LISTINGS	12.00
VOLUME III	885-1015	N/A	SOFTWARE LISTINGS	9.00
VOLUME IV	885-1037	N/A	SOFTWARE LISTINGS	12.00
WATZMAN ROM SOURCE & DOC	885-1221-[37]	CPM	H19 FIRMWARE	30.00
WATZMAN ROM	885-4600	N/A	H19 FIRMWARE	45.00
WHEW UTILITIES	885-1120-[37]	HDOS	UTILITY	20.00
XMET ROBOT X-ASSEMBLER	885-1229-[37]	CPM	UTILITY	20.00
Z8C 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

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
REMARK VOL 7 ISSUES 72-83	885-4007	N/A	1986	25.00
RF CAD	885-8020-[37]	CPM	UTILITY	30.00
SEA BATTLE	885-1211-[37]	CPM	GAME	20.00
UTILITIES BY PS	885-1226-[37]	CPM	UTILITY	20.00
UTILITIES	885-1237-[37]	CPM	UTILITY	20.00
X-REFERENCE UTILITIES FOR MBASIC	885-1231-[37]	CPM	UTILITY	20.00
ZTERM	885-3003	CPM	COMMUNICATION	20.00
H/Z-100 (Not PC) Only				
ACCOUNTING SYSTEM	885-8048-37	MSDOS	BUSINESS	20.00
CALC	885-8043-37	MSDOS	UTILITY	20.00
CARDCAT	885-3021-37	MSDOS	BUSINESS	20.00
CHEAPCALC	885-3005-37	MSDOS	SPREADSHEET	20.00
CHECKBOOK MANAGER	885-3013-37	MSDOS	BUSINESS	20.00
CP/EMULATOR	885-3007-37	MSDOS	CPM EMULATOR	20.00
DBZ	885-8034-37	MSDOS	DBMS	25.00
ETCHDUMP	885-3005-37	MSDOS	UTILITY	20.00
EZPLOT	885-3023-37	MSDOS	PRINTER PLOTTING UTILITY	20.00
FAST EDDY	885-8025-37	CPM	TEXT PROCESSOR	20.00
FAST EDDY	885-8029-37	MSDOS	TEXT PROCESSOR	20.00
GAMES CONTEST PACKAGE	885-3017-37	MSDOS	GAMES	25.00
GAMES PACKAGE II	885-3044-37	MSDOS	GAMES	25.00
GRAPHICS	885-3031-37	MSDOS	ENTERTAINMENT	20.00
HELPSCREEN	885-3039-37	MSDOS	UTILITY	20.00
HEPCAT	885-3045-37	MSDOS	UTILITY	35.00
HUG BACKGROUND PRINT SPOOLER	885-1247-[37]	CPM	UTILITY	20.00
HUG BACKGROUND PRINT SPOOLER	885-5009-37	CPM86	UTILITY	20.00
HUGPBBS	885-5006-37	CPM86	COMMUNICATION	40.00
HUGPBBS SOURCE LISTING	885-5007-37	CPM86	COMMUNICATION	60.00
ICT 8080 TO 8088 TRANSLATOR & HFM	885-5008-37	CPM86	UTILITY	20.00
KEYMAP	885-3010-37	MSDOS	UTILITY	20.00
KEYMAP	885-5001-37	CPM86	UTILITY	20.00
KEYMAP CPM-85	885-1245-37	CPM	UTILITY	20.00
MAPLE	885-8023-37	CPM	COMMUNICATION	35.00
MATHFLASH	885-8030-37	MSDOS	EDUCATION	20.00
ORBITS	885-8041-37	MSDOS	EDUCATION	25.00
POKER PARTY	885-8042-37	MSDOS	ENTERTAINMENT	20.00
SCICALC	885-8028-37	MSDOS	UTILITY	20.00
SKYVIEWS	885-3015-37	MSDOS	ASTRONOMY UTILITY	20.00
SMALL-C COMPILER	885-3026-37	MSDOS	LANGUAGE	30.00
SPELL5	885-3035-37	MSDOS	SPELLING CHECKER	20.00
SPREADSHEET CONTEST PACKAGE	885-3017-37	MSDOS	VARIOUS SPREADSHEETS	25.00
TERM86 & DSKED	885-5004-37	CPM86	COMMUNICATION & UTILITIES	20.00
TREE-ID	885-3036-37	MSDOS	TREE INDENTIFIER	20.00
USEFUL PROGRAMS I	885-3022-37	MSDOS	UTILITIES	30.00
UTILITIES BY PS	885-5003-37	CPM86	UTILITY	20.00
UTILITIES	885-3008-37	MSDOS	UTILITY	20.00
ZBASIC DUNGEONS & DRAGONS	885-3009-37	MSDOS	GAME	20.00
ZBASIC GRAPHIC GAMES	885-3004-37	MSDOS	GAMES	20.00
ZBASIC GAMES	885-3011-37	MSDOS	GAMES	20.00
ZPC II	885-3037-37	MSDOS	PC EMULATOR	60.00
ZPC UPGRADE DISK	885-3042-37	MSDOS	UTILITY	20.00
H/Z-100 — PC Compatibles				
ADVENTURE	885-3016-37	MSDOS	GAME	10.00
ASSEMBLY LANGUAGE UTILITIES	885-8046-37	MSDOS	UTILITY	20.00
DEBUG SUPPORT UTILITIES	885-3038-37	MSDOS	UTILITY	20.00
DOCUMAT & DOCULIST	885-8035-37	MSDOS	TEXT PROCESSOR	20.00
DPATH	885-8039-37	MSDOS	UTILITY	20.00
HADES	885-3040-37	MSDOS	UTILITY	40.00
HELP	885-8040-37	MSDOS	CAI	20.00
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/Z	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***.

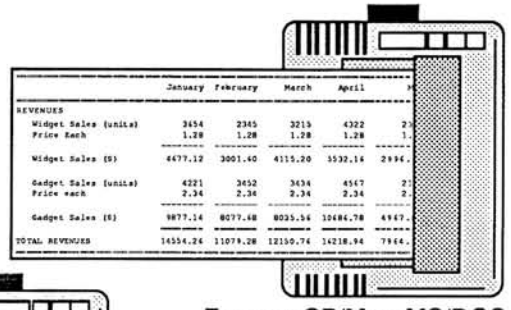
"Thank Heaven for *HADES*!" That's what a lot of MS-DOS users are saying when ***HADES*** rescues a file that just got accidentally erased. Erased file recovery is only a small part of the capabilities of this program. *HADES* is *HUG's Absolute Disk Editing System*. Within the realms of MS-DOS, *HADES* 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***.

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.

LONG & LOUD!

Sideways & Banner Printing Utility for Dot-Matrix Printers



	January	February	March	April
REVENUES				
Widget Sales (units)	3454	2345	3215	4322
Price Each	1.28	1.28	1.28	1.28
Widget Sales ($)	4677.12	3001.40	4115.20	5532.16
Gadget Sales (units)	4221	3652	3434	4547
Price each	2.34	2.34	2.34	2.34
Gadget Sales ($)	9877.14	8077.68	8035.56	10686.78
TOTAL REVENUES	14554.26	11079.28	12150.76	16218.94

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

$34.95

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

$49.95

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

PRESTO!™

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

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

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

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

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

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

Rembrandt

Complete Business Graphics Toolkit™

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

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

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

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

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

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

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

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

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

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

T E C H N O L O G I E S, I N C.

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%)

10 – Very Good
9 – Good
8 – Average

TABLE C
Product Rating

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

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

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

ORDERING INFORMATION

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

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

NOTES

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

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

HUG P/N 885-3046-37
KEYMAC Keyboard Macro Processor $20.00

Introduction: KEYMAC is a keyboard macro processor for H/Z-100 (not PC) series computers that is similar to such programs as PROKEY (tm) or SUPERKEY (tm) that are available for PC-compatible computers. With KEYMAC, you can program any key on the keyboard to produce up to 100 characters when it is pressed. For example, you could program the F0 key to produce "Acme Software Company" each time it is pressed. Defined keyboard macros are stored in files, and you can prepare any number of them for use in different situations. You can prepare macro definition files using a special utility, or you can store up keystrokes "on the fly", and store them in a definition file later. KEYMAC will work with just about any program, including difficult ones like the Z-100 version of Lotus 1-2-3 (tm).

Requirements: KEYMAC requires an H/Z-100 series (not PC) computer or an expanded ET-100 computer, any version of MS-DOS or Z-DOS, and at least 128k of system memory. KEYMAC itself uses less than 8k of memory.

Author: Patrick Swayne

The KEYMAC disk contains the following files:

README	.DOC	KEYMAC	.DOC
KEYMAC	.COM	MAKEMAC	.COM
WS	.KM	DOS	.KM
BASIC	.KM	NULL	.KM
KEYMAC	.ASM	MAKEMAC	.ASM

Here is an explanation of the files:

KEYMAC.DOC — Instructions for using KEYMAC.

KEYMAC.COM — The KEYMAC program. This program installs itself into memory the first time you run it. After that, it can be used to load macro definition files as they are needed.

MAKEMAC.COM — This program is used to create macro definition files. In addition to allowing you to define the keys, it allows you to create a prompt line for the function keys that will appear on the 25th screen line while your macro file is loaded and active.

WS.KM — A ready-made macro definition file for use with WordStar.

BASIC.KM — A macro definition file for use with BASIC, that contains many BASIC keywords programmed into the function keys.

DOS.KM — A macro definition file with DOS commands programmed into the function keys, including commands to load the other macro definition files on the disk.

NULL.KM — A special macro definition file that simply removes any previous definitions so that all keys work as they do when KEYMAC is not installed.

KEYMAC.ASM, MAKEMAC.ASM — These are the assembly source files for KEYMAC and MAKEMAC.

TABLE C Rating: (2,3,10)

*

NEW PRICES!
NEW PRODUCTS!
NO SURCHARGES
SERVICE & SUPPORT

DYNAMIC RAM MEMORY

Includes Installation Instructions. **ONE YEAR WARRANTY.** Prices current 7/1/87 and subject to change; call toll free for current prices.

QUANTITY PRICES AVAILABLE ON REQUEST.

4164-150 : 64K 150 ns. DRAM..... **$ 1.40**
41256-150: 256K 150 ns. DRAM..... **$ 3.28**
41256-120: 256K 120 ns. DRAM..... **$ 3.55**
41256-100: 256K 100 ns. DRAM..... **$ 4.40**

AT COMPATIBLE MEMORY UPGRADES

All Memory Expansion Cards come with Hardware, Installation Instructions, LIM/EMS Software, and a **ONE YEAR WARRANTY.** Fully populated boards available from 2MB. to 16MB., for systems up to 16MHz. The following is a sample:

2 MB Memory Expansion Board: Fully populated, expandable to 4 MB. with optional 2 MB. Daughter Board. Compatible with systems up to 8 MHz ..... **$ 650.00**
Compatible with systems up to 12 MHz ..... **$ 725.00**
2 MB. Daughter Board..... **$ 50.00**

4 MB. Memory Expansion Board: Fully populated. Requires one slot only!
Compatible with systems up to 8 MHz..... **$ 725.00**
Compatible with systems up to 12 MHz ..... **$ 799.00**

6 MB. Memory Expansion Board: Fully populated with 1 MB. DRAM's.
Compatible with systems up to 16 MHz **$1650.00**

PC COMPATIBLE UPGRADE ACCESSORIES

All AST Memory Expansion Cards come with Hardware, Installation Instructions, DesqView and SuperPak Software, and a **ONE YEAR WARRANTY.**

AST RAMPAGE: EMS/EEMS. Expandable up to 2 MB. 64K Memory Installed..... **$ 269.00**

AST ADVANTAGE: EMS/EEMS, expandable to 1.5 MB. One Serial and One Parallel Port. 128K Installed ..... **$ 399.00**

FBE RESEARCH's SMARTWATCH: No Slot Clock/Calendar, 10-year lithium battery. Easy plug-in installation. Includes Installation Instructions, software with source code.
120 DAY WARRANTY..... **$ 44.00**

MICROSPEED'S FAST 88: Speed upgrade for PC's and Compatibles. **RAM Technology** guarantees that it will work in your H/Z 150/160 or PC, PC/XT series computer or your money back! Increases CPU speed up to 8 MHz. Speed is switchable via software or external speed switch module.
Includes Installation Manual, software, and hardware; no CPU included. **ONE YEAR WARRANTY**..... **$ 120.00**
With Optional 8088-2 8 MHz. Processor..... **$ 124.00**
With Optional NEC V20 8 MHz. Processor..... **$ 135.00**

RAM Technology

All **RAM Technology** products include Hardware, Installation Instructions, and **ONE YEAR REPLACEMENT WARRANTY. No Soldering Required.**

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

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

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

Z-MEG 171: 1 MB. Memory Upgrade for the ZFL-171. Up to 640K of addressable memory and a 360K RAM DISK can be installed on your existing memory board. Software included. Developed jointly by **FBE Research** and **RAM Technology** ..... **$ 89.95**

Z-181 External 5¼" Disk Drive: 360 DS/DD 48 TP Disk Drive mounted in Half Height Cabinet/Power supply for the ZFL-181-92 ..... **$ 199.00**
For ZFL-181-93 or ZWL-183-92..... **$ 219.00**

INTEL PROCESSORS

Includes Installation Instructions. **FIVE YEAR WARRANTY.** Prices current 7/1/87 and subject to change; call toll free for current pricing.

QUANTITY PRICING AVAILABLE ON REQUEST.

8088-2 : 8 MHZ. Processor..... **$ 8.95**
8087-3 : 5 MHz. Math Co-Processor ..... **$ 119.00**
8087-2 : 8 MHz. Math Co-Processor ..... **$ 169.00**
8087-1 : 10 MHz. Math Co-Processor ..... **$ 225.00**
80287-6 : 6 MHz. Math Co-Processor ..... **$ 195.00**
80287-8 : 8 MHz. Math Co-Processor ..... **$ 269.00**
80287-10: 10 MHz. Math Co-Processor ..... **$ 325.00**
80387-16: 16 MHz. Math Co-Processor ..... **$ 595.00**

NEC V20: 8 MHz. 8088 replacement Microprocessor. Increases CPU efficiency 10-30%. Includes Installation Instructions, **ONE YEAR WARRANTY** ..... **$ 12.95**

CABLES

RAM Technology carries a large selection of quality molded cables in stock. The following is just an example of our product line. All cables come with a **ONE YEAR WARRANTY.** Special orders accepted; call us toll free for a quotation on your cable requirements.

QUANTITY PRICING AVAILABLE ON REQUEST.

Parallel Printer Cable, 6 ft..... **$ 18.50**
Serial Cable, DB25M-DB25F, 1-8 & 20, 9 ft..... **$ 15.50**
Serial Cable, DB25M-DB25F, 1-25, 9 ft..... **$ 18.50**
RS-232 Gender Changer, Specify DB25M-M or DB25F-F ..... **$ 15.50**
AT Serial Adaptor Cable (DB9-DB25), 2 ft ..... **$ 12.95**
Keyboard Extension Cable, 6 ft ..... **$ 9.50**
RGB/TTL Monitor Extension Cable, 6 ft ..... **$ 14.50**

Need a cable for your APPLE or MACINTOSH? CALL US!

DISKETTES

RAM Technology carries a full line of quality diskettes by NASHUA, SONY, MAXELL, 3M and other major manufacturers. Call us toll free for pricing on your diskette needs. Bulk diskettes available; call toll free for price and availability.

QUANTITY PRICING AVAILABLE ON REQUEST

NASHUA DISKETTES

Packaged in Boxes of 10. **LIFETIME REPLACEMENT WARRANTY.**

MD2D: DS/DD 48 TPI Soft Sector, 5¼"..... **$ 8.95**
MD210D: DS/DD 48 TPI Hard Sector, 5¼"..... **$ 9.95**
MD2F: DS/DD 96 TP Soft Sector, 5¼"..... **$ 14.95**
MD210F: DS/DD 96 TPI Hard Sector, 5¼"..... **$ 15.95**
MD2HD: DS/High Density Soft Sector (AT), 5¼"..... **$ 21.95**
MF2: DS/DD 135 TPI, 3½"..... **$ 20.95**
FD2D: DS/DD Soft Sector, 8"..... **$ 20.95**

HUGCON SPECIAL

Seagate ST-251 Hard Disk Drive
40 Ms. Average Access Time
With AT Rails and Cable
$499.00
Offer Expires August 31, 1987

U.S. ROBOTICS MODEMS

Sportster, 1200 Baud External ..... **$ 129.00**
Password, 1200 Baud External ..... **$ 219.00**
Courier 1200, External ..... **$ 235.00**
Microlink 1200, Internal, with Telpac Software..... **$ 219.00**
Varmodem 1200, Internal ..... **$ 119.00**
Courier 2400E, External..... **$ 475.00**
Microlink 2400, Internal, with Telpac Software..... **$ 435.00**
Varmodem 2400, Internal ..... **$ 205.00**
Courier HST 9600, External ..... **$ 785.00**

Z100 UPGRADE ACCESSORIES

C.D.R. ZS-100, Speed Upgrade for 4.7 MHz. Z-100's.
Easily installed, no soldering required..... **$ 44.95**

FBE RESEARCH ZMF-100a: Install up to 768K on old Motherboard Z-100's, #181-4917 or lower; no soldering required ..... **$ 59.95**

Z-MAX 100 768K Memory upgrade for new Motherboard Z-100's, #181-4918 or greater; simple plug-in installation, no soldering required ..... **$ 59.95**

UCI EASYRAM plug-in Memory board for Z-100 w/192K-768K. Expandable to 2 MB. 2/256K DRAM. With 0K ..... **$ 299.00**
RAMDRIVE Software ..... **$ 45.00**
With RAMDRIVE Software and 1 MB. RAM ..... **$ 459.00**
With RAMDRIVE Software and 2 MB. RAM ..... **$ 575.00**

Z-100Winchester Disk Drive Controllers: Includes Installation Instructions, Cables, Hardware, Software and **ONE YEAR REPLACEMENT WARRANTY**

C.D.R. 317 Controller (Z-217 Compatible) ..... **$ 550.00**
U.C.I. EASYWIN S-100 Interface ..... **$ 245.00**
With Winchester Controller ..... **$ 355.00**
External Hard Disk Sub-system: Cabinet and Cables, Add ..... **$ 139.00**

For a complete selection of Hard Disk Drives, please see HARD DISK section. Or Call Toll Free for assistance!

Z-100 1.2 MB External Floppy System: Utilize your Z-100's built-in 8" controller for high capacity storage and back-up. **ONE YEAR REPLACEMENT WARRANTY**
DRSZ101-12: One 5¼" 1.2 MB High Density Floppy Disk Drive, cables, external power supply and cabinet, and instructions ..... **$ 249.00**
DRSZ102-12: Two 5¼" 1.2 MB High Density Floppy Disk Drive system ..... **$ 439.00**

HARD DISK DRIVES & CONTROLLERS

All Hard Disk Drives and Controllers come with installation Instructions and a **ONE YEAR REPLACEMENT WARRANTY.** System Orders (hard disk & controller) come with Cables and Hardware. All controllers capable of handling two drives unless otherwise specified.

