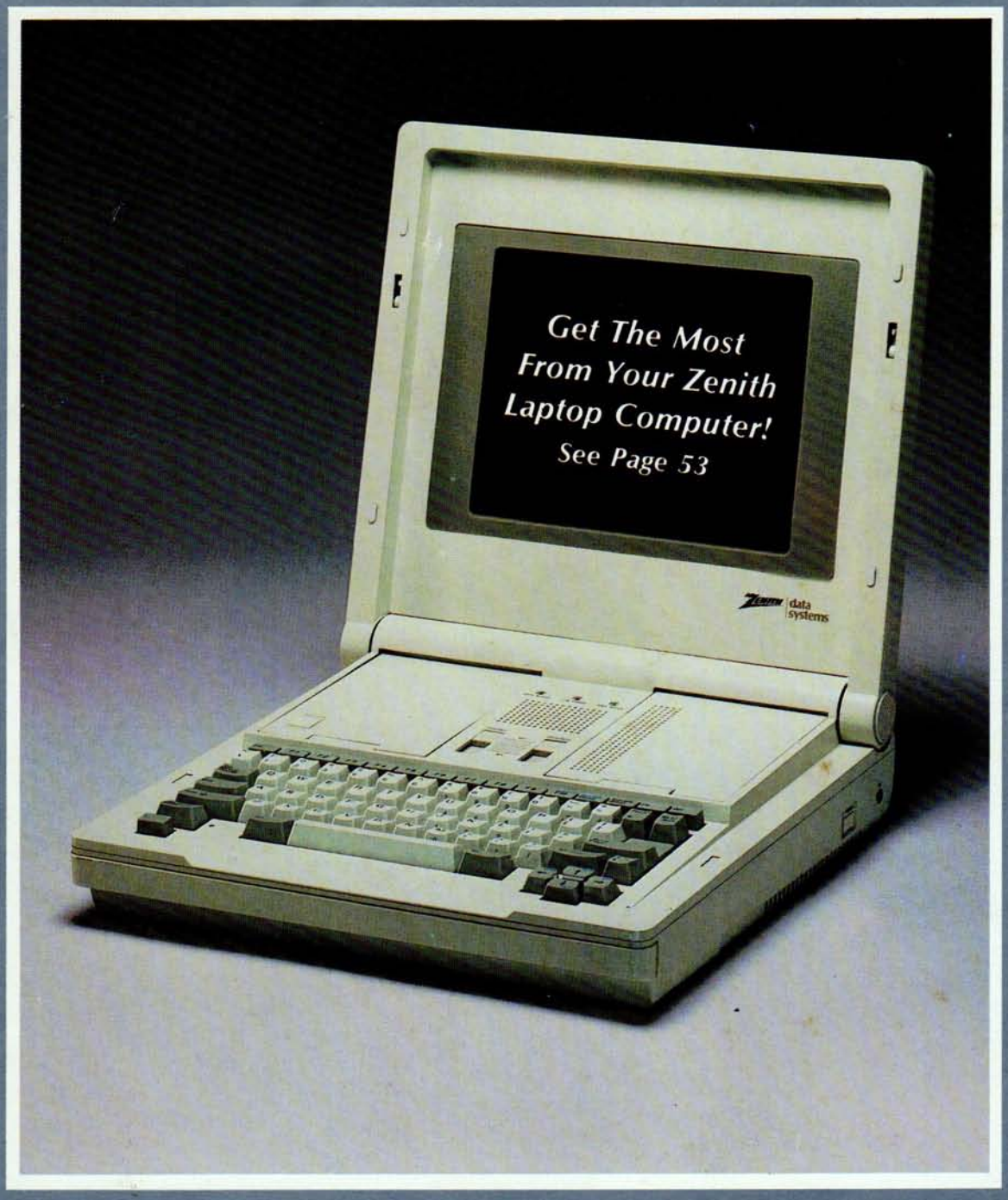
REMark®

Volume 9, Issue 6 • June 1988
P/N 885-2101 Issue 101

POWERING-UP
Adney's Beginners' Series
See Page 9

$2.50



A Publication of Heath/Zenith


"Why Haven't You Called?"
"I've stayed up 24 hours a day waiting to hear from you. I know that for most of you, this may be your first time, but I promise, I'm very gentle! If you're still unsure, let me tell you what I have to offer. First, there's my message base. Here's where my users exchange ideas, receive assistance, and sell things. Next, is my database. Most of my users will tell you, it's second to none! With over 30 megabytes of hand-picked software it caters to all Heath/Zenith computers, but mostly PC compatibles. Finally, I have something new for you, my 'Bargain Centre'. Here you can buy surplus software and hardware, at unheard of prices. Interested? I hope so! Set your modem to either 300, 1200, or 2400 baud, and call me right now at (616) 982-3956, and register today. If you're still a bit shy, you can still call my human at (616) 982-3837 and register with him. Although he talks at 150 baud, he's gentle, too!"
McC

The other cats get to sing along!

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

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

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

HEPCAT gets along with everyone . . .

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

HEPCAT works harder . . .

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

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

HEPCAT eats less . . .

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

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

Heath/Zenith Users' Group

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

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

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

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

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

HUG Parts Ordering ..... (616) 982-3838

Contributing Editor .... William M. Adney

Contributing Editor ..... Joseph Katz

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

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

*U.S. Funds

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

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

REMark®

Official magazine for users of HEATH ZENITH computer equipment.

Volume 9, Issue 6 • June 1988

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: *Pictured is the Z-183 Laptop Computer from Zenith. On Page 53 of this issue of REMark, you will find a series of articles on the Zenith Laptop Computers by Joseph Katz.*

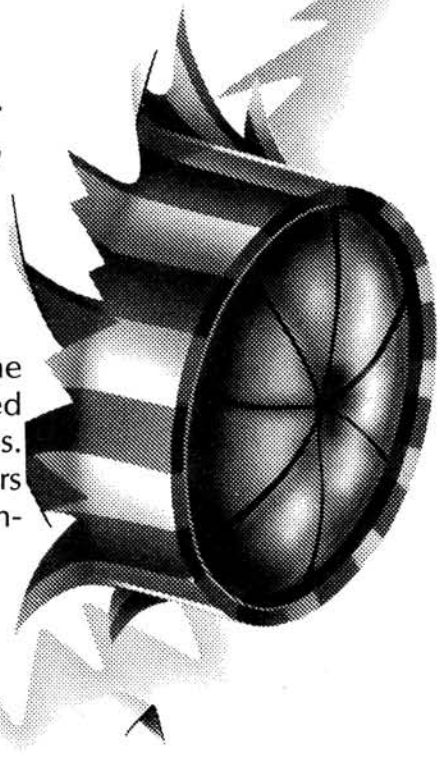
SEATTLE HUGCON

ONE GIANT DAY ONLY!!

July 9th, 1988 (10-4 PM PST)
Seattle Center (Lopez Rooms)
(Northwest Rooms)

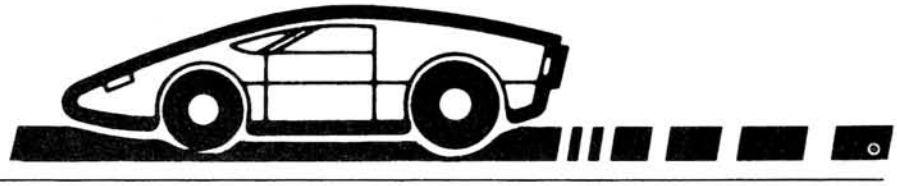
- ***Free Admission!***
- ***$100,000 in H/Z computer equipment at blow-away prices from the Heath Store***
- ***Vendor Booths***
- ***Speakers***

The Seattle Center is within 20 minutes of the SEA-TAC International Airport, and is situated in the heart of all major Hotel/Motel chains. Vendor booths are free, and interested vendors should contact Les Hall at (206) 234-2851 immediately! Don't miss this one!!



Please let us know 3-4 weeks in advance, so you won't miss a single issue of REMark!

MOVING?



Keeping Your System Healthy

Patrick Swayne
HUG Software Engineer

It seems that in just about every field of endeavor, there are "bad guys" who are constantly trying to mess things up. The realm of computers is no different. Of the various kinds of computer bad guys, the worst is the one who distributes software that harms another person's data or system. With the spread of "shareware" software and bulletin board systems, harmful software in the guise of functional programs is becoming a problem that can affect many computer users.

Recently, a new kind of harmful software has surfaced called the "virus". A virus is originally transferred to your system via a seemingly functional shareware or public domain program. When you run the program, the virus somehow attaches itself to your operating system. Then, it maintains a counter of some kind until a period of time has passed, which can be months or even years. When the count reaches a pre-selected value, the virus does its "dirty work". It may erase files, or it may even attempt to harm your hardware. I have heard of a virus that changed the video mode of the computer's EGA card to one incompatible with the monitor, causing the monitor to burn out.

The virus program also propigates itself to other disks. If it is attached to a component of your operating system, it will be transferred to any disk you format with the /S option, or any disk that you use the SYS command on. It could also be spread by copying the infected program to other disks.

A Simple Virus Watchdog

There are a number of simple procedures you can perform that will protect your system from computer viruses. One procedure is to make backup copies of the components of your operating system, and then use the FC program (File Compare) supplied with MS-DOS to compare the copies with the originals from time to time. But before you make the copies, you should verify that your operating system components are ok to start with.

First, you need to determine what the system component files are called. The three system components are the I/O processor, the MS-DOS processor, and the command processor. The command processor is always called COMMAND.COM. The I/O processor is called IO.SYS on MS-DOS versions 3.10 and below, and IBMIO.COM on versions 3.20 and above. The MS-DOS processor is called MSDOS.SYS on versions 3.10 and below, and IBMDOS.COM on versions 3.20 and above. (This discussion assumes that you have Zenith MS-DOS. The names may be different with MS-DOS from other vendors.)

To check your operating system components, place your original MS-DOS distribution disk in a drive, and enter commands as in this example:

```
fc /b ibmio.com a:ibmio.com
fc /b ibmdos.com a:ibmdos.com
fc /b command.com a:command.com
```

The above example assumes that the distribution disk is in drive A:, and the default disk is another drive (C:, for example), and that the operating system is version 3.20 or above. Notice that the commands were entered in lower case letters. The FC program, for some unknown reason, does not accept switch characters in upper case.