OMTI CONTROLLERS

5520: PC Half Card ..... **$ 105.00**
5527: PC RLL Half Card ..... **$ 145.00**

WESTERN DIGITAL CONTROLLERS

WD-1003-WA2: AT Hard Drive and Floppy ..... **$ 210.00**
WD-1003-RA2: AT RLL Hard Drive and Floppy ..... **$ 249.00**
WD-1003-RAH: AT RLL Hard Drive Only ..... **$ 231.00**

MINISCRIBE HARD DISK DRIVES

MS-3438: 30 MB. RLL, 65 ms., Half Height ..... **$ 325.00**
MS-3650: 30 MB., 28 ms., Half Height ..... **$ 480.00**
MS-6085: 71.4 MB. (Formatted), 28 ms., Full Height ..... **$ 990.00**

SEAGATE HARD DISK DRIVES

ST-225: 20MB., 65 ms., Half Height ..... **$ 299.00**
ST-238R: 30MB. RLL, 65 ms., Half Height ..... **$ 360.00**
ST-251R: 44 MB. RLL, 40 ms., Half Height ..... **$ 545.00**
ST-277R: 65 MB. RLL, 40 ms., Half Height ..... **$ 610.00**
ST-4038: 30 MB., 40 ms., Full Height ..... **$ 605.00**
ST-4053: 45 MB., 28 ms., Full Height ..... **$ 715.00**
ST-4096: 80 MB., 28 ms., Full Height ..... **$ 995.00**

HUGCON SPECIAL

Seagate 20 MB. Hard Disk Drive Kit With OMTI Controller and Cables

$329.00

Offer Expires August 31, 1987

RAM Technology Hard Cards

OLHC-20: 20 MB., Hard Card featuring AUTO-START and AUTO-PARK heads, 65 ms ..... **$ 425.00**
OLHC-30: 30MB Hard Card featuring AUTO-START and AUTO-PARK heads, 65 ms ..... **$ 475.00**

HUGCON SPECIAL

Free Shipping Within the Continental United States on All Purchases During the Month of August

FLOPPY DISK DRIVES

All Disk Drives come with Installation Instructions, hardware and **ONE YEAR REPLACEMENT WARRANTY.**

MITSUBISHI FLOPPY DISK DRIVES

MF501A: 360K, 5¼" Half Height, 48 TPI ..... **$ 115.00**
MF504A: 1.2MB., 5¼" Half Height, 96 TPI ..... **$ 159.00**

PANASONIC FLOPPY DISK DRIVES

PAN-360: 360K, 5¼" Half Height, 48 TPI ..... **$ 99.00**
PAN-120: 1.2 MB., 5¼" Half Height, 96 TPI ..... **$ 135.00**
PAN-720: 720K, 3½" ..... **$ 110.00**

TOSHIBA

3½" Disk Drive Upgrade: For PC's and Compatibles including all Z-100 PC and Z-200 PC series computers. Maintain compatibility with the new PS/2 and Portable software formats and increase storage capacity to 720K per diskette utilizing your existing floppy disk controller. Includes mounting hardware for 5¼" enclosures. Requires MS-DOS 3.2 or graters. ..... **$ 169.00**

MULTI-PORT SERIAL BOARDS

For applications requiring multiple serial ports, ie. Xenix, Unix, BBS etc. All boards include necessary hardware. Order Drivers Separately. **ONE YEAR WARRANTY.**

DBCOM4S: Four Serial Ports. Compatible with systems up to 16 MHz ..... **$ 409.00**
DBCOM8S: Eight Serial Ports. Compatible with systems up to 16 MHz ..... **$ 640.00**
DBCOM4I: Four Serial Port Intelligent Controller, 80186 Processor, On Board Buffering. Compatible with systems up to 16 MHz ..... **$ 825.00**
DBCOM8I: Eight Serial Port Intelligent Controller, 80186 Processor, On Board Buffering. Compatible with systems up to 16 MHz ..... **$ 995.00**
DBMSDOS: MS-DOS Device Driver ..... **$ 25.00**
DBMDV22: Microport V/286 2.2 Device Driver ..... **$ 35.00**
DBSDV22: SCO Xenix V/286 2.2 Device Driver ..... **$ 35.00**

SOFTWARE

Microport Unix V/286 Version 2.2
Run-Time System (2 User License) ..... **$ 180.00**
Software Development System ..... **$ 225.00**
Text Preparation System ..... **$ 180.00**
Complete System ..... **$ 495.00**
DosMerge 286 ..... **$ 135.00**
Unlimited User License ..... **$ 225.00**

SCO Xenix V/286 Version 2.2
Operating System (14 User License) ..... **$ 465.00**
Software Development System ..... **$ 445.00**
Text Preparation System ..... **$ 155.00**
Complete Xenix System ..... **$ 970.00**

RAM Technology carries a complete line of Software for PC and AT Compatible Systems. We also carry a complete selection of Unix and Xenix Application Software and Languages. Call us toll free for your software requirements or write for our free catalogue.

TERMS

Prices and specifications subject to change. Personal checks held 10 working days; money orders accepted as cash. Please add 2% (minimum $2.50) for shipping. COD orders accepted; cash or cashier's check only. VISA/MC accepted; NO surcharge. Purchase orders accepted from qualified businesses, government agencies, and educational instutitions.

HOURS OF OPERATION: Monday-Friday, 9:00 AM-5:00 PM EST

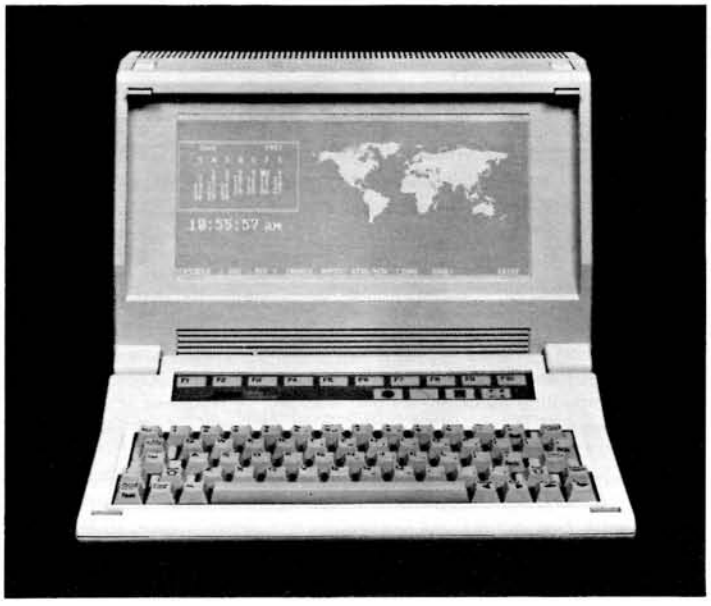
RAM Technology

427-3 AMHERST STREET, SUITE 265
NASHUA, NEW HAMPSHIRE 03063
ORDER LINE: 1-800-662-0070
INFORMATION: (603) 889-0633

MasterCard

VISA

Give Your Z-171 A New Twist



Pat Swayne
HUG Software Engineer

Caution:

The modification presented in this article requires nearly a complete disassembly of your computer and a small amount of desoldering and soldering. Do not attempt this modification unless you feel comfortable doing such work. If you perform this modification, you do so at your own risk.

The display originally supplied on the Z-171 portable computer is pretty good compared to other "lap top" computers that were being made at the time it was introduced. But when you compare it to the super twisted crystal display on the newer Z-181, it doesn't look so good any more. You may have heard that Zenith Data Systems is now manufacturing the Z-171 with a super twist display, which looks almost as good as the Z-181's display. In this article, I will describe the procedure for upgrading an older Z-171 with the new display. The super twist display is quite expensive ($341 if purchased from the Heath Company Parts Dept.), but you may consider the improvement in display quality worth it, especially if you bought your Z-171 at a discount.

The new type of display is available from the Heath Company parts department as part no. 411-888. You may also need to purchase three mounting brackets, which are part no. 204-2970. The mounting brackets will be described later in this article. To perform the procedure described here, you will need a medium philips screwdriver, a smaller philips screwdriver, a light wattage soldering iron and solder, and possibly a flat bladed screwdriver and/or 1/4" nut driver and a 3/16" nut driver. You may also need a reamer or small pointed knife, to enlarge the mounting holes in a printed circuit board.

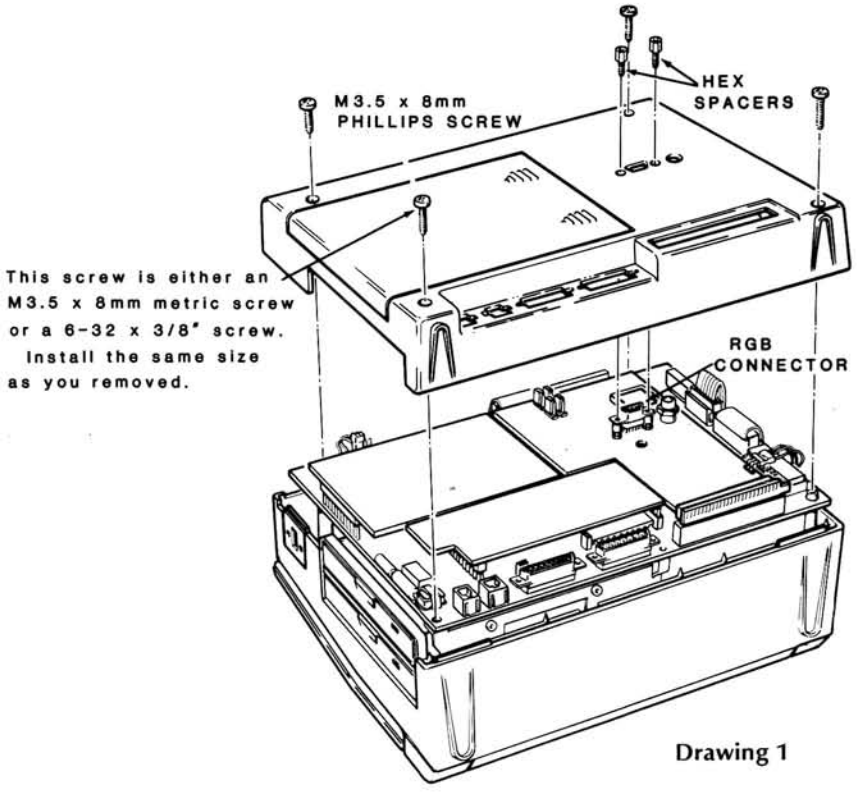
The modification procedure will be described in sections, with a drawing accompanying each disassembly section. The sections will be further divided into individual steps.

Note: Remove the power connector and battery pack from your computer before you begin disassembly.

Back Cover Removal

Refer to Drawing 1 as you perform the following steps.

( ) Close the keyboard and lay the computer face down on a clean, padded surface.



Drawing 1

( ) Completely loosen the four philips head screws at each corner of the back cover. Do not remove the screws from their holes. As indicated in the drawing, one of the screws may be different from the others, and the best way to keep from getting them mixed up is to leave them in the holes.

( ) If you have a color video board installed in your computer, remove the two hex spacers connected to the RGB connector.

( ) Lift the back cover straight up and set it aside without turning it over, so that the screws do not fall out.

( ) If you have a color video board installed, replace one of the hex spacers you removed to temporarily hold the RGB connector in place.

Main Board Removal

Refer to Drawing 2 for these steps.

( ) Unplug the connectors at J12, J3, and J4 at the top of the main board.

( ) Remove the connectors at J5, J6, and J7 at the right side of the main board.

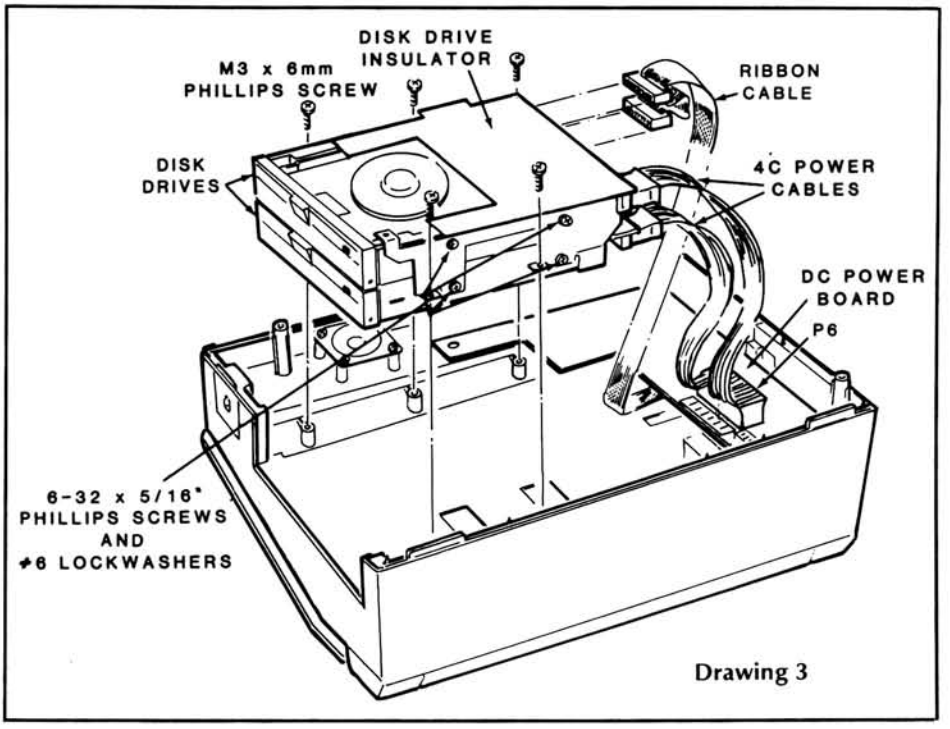
( ) Remove the screw at location A as shown in the drawing.

( ) Lift the main board from the computer and set it aside.

Disk Drive Assembly Removal

Refer to Drawing 3 for these steps.

( ) Unplug the drive power cable at P6 on the DC power board.



Drawing 3

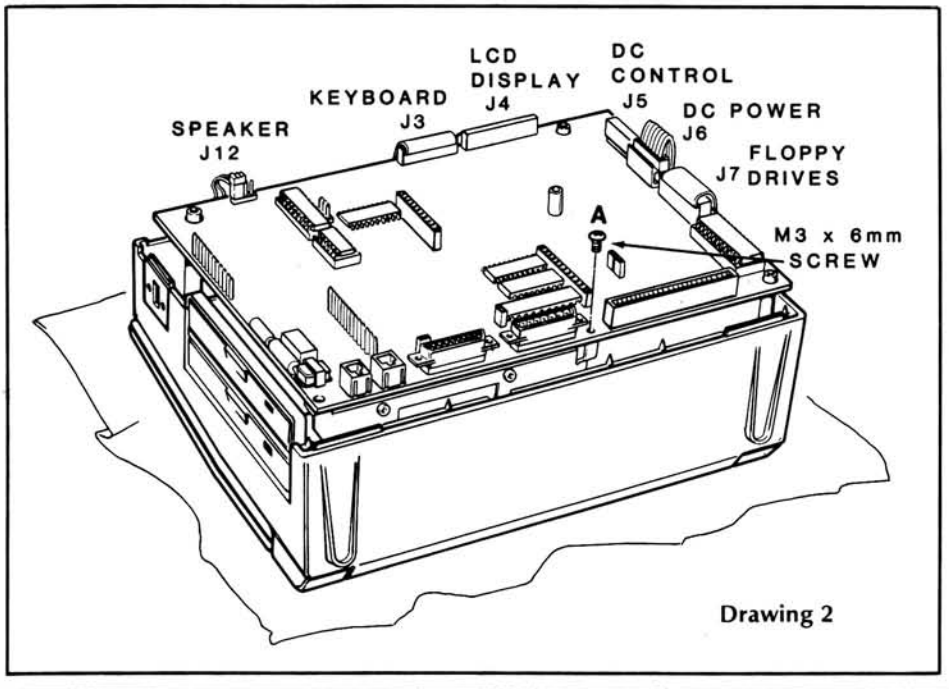
( ) If the drive ribbon cable is routed under the DC power board, unplug it from both drives. If it was routed above the DC power board, you can leave it connected.

( ) Remove the five screws that secure the disk drive assembly to the computer.

( ) Lift the drive assembly out of the computer and set it aside.

DC Power Board Removal

Refer to Drawing 4 for these steps.



Drawing 2

( ) Unplug the cables at P3, P5 and P6 on the DC power board.

( ) Remove the four screws that hold the power board in place. If one of the screws is different, be sure to note its position.

( ) Remove the board by sliding it toward the disk drive opening until the contrast knob and power switch are clear, and then lift the board straight up and set it aside.

Liquid Crystal Display Assembly Removal

Refer to Drawing 5 for these steps.

( ) Remove the six screws that hold the chassis assembly in place.

( ) Lift up the chassis assembly while holding the cables along the top out of the way.

( ) On your computer, the keyboard connector may be routed through the hole that the battery connector wires pass through. If this is the case, pass the end of the cable down through the hole.

( ) Set the chassis assembly aside temporarily.

( ) Remove the three screws and flat washers from the top edge of the LCD assembly.

Note: On some Z-171 models, the bottom edge of the LCD assembly is held in place by screws passing

through mounting holes in the edge of the assembly. On other models, the bottom edge is held in place by small right angle metal brackets through which the screws pass. If your computer does not use the brackets, you may have to purchase three of them, because the new LCD assembly may not align with the mounting holes. After you remove the old assembly, place the new one in position with the top mounting holes aligned and note whether the bottom edge comes short of the mounting holes. If it does, you will need the brackets.

( ) Remove the screws and brackets (if any) that hold the lower edge of the LCD in place.

( ) Lift out the LCD assembly and place it face down on a smooth padded surface.

LCD Assembly Preparation

( ) Unsolder the black and yellow twisted cable that is connected to the right edge of your old LCD assembly and solder it to the same location on the new one, making sure that the black wire goes to the top hole. The middle of the three holes is not used. If there is a small black wire on the old connector leading from the top hole to one of the bezel mounting tabs, do not transfer it to the new assembly.

( ) If there is a capacitor soldered to the top edge of your old assembly by pins 16 and 17 of the ribbon cable connector, unsolder it and solder it to the new LCD assembly at the same location.

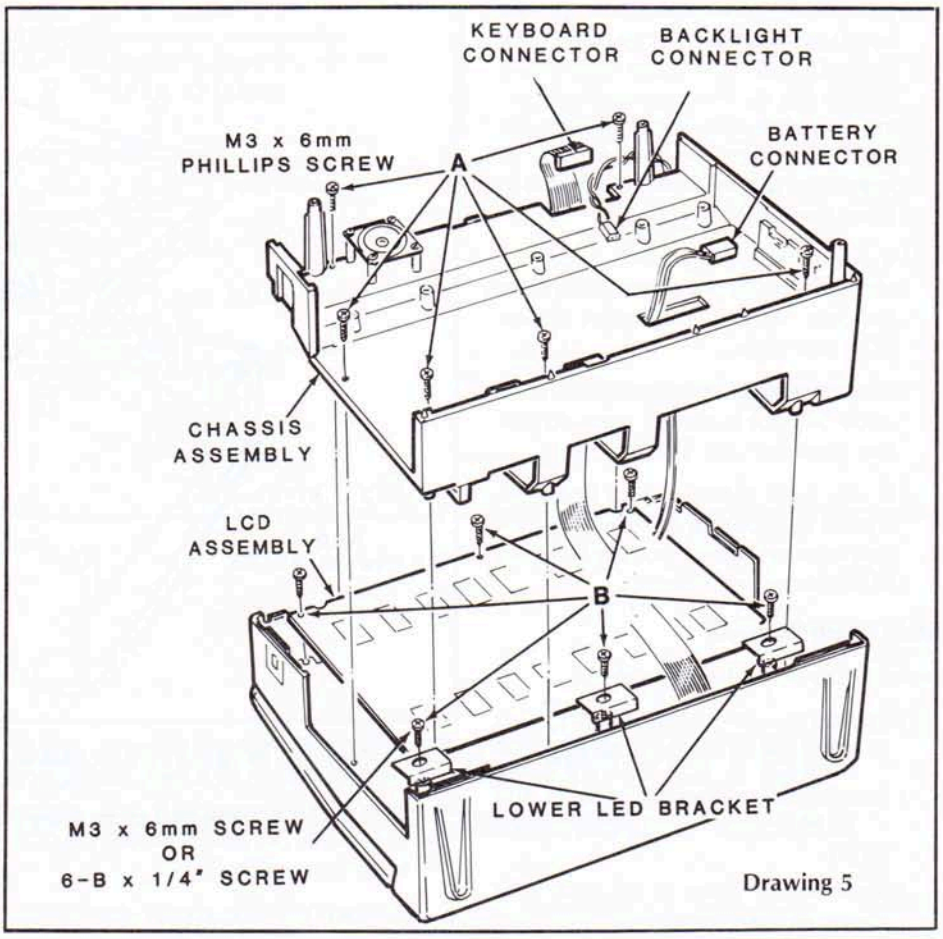
( ) Remove the ribbon cable from the old LCD assembly and plug it onto the new one.

( ) Place the new LCD assembly in position in the computer and check to see if the upper mounting holes align with the holes in the mounting bracket in the computer. You will probably have to enlarge the holes in the top edge of the LCD assembly. They should be enlarged to about 3/16" (4 mm) if it is needed.

LCD Assembly Installation

( ) Place the new LCD assembly in the computer and replace the top three screws and flat washers. Do not tighten the screws yet.

( ) Install the lower mounting brackets (if required) and the lower screws. Do not tighten them.



Drawing 5

( ) Tighten the upper screws just enough to keep the washers from moving freely. Do not tighten them enough to bend the LCD assembly printed circuit board at all.

( ) Tighten the lower screws just enough to hold the brackets in place. Do not bend the circuit board.

Computer Reassembly

Reassembly of the computer can be done by reversing the disassembly steps with a few exceptions, which will be noted here.

( ) Ensure that all cables are routed properly as you replace each assembly.

( ) Replace the chassis assembly.

( ) Replace the DC power board, but do not replace the mounting screws yet.

( ) Replace the disk drive assembly. You may have to use a magnetic screwdriver to replace the lower screws. If you do not have one, try placing the screws in the mounting bracket holes before you insert the assembly, then lower it carefully and start the screws while holding it up about 1/8" (3 mm) from its final position.

( ) Replace the disk drive power connector at P6 on the DC power board. The connector is hard to push down, so lift up the DC board slightly and hold two fingers under it while you push the connector down.

( ) Replace the DC power board screws and the other connectors removed from it, and the disk drive ribbon connectors.

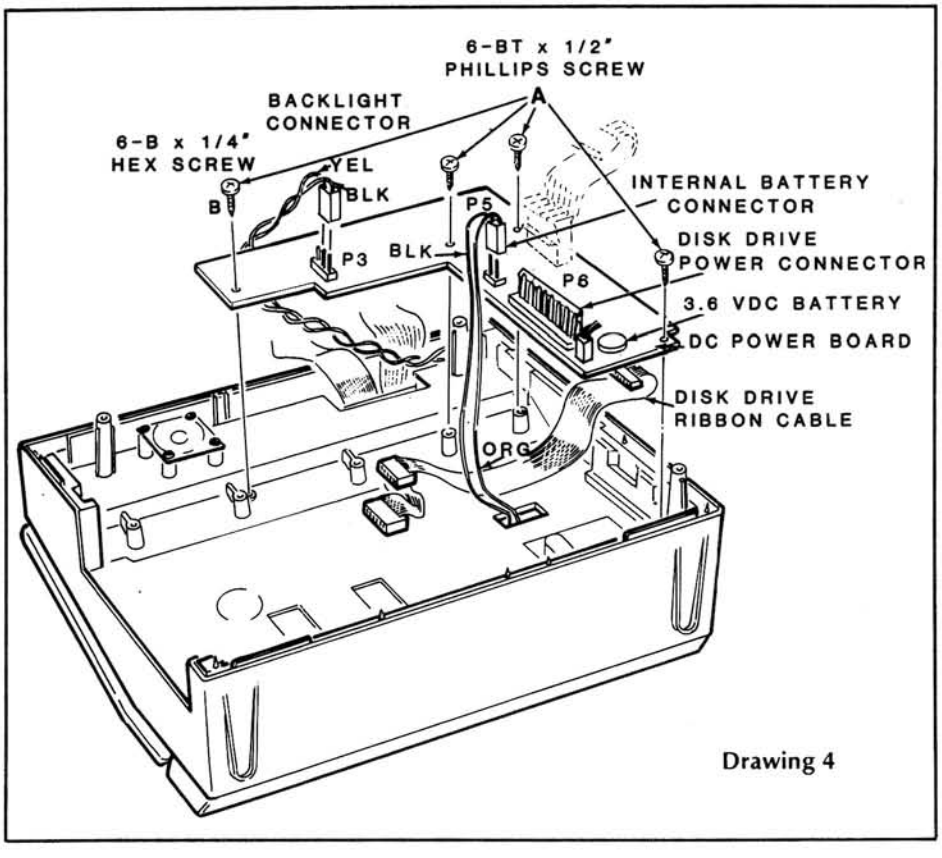
( ) Replace the main board and ensure that all connectors are re-installed properly. Also be sure to remove the hex spacer at the RGB connector (if you have one) before you replace the back cover.

( ) Replace the back cover.

Final Testing

Replace the power connector and/or battery pack in your computer, turn it upright, and turn the power on. If you do not see the "Testing Memory" message right away, try adjusting the contrast knob. If you still do not see it, something is wrong. Turn off your computer immediately, and re-check your work.

After you have ensured that your computer is operating correctly, you will have to run Setup and reset the date and time and the modem configuration, etc. *



Drawing 4

Introducing Enable 2.0 For The Z-100

Earl R. Zimmerman, Jr.
169 Spinning Road
Dayton, OH 45431

If you're a Z-100 Series computer owner and are in the market for a good integrated package, you're probably disappointed in the limited selection of software available for it. It was times like these when I wanted to trade my 100 in for an IBM or IBM compatible computer. Luckily, through the Dayton Heath Users Group (DAYHUG), I discovered Enable, a product of The Software Group (TSG). Before I discuss some of the features of this super application package let me give you some background on how I began using it.

DAYHUG made special arrangements with TSG to purchase over 100 copies of the beta version of Enable 1.1Z in October 1986. TSG agreed to sell us the beta version at a substantially reduced price if the group tested the program and provided them with feedback on bugs, annoyances, and suggested improvements. This we gladly did, as they also agreed to provide at no additional cost, the final 2.0 version. The only drawback to this arrangement has been that as this article is being written TSG is going to charge us $55.00 for the updated documentation. Purchasing the updated documentation appears to be a necessity as the 2.0 version has substantial improvements and many additional features. Hopefully, they will consider providing the documentation at no added cost.

Enable 2.0 has numerous applications all rolled into one powerful and relatively user-friendly package. Applications include word processing (with spelling checker and mail merge capability), a Lotus 1-2-3-like spreadsheet, database management system, graphics, and telecommunications. All these applications are tied together by a Master Control Module (MCM). What else could you possibly ask for in one package? Some of the trade publications, when talking about the IBM version, say it is the only package you will ever need, and after using it you will give up WordStar, Lotus 1-2-3, and dBase. You can judge for yourself after reading this article and purchasing the program. I stated the package was relatively user-friendly because it's not a program you can learn in a day or two or begin using right out of the box. It does require that you read the documentation, especially if you wish to take advantage of the advanced program features. But in all fairness, how many people could use Lotus 1-2-3 right out of the box. Because Enable is an integrated package, I cannot cover all of the packages with any detail in this article. I will cover installation and documentation, establishing profiles, word processing features, and the spreadsheet package.

Installation And Documentation

Installation. Enable 2.0 is a large program — six 5-1/4" disks to be exact. So while I would recommend the program for everyone's use, I would also recommend the person have a hard drive, or more than two disk drives, or an installed UCI (or equivalent) ram board. I say this because I don't like to swap disks to take advantage of all the features. Regardless of what type of system you have, the installation procedure is very easy. Just place the Install Disk in Drive A and have sufficient formatted disks ready. All that's necessary is to type INSTALL and follow the prompts for your system (hard disk or dual floppy).

Documentation. Documentation for Enable is rather extensive, but could be improved. Documentation consists of five spiral bound books. There is a system overview book which provides a general description of Enable and its application packages along with information on the MCM which controls how the package is integrated. It also explains how to use the windows feature. Understanding the windows feature is extremely important when copying information and/or graphs from one package to another. In addition, it also explains the Tool Menu Option and the Menu Generator features. There are also separate books for word processing, spreadsheet/graphics, database management, and telecommunications. In addition to the written documentation, TSG also provides a large plastic template that summarizes the commands for all packages, and the program itself comes with a tutorial and an on-line help function which

is activated by depressing the F1 or HELP key.

While I can't tell you about the 2.0 documentation as I haven't received it yet, I can say that the 1.1Z documentation could have been improved. Information on the windows feature would have been more meaningful if placed in the application books. Information on using the advanced features such as macros, tools, and menus should be more detailed.

Profiles And Main Menu

Profiles. One essential element of the MCM feature is the ability to establish profile definitions. These definitions control numerous system features. (See Figure 1) These features include:

Hardware — Printer and plotter options.

Page Form — Papersize, font, margins, line spacing, header, footer location, and other layout options.

Special Text — Screen color options and the way text attributes (italics, underline, boldface, compressed print, etc.) are displayed.

Word Processing Options — How you want various format options displayed, document printing and backup information.

Spreadsheet Options — Whether or not you want all cells protected, or backup spreadsheets created, and conversion of Lotus 1-2-3 macros to Enable command sequences.

Database Options — How you want records displayed (full screen or individually) and other options.

Telecommunications Options — Quick connect and terminal options.

System Options — Key operation, such as what character the BACKSPACE key deletes.

After you create these profiles, they are stored in file PROFILE.$PR and can be changed as desired. Whenever you enter Enable, the Sign-On screen appears (See Figure 2) and you enter the profile you wish to use (if any). Two profiles have already been created — Default and Color (used with color monitor) — so you can modify these or create your own. In addition, you can rename or delete definitions, if you wish.

Main Menu. After the profile is selected, the Main Menu will appear (See Figure 3). From this, you select which system you want to use and whether you want to create, revise, or print. You can also update your user created dictionary or enter the Mail Merge or MCM features. Help is also available by selecting the tutorial or calling up a help screen that lists a toll free customer service number.

Word Processing Features

I chose the word processing package first because it's the one I use most often and my favorite package. At work, I've used WordStar Professional and Peachtext, and in my opinion they are greatly inferior to Enable. After selecting word processing and telling Enable to create a new document (See Figure 3), you can title your document by typing in what you want it called. In Figure 4, I decided to call mine Demonstration Document. This title will be printed on a separate page. If you don't find this feature necessary, simply tell Enable not to prompt for it when you set up your profile.

Top Line and Status Line. You can now begin typing your document or call up the top line menu by depressing F10. Figure 4 indicates I wish to use the edit option category. The top line menu organizes the functions available into the different general categories shown in Figure 4. After one of these categories is selected, another sidebar menu will be displayed in reverse video so it can be distinguished from other material on your screen. You then select the specific feature from the sidebar menu. When you are an experienced Enable user, you can skip over some menus by depressing the F9 key and one or more other keys. Using the F9 key and other keys is referred to as using an expert command.

In addition to the top line menu, there is a reverse video line at the bottom of the screen. This is the status line and it contains information about the file your using. It displays the window number, disk drive, file name, display mode (draft or final), text attributes in use (i.e. "u" for underline), and cursor location by line and column.

Edit Options Menu

The first function in the word processing package is the Edit Options Menu. This function basically serves as a toggle switch to turn sub-features on and off; switch from draft to final mode; insert comments or paper clips and to move previously marked blocks of text.

Features. Some of the specific features include the ability to edit documents in both the draft and final modes. Editing is faster in the draft mode because Enable is not continually adjusting page breaks as text is added or deleted. You can also add comments or paper clips. Comments are just notes to yourself, while paper clips mark a spot in your file. Even though they are displayed they are not printed.

Text Enhancements. Using the edit options, you can activate the help screen for text attributes (enhancements). These enhancements allow the user to create professional looking documents and clarify important points with special type. You can also change the size and style of characters in your document. Boldface or italics can

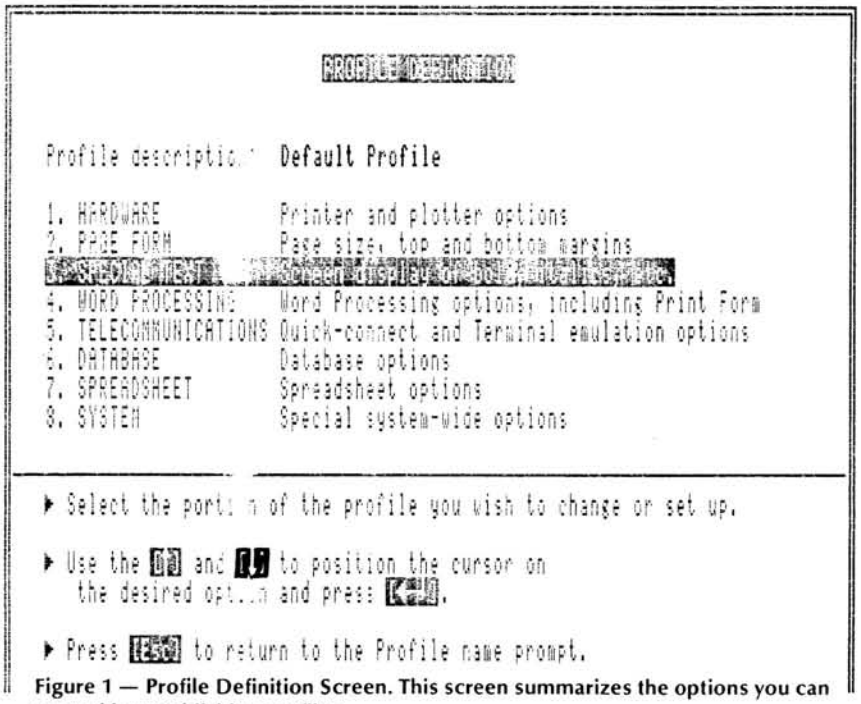


Figure 1 — Profile Definition Screen. This screen summarizes the options you can control by establishing profiles.

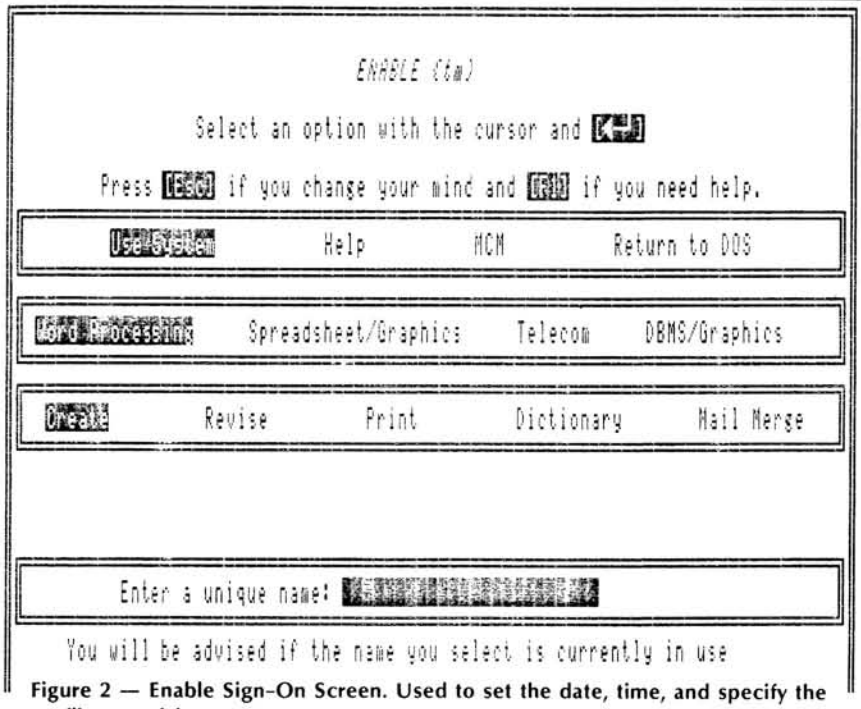


Figure 2 — Enable Sign-On Screen. Used to set the date, time, and specify the profile you wish to use.

be applied; phrases can be underlined; superscript and subscripts can be added; and compressed, wide, elite, or quality text can be used for emphasis.

Layout Options Menu

The Layout Option Menu is one of the most important menus in the word processing system. It's used to insert rulers; set line spacing; insert a page break, header, or footer; or create a shorthand entry or footnote. In addition, special character sets can also be used. The two most unique features under this menu are the shorthand feature and the special character sets.

Shorthand Feature. The shorthand feature is designed to save you time when you use the same word or phrase often within the same article. Instead of typing the word or phrase (string), you can use a code to identify that string. For instance, in this article, I use the code ?E to identify the word Enable and $M to identify menu. See Figure 5 to see how this feature is displayed. (You can also see what a ruler looks like.) After "compiling" these entries, the code is converted to text.

Special Character Sets. Enable contains several character sets that allow you to do some graphics, i.e., create sharp looking tables, or use some miscellaneous characters that are not on the keyboard, such as the ^X ^Y [\ ] characters. In addition to the character sets that can be used frequently, Enable also has Greek characters and two foreign language sets. These special characters are tied to the first row of grey keys on a Z-100, and include their shift character. As the Z-100 has no NUM LOCK key, you can't type in numbers when the special characters are in use like you can in the IBM version of Enable. You should also be aware that certain printers will not print certain special characters. See Figure 4 for an example of a box drawn by Enable.

Copying/Deleting/Finding Text

Enable also has menus to copy, delete, or find text. These menus are briefly explained in the following paragraphs.

Copying Text. Enable can duplicate any portion of your file or move a section from one location to another within your file. The COPY command duplicates text; the MOVE command relocates text. Text can't be moved between two windows — only copied. You can copy or move words, sentences, paragraphs, graphs, and columns.

Deleting Text. Using the Delete Options Menu, you can delete a marked block of text from the cursor position to the end of a sentence, paragraph, page, beginning or end of a file, white space between words, and delete a paragraph marker. Deleting a paragraph marker allows you to combine two or more paragraphs into one without retyping.

Finding And Replacing Text. Enable's FIND option will search an entire file to locate characters, words, phrases, numbers, or symbols. Three options are available — find only, find and replace, and find and mark. Enable can be instructed to ignore the letter case while searching; where to begin the search from; and what columns to search. After the first occurrence, you can depress F5 and Enable will search for the next occurrence. The find and replace feature has an additional option called occurrences, that instructs Enable to search the number of occurrences you specify or you can specify all. If you aren't sure of the correct spelling of a word (or

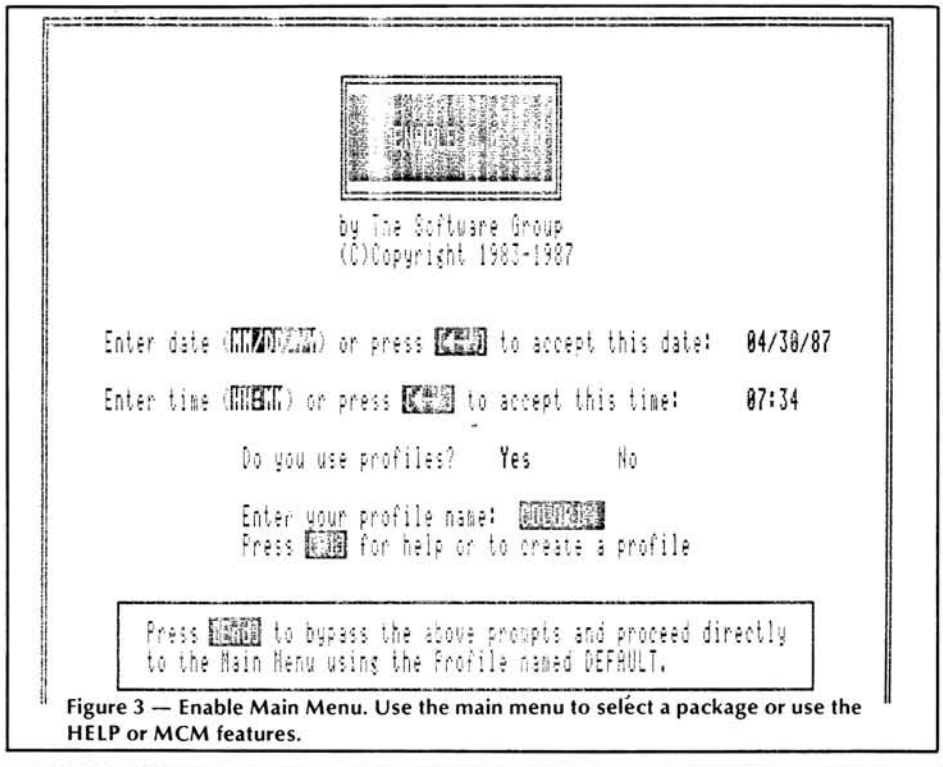


Figure 3 — Enable Main Menu. Use the main menu to select a package or use the HELP or MCM features.