The FC program will display the message "no differences encountered" if the two files being compared match. It will display the addresses and values of any mismatching bytes otherwise. If you have changed anything from the default settings with the CONFIGUR program, there will be a few bytes that do not match between your I/O processor (IBMIO .COM or IO.SYS) and the one on the distribution disk. If you are not sure that the mis-match is caused by configuration changes, boot up on the distribution disk

and use the SYS command to put a fresh copy of the system component files on your operating system disk. If there are any mis-matched bytes at all in your MS-DOS component or in COMMAND .COM, and you have not made any patches to them yourself, you should suspect foul play and use the SYS command as described above.

After you have verified your system component files, you can make back up copies of them. Since the I/O processor and the MS-DOS processor are hidden files, you cannot use the COPY command to make back-up copies, but you can use DEBUG to make them. To do this, follow this example:

```
C>DEBUG
-NIBMIO.COM
-L
-NDOG.X
-W
-NIBMDOS.COM
-L
-NCAT.X
-W
-Q
```

The above example is for versions 3.20 and above. If you have version 3.10 or below, replace IBMIO.COM and IBMDOS .COM with IO.SYS and MSDOS.SYS. In this example, we have copied IBMIO .COM to a file called DOG.X, and we have copied IBMDOS.COM to CAT.X. For those who are not familiar with DEBUG, the N before CAT.X and the other file names is the envocation of DEBUG's Name command. You can read about the DEBUG commands in your MS-DOS manual.

After you have copied your I/O processor and MS-DOS processor with DEBUG, use the COPY command to copy COMMAND.COM, as in:

```
COPY COMMAND.COM MOUSE.X
```

When you have copied the component files, you can make a batch file that will run FC and check each of them. To make the batch file, enter these commands:

```
copy con vcheck.bat
fc /b ibmio.com dog.x
fc /b ibmdos.com cat.x
fc /b command.com mouse.x
^Z
```

Press Return after each line. The ^Z means to type Control-Z (hold down Ctrl and type Z). If the system prompt does not re-appear when you press Return after typing Control-Z, press Return again. This example assumes MS-DOS version 3.20 or above, and that the files were named as in our copy examples. You should make up your own unique names for the files, and you may want to copy them to another directory or partition. Just be sure to put the proper directory name or drive designation in your batch file. After you have made the batch file, you can check your system components for unauthorized modifications any time you want to by entering:

```
VCHECK
```

You should perform this check each time you copy a new piece of public domain or shareware software to your system, after you have run the software for the first time.