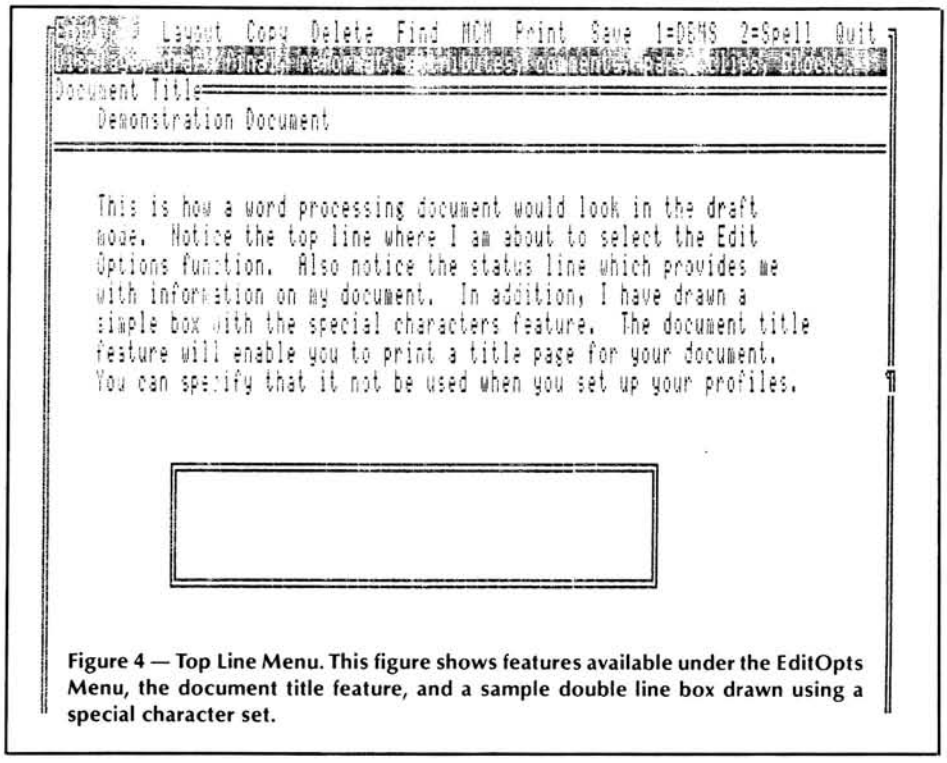


Figure 4 — Top Line Menu. This figure shows features available under the EditOpts Menu, the document title feature, and a sample double line box drawn using a special character set.

name), you can use wildcards (? or $) in the string.

Saving/Revising Files

One of the nice features of Enable is the ability to save and revise documents to different formats. Files can be saved and revised in Enable, ASCII, Volkswriter, Easywriter, Multimate, WordStar, Peachtext, DCA, and DIF format. The entire document, marked text, or marked blocks can be saved and a different name assigned.

Spelling Checker

Enable also contains an 80,000 word Proximity/Merriam-Webster Linguibase Spelling Checker which is accessed by selecting 2 from the Top Menu Line. You can instruct the spelling checker where to begin checking (top of file or from cursor position) and whether or not you want it to check for duplicate words that are next to each other. The spelling checker will stop at a word it doesn't recognize and a sidebar menu will appear as in Figure 6. Select one of the six actions. When it is finished spell checking the document, a summary screen will appear (See Figure 7) and provide you with information on what changes were made to your file.

The spelling checker is very good, except it doesn't handle contractions or hyphenated words well. It will always stop on a hyphenated word instead of checking each word individually.

Spreadsheet Feature

Due to space limitations of this article I can't cover every feature of the Enable spreadsheet package, so I'll compare it with Lotus 1-2-3 (Version 1A) and just highlight some of the best features.

Enable/Lotus 1-2-3 Differences. The spreadsheet is almost an exact duplicate of Lotus 1-2-3 right down to the macro feature. If you ever used Lotus 1-2-3, learning the Enable spreadsheet should be a snap. Like the word processing package, the spreadsheet top line menu can be called up by depressing the F10 or SLASH key (/). See Figure 8. Lotus users will immediately notice the similarity between the two spreadsheets. Lotus and Enable both have the Worksheet, Print, Graph, DBMS (Data in 1-2-3), and Quit options. The Save option in Enable is similar to the File/Save option in Lotus. In Enable, the Move feature is executed through the Worksheet option, but in Lotus the Move feature is a separate option. Conversely in Enable, Combine is a feature, while in Lotus this feature is executed through the File option.

Best Features. Some of the best features in my opinion are the transpose, spreadsheet dimension, highlight, and save features.

Transpose. This feature will allow you to take a row of figures and formulas and make them into a column or vice versa, and at the same time adjust the formula to the new cell locations. This feature is very helpful if you want to adjust the layout of your spreadsheet.

Spreadsheet Dimension. The size of your spreadsheet can be adjusted to suit your needs through the use of the WORKSHEET/GLOBAL/DIMENSION command. There are six different sizes to choose from; one is the default 255 rows and columns; others are 511 × 127, 1023 × 63, 2047 × 31, 4095 × 15, and 127 × 511.

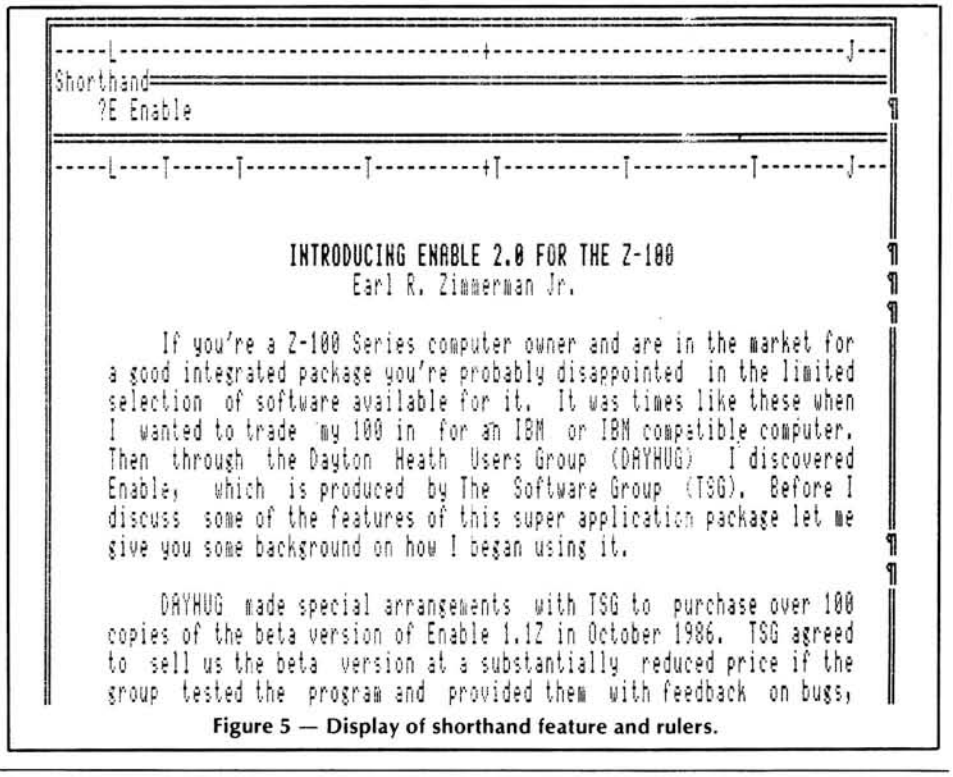


Figure 5 — Display of shorthand feature and rulers.

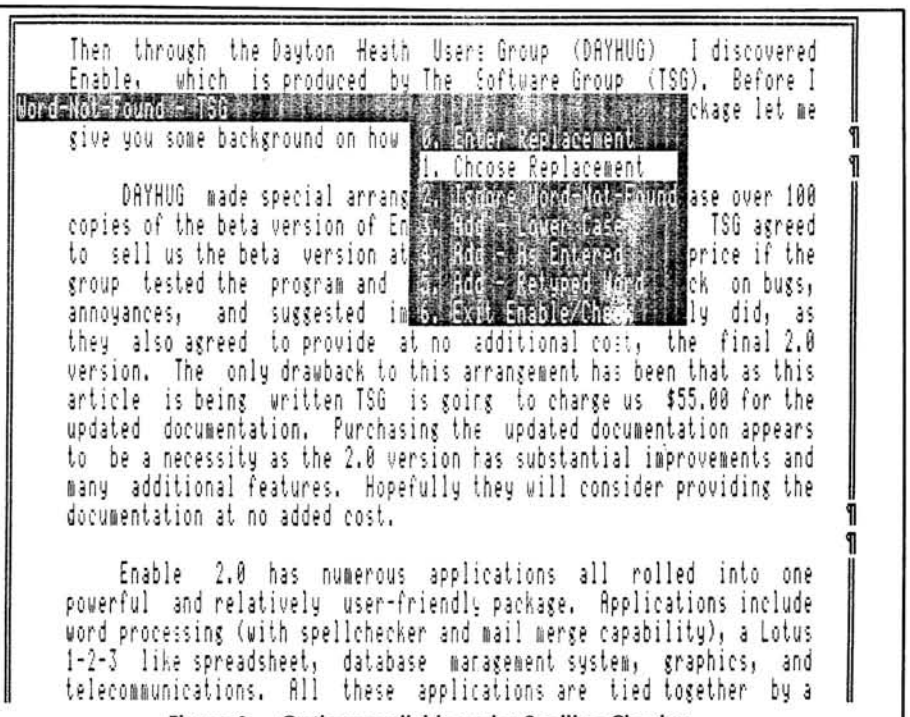


Figure 6 — Options available under Spelling Checker.

Highlighting. The HIGHLIGHT command allows you to highlight all the cells that refer to a specified cell. This command is especially useful when you consider making a change to the spreadsheet and you want to know what cells will be affected by the change. All that you have to do is tell Enable to highlight all cells with formulas that refer to that cell. At the prompt, all that's necessary is to enter the cell number and Enable does the rest.

Saving. Enable will save spreadsheets in Lotus 1-2-3 version 1A (that's not surprising is it?), Lotus 1-2-3 version 2, DIF, ASCII, and Supercalc 3 formats, also.

Conclusion

Enable 2.0 is a powerful program and it has more features than an average home user will ever need and most of, if not all, the features a business needs. It is a long overdue addition to the software available for the Z-100 computer. My next article on Enable will cover the graphics and DBMS features of Enable. As a little preview, let me tell you that Enable has the best graphics package I've ever seen in an integrated package and all others run a poor second. How many graphics packages allow you to do graphs in 3-D?

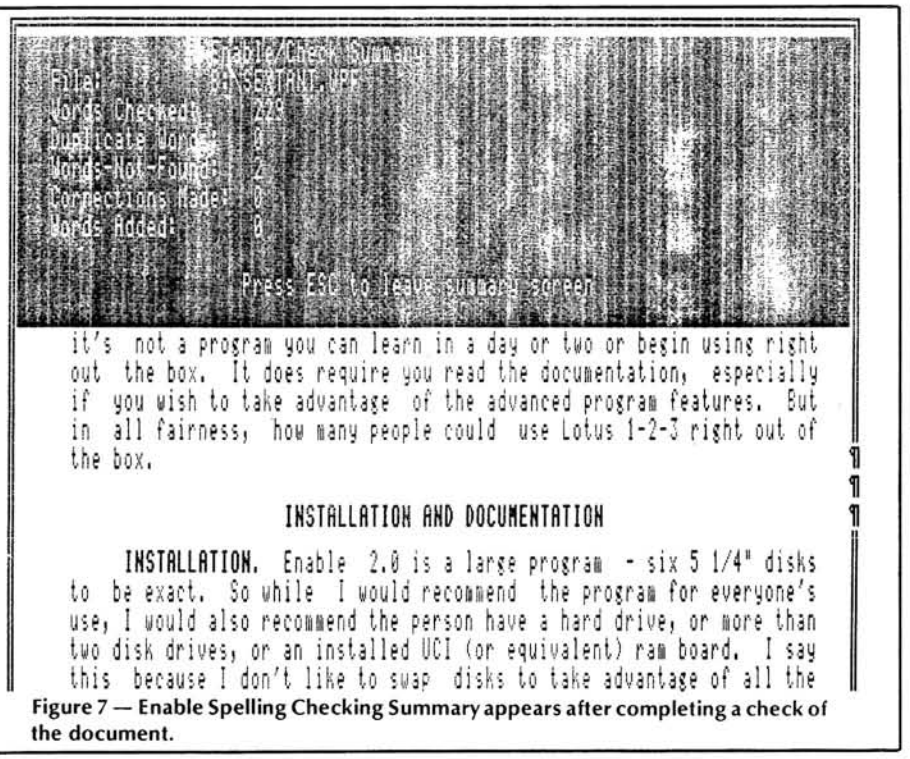


Figure 7 — Enable Spelling Checking Summary appears after completing a check of the document.

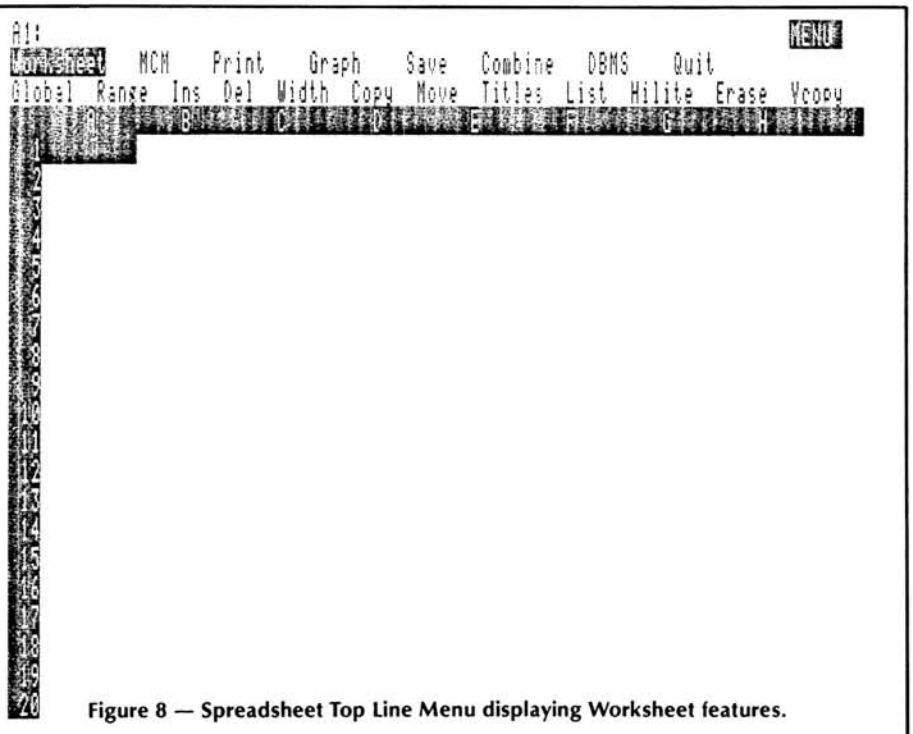


Figure 8 — Spreadsheet Top Line Menu displaying Worksheet features.

Good Guy Review Of Enable

George Elwood
1670 N. Laddie Court
Beavercreek, OH 45432

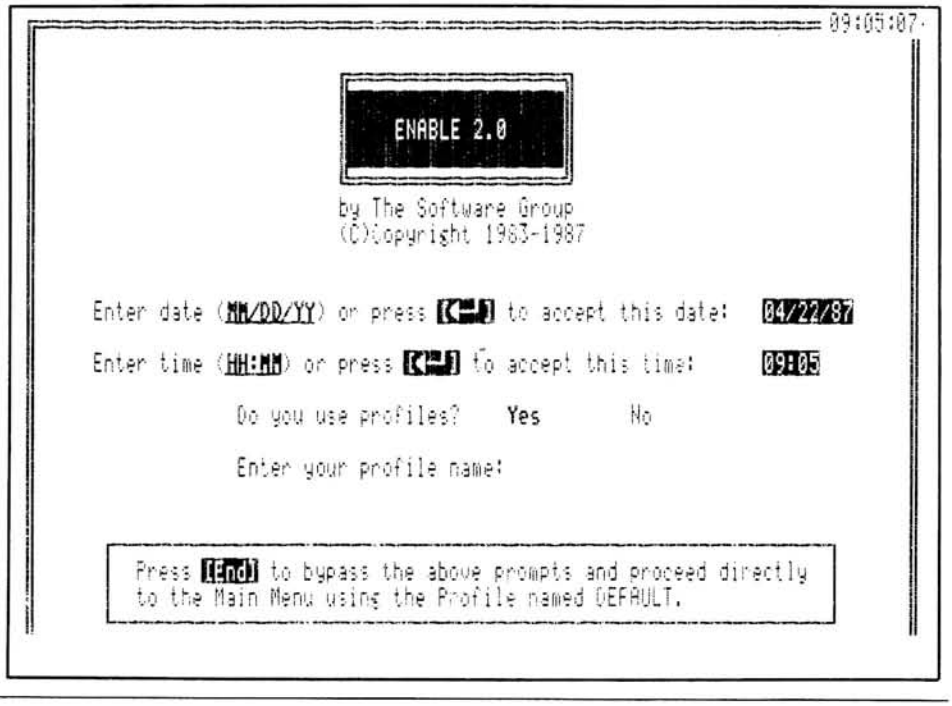
It is unusual to add a good guy to the list of Heath/Zenith supporters, but I have found one that deserves to be noticed. Since "Big Blue" arrived on the scene with their "standard system," the Z-100 has been left out of the main stream of new programs. By down grading the Z-100 with one of the emulators or ZPC, IBM compatibility could be obtained and "standard" programs could be made to run on the machine.

We have been lucky in that there are programs for the Z-100 that will do about everything that is important. WordStar, Lotus, and dBase II are available to handle the basic office functions, along with communications and graphics packages that provide added capability. There is even an integrated package of sorts in the form of Peachtext 5000. Until now, there has not been a good integrated package in the form of Symphony or Framework.

The Software Group has corrected this with their Z-100 version of ENABLE. This one package provides a word processor, a Lotus-like spreadsheet, a relational database and a communication package. ENABLE has been around a while in the PC world and with its selection as the integrated package for the Department of Defense standard microcomputer contract (Z-248) and other major contracts, it is becoming better known.

In the fall of 1986, the Software Group completed the Z-100 beta version of ENABLE, version 1.1Z. The Dayton Heath/Zenith Users' Group purchased a substantial number of copies of this program and proceeded to test all phases of this package. We found and reported bugs to the Software Group, which corrected them in the production version 2.0Z. We also made suggestions for improvements on things we would like to see.

ENABLE, like other integrated packages, is large. It comes on six disks and about five pounds of documentation. Each separate function has its own manual with a system overview as the link. It works better if it can be loaded on a hard disk (upgrade your Z-100 with the UCI EasyWIN), although it can be run from a dual floppy system. One unique feature permits different modules on different disks. I have two high capacity drives (1.2 Meg) on my Z-100 with parts of the program on both with the data going to a third disk.

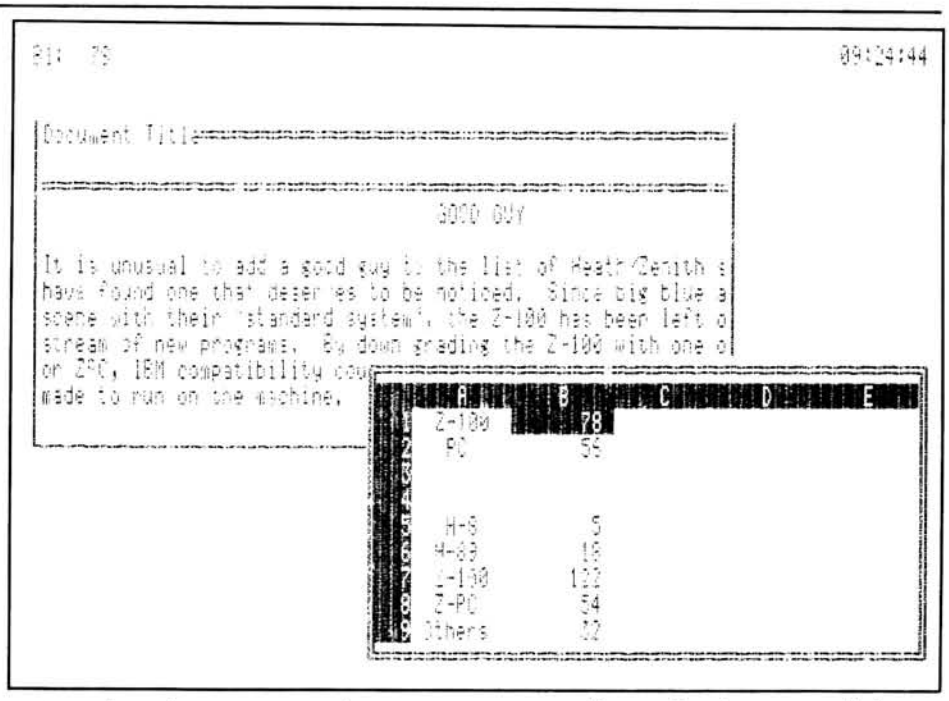


Installing ENABLE has been improved in version 2.0. All you have to do is place the installation disk in a floppy drive and type "install." It will first prompt you for printer selection(s). The Z-100 version has 46 printer drivers available, while the PC version has over 60. After selecting your printer type(s), the installation procedure continues with questions as to the type of media you wish to install, (h)ard or (f)loppy. The floppy procedure is further broken down into 360K, 720k 3-1/2 inch, or 8 inch.

The procedure is easy to follow with instructions as to which disk to place where. When you install ENABLE on a dual disk system, you will have to boot off another disk as ENABLE fills up the system utility disk.

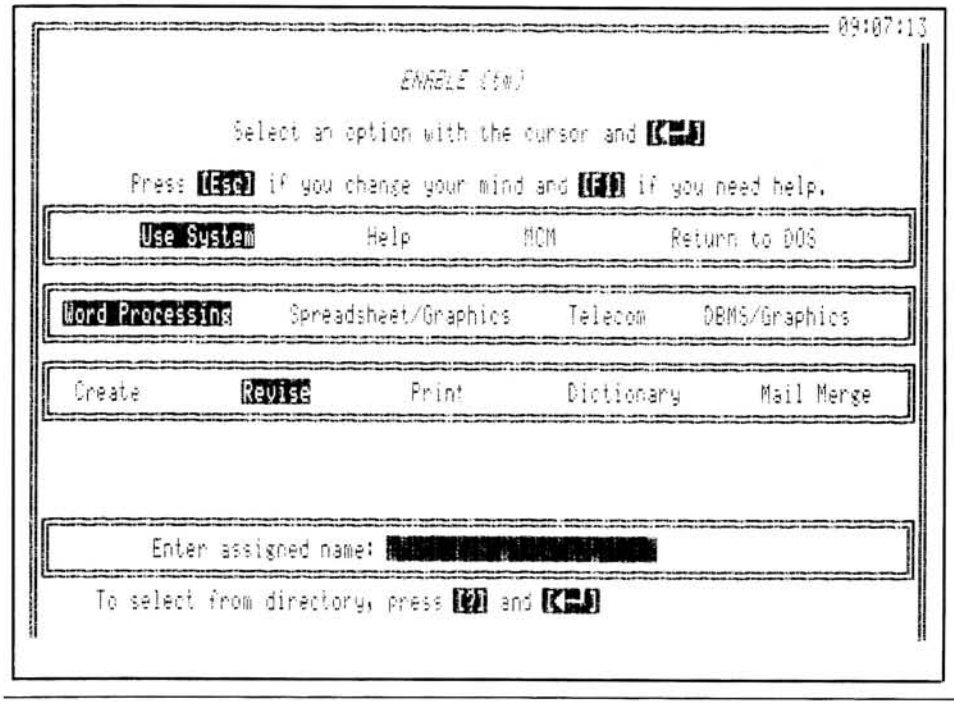
Once installed, it is best to use a batch (.BAT) file to instruct ENABLE how to operate. The ENABLE setup can be used to direct the basic program to look on other drives or directories for subprograms. These are shown by using the following: enablez (system disk drive:\path, operation disk drive:\path, tutorial disk drive:\path, utility disk drive:\path, data disk drive:\path) The basic system would be enablez (,,,,e:). This would indicate that ENABLE is located on partition E: and that you want the data to be placed in the same place. Notice that subdirectories may be added. To add the data files to a subdirectory called "enableda" on e: drive the batch file would read enablez (,,,,e:\enableda).

The first time through, you have to define profiles for the system. The profile tells the system the printer, screen colors, communication setups, word processing defaults, and other basic system functions. Once these have been defined, simply pressing SHIFT and 1 on the keypad (remapped END) at the same time will bring up the default system. See Figure 1 for the opening ENABLE screen and Figure 2 for the menu screen. You are now in the opening menu of the system. You can make selections by using the arrow keys or by typing in the first letter of your choice.



The Tutorial

ENABLE's tutorial makes use of the macro capability of the program. The tutorial is a set paced instruction that covers all phases of the program. Each of the ten lessons is about 30 minutes long and covers the section quite well. You select where the tutor will run by typing in "tutor h" or "tutor f." The macro then takes over and puts you into the basic tutor program. From there, you can go to any lesson that you wish.



MCM

In the opening menu there is an option called "MCM." This is the Master Control Module and it controls ENABLE from all other modules. It provides for windows, file management, macros, profiles, graphics, MCM-DOS interface, and a menu generator. Using the file management part you can copy, delete and get summaries of the files in the directory. You can change directories to check other status, also.

Windows

Enable supports up to eight windows at one time. By opening several windows, data can be transferred rapidly between modules. Moving graphs from the graphics module of the spreadsheet to the word processor requires three windows, the spreadsheet, the graph, and the word processor. By reducing the size of the windows, you can have several visible at one time. The tutorial package uses this capability. Figure 3 is the screen presentation with parts of a word processing and spreadsheet window open.

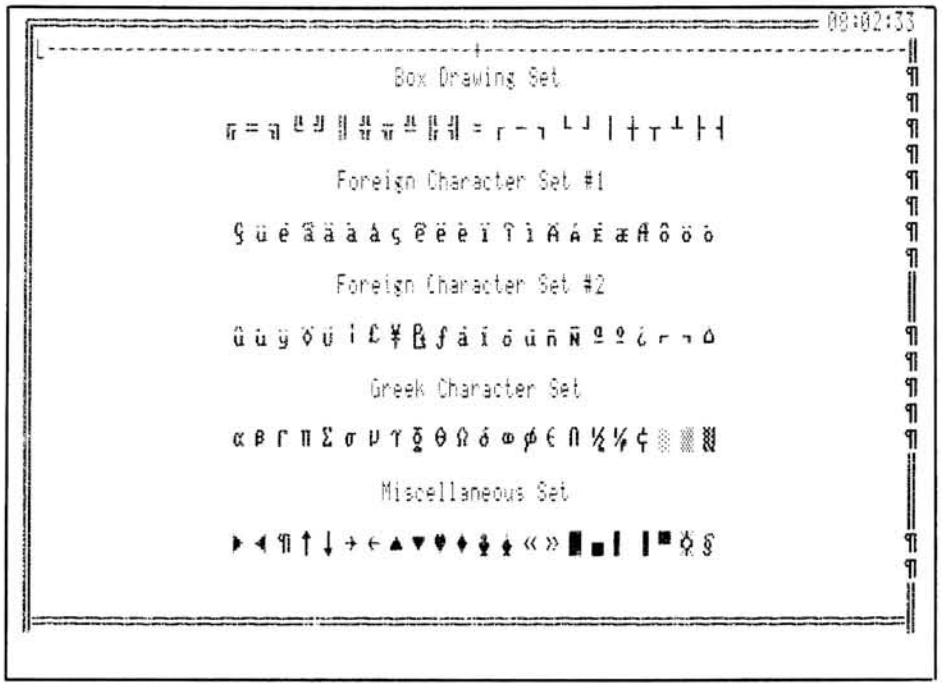
The Word Processor

To create a word processing document, you type "U W C" and ENABLE prompts you for a name. If you are revising a docu-

ment, you would type in "U W R" and then the name of the document. If you did not remember the name, you would enter "drive:\path ?." This will then display all ENABLE word processing documents in the selected directory. The same procedure could be used in any of the modules of ENABLE. The document opens with the option to put in a title, which is non-printing. The default ruler and tab settings appear on the top of the screen, if selected. The document is then input in the normal manner. By using the "F0" key (the remapped "ALT"), different functions can be added. "F0" with i, b, and/or u add italics, bold, and/or underline which are also displayed on the screen. The "F10" key provides a menu list at the top of the screen for additional functions. These same functions can be obtained using the "F9" key or "EXPERT COMMANDS" and up to three other keys. These functions include deletions, displays, inserts, finds, and save. ENABLE does use the keys of the Z-100. "HELP" will get you help, "D CHR" will delete the character under the cursor, "INS Line" will insert a line.

Once your document is completed, you can look at it in final mode by typing "F0 O D" with all printed features added. To change a printing feature from the default or selected profile type "F10 P". This brings up a five page printer menu which permits you to select printers, change characters per inch, change line spacing, select number of copies, etc. "F0 F10" brings you back. You can print from the document by typing "F0 F2" or save the document by typing "F0 F10." You can also print from the main menu. ENABLE V2.0 has added a print buffer so you can have several documents set to print while you are doing something else. The save menu permits you to change the name or format. You can save the document as an ENABLE document or WordStar, Easy, Multimate, Volkswriter, Peachtext, ASCII, DCA, or DIF formats. ENABLE will also read these formats.

ENABLE also provides special characters. There is a box drawing set available along with Greek characters and three other foreign character sets which can be printed if your printer supports them. These are easily turned on or off using the menu or the EXPERT COMMANDS "F9." Figure 4 is a presentation of these special character sets. One thing that saved me when I first started using ENABLE was the undelete key. Having been a long time WordStar user, I continued to use the Z-100 "DELETE" key. In ENABLE, this deletes the line from the cursor. By selecting the top menu



"F10" and "(D)elete" and then "0," you can restore what you last deleted. The word processor also permits a limited column calculator capability within the document. The spell checker with 80,000 words is also available from the top menu. You can add words easily and this new dictionary can be checked using the word processor. Like CorrectStar 3.31, ENABLE provides some suggested spellings. It not only checks the document for misspelled words, but also checks for duplicate words.

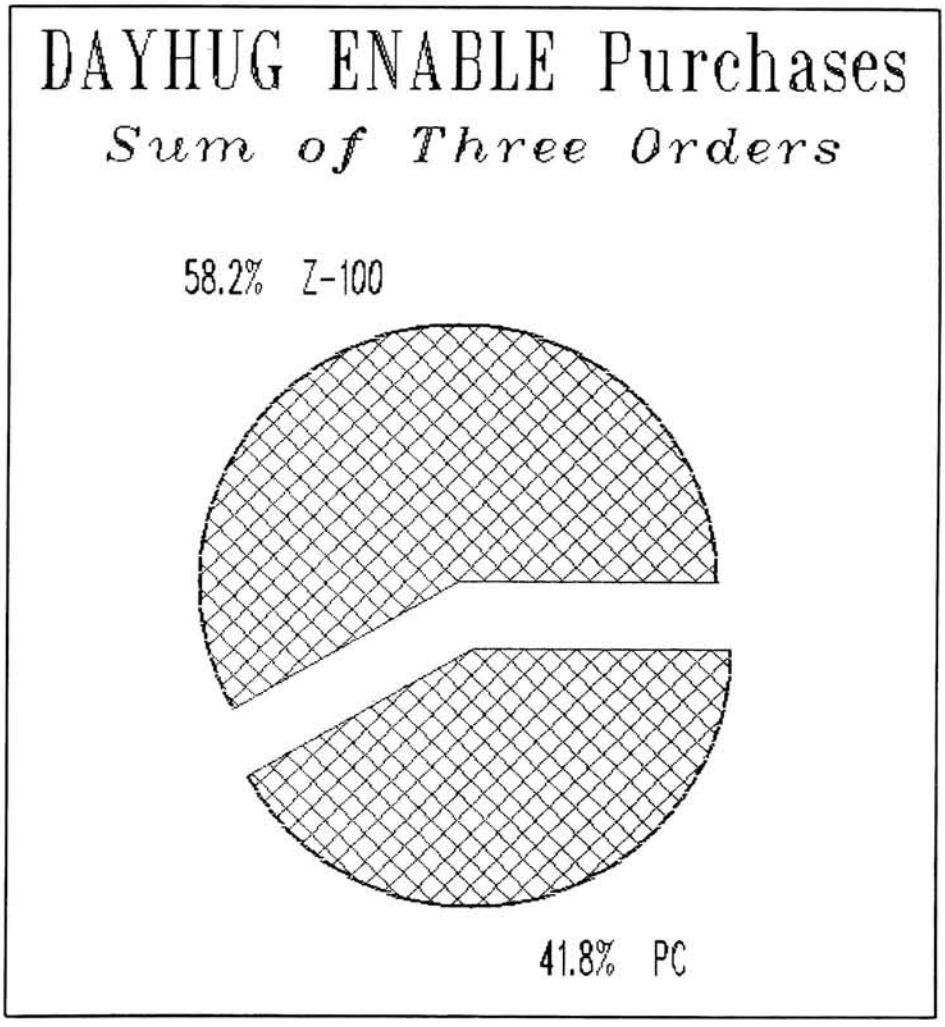
The SpreadSheet

The spreadsheet is Lotus-like. If you know Lotus, you can use ENABLE's spreadsheet without a problem. ENABLE will import/export LOTUS 1-2-3, SuperCalc 3, Symphony, and ASCII files and will run LOTUS macros. The spreadsheet is adaptable in size with six selections. Row and column sorting has been added in version 2. The graphics function is much improved over Lotus with the capability to store up to eight graphs so when you change data in the spreadsheet all charts will be updated. Unlike Lotus, you build the graphs in the spreadsheet and can print them from there. ENABLE has the normal graphs along with 3D bars, exploded pies, and High-Low-Close charts. The PC version has additional 3D graphing available. The graph size and orientation can be changed to meet your needs with both landscape and portrait modes available. Automatic scaling of the axis along with fixed scaling is available. You have the option of selecting four different font types and can use the different types on the same graph. Figures 5 and 6 are two examples of the graphing capability of ENABLE.

The part of the spreadsheet/graphics that makes ENABLE great is the capability to transfer a graph into a word processing document. By opening a window, you can transfer the chart into the document and print it if your printer supports graphics. Sections of the spreadsheet can also be transferred to the word processing window. The spreadsheet can be used to build a database and then transfer this information to the database window. To aid you, it has macro capabilities not only in the spreadsheet, but throughout the package.

The Database Management System

The relational database management system is easy to use. Again, it is accessed through the main menu. Like the other two modes, the database management system (dbms) will read other program files like dBase II, dBase III, PC-File III, and Condor 3. Going into the system, you are prompted to create or use an available database. Building a new database is made easy by the prompts from ENABLE. It permits quick or detailed definitions of fields. The detailed definitions permit assigning dates, telephone numbers, state codes, and ZIP codes. If you use the detailed definitions, ENABLE does error checking to make sure the area code matches the state codes which match the ZIP codes. Date definition permits assigning earliest and latest date. Building the input/output form uses the word processing function. You type whatever you want and use a "here is where I want the field" to finish it off. The



output forms are built the same way. Once built, you can call them up to display data in whatever form you wish. Again, data can be transferred between windows from the database. Graphing is also available. Because the dbms is relational, you can update several databases using one input form. An example of one use of the dbms is in tracking telephone bills. One database would contain the names, company, telephone number, and other required information. A second database would contain the monthly telephone bill. As you input the telephone number, ENABLE would check the contact database and place the requested data on the screen. This would then relate the cost of the calls and time spend to the company contacted.

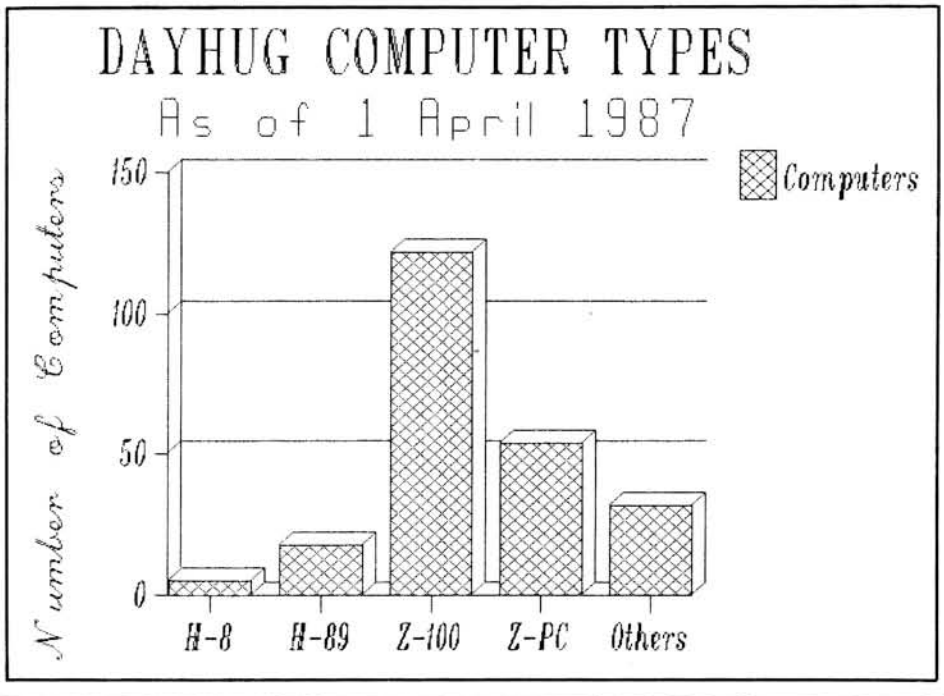
Telecommunications

The telecommunications mode is the last mode in this four-mode program. It does provide a convenient means of transferring documents to and from ENABLE. It provides a VT52, VT100, and AT&T 4410 emulation. The program provides several file transfer options, XModem, Kermit, ENABLE's own protocol, and buffer capture with only XON/XOFF. Using the macro capability, fully automatic transfers are possible. The capture mode is a single key, "F7." When pressed, all data displayed is captured. ENABLE capture can send files directly to the word processor. When using the ENABLE to ENABLE protocol, complete files, including special characters, can be sent. The ENABLE protocol provides more error checking than the standard XModem. One nice feature in the ENABLE telecommunications mode is the on screen time and cost display. You can select this option from the menu which provides a running log on time and cost on the 25th line.

All in all, if you want one software package that does everything, ENABLE is for you. It is not perfect in all areas, but it is good enough to get everything done. The capability to transfer between modules has been an asset in several projects I have been working on and is a key to the success of this program. I have not covered all areas of ENABLE, but have provided an overview which should provide you some insight into the capabilities of the program.

ENABLE is the trademark of the Software Group. Volkswriter is the registered trademark of Lifetree Software, Inc. Multimate is the trademark of Softword Systems, Inc. WordStar and CorrectStar are the registered trademarks of MicroPro International Corporation. dBase II, dBase III and Framework are the registered trademarks of Ashton-Tate. 1-2-3, Lotus, Symphony and VisiCalc are the registered trademarks of Lotus Development Corporation. Condor 3 is the registered trademark of Condor Computer Corporation. Peachtext 5000 is the registered trademark of Peachtree Software. SuperCalc 3 is the registered trademark of Sorcim/IUS Micro Software.

*



Mastering Equations With Equation Master

Christopher A. Feuchter
Suite 69
160 Washington SE
Albuquerque, NM 87108

Until recently, equations and text didn't mix in the world of word processing. If you wanted to create equations, such as appear in Figure 1, you probably did them by hand. There are several word processing packages now on the market which provide such a capability. The problem is that they are expensive — $200 and up. Most people with only an occasional need for this capability, myself included, are going to be hesitant to make such an investment. But an occasional need is still a need. I started thinking about a companion program for my word processor which could assemble and print equations. In a two step process, my word processor could create and print the text; the companion program would create and print the equations. Thinking led to experiments, experiments to a code. The result was the immodestly named Equation Master, henceforth referred to as EM, a GW-BASIC code (see listing) for my IBM PC compatible Zenith 150 and NEC 8023A-C dot matrix graphics printer.

EM BASICS

The basic goal of EM is to be able to construct virtually any equation quickly and easily on the monitor and then be able to print the equation where and when desired. To increase its usefulness, EM allows editing and positioning of the equation, and provides for storing and retrieving it from disk. And since a user's needs cannot be completely foreseen, it allows the user to easily add any additional symbols that are required.

The first line of the screen is used to communicate with the user. All prompts are displayed there. Lines 4 through 21 are used for equation construction. This area is defined top and bottom with lines of dashes marked every 5 columns as a positioning aid. The 10 function keys along with Ctrl-B, Ctrl-C and Ctrl-D initiate all of EM's functions. These functions are discussed in the text below and summarized in Table 1. The Escape key is used throughout the program to cancel functions in progress and return to an "Enter command" mode.

Zenith Z-150 Screen Graphics

The Zenith Z-150 (also known as the Z-100 PC) has a reputation for being the most compatible of the many IBM PC clones. Its high resolution monochrome graphics are functionally identical to those of the IBM PC. The screen can display 640 horizontal pixels (numbered left to right, 0 - 639) by 200 vertical pixels (numbered top to bottom, 0 - 199). Each pixel can be individually turned on or off, and the "on" or "off" status of each pixel can be individually determined. Characters are based on an 8 pixel by 8 pixel square — thus 25 lines of 80 characters.