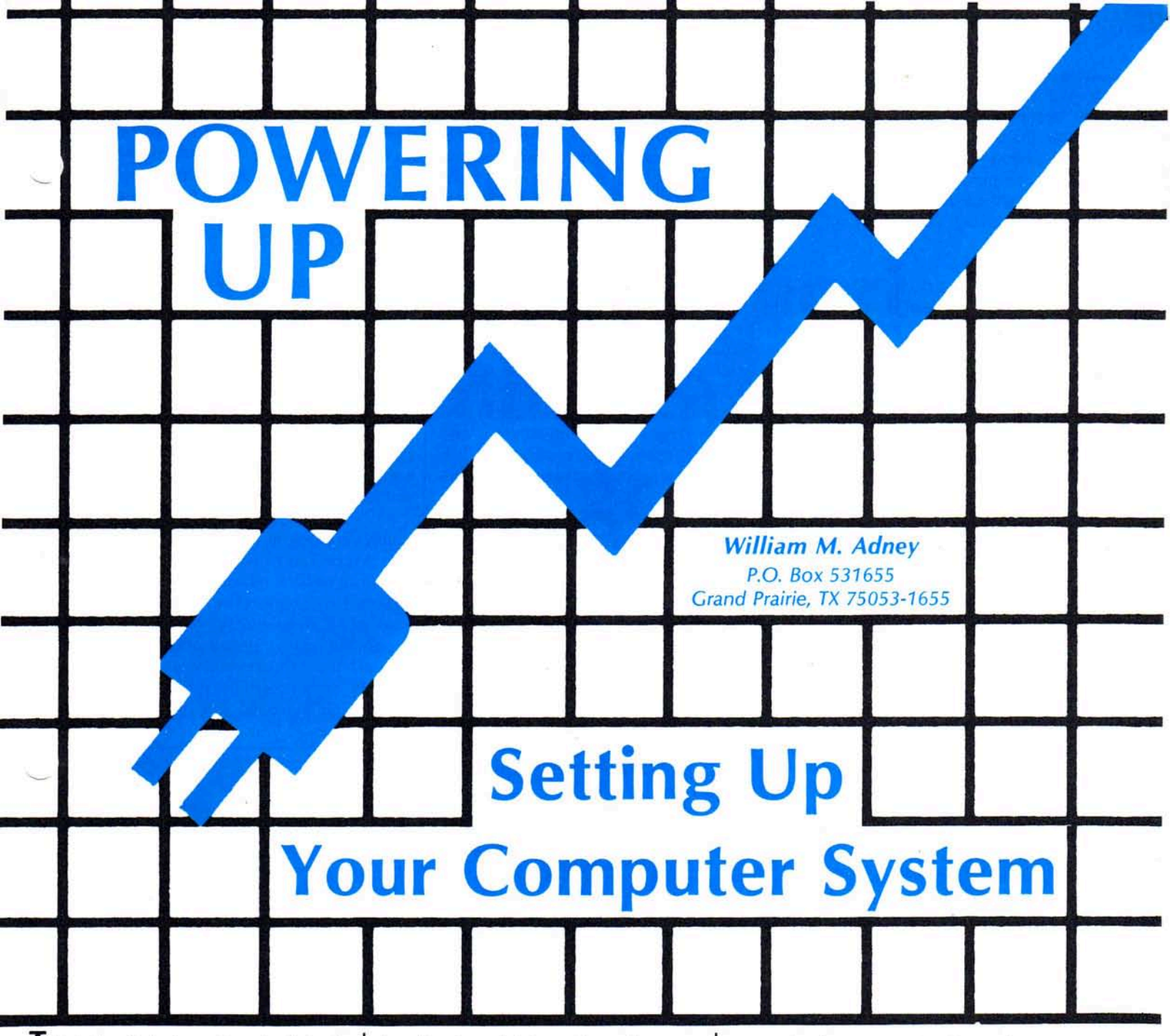
Other Health Checks

There are other checks you can perform to guard your system disks against unauthorized modifications. You should periodically check your CONFIG.SYS and AUTOEXEC.BAT files, if you have any, for changes. You should also check your directories, looking for any files that you cannot account for. You should use a directory program that can see hidden files and hidden directory names to make this check. My own D.COM, from HUG disk 885-6011-37, is a good directory program to use for checking your files. Just use the /S switch, and any hidden files or directories will show up.

If you find a file that you cannot account for, it will probably be a legitimate data file created by a new program. You should examine the instructions for the new program to see if the files created by the program are listed. If they are not, you can always back up the file to another disk, and then delete it. Later on, if a program complains with a message like "Cannot find file.dat", you can restore the file from the back-up disk.

*

POWERING UP



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

Setting Up Your Computer System

This is the first of a series of articles for beginning computer users on various topics that you need to know for effective, efficient, and successful use of your system. I have chosen the basic column title as "Powering Up" to answer the question: "What do I do after I turn on the computer?"

The primary objective of all articles in this column is to discuss, in non-technical terms, various topics that are critical to the successful use and operation of your microcomputer system. In particular, each article will provide an introduction to a wide variety of computer-related topics that are particularly useful to first-time users of Heath and Zenith microcomputers. These articles will focus on the "need-to-know" information with a specific emphasis on the practical aspects of microcomputer use. In addition, appropriate columns will include practical examples of specific DOS commands, with explanations, so that even a novice or beginning user can immediately take advantage of advanced features without a lot of technical background or knowledge. Even advanced microcomputer users can learn about some practical tricks that can help improve their productivity significantly.

Another important objective is to include suggestions, "tricks-of-the-trade", hints and kinks, and ideas for more efficient and effective use of your microcomputing system and its peripherals. I hope that this will give you a starting point so that you can customize your system to meet your particular needs. Each article will discuss a specific subject for your computer system and provide the basic information you MUST know about the subject. But before we get into too much discussion about setting up your system, let's review one subject that seems to be quite confusing to many computer owners.

What Is PC Compatible?

In general, these articles will discuss the various aspects of PC compatible computers that are available from Heath and Zenith. Today, the only models available ARE compatibles, but in order to avoid confusion, PC compatible includes the Z-138, Z-148, Z-150 series (i.e., 151, 157,

158, and 159), the Z-171 portable, the Z-180 laptop series (i.e., 181, 182, 183, and 184), the Z-200 series (i.e., the 241 and 248), and the Z-386. Given the focus of today's computer interest, it is nearly certain that all future models sold by Heath and Zenith will also be PC compatible.

Since some of these articles will address hardware features on the PC compatibles, it is important to note that the discontinued Z-100 computer (i.e., 11x and 12x models) hardware is NOT included in these articles. For example, the Z-100 does not have the wide variety of video display adapters or memory cards that are available for the PC compatibles. If you have an interest in topics of that nature, most of the Z-100's hardware features have been discussed in older issues of REMark. For the most part, however, these articles will include features available on the Z-100 computer under the current MS-DOS 3.10 version. And this article applies to all computer systems.

Where To Begin?

At this point, I assume that you have a computer, CRT monitor, and a place to work. Perhaps you have decided to set up your system on a card table in a spare bedroom or have commandeered part of another room to work. After working for a while, you notice that your computer system is quite literally giving you a "pain in the neck". Since I have been known to spend as much as 18 hours a day at my computer, it's important to me to have a comfortable and well organized work station. As a result, I have spent some time learning about ergonomics.

What Is Ergonomics?

ERGONOMICS, sometimes called biotechnology or human factors engineering, is concerned with the application of biological and engineering data to problems relating to the interface of humans and machines — in this case, computer systems. It includes consideration of the anatomical, physiological, and psychological aspects of that interface. Ergonomics includes just about all factors that you are likely to find in a working place for a microcomputer system: lighting, furniture, noise, power and cabling, environment, and storage. Let's begin by looking at the basic setup for a good work station by looking at computer furniture.

Computer Furniture

It seems that you can buy computer furniture just about anywhere, but there is more to it than you might expect. I have listed several sources at the end of this article, but you will probably be able to find much better prices in local home improvement centers and office furniture stores.

The furniture must be considered as a function of the individual who will use it. Ergonomically designed furniture will be comfortable, and your comfort will improve your productivity. You will be able to concentrate more effectively, and therefore, improve your productivity. It is important to consider whether the furniture will be primarily used by a man or woman, or a younger or older person. This is not a matter of age or sex discrimination — it is simply a recognition of the physiological differences in these groups. Men, for example, are on the average physically bigger than women.