The Zenith graphics video card ROM contains a set of 128 characters usable in graphics mode. Except for the upper and lower case alphabets and the numerals 0 - 9, only a few of these characters are useful in equation building. However, up to 128 additional graphics characters may be supplied by the user. This is done by altering the address pointed to by interrupt 1FH to point to the memory containing the user-defined characters. The additional characters presently included in EM are displayed in Figure 2 with their character codes. Character code use is discussed below. This display is obtained from EM by typing Ctrl-D. Additional symbols can be defined by the user as explained in the first box. While most of EM's additional characters are complete symbols, some multicharacter symbols are present. For example, characters 214 - 217 constitute a large upper case sigma and 218 - 221 combine to form a large upper case pi. Additionally, the integral sign (188 - 189), brackets (203 - 206) and braces (207 - 212) are formed of at least two characters and may be augmented with the vertical line with code 202.

Building Equations

The key to an easily understood, visually pleasing formula is proper horizontal and vertical symbol placement. This implies that a measure of aesthetic judgement is required in building equations. Because placement is critical, equation building can not be limited by the restrictive 25 line, 80

Table 1
A summary of EM's functions and key assignments.

FUNCTION KEY	FUNCTION	REMARKS
1	Enter Character	Keyboard Character
2	Enter Character Code	0 - 255
3	Move Character to Cursor	Ctrl-F3 to OR with Screen Contents
4	Repeat Character	e.g., Horizontal Lines
5	Enable Edit Mode	Move Cursor with Arrows, Use Insert, Delete Keys
6	Copy Character	Bypass character code
7	Save Equation	File Name Required
8	Read Equation	File Name Required
9	Enable Equation Shift	Use arrows
10	Print Equation	Escape to interrupt
Ctrl-B	Return to BASIC	
Ctrl-C	Clear Construction Area	
Ctrl-D	Display User-Defined Characters	

column screen format. Equations containing super- and subscripts, special multicharacter symbols, and horizontal and vertical dividing lines require more flexibility. The issue is how precise horizontal and vertical symbol placement must be. Too little precision will result in improperly or unattractively spaced symbols; too great a precision will result in extra work and difficulties in maintaining consistent spacing. The maximum possible precision is one horizontal or vertical pixel. However, experiments have shown that 4 pixels — one half a line or character — is a quite satisfactory minimum.

It is obvious that the need to position symbols to the nearest half line or half character precludes directly typing the symbols into their final positions. A typed character can only occupy one of the 2000 positions defined by rows and columns. EM solves this problem by displaying each symbol at the top of the screen before it is placed in the equation. By using BASIC's GET and PUT graphic image transfer functions, the cursor can be moved up, down, left and right in steps of 4 and 8 pixels. With the cursor properly positioned, the PUT function can transfer the symbol to the cursor's location. This procedure allows the cursor to be positioned with full knowledge of the symbol's size and approximate location within the eight pixel by eight pixel matrix.

User-defined characters cannot be typed from the keyboard. Each of them must be referenced by its character code. This leads to two character entry modes — 'Enter character' and 'Enter character code'. The first is initiated by F1; the second by F2. Any key (or combination of keys) which produces a character can be typed in response to 'Enter character'. Any character code from 0 to 255 can be entered in response to 'Enter character code'. Character codes 0 - 127 refer to the ROM character set. Most of these correspond to the keyboard characters or a key pressed simultaneously with the Control key as defined in the BASIC manual. They are normally entered directly. User defined character codes begin at 128. User characters are assigned codes according to the order in which they are defined in DATA statements.

When the character is to be transferred from the top line into the equation, pressing ENTER causes the cursor to appear and blink. The cursor can be positioned in 8 pixel increments using the numeric pad arrows. If the shift key is depressed simultaneously with the arrows, the increment size is reduced to 4 pixels. When the cursor is in the desired location, the character can be transferred by pressing F3. The cursor disappears after the transfer, although its position moves one space to the right. Pressing F4 after F3 causes the character to be repeated at the new cursor location. Holding F4 down results in multiple repetitions. This is useful, for example, in constructing horizontal lines or decorative borders.

The character transfer normally destroys the screen's contents at the cursor position. However, if the Control key is depressed at the same time as either F3 or F4, the transferred character is ORed with the screen contents at the cursor's position. This allows placing symbols in situations where destructive transfer would destroy all or parts of existing symbols, a frequent possibility when building equations.

Editing Equations

The ability to edit an equation at any stage of its construction is a must. A special EM editing mode is entered by depressing F5. The cursor appears and may be moved in 4 or 8 pixel increments with the numeric pad arrows as described above. In this mode, blanks may be inserted at the cursor with the Insert key. Half blanks may be inserted by pressing Shift-Insert. Characters may similarly be deleted at the cursor with the Delete key. Because symbols may occupy more than one line, the insert/delete operations may have to be performed on two or more lines to maintain character or equation integrity.

While in the F5 edit mode, F6 transfers the character at the cursor to the top of the screen. From here it can be transferred back into the equation in a new position in the usual manner. This permits repeated use of a character in an equation without having to repeatedly resort to finding its character code. It also permits easy duplication of composite or partial characters which may have been created.

An equation positioning mode is enabled with F9. In this mode, the entire construction area (lines 4 - 21) can be shifted in 4 pixel increments in any direction using the arrows. Text which reaches a boundary of the construction area is wrapped around, appearing on the opposite side of the screen. This allows positioning the equation prior to printing.

Saving The Equation To Disk

The F7 and F8 keys control saving and retrieving the construction area to and from

disk, respectively. F7 produces a message asking the user for a file name in which to save the entire construction area. When a name has been typed and the 'Enter' key pressed, the GET function transfers the contents of the construction area to an array. Zero bytes within the array represent background and aren't saved to disk. Non-zero bytes of the array are written to the file along with their positions within the array. These files are approximately 2000–3000 bytes in length.

Equations saved in this manner can be restored to the screen using the F8 function key. The file name is entered and the file is read into the storage array. The equation is then immediately transferred to the screen with the PUT function.

Printer Basics

Dot matrix graphics printers are a diverse lot. While their basic capabilities are generally similar, in that they can print characters in text mode and dot images in graphics mode, each model seems to have its own peculiarities. In this section, the principles of printing a screen graphics image (viz., the equation) will be presented using the NEC 8023A-C as a model. The second box discusses the changes which must be considered when using some other popular printers.

The NEC printer has a print head which contains a vertical arrangement of nine pins. If a pin is activated, it strikes the ribbon leaving a dot on the paper approximately 1/72 of an inch in diameter. Otherwise, the paper beneath the pin remains blank and appears as background. Columns composed of dots and background spaces are printed one after the other as the print head moves across the paper.

Characters printed in text mode are composed from a rectangular matrix of dots and spaces. Which pins are activated to print a character is determined by the printer ROM depending upon which byte is sent. For example, an "A" is printed when the decimal value of the byte sent is 65. Two character matrix sizes are used by the NEC, 7 horizontal by 9 vertical and 8 by 8.

In graphics mode, printer operation is entirely different. For most printers, including the NEC, a graphics byte represents one

This figure shows some of the equations which can be created with the Equation Master code described in this article. All that is required, in addition to the code, is a suitable dot matrix graphics printer. Below is a description of the carbon cycle taken from Martin Schwarzschild's 1959 edition of "Structure and Evolution of the Stars".

$$C^{12} + H^{1} \rightarrow N^{13} + \gamma \qquad +1.95\ \text{Mev}\ (1.3 \times 10^{7}\ \text{yrs})$$
$$N^{13} \rightarrow C^{13} + e^{+} + \nu \qquad +2.22\ \text{Mev}\ (7\ \text{min})$$
$$C^{13} + H^{1} \rightarrow N^{14} + \gamma \qquad +7.54\ \text{Mev}\ (2.7 \times 10^{6}\ \text{yrs})$$
$$N^{14} + H^{1} \rightarrow O^{15} + \gamma \qquad +7.35\ \text{Mev}\ (3.2 \times 10^{8}\ \text{yrs})$$
$$O^{15} \rightarrow N^{15} + e^{+} + \nu \qquad +2.71\ \text{Mev}\ (82\ \text{sec})$$
$$N^{15} + H^{1} \rightarrow C^{12} + He^{4} \qquad +4.96\ \text{Mev}\ (1.1 \times 10^{5}\ \text{yrs})$$

Next comes the definition of the vector cross product,

$$\vec{A} \times \vec{B} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} = (a_2 b_3 - b_2 a_3)\,\hat{i} + (a_3 b_1 - b_3 a_1)\,\hat{j} + (a_1 b_2 - b_1 a_2)\,\hat{k}$$

followed by an indefinite integral,

$$\int \frac{x\,dx}{(a+bx)^2} = \frac{1}{b^2}\left[\log(a+bx) + \frac{a}{a+bx}\right]$$

and a series expansion of the exponential.

$$e^{x} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

Last is Schrodinger's wave equation in spherical coordinates accented with a border.

$$\nabla^2\psi + \frac{8\pi^2 m}{h^2}(H - U)\,\psi = 0$$

Figure 1

An example of EM's capability to construct and print equations. Note that the letters and numbers appearing in the equations are different from those in the text. Those in the equations are copied exactly from the screen graphics characters, which differ from the corresponding printer characters.

column of eight dots and spaces (the ninth pin is not used in graphics mode). Each zero bit in the byte corresponds to a space. Each set bit corresponds to a dot. NEC 8023A-C vertical spacing between dots is 1/72 inch. Normally, the printer prints six lines of characters per inch. Seventy-two dots per inch divided by six is 12 dots per line, or a vertical background spacing between lines of 12 - 8 = 4 dots. Since the screen is to be copied exactly, these four dots between lines must be eliminated. This is done by sending the printer the correct line spacing. With the NEC printer, spacing can be specified in 144ths of an inch. The control sequence to do this is &H1B and &H54 followed by the two digit ASCII representation of the spacing in 144ths of an inch. In this case, the correct spacing is "16".

With the line spacing properly set, the graphics mode is entered by sending the printer another control sequence. This sequence is &H1B and &H53 followed by a four digit ASCII representation of the number of graphics bytes to follow. Because entire screen lines are being printed, this number is set to "0640". Each of the graphic bytes must be determined by examining the status of each pixel in the vertical line of eight pixels which it represents. There are several possible ways of doing this. Using the POINT function to determine pixel status is the most easily understood method, if not the fastest. Since this part of the program must be adjusted to accommodate different printers, simplicity dictates its use. In the case of the NEC 8023A-C, the highest order bit of the byte corresponds to the bottom pin.

Printing The Equation

Users of EM will invent their own methods of measuring the equation and properly aligning the paper to insure that the equation goes in the desired position within the previously printed text. As an aid to correctly inserting the equation into the text, EM spaces one line (1/6 inch) before actually printing. This facilitates using the text as a spacing guide. The printing always starts with the topmost line of the construction area, proceeding down line-by-line until the escape key is struck or the entire area is printed. Unfortunately, interpreted GW-BASIC requires about one minute to process a line. Thus, in the interest of saving time, it is usually best to move the equation being printed to the first line of the construction area. The equation may also be moved right or left at this time to properly align it with the text into which it is being inserted. The number of the line being printed is displayed at the top of the display. Printing is done only left-to-right to insure alignment of the equation's symbols.

Because GW-BASIC, like most BASICs, gratuitously sends line feeds to the printer after a certain number of characters have been printed using LPRINT CHR$(N);, the graphics characters are sent to the printer via a simple machine language program which is poked into memory. DOS function 5 is used for this purpose to send bytes to the parallel port.

Summary

EM should adapt almost effortlessly to any combination of IBM compatible PCs and graphics printers. For people with an occasional equation or two, it should be all that's ever needed. Users with equation-filled manuscripts to write, however, should look into the several WP packages on the market which include equation building features.

Box 1
User-Defined Characters

All standard ASCII screen graphics characters fit within an 8 pixel by 8 pixel matrix. To provide background between characters, the rightmost column of pixels is not used except for the underline. The bottom row of pixels is used only for descenders (the letters g,j,p,q,y) and for the underline. Generally, these conventions should be followed for user-defined characters as well. This insures that there is no problem mixing standard ASCII and user defined characters.

A user-defined symbol is defined in EM by an ordered sequence of 8 bytes. Each byte represents one row of 8 pixels. The first byte corresponds to the top row, the second byte to the next row, etc. Within each byte, each bit corresponds to one of the 8 columns of pixels; the most significant bit corresponds to the leftmost column. A bit which has a value of one represents a pixel which is part of the character. A bit which has a value of zero is part of the background. An example is given below for the capital Greek letter psi. The character is shown with the corresponding binary and hexadecimal row values.

```
XX X XX   11010110 = D6
 X X X    01010100 = 54
 X X X    01010100 = 54
  XXX     00111000 = 38
   X      00010000 = 10
   X      00010000 = 10
  XXX     00111000 = 38
```

For each user-defined character, the 8 bytes are stored sequentially in memory using DATA, DEF SEG, READ and POKE statements. By placing the starting address of the user-defined character data in INT 1F, the characters become "standard" graphics characters. Because the segment used by BASIC for the POKE function is not available to the programmer, the DEF SEG statements are required. The first (DEF SEG=&H2000) allows the characters to be placed at a known absolute address in memory. The second (DEF SEG=0) allows that address to be placed in INT 1F, and the third (DEF SEG) simply returns to BASIC's default value. The value &H2000 works fine for a 320K memory, but the value may have to be altered for systems with different amounts of memory.

The order of the characters (i.e., their characters codes) is determined by the order in which they appear in the DATA statements. Additional symbols can be added to those already provided. Simply add the necessary DATA statements and increase by eight per user character the upper bound of the FOR-NEXT loop which reads and pokes bytes. The maximum number of user characters permitted in EM is 128.

Box 2
Printer Variations

There are four areas which may require attention when adapting EM to work with other printers. The first concerns the control sequences necessary to change line spacing and enter the graphics mode. Line spacing generally presents little problem. The existing lines in EM which control spacing can simply be replaced with other lines appropriate to the new printer. There are two distinct philosophies, however, associated with entering (and leaving) the graphics mode. The first and simplest philosophy is that used by the NEC 8023A-C, Epson printers and others. When entering the graphics mode, the number of bytes to be printed before returning to the text mode is declared. The second philosophy is typified, for example, by the Okidata Microline printers. This philosophy requires sending a control sequence to the printer while in the graphics mode to return to the text mode — a sequence which could possibly occur in the normal course of sending bytes to be printed. In the case of the Okidata printers, return from the graphics mode is accomplished by sending the equivalent of CHR$(3); CHR$(2). Every CHR$(3) which is to be treated as a graphics byte must be sent twice as CHR$(3);CHR$(3). This requires putting a test into the print loop to handle those situations, as well as sending the exit code.

The second area concerns the correspondence of the bits within the graphics byte to the print head pins. Many printers, such as the Epson printers, reverse the correspondence described for the NEC printer, i.e., the highest order bit in the graphics byte corresponds to the top pin. This is easily handled. Simply reverse the limits of the JJ% FOR-NEXT loop in the 'PRINT FORMULA' section of the code and change the step from -1 to 1. This causes the screen pixels to be examined in the opposite direction, inverting their position within the graphics byte.

Finally, some printers print seven vertical dots instead of eight (Okidata) or, in at least one case, print the dots horizontally, as with the PR 2300 Olivetti. The first of these can be handled easily by printing 20 lines instead of 18 and only examining 7 pixels instead of eight in the JJ% loop mentioned above. In the later case, extensive changes would be needed.

Although EM includes no provision for adjusting the horizontal spacing of dots, some users might like to experiment. Many printers support different spacings in the graphics mode.

Listing

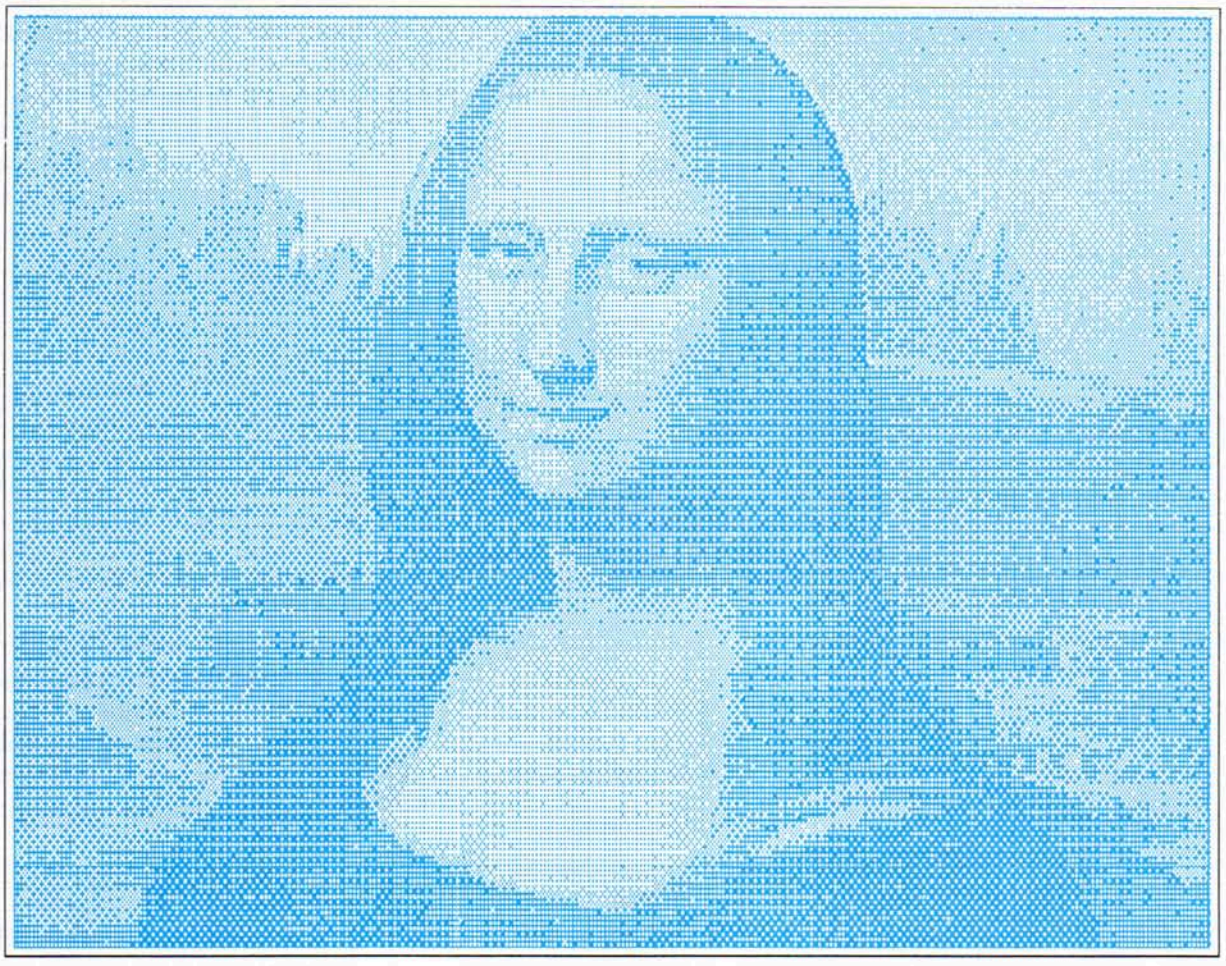
```
10 CLEAR : SCREEN 2
20 DEFINT W,Y
30 DIM L(3),LL(3),LLL(3),LLLL(3),W(1000),WP(15),X(161),XX%(640),Y(5762)
40 '      enter printer program
50 DEF USR0=&HF800
60 FOR I=0 TO 6
70 READ WP(I): POKE &HF800+I,WP(I)
80 NEXT I
90 '      transfer user characters to memory and set INT 1F to point to them
100 DEF SEG=0
110 POKE &H7C,0: POKE &H7D,0:  'INT 1F offset
120 POKE &H7E,0: POKE &H7F,&H20:  'INT 1F segment address(from DEF SEG below)
130 DEF SEG=&H2000
140 FOR I=0 TO 751:  'add 8 to upper limit for each additional user character
150 READ W(I): POKE I,W(I)
160 NEXT I
170 DEF SEG
180 '      define an 8 x 8 pixel blank and cursor
190 LOCATE 1,1: PRINT " "
200 GET (0,0)-(7,7),LL
210 GET (0,0)-(7,7),LLLL:  'blank
220 GET (0,0)-(7,7),LLL
230 K=VARPTR(LLL(2))
240 FOR I=-4 TO 3: POKE K+I,255: NEXT I
250 '      set up screen
260 CLS: FOR I=1 TO 10: KEY I,"": NEXT I
270 C$="": FOR I=1 TO 16: C$=C$+"----+": NEXT I
280 LOCATE 3,1: PRINT C$
290 LOCATE 22,1: PRINT C$
300 BLANKS$=STRING$(40," ")
310 KROW=13: KCOL=20:  'initial cursor position
320 '
330 '      BRANCH ON FUNCTION CODES
340 AA$="  ": LOCATE 1,1: PRINT BLANKS$: LOCATE 1,1: PRINT "Enter Command"
350 A$=INKEY$:IF A$="" THEN GOTO 350
360 IF LEN(A$)=1 AND ASC(A$)<>2 AND ASC(A$)<>3 AND ASC(A$)<>4 THEN GOTO 350
370 A$=RIGHT$(A$,1):A=ASC(A$)
380 IF A=2 THEN CLS: END:  'Ctrl-B, return to BASIC
390 IF A=3 THEN GOTO 2090:  'Ctrl-C, clear formula
400 IF A=4 THEN GOTO 2130:  'Ctrl-D, display secondary characters
410 ON A-58 GOTO 460,590,740,1210,1260,350,1370,1480,1620,1880: 'F1 - F10 keys
420 IF A=97 THEN GOTO 1220:  'Ctrl-F4
430 GOTO 350
440 '
450 '      F1 - ENTER CHARACTER
460 LOCATE 1,1: PRINT BLANKS$: LOCATE 1,1: PRINT "ENTER CHARACTER"
470 GX1=128: GY1=0: GX2=135: GY2=7: DGY=0: DPY=0:
    'character transfer parameters
480 A$=INKEY$: IF A$="" GOTO 480
490 IF ASC(A$)=27 THEN GOTO 340:  'escape
500 IF ASC(A$)=95 THEN DGY=7: DPY=7:  'underscore
510 LOCATE 1,17: PRINT A$: LOCATE 1,18: PRINT "_"
520 B$=INKEY$: IF B$="" THEN GOTO 520
530 IF ASC(B$)=27 THEN GOTO 340:  'escape
540 IF ASC(B$)=13 THEN GOTO 740:  'enter
550 IF ASC(B$)=8 THEN GOTO 460:  'backspace
560 GOTO 520
```

```
570 '
580 '     F2 - ENTER CHARACTER CODE
590 LOCATE 1,1: PRINT BLANKS$: LOCATE 1,1: PRINT "ENTER CHARACTER CODE _"
600 NC=22: GOSUB 2280:  'get character code
610 B=VAL(R$)
620 IF B<0 OR B>255 GOTO 590
630 GX1=168: GX2=175: GY1=0: GY2=7: DGY=0: DPY=0:
    'character transfer parameters
640 LOCATE 1,22: PRINT BLANKS$
650 LOCATE 1,22: PRINT CHR$(B)
660 LOCATE 1,23: PRINT "_"
670 B$=INKEY$:IF B$="" THEN GOTO 670
680 IF ASC(B$)=27 THEN GOTO 340:  'escape
690 IF ASC(B$)=13 THEN GOTO 740:  'enter
700 IF ASC(B$)=8 THEN GOTO 590:  'backspace
710 GOTO 670
720 '
730 '     F3 - POSITION CURSOR & PLACE CHARACTER
740 GET (GX1,GY1+DGY)-(GX2,GY2+DGY),L:  'character to be moved
750 GET ((KCOL-1)*8+1,(KROW-1)*8)-(KCOL*8,(KROW-1)*8+7),LL
760 A$=INKEY$
770 PUT ((KCOL-1)*8+1,(KROW-1)*8),LLL:  'display cursor
780 FOR I=1 TO 50: NEXT I:  'blinking delay loop
790 PUT ((KCOL-1)*8+1,(KROW-1)*8),LL,PSET:  'display original character
800 IF A$="" THEN GOTO 760 ELSE A=ASC(A$)
810 IF A=27 THEN GOTO 330:  'escape
820 IF ASC(A$)=0 THEN GOTO 860:  'extended character code
830 IF A=46 OR A=48 THEN GOTO 950:  'half character insert/delete?
840 '     move cursor?
850 IF A<>50 AND A<>52 AND A<>54 AND A<>56 THEN GOTO 760:  'half space?
860 A=ASC(RIGHT$(A$,1))
870 IF A=72 THEN KROW=KROW-1 ELSE IF A=56 THEN KROW=KROW-1/2:  'cursor up?
880 IF A=80 THEN KROW=KROW+1 ELSE IF A=50 THEN KROW=KROW+1/2:  '     down?
890 IF A=77 THEN KCOL=KCOL+1 ELSE IF A=54 THEN KCOL=KCOL+1/2:  '    right?
900 IF A=75 THEN KCOL=KCOL-1 ELSE IF A=52 THEN KCOL=KCOL-1/2:  '     left?
910 '     edit mode?
920 IF ASC(AA$)<>63 THEN GOTO 1120:  ' edit mode if = 63
930 IF A=64 THEN GOTO 1310:  '  F6
940 '     insert space or half space?
950 IF A<>82 AND A<>48 THEN GOTO 1030: 'insert space if = 82, half space if 48
960 IF A=48 THEN GET ((KCOL-1)*8+1,(KROW-1)*8)-(629,(KROW-1)*8+7),X
970 IF A=82 THEN GET ((KCOL-1)*8+1,(KROW-1)*8)-(625,(KROW-1)*8+7),X
980 PUT ((KCOL-1)*8+1,(KROW-1)*8),LLLL,PSET
990 IF A=48 THEN PUT ((KCOL-1)*8+5,(KROW-1)*8),X,PSET
1000 IF A=82 THEN PUT (KCOL*8+1,(KROW-1)*8),X,PSET
1010 GOTO 750
1020 '     delete space or half space?
1030 IF A<>83 AND A<>46 THEN GOTO 1120:
     'delete space if = 83, half space if 46
1040 PUT ((KCOL-1)*8+1,(KROW-1)*8),LL
1050 XX=((KCOL-1)*8-3-627): XX=INT((XX+7)/8)*8
1060 IF A=46 THEN GET (KCOL*8-3,(KROW-1)*8)-(627,(KROW-1)*8+7),X
1070 IF A=83 THEN GET (KCOL*8+1,(KROW-1)*8)-(625,(KROW-1)*8+7),X
1080 PUT ((KCOL-1)*8+1,(KROW-1)*8),X,PSET
1090 PUT (625,(KROW-1)*8),LLLL,PSET:  'blank at end of line
1100 GOTO 750
1110 '
1120 IF A=61 OR A=96 THEN GOTO 1210:  'F3 or Ctrl F3?
1130 '     wrap cursor around screen?
1140 IF KROW<4 THEN KROW=21:  'cursor wraparound top to bottom
1150 IF KROW>21 THEN KROW=4:  '                  bottom to top
1160 IF KCOL>79 THEN KCOL=1:  '                  right to left
1170 IF KCOL<1 THEN KCOL=79:  '                  left to right
1180 GET ((KCOL-1)*8+1,(KROW-1)*8)-((KCOL-1)*8+8,(KROW-1)*8+7),LL
1190 GOTO 760
1200 '      F3 or Ctrl-F3
1210 IF A=61 OR A=62 THEN PUT ((KCOL-1)*8+1,(KROW-1)*8+DPY),L,PSET
1220 IF A=96 OR A=97 THEN PUT ((KCOL-1)*8+1,(KROW-1)*8+DPY),L,OR
1230 KCOL=KCOL+1: IF KCOL>79 THEN KCOL=1:  'move cursor right
1240 GOTO 330:  'next command
1250 '
1260 '     F5 - EDIT MODE
1270 LOCATE 1,1: PRINT BLANKS$: LOCATE 1,1: PRINT "EDIT MODE, (ESC) TO END"
1280 AA$=A$:  'for later test
1290 GOTO 750
1300 '
1310 '     F6 - DUPLICATE CHARACTER (only in edit mode)
1320 GET ((KCOL-1)*8+1,(KROW-1)*8)-(KCOL*8,KROW*8),L: 'get character at cursor
1330 LOCATE 1,1: PRINT BLANKS$: LOCATE 1,1: PRINT "ENTER CHARACTER"
1340 PUT (129,0),L,OR:  'move character to top line
1350 GOTO 750
1360 '
1370 '     F7 - SAVE FORMULA TO DISK
1380 LOCATE 1,1: PRINT BLANKS$: LOCATE 1,1: PRINT "SAVE FILENAME _"
1390 NC=15: GOSUB 2280:  'get filename
1400 OPEN "O",#1,R$:  'open file for output
1410 GET (0,24)-(639,167),Y:  'GET formula
1420 FOR I=0 TO 5759
1430 IF Y(I)<>0 THEN WRITE #1,I,Y(I):  'save only nonzero values
1440 NEXT I
1450 CLOSE #1
1460 GOTO 330:  'next command
1470 '
1480 '     F8 - READ FORMULA FROM DISK
1490 LOCATE 1,1: PRINT BLANKS$: LOCATE 1,1: PRINT "READ FILENAME _"
1500 NC=15: GOSUB 2280:  'get filename
1510 GET (0,24)-(639,167),Y:  'GET formula (no PUT without GET)
1520 OPEN "I",#1,R$:  'open file for input
1530 FOR I=1 TO 5760: Y(I)=0: NEXT I:  'zero Y(I) as only nonzero values read
1540 FOR I=1 TO 5760
1550 IF EOF(1) GOTO 1580
1560 INPUT #1,J,Y(J)
1570 NEXT I
1580 PUT (0,24),Y,PSET:  'display formula
1590 CLOSE #1
1600 GOTO 330:  'next command
1610 '
1620 '     F9 - MOVE FORMULA
1630 LOCATE 1,1: PRINT "MOVE ENTIRE TEXT, (ESC) TO END"
1640 A$=INKEY$: IF A$="" THEN GOTO 1640
1650 IF ASC(A$)=27 THEN GOTO 330:  'escape
1660 IF ASC(RIGHT$(A$,1))<>72 THEN GOTO 1710:  'move text up?
1670 GET (0,24)-(639,27),X
1680 GET (0,28)-(639,167),Y
1690 PUT (0,164),X,PSET
1700 PUT (0,24),Y,PSET
1710 IF ASC(RIGHT$(A$,1))<>80 THEN GOTO 1760:  'move text down?
1720 GET (0,164)-(639,167),X
1730 GET (0,24)-(639,163),Y
1740 PUT (0,24),X,PSET
```

```
1750 PUT (0,28),Y,PSET
1760 IF ASC(RIGHT$(A$,1))<>77 THEN GOTO 1810:  'move text right?
1770 GET (636,24)-(639,167),X
1780 GET (0,24)-(635,167),Y
1790 PUT (0,24),X,PSET
1800 PUT (4,24),Y,PSET
1810 IF ASC(RIGHT$(A$,1))<>75 THEN GOTO 1860:  'move text left?
1820 GET (0,24)-(3,167),X
1830 GET (4,24)-(639,167),Y
1840 PUT (636,24),X,PSET
1850 PUT (0,24),Y,PSET
1860 GOTO 1640
1870 '
1880 '     F10 - PRINT FORMULA
1890 LPRINT CHR$(13);
1900 FOR I%=3 TO 20:  'line to be printed-1
1910 LOCATE 1,1: PRINT BLANKS$: LOCATE 1,1: PRINT "Printing Line";I%-2
1920 A$=INKEY$: IF A$="" THEN GOTO 1930 ELSE IF ASC(A$)=27 THEN GOTO 330
1930 FOR J%=0 TO 639:  'columns
1940 TEST%=0
1950 FOR JJ%=7+I%*8 TO I%*8 STEP -1:  'construct graphics byte
1960 TEST%=2*TEST%+POINT(J%,JJ%)
1970 NEXT JJ%
1980 XX%(J%)=TEST%:  'save byte for printing
1990 NEXT J%
2000 LPRINT CHR$(&H1B);CHR$(&H54);"16";
2010 LPRINT CHR$(&H1B);CHR$(&H53);"0640";
2020 FOR N%=0 TO 639: POKE &HF801,XX%(N%): X=USR0(0): NEXT N%
2030 POKE &HF801,13: X=USR0(0)
2040 LPRINT
2050 LPRINT CHR$(&H1B);CHR$(&H54);"24";
2060 NEXT I%
2070 GOTO 330:  'next command
2080 '
2090 '     CTRL-C  (CLEAR CONSTRUCTION AREA)
2100 CLS: LOCATE 3,1: PRINT STRING$(80,"-"):
     LOCATE 22,1: PRINT STRING$(80,"-")
2110 GOTO 330:  'next command
2120 '
2130 '     CTRL-D  (DISPLAY USER-DEFINED CHARACTERS)
2140 FOR I=1 TO 8
2150 FOR J=1 TO 16
2160 A$=INKEY$: IF A$="" THEN GOTO 2170 ELSE IF ASC(A$)=27 THEN GOTO 330
2170 LOCATE J+4,(I-1)*10+1:  'locate print position
2180 B=(I-1)*16+J+127: PRINT B;" ";CHR$(B)  'character code
2190 'LOCATE 1,22: PRINT CHR$(B)
2200 'GET (168,0)-(175,7),L:  'move character
2210 'PUT (((I-1)*10+5)*8+1,(J+3)*8),L,PSET
2220 'LOCATE 1,1: PRINT BLANKS$
2230 NEXT J,I
2240 GOTO 330:  'next command
2250 '
2260 '     ***** SUBROUTINE *****
2270 '
2280 '     ENTER STRINGS
2290 R$="_"
2300 A$=INKEY$:IF A$="" OR LEN(A$)=2 THEN GOTO 2300
2310 IF ASC(A$)=27 THEN GOTO 330:  'escape
2320 IF ASC(A$)=13 THEN LOCATE 1,NC+LEN(R$)-1: PRINT BLANKS$: RETURN
2330 IF ASC(A$)<>8 THEN GOTO 2370:  'backspace
2340 IF LEN(R$)=1 THEN GOTO 2300
2350 R$=LEFT$(R$,LEN(R$)-2)+"_":  'perform backspace function
2360 LOCATE 1,NC: PRINT R$;" ": GOTO 2300
2370 LOCATE 1,NC: R$=LEFT$(R$,LEN(R$)-1)+A$+"_":  'add character to string
2380 PRINT R$: GOTO 2300
2390 '
2400 '     PRINTER PROGRAM
2410 DATA &HB2,0,&HB4,5,&HCD,&H21,&HCB
2420 '
2430 '     DEFINE USER CHARACTERS
2440 '
2450 '     LOWER CASE GREEK (128-151)
2460 DATA 0,0,&H72,&H8C,&H88,&H88,&H76,0,0,&H3C,&H42,&H7C,&H42,&H7C,&H80,0
2470 DATA &HC0,&H24,&H28,&H30,&H20,&H60,&H60,0
2480 DATA &H38,&H40,&H40,&H20,&H38,&H44,&H38,0,0,&H1C,&H20,&H40,&H7C,&H20,
     &H1C,0
2490 DATA &H2C,&H1C,&H20,&H40,&H38,4,&H18,0,0,&H58,&HA4,&H24,&H24,4,4,0
2500 DATA &H30,&H48,&H84,&HFC,&H84,&H48,&H30,0,0,0,&H40,&H40,&H40,&H48,&H30,0
2510 DATA 0,&H44,&H48,&H50,&H60,&H54,&H48,0,0,&HC0,&H20,&H20,&H30,&H48,&H86,0
2520 DATA 0,&H48,&H48,&H48,&H74,&H40,&H40,0
2530 DATA 0,&H84,&H44,&H48,&H50,&H60,&H40,0
2540 DATA &H60,&H78,&H80,&H78,&H80,&H78,4,&H18,0,0,&H38,&H44,&H44,&H44,&H38,0
2550 DATA 0,2,&H7C,&HA8,&H28,&H28,&H26,0
2560 DATA 0,&H38,&H44,&H44,&H78,&H60,&H80,0,0,0,&H7E,&H88,&H88,&H88,&H70,0
2570 DATA 0,2,&H7C,&H90,&H10,&H10,8,0,0,&HC6,&H24,&H24,&H24,&H24,&H18,0
2580 DATA &H10,&H38,&H54,&H54,&H54,&H38,&H10,0,0,2,&H64,&H28,&H10,&H28,&H46,0
2590 DATA 0,&H92,&H92,&H92,&H7C,&H10,&H10,0,0,0,&H44,&H82,&H92,&H92,&H6C,0
2600 '     UPPER CASE GREEK (152-159)
2610 DATA 0,6,&HE,&H16,&H26,&H46,&HFE,0
2620 DATA &HFE,&H44,&H20,&H10,&H20,&H44,&HFE,0
2630 DATA &HFE,&HC4,&HC0,&HC0,&HC0,&HC0,&HC0,0
2640 DATA &H38,&H44,&H82,&H82,&H82,&H44,&HC6,0
2650 DATA &H38,&H44,&HC6,&HFE,&HC6,&H44,&H38,0
2660 DATA &H30,&H38,&H38,&H4C,&H4C,&H84,&H84,0
2670 DATA &H38,&H10,&H7C,&H92,&H7C,&H10,&H38,0
2680 DATA &HC6,&H54,&H54,&H54,&H38,&H10,&H10,0
2690 '     SUPERSCRIPTS/SUBSCRIPTS (160-170)
2700 DATA &H30,&H10,&H10,&H10,&H38,0,0,0,&H38,&H44,&H18,&H20,&H7C,0,0,0
2710 DATA &H38,&H44,&H18,&H44,&H38,0,0,0,&H18,&H28,&H48,&H7C,8,0,0,0
2720 DATA &H7C,&H40,&H78,4,&H78,0,0,0,&H38,&H40,&H78,&H44,&H38,0,0,0
2730 DATA &H78,8,&H10,&H20,&H40,0,0,0,&H38,&H44,&H38,&H44,&H38,0,0,0
2740 DATA &H38,&H44,&H3C,4,&H38,0,0,0,&H38,&H44,&H44,&H44,&H38,0,0,0
2750 DATA 8,&H10,&H20,&H40,&H80,0,0,0
2760 '     ALGEBRA (171-184)
2770 DATA &H10,&H10,&HFE,&H10,&H10,&HFE,0,0,0,&HFE,&H10,&H10,&HFE,&H10,&H10,0
2780 DATA 0,&H60,&H92,&HC,&H60,&H92,&HC,0,0,&H60,&H92,&HC,0,&HFE,0,0
2790 DATA 0,&HC,&HFE,&H10,&HFE,&H60,0,0,0,&H44,&HAA,&H90,&HAA,&H44,0,0
2800 DATA &H20,&H10,8,&H10,&H20,0,&H38,0,8,&H10,&H20,&H10,8,0,&H38,0
2810 DATA 0,&H18,0,&H7E,0,&H18,0,0,&H24,&H24,&H24,&H24,&H24,&H24,&H24,&H24
2820 DATA 8,8,8,8,&H48,&HA8,&H18,8,&H38,&H54,&H92,&HFE,&H92,&H54,&H38,0
2830 DATA &H38,&H44,&HAA,&H92,&HAA,&H44,&H38,0,0,&HFE,0,&HFE,0,&HFE,0,0
2840 '     VECTOR NOTATION (185-187)
2850 DATA 0,0,0,0,8,6,&HFE,0,0,0,0,&H18,&H18,0,0,0,0,0,0,0,&H10,&H28,&H44,0
2860 '     CALCULUS (188-194)
2870 DATA 6,&HA,8,8,8,8,8,8,8,8,8,8,8,8,&H28,&H30
2880 DATA &HC0,&H30,&H18,&H7C,&H86,&H86,&H78,0
2890 DATA 0,0,0,0,0,&H30,&H30,0,0,0,0,0,0,&H66,&H66,0
2900 DATA 0,&H44,&HAA,&H92,&HAA,&H44,0,0,&HFE,&H86,&H8C,&H98,&HB0,&HE0,&HC0,0
```

```
2910 '     GEOMETRY (195-197)
2920 DATA &H10,&H10,&H10,&H10,&H10,&H10,&HFE,0,2,4,8,&H10,&H20,&H40,&HFE,0
2930 DATA 0,0,&H18,0,0,&H66,0,0
2940 '     LOGIC (198-201)
2950 DATA 0,0,&H82,&H82,&H82,&H44,&H38,0,0,0,&H38,&H44,&H82,&H82,&H82,0
2960 DATA 0,&H3E,&H40,&H80,&H80,&H40,&H3E,0,0,&HF8,4,2,2,4,&HF8,0
2970 '     BRACKETS/BRACES/HORIZONTAL LINE (202-212)
2980 DATA 8,8,8,8,8,8,8,8,0,0,0,&HE,8,8,8,8
2990 DATA 8,8,8,8,&HE,0,0,0,0,0,0,&H38,8,8,8,8
3000 DATA 8,8,8,8,&H38,0,0,0,8,8,&H10,&H20,&H10,8,8,8
3010 DATA 0,0,0,6,8,8,8,8,8,8,8,6,0,0,0
3020 DATA 8,8,4,2,4,8,8,8,0,0,0,&H30,8,8,8,8
3030 DATA 8,8,8,8,&H30,0,0,0,0,&HFF,0,0,0,0,0,0
3040 '     LARGE CHARACTERS (214-221)
3050 DATA &HFF,&HFF,&H60,&H30,&H18,&HC,6,3,&HFE,&HFE,&HE,4,0,0,0,0
3060 DATA 7,&HC,&H18,&H30,&H60,&HFF,&HFF,0,0,0,0,4,&HE,&HFE,&HFE,0
3070 DATA &HFF,&HFF,&H18,&H18,&H18,&H18,&H18,&H18
3080 DATA &HFE,&HFE,&H30,&H30,&H30,&H30,&H30,&H30
3090 DATA &H18,&H18,&H18,&H18,&H18,&H18,&H18,0
3100 DATA &H30,&H30,&H30,&H30,&H30,&H30,&H30,0
```

*



SymSoft's HotShot Shoots Screens

When even a thousand words can't match the value in one picture of a computer's screen display, SymSoft's HotShot package comes into its own. It's such a remarkable performer that I'm hard pressed to think of it as a medley of programs built around the mundane screen dump. I think of HotShot instead as a specialized desktop publishing toolkit. Each module in this package contributes a set of functions that take special advantage of the PostScript laser printer's remarkable graphics potential. Together they allow not only standalone screen shots but also fullblown illustrations, the kind that can be imported into page makeup programs for incorporation into the text of brochures, manuals, books, and—as you can see from the smiling lady above—even magazines.