Perhaps the biggest need is for a "computer table" that provides a large stable work area, such as shown in Figure 1. Since you will have your computer keyboard on this table, it is important to get one that is the proper height. Tables designed for typewriters and the so-called secretarial desks are generally designed for women and are 24 inches from the floor for proper keyboard height. Many men find this uncomfortable and prefer a minimum of 25 inches.

If you have the space in your home, I recommend that you get the largest computer table you can find, since there never seems to be enough room for everything. Don't forget that you will need a flat work space to look at printouts, store floppy disks, and keep the software manuals handy for reference. As you accumulate various kinds of computer goodies, you will find that all of the space you thought you had seems to evaporate.

When you look at a computer table, be sure that you get one that is deep enough (i.e., front to back measurement) for your system. I recommend an absolute minimum of 24" deep and 30" or more is best depending on your system and keyboard. You will have the keyboard and a cable from the computer to the printer that also takes some space, so it is not sufficient just to measure your computer. In most

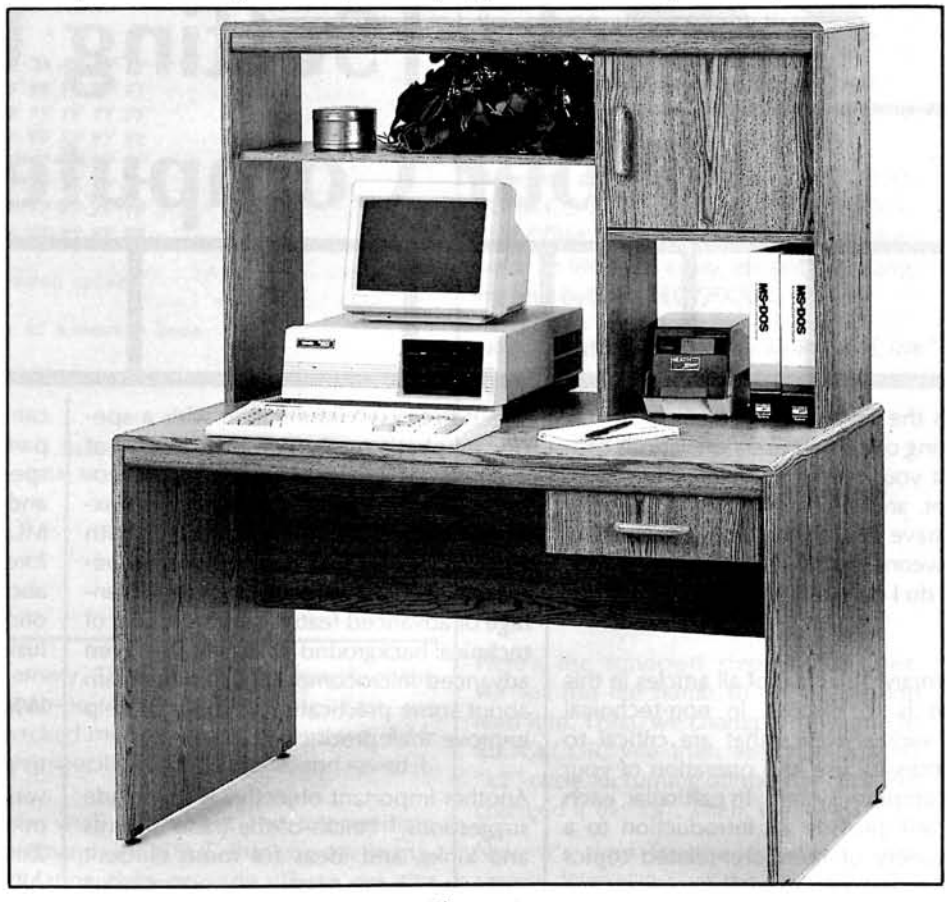


Figure 1
A Typical Computer Desk

cases, you will probably want to have the CRT sitting on top of the computer, so be sure to check the height of any shelves in the furniture. Is the "opening" wide enough to allow your computer to fit? Check all of the measurements of your system BEFORE you buy any furniture. You can find computer tables that range in price from $100 or so up to over $1,000.

Now that you have found a prospective work table (don't buy it yet), the second component of your work station is a comfortable chair. From a comfort point of view, this is perhaps the most critical item, and a dining room or kitchen chair simply was not designed for this kind of use. You can find any number of inexpensive chairs that are specially designed for this purpose, such as shown in Figure 2.

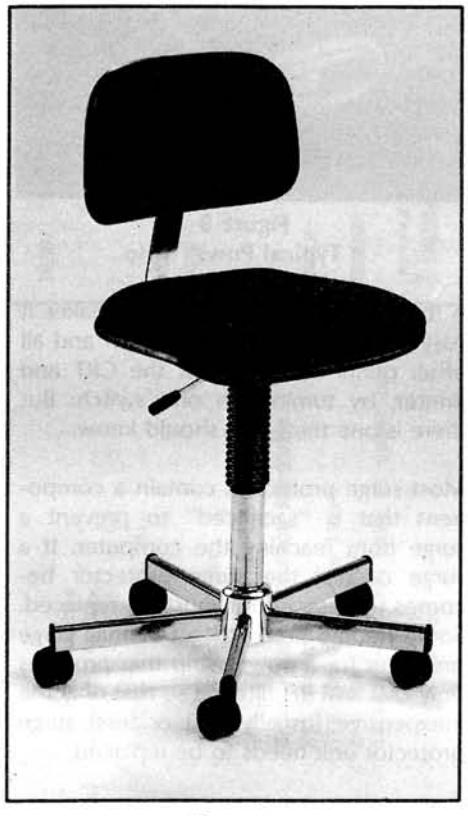


Figure 2
A Typical Workstation Chair

Some of the newer and more expensive chairs feature synchronized articulation so that the seat and back moves as the body moves. Most chairs provide a back rest control that locks the chair into several different positions from a totally upright position to near reclining in some models. Virtually all of today's chairs, such as shown in Figure 2, provide an easy means of controlling the height of the seat, usually by means of a gas cylinder under pressure. When a lever is moved, the gas pressure raises the seat height, but an individual's weight makes it easy to adjust the chair to a comfortable height while still seated. If more than one person will be using the chair, it is best to get one that is most easily adjustable. Also, I have found that I prefer a chair with arms, and you may also want to consider a chair model that has arms as an option.

From an ergonomic perspective, it is usually agreed that 16-17 inches above the floor is the "average" seat height through which a seat should be adjustable. Many of the newer gas cylinder chairs provide a much wider range of seat adjustment ranging from 13 to 20 inches. Some also provide for a dynamic adjustment in the position of the seat back so that it can be quickly adjusted for back support.

In order to prevent inhibiting the blood circulation in the thighs, many people find that a chair with a "waterfall front" on the seat is most comfortable. A 6 degree pitch forward from the horizontal is average. Some chairs can be adjusted to accommodate a 5 degree pitch for shorter people to an 8 degree pitch for the tallest.

While all of these ergonomic details are nice to know, perhaps the most important part of your chair selection should be, what I call, the Sit Test. Try to adjust the chair for your comfort. Does it "feel right". If it doesn't, continue looking. If it does, move it over to your prospective work table, and imagine that you are typing on your keyboard. Does it still "feel right"? For obvious reasons, it is best to buy the chair at the same place where you find a work table. But there is one other item of computer furniture that you will need.

A printer table. If you have a hard disk system, do NOT, under any circumstances, place the printer on the same table as your computer. Even in a floppy disk system, it is not advisable to have your computer and printer share the same table because the printer will obviously shake the computer while printing. If you shake a hard disk system during use, such as when printing, you run the risk of destroying data on your hard disk with a head crash. Hard disks can be temperamental enough without adding a foolish risk during operation. Besides, it is much cheaper to buy a separate printer table than it is to replace a hard disk.

If you are aesthetically inclined, you might want to choose a printer table that matches your computer table. One example of a good printer table is shown in Figure 3.

Figure 3
A Typical Printer Table

Most tables designed specifically for printers have one or more slots in the top so that you can route paper to and from the printer. If you are interested in saving some money, I have found that a "microwave cart" with casters is available in knockdown form, and I use one as a stand for my letter quality printer. It has a nice storage area underneath the printer that I use for ribbons, daisy wheels, etc.

Lighting And CRTs

Good lighting is essential at a computer for various tasks including reading the manual when all else fails. Too little or too much lighting can cause eye fatigue and eyestrain. Even the color of the light can be important, but my experience is that this is primarily a matter of personal preference. You can use either an incandescent (the light bulb) that provides "warm" light or a fluorescent (tube) that provides "cool" light. Most people prefer an incandescent light because they feel warmer and look better than with a fluorescent. My personal preference is a fluorescent desk lamp because it runs cooler, and I have more than enough heat in my study generated by the computer and its peripherals.

Regardless of which type of light you prefer, it is best to get one that is adjustable for a wide variety of tasks. I have found that any light with an adjustable "gooseneck" seems to be the best because it is easy to adjust and move out of the way when it is not needed.

But there is another indirect lighting consideration. You can easily get eyestrain from glare on the CRT from a window, and I have found that I can minimize the problem by keeping the miniblinds on my study window closed so that the room is fairly dark. Although you can find CRTs with a special coating or surface that is "glare resistant", that still does not solve the problem. You can even buy special CRT screen overlays that are supposed to minimize the problem, but I have found it is easier, better, and much cheaper (i.e., reduced electric cost) to keep the room darkened. Then, an adjustable light is an easy way to get the light where you want it without having the glare on the CRT.

When you are using any kind of a CRT, the eye-screen distance is quite critical for many people. Some must sit closer than others in order to see the display properly. People who wear bifocals can have unique problems since the lower lens is used for close work. If the CRT is at an inappropriate angle, the bifocal wearer must maintain a very uncomfortable position in order to see the display. That can, quite literally, give you a pain in the neck.

For those who do not wear bifocals, it is usually agreed that the center of the display should be between 10 and 20 degrees below the horizontal plane of the eye for comfortable viewing. Since different people will have individual preferences for the best viewing angle, an adjustable base, such as shown in Figure 4, allows the display to be tilted to suit nearly any preference.

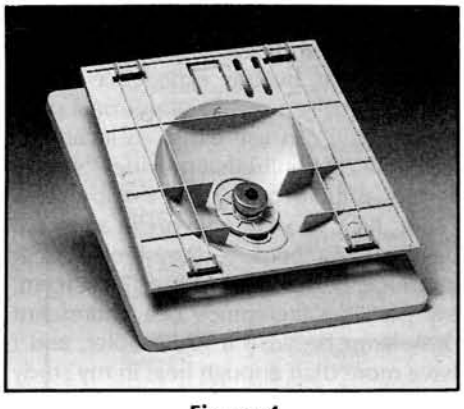


Figure 4
A Typical Tilt/Swivel CRT Base

This feature has become so important that some manufacturers are including a tilt/swivel base as part of the product. For example, my NEC MultiSync monitor has a built-in base that can tilt and swivel for easiest viewing. A base like this can also help reduce glare on the CRT aside from making it easier for multiple users to adjust the screen to their preference.

At this point, we have talked about how to set up a comfortable place to work with your computer. A good workstation, a comfortable chair, and proper lighting can determine how useful your computer really is. Although you may think that such comfort is a luxury, you will find that your computer is really more useful than you thought. That is only because you will comfortably be able to spend the time required to properly learn the system and its hardware without the annoying aches, pains, and eyestrain. Some people spend thousands of dollars on hardware and software, but haven't thought of spending a little more to set up a good working environment to use it effectively. By the way, I hope that nobody told you that getting into the world of personal computing is cheap . . . it isn't.

Now that we have completed the physical setup of the workstation, it's time to take a look at some other helpful items.

Power For Your System

Electronic equipment of all kinds is sensitive to the type of electricity that is used to power it, and computers are especially sensitive in this regard. Unfortunately, most of the power available at a standard wall socket is not as "pure" as you might think. For example, the voltage can vary from 105 volts to 130 volts, and you may not ever know about it unless your computer does some strange things. And there are all kinds of interference that can be transmitted along a power line because it can act as a very large antenna.

The first kind of power problem is called a surge which is a sudden increase in the voltage that can exceed normal limits. Lightning is perhaps the most striking (pun intended) example of this. Excessive voltage coming through the wall socket can cause serious and expensive damage to your system. You can help protect your system from this with various kinds of units called surge protectors.

The two other major kinds of interference are elctromagnetic interference (called EMI) and radio frequency interference (called RFI). Without going into all of the gory technical details of these kinds of interference, suffice it to say that it can be caused by an electric motor, such as in a refrigerator or an air conditioner, as well as other similar devices.

Perhaps the best solution for this problem is to get a power strip, such as shown in Figure 5.

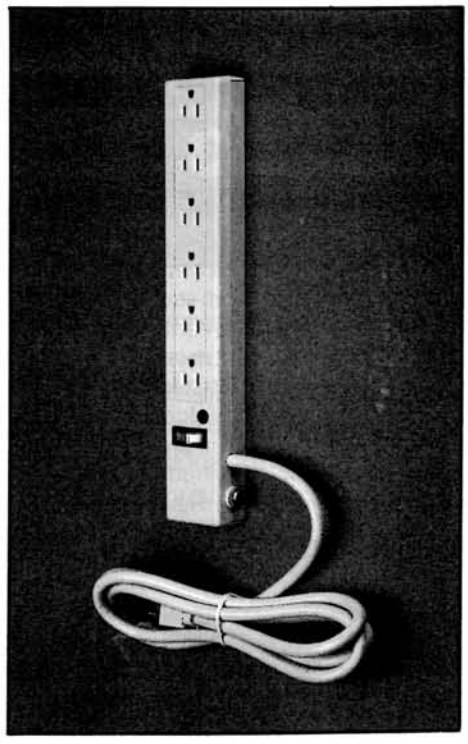


Figure 5
A Typical Power Strip

A multi-socket power strip also makes it easy to power-up your comptuer and all other peripherals, such as the CRT and printer, by turning on one switch. But there is one thing you should know.

Most surge protectors contain a component that is "sacrificed" to prevent a surge from reaching the computer. If a surge occurs, the surge protector becomes useless, and it must be replaced. Some people prefer to get a small surge protector for a power strip that provides only EMI and RFI filtering so that only the inexpensive (usually $10 or less) surge protector unit needs to be replaced.

Attaching Peripherals

One of the most difficult problems in setting up the system is to cope with the bewildering variety of cables and connectors that are part of the game. You will find it much easier in the long run to buy a cable for your system at the same time that you buy the peripheral, such as the printer. Since many computers have 25-pin connectors for these connections, I have found it to be most effective to buy 25 conductor cables even though they are more expensive as an initial investment. While you never need all 25 pins for any

computer peripheral, the advantage is that you can generally use the same cable for a printer as for a modem even though the wires used by each peripheral are different. There are, of course, some exceptions to this as you will see when we discuss the various kinds and connections for printers and other peripherals in this series.

There is one other interesting problem that can occur. You will find that certain kinds of devices must be connected to a specific outlet (called a port) on the back of your computer. For example, both a mouse and modem must be connected to a serial port that is usually identified as COM1 on most PC compatible computer systems. How can you connect both devices to your computer when most only have a single connector?

The easiest and least expensive way to fix this problem is to get something called an A-B box which has a two-position switch cleverly labeled "A" and "B". You connect one end of the box to your computer with a cable. There are two other connectors on the back of the box, also cleverly labeled "A" and "B", so you can connect a mouse to one and a modem to the other. One example of an A-B box is shown in Figure 6.

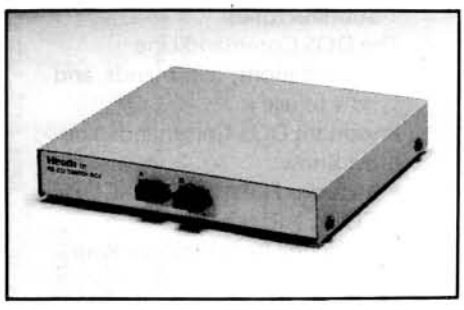


Figure 6
A Typical A-B Box

This is an easy and inexpensive way to connect two devices to your computer, but there is one caution. You should NEVER switch devices when your computer or the device is powered-on. Always turn the power off before changing the position of the switch.

One other interesting point about the A-B box is that most of them have standard 25-pin connectors on the back, but not all have a switch that will switch all 25 pins. If you need one, I recommend that you find one that can switch all 25 pins so it can be used for other equipment if you buy new peripherals. You will also want to check to make sure that the connectors are the standard 25-pin connectors because they are the most flexible. Some units have the 36-pin Centronics connectors.

Now that you have most of the hardware aspects under control for your system, there is one other important thing you can do to help keep your work organized.

Organizing Floppy Disks

When you begin to work with your computer, it is easy to think that you can remember what is on all of the floppy disks. As you get more software and use your system to create various files, you will find that the floppy disks seem to be capable of reproducing themselves. For example, I began working with my computer over six years ago, and I now have over 500 floppy disks in my library.

One of the best ways to help keep floppy disks organized is with a special cabinet made for that purpose, such as shown in Figure 7.

Figure 7
A Typical Floppy Dual Storage Cabinet

These cabinets typically hold up to 50 floppies, although you can find some that hold more. This is a good investment since it will help you keep your disks organized.

Each box of disks usually includes multicolored labels that can be quite helpful in organizing your programs and data. For example, a red label can indicate a bootable DOS disk created with the FORMAT/S command that is required to start the computer. You might choose blue for word processing files and green for spreadsheet or financial data. But there is one other trick that can help you set up an effective disk library. It uses a concept that includes a discussion of a SYSTEM DISK and a DATA DISK.

A SYSTEM DISK contains programs (i.e., software) of one kind or another and that's all. You will need to create a disk to start your system with DOS, and this is easily created with the FORMAT/S command to transfer the System to the disk. In addition, you will probably want to create a "Working DOS Master" disk that is set up for your particular system. We will discuss that in more detail in the next couple of articles.

A system disk also contains various programs that you use in your system, such as a word processor or a spreadsheet. If you use any of today's work processors, such as WordStar or Microsoft Word, you will also find that you need multiple system disks for those programs. One disk may contain the basic program files, one may contain the dictionary for spell checking, and another may contain various utilities for that specific word processor. The point is that none of these system disks should contain any of your data. In most cases, a lot of today's software REQUIRES the use of this concept because there are many large program files that just will not fit on a single disk. And in many cases, the software includes an installation procedure or program that automatically sets up the programs for use on your system. All you need to do is follow the instructions in the manual, and these disks are frequently initialized with the FORMAT command alone (i.e., no /S). When this procedure is used, there is often no room at all for data on the disk anyway.

On the other hand, a DATA DISK only contains data for a specific program, such as a word processor. And you will find it helpful to keep the data disks organized and restricted to a specific subject. For example, you might have word processing data disks for Personal Letters, Business Letters, Personal Projects, Business Projects, and Miscellaneous Correspondence.

The System Disk and Data Disk concept works quite well on most systems, even for a hard disk system. The idea is that the system disk always runs in the system drive — typically drive A on a floppy disk system or drive C on a PC compatible hard disk system. The data disk always runs in the "data" drive which is usually drive B for a floppy or drive D for a hard drive system, it is usually worth it because the idea is easy to use. But there is one other significant reason for using this concept with any floppy or hard disk system.

At some point, you will want to update your software to a new version. It may be

a new version of DOS or it may be the latest and greatest release of your favorite word processor. Regardless of what software it is, it is MUCH easier to update software when all of it is clearly located (and labeled) on a system disk. That limits the number of updates you must make, and then you KNOW that you have updated everything.

For a hard disk system, this approach has another side benefit. If all of your programs, including DOS, are located on drive C, you only need to backup that drive when you update or add software. Depending on how frequently you buy new or updated software, that can save a significant amount of time. The data drive, drive D, should be backed up more often since something on that drive may be updated every time you power-up the system.

Since your disk library is now fairly well organized, all you have to do is label the dividers for the floppy disk storage cabinet. That should be easy since you have already determined the functional categories of the floppy disks. Creating labels for word processing, spreadsheets, and games is easy. My preference is to place the appropriate system disk(s) in the front of each group and the backup disk(s) follows the data disks.

I also have a divider which is labeled as "Blanks" which means that the disks are formatted, but no data is recorded. I have also found it very useful to sequentially number each disk and include the date that the disk was first formatted. By now you are so well organized that you probably can't stand it.

There is one other trick that can save you a lot of time in setting up your disk library. When you got the storage cabinet for your floppy disks, you did not throw away the box that the floppy disks came in, did you? That is an ideal storage container for ALL of the distribution disks that were supplied with your software. NEVER use these masters for normal program operation unless the instructions tell you to do so. This is only required when you have a copy-protected program, such as Lotus 1-2-3.

In addition