The screen shots may be either simple text displays or graphics displays as complex as a digitized "Mona Lisa," depending upon which HotShot modules you use. The HotShot program itself captures, edits, and prints only text; the Grab program only captures graphics, with no provision for their immediate printing or editing. Run the two programs in sequence (Grab before HotShot), and you won't even need to know the display mode being captured. That combination handles anything except displays from within Microsoft Windows. Wingrab, a third module, is supposed to capture Windows displays but I can't get it to work. SymSoft tells me that no one else has reported a problem with Wingrab and I have reason to believe them. I've seen Windows screen shots published with credit to HotShot.

Cvrt, a conversion utility added to Version 1.1 of the HotShot package, converts the text or graphics capture files to either Altsys's EPSF ("Encapsulated PostScript Format") or Z-Soft's PCX format. Those two formats are becoming standards for graphics exchange. EPSF files can be transferred to a Macintosh or printed on any PostScript printer, and PCX files are becoming a medium for moving graphics among many different MS-DOS programs. In fact Cvrt can produce two kinds of PCX files from a HotShot screen capture: one retains the original color values, for importa-

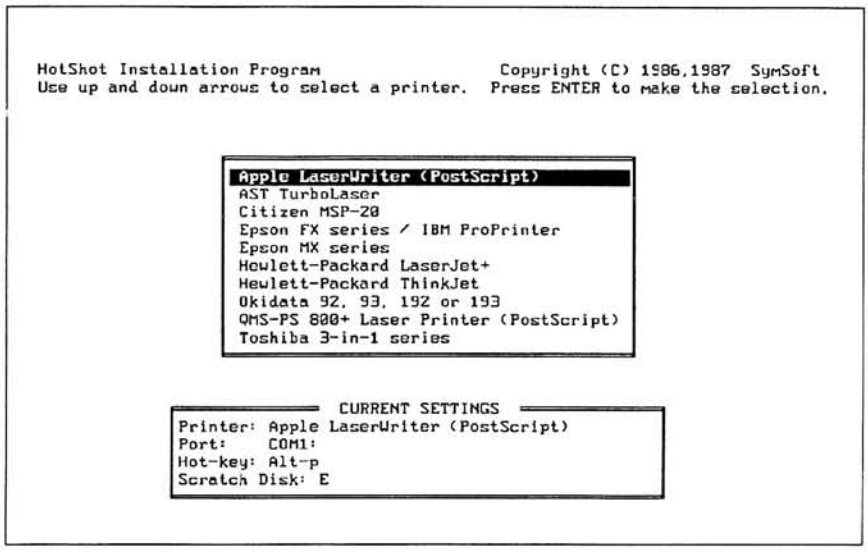


Figure 1. Installing HotShot for a supported printer.

```
This is the HotShot Grab program.
It is used to capture graphics screen images.
It should be installed before HotShot.

To install it, enter:

 grab X

X indicates which of the following video
 adapters you are using:

  1 - AT&T 6300 (monochrome)
  2 - Hercules Monochrome graphics
  3 - IBM Color Graphics adapter
  4 - IBM Enhanced Graphics adapter
  5 - Video-7 Vega Deluxe adapter
```

Figure 2. Grab's error message, and the video cards it supports.

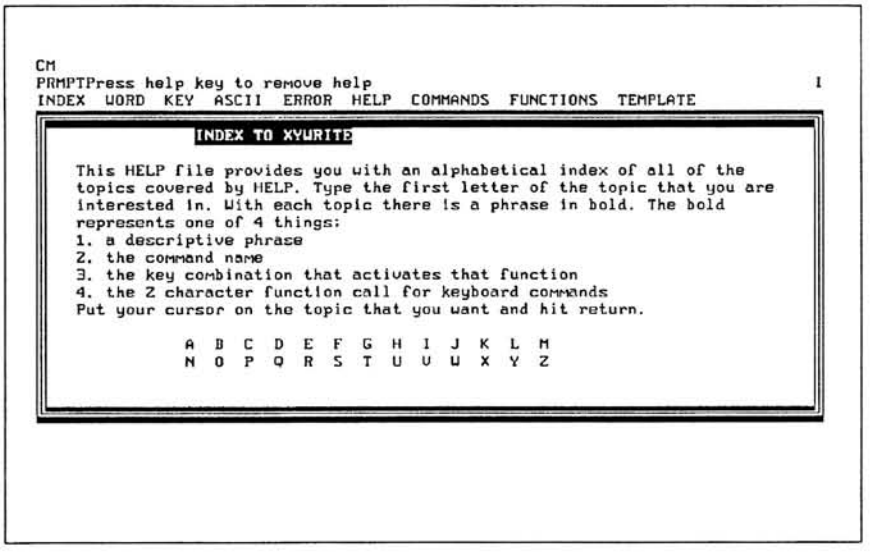


Figure 3. HotShot passes the acid test: it shoots XyWrite III, don't it?.

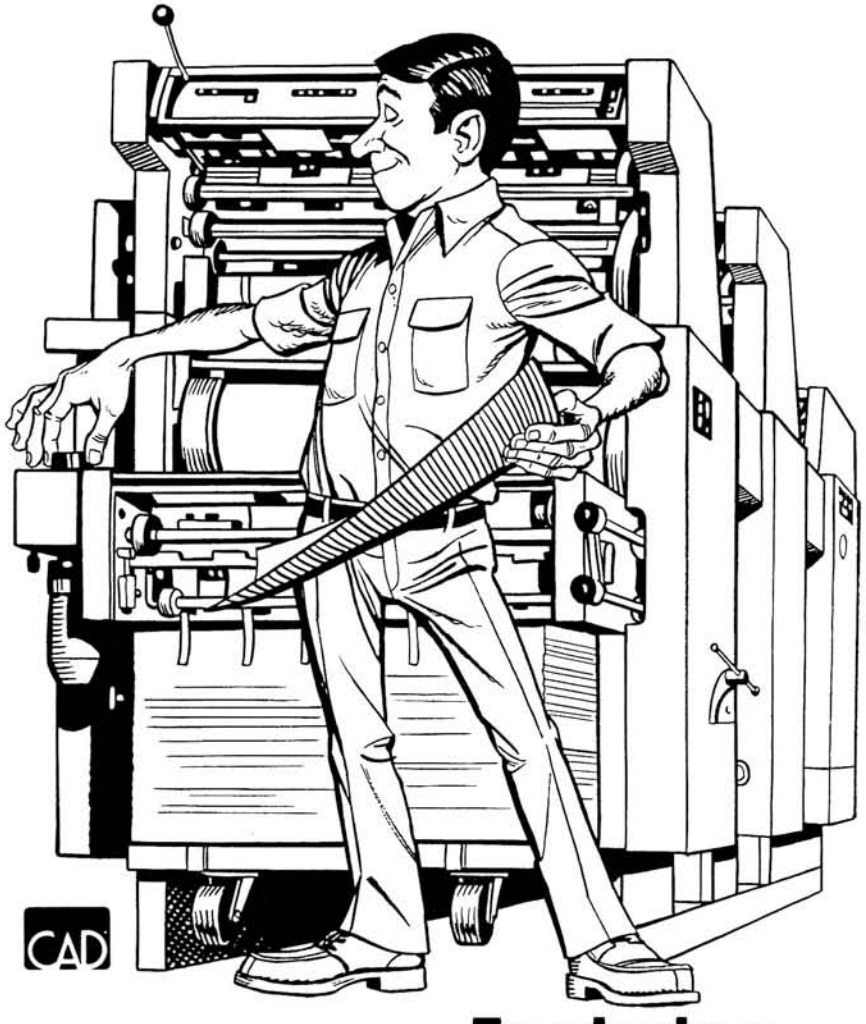
tion into one of Z-Soft's Paintbrush programs; the other grayscales the colors for better monochrome reproduction in the PageMaker and Ventura Publisher page makeup programs. The image of "Mona Lisa" started digital life as a color image and had to be grayscaled before PageMaker could print it. I intentionally used a killer example: the HotShot grayscaled image is absolutely representative of the original color image. I didn't think it could be done. I am impressed.

Wingrab, another utility in the HotShot package, is intended to capture screens from Microsoft Windows. MS-DOS TSRs ("Terminate and Stay Resident" programs) like Grab and HotShot simply won't work with Windows: it takes over the computer. Unfortunately, as I said, I can't get Wingrab to work either. I suspect my problem is not Wingrab but some obscure trouble with my hardware. I'd love for Wingrab to work, because I desperately need the capability to capture screens from Microsoft Windows' application programs. But even though Wingrab won't work for me, yet, I still consider the HotShot package extraordinarily valuable.

It's also easy to install. Wingrab is configured like other Microsoft Windows applications, through Windows' WIN.INI file, but the HotShot and Cvrt modules require separate MS-DOS installation. Hinstal is a single installation program that produces an HS.CFG configuration drawn on by both the HotShot and Cvrt modules. First you copy the distribution disk to the computer's hard disk or, if you don't mind willy-nilly creation of an \HS subdirectory for the HotShot files, after using the package's Setup program to do it for you. Then you run Hsinstal.

Hsinstal needs to know three things about the system: the printer, the port to which the printer is connected, and a "scratch disk" to which the HotShot programs can write temporary files. Figure 1 shows the currently-supported printers, which can be connected to any standard port. Since I am committed to getting the best possible printed copy from my system, I'm not much interested in any but the PostScript printers.

Hot off the Press.



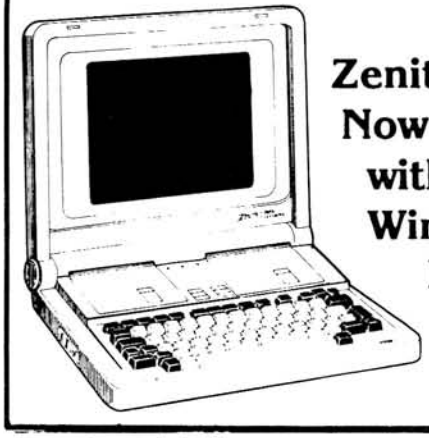
First Capitol Computer's Summer/Fall 1987 Catalog

It's here at last and better than ever! Our new Summer/Fall 1987 Catalog has it all - from Amstrad to Zenith and Ashton-Tate to Xerox. There are 52 pages packed with computer products for any user.

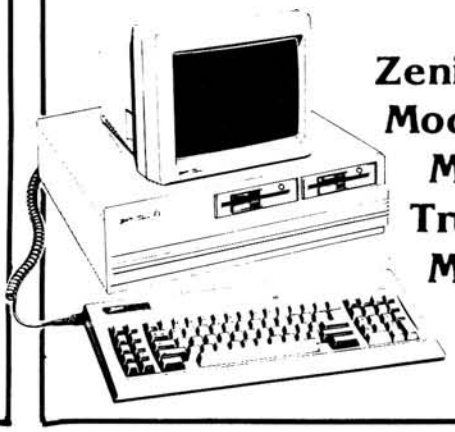
And if a special system is what you need, we've got it. You'll find special prices on our customized LAN packages and CAD systems. Don't forget to check out the Zenith SuperSystems that have given us our reputation. We can also put together a Custom Quote on your own system configuration.

Call or write to get your free catalog. (Note our new address and phone number.) Grab one while it's hot off the press!

Featuring . . .



Zenith's Z-183 Now available with 10 mb Winchester Drive



Zenith's Z-386 Model-40 and Model-80 True 32-bit Machines

First Capitol Computer

16 Algana Dr.
St. Peters, MO 63376-3930

1-800-TO-BUY-IT (800-862-8948) orders and quotes
1-314-447-8697 technical support and order status

Before you buy . . . . . .

Call Us First. And Last.

Although the manual says the default hotkey to trigger the HotShot and Grab programs is ALT-H, it's now ALT-P. That hotkey and any of the other configuration specs are easily changed at any time with Hsinstal. When Hsinstal is finished, it automatically writes the HS.CFG configuration file. There's nothing to installation.

The HotShot package is also easy to use. After the package is installed, you run Grab alone to capture graphics screens, HotShot alone to capture text screens, or Grab and then HotShot to be ready for either. Both are TSRs: Grab requires about 25KB of RAM and HotShot takes about 60KB. Although SymSoft recommends a system minimum of 256KB RAM, what the recommendation really means is that this package will run on any IBM compatible computer: I can't imagine one with less RAM.

There are only two nuisances I've found in the entire package. Unfortunately, you have to run HotShot and Cvrt from the directory in which they are installed, because otherwise they can't find the HS.CFG file—even with the aid of a data file pathfinder like Dpath. Unfortunately, too, Grab doesn't consult the HS.CFG file for information about the video card in the computer. You must provide Grab a command-line switch to supply that information. Figure 2 shows what happens when you forget: it's a reminder of the supported video cards. Those are nuisances.

Once either or both of the capture programs are resident, though, everything is easy. All you do is press the hotkey to capture the current screen. I haven't yet found an MS-DOS program, except HotShot itself, that defeats HotShot or Grab. Of course I tried, but HotShot even snapped screens from XyWrite III, which is the heavyweight in any TSR demolition derby because XyWrite's standard keyboard handler elbows everything else out of the way. Look at Figure 3.

Grab and the HotShot program behave differently from one another once a screen is captured. Grab simply pops up a prompt for the name of a capture file to use. Since Grab produces PCX files, any extensive manipulation of them must be with one of Z-Soft's Paintbrush packages. (One of them belongs in a desktop publishing system anyway.) HotShot, however, superimposes a display of the capture image with a menu of four choices: print, modify, files, and exit. Exiting at this point results in loss of the captured screen. The Print option invokes a submenu with choices that permit minor alterations of the screen image before printing: you can do things such as change the size, negative the image, and add a caption and borders. Choose the Modify option for the ability to make more significant changes—including editing the screen text, adding shades and symbols, and shifting the image—that can be saved permanently. Saving is done with the Files option, which allows either saving the current screen to a disk file or loading a previously-saved image from an extant disk file. Screens saved by HotShot are in its own compressed format and are stored in files with the extension SCR.

Cvrt works on both SCR and PCX. It will convert SCR files to EPSF or PCX files, and PCX files to EPSF or grayscaled PCX files. It also will print PCX files directly. PCX files can be imported into Z-Soft's PC Paintbrush, PC Paintbrush Plus, or Publisher's Paintbrush for massaging. Grayscaled PCX files cannot be imported into the Z-Soft programs and therefore should be the last transformation before a captured screen is imported into a page makeup program. EPSF files cannot be edited but can be either sent directly to any PostScript printer or imported into page makeup programs. None of the preceding formats can be converted to SCR format, nor can I conceive of any reason to do so. No matter what transformation Cvrt performs, it leaves the original SCR files untouched and available for further conversions. You therefore should not discard SCR files lightly.

Documentation for the HotShot package is adequate. There's a nicely-produced manual for the HotShot program itself, and a seven-page supplement covering all the other programs. My impression is that Version 1.1 ran ahead of its documentation, but I'm not particularly shaken by that impression. The HotShot manual has a tutorial keyed to sample SCR files on the distribution disk and the tutorial is all right. But I went back to look at it instead of forward to learn from it. The programs are designed so well that anyone with any experience at all of screen dumps ought to need only a brief explanation of what each module does. In fact my hunch is that what I've said here is most of what you need to use most of the package.

SymSoft's support is excellent. Although you'll have to pay for the telephone call should you need support, there's no problem getting through—in either sense of that term. I didn't get busy signals from the phone or the tech support staff. What I did get was fast, well-informed help from people who knew the product well and wanted it to work well. SymSoft has expressed special interest in the Heath and Zenith market. They know who you are and desire your custom.

At $199, Version 1.1 of HotShot seems pricey. Whether it is or not will depend on your standards as well as your needs. HotShot is first class. It has no competition because there simply are no similar packages anywhere near its league, certainly none at all that support PostScript printers. I would be seriously inconvenienced if I had to do without HotShot now. It did the screen shots in The LaserWriter Connection manual and it's what I've been using these past couple of months to illustrate my "Mainstream Computing" column in *REMark* and my "C Notes" column in *Sextant*. HotShot has replaced my $1,500 computer camera in all instances except those that absolutely demand the color reproduction of screen shots.

Recommended.

For more information:
HotShot. Version 1.1. $199.
SymSoft
444 First Street
Los Altos, CA 94022
415/941-1552

*

Z-100 PC Expansion with ZPAL™

Z-151/Z-161 Users
Use 256K bit memory chips on the original memory board. Extend the memory to 640K or 704K bytes with ZPAL-2 Decoder **$29.00**

Z-158 Users
Extend your system to 768K bytes with 3 banks of 256K bit memory chips and ZPAL-158 decoder.
ZPAL-158 Decoder **$36.00**

Z-138/Z-148 Users
Extend your system to 768K bytes with 3 banks of 256K bit memory chips and ZPAL-148 decoder.
ZPAL-148 Decoder **$36.00**

Z-100 PC Expansion with COMPAL™

Z-151/Z-161 Users
Instead of disabling COM2 when using an internal modem or multifunction board, MAP it to COM3. Package includes driver software.
COMPAL-3 decoder **$39.00**

ZSTEMpc-VT220 — latest release supports COMPAL-3.

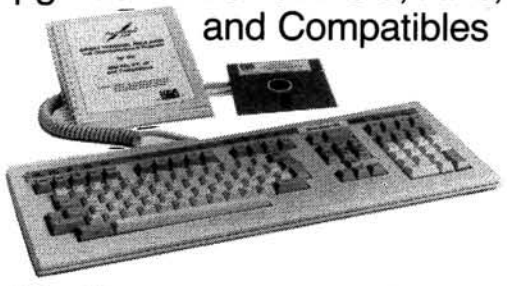
KEA Systems Ltd.

2150 West Broadway, Suite 412,
Vancouver, B.C. CANADA V6K 4L9
Technical Support (604) 732-7411

Order desk (800) 663-8702 Toll Free
Telex 04-352848 VCR
VISA and Mastercard Accepted
30 day money back guarantee.
Quantity and dealer discounts available.

PowerStation220™

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



"You'll never know you are not on a true VT220 — unless you take advantage of our many extended features."

PowerStation™ 220 **$289**
VT220 style keyboard and ZSTEMpc™-VT220 emulation Software.

ZSTEMpc™-VT220 Emulator Emulation Software only. **$150**
All the features of ZSTEMpc-VT100 plus 8-bit mode, downloadable fonts, user defined keys, full national/multi-national modes. Extended macros/script language. True 132 columns on enhanced EGAs and on most standard EGA's using the EGAmate option. 43 line support on EGA's. Enhanced keyboard support.

EGAmate™ daughterboard option for 132 columns on most standard EGA adaptors. **$39**

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

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

KEA Systems Ltd.

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

S & K Technology, Inc.

Quality Software for Heath/Zenith Microcomputers

Announcing PC WatchWord® Version 3

New features:

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

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

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

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

Texas Residents add state sales tax.

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

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 885-2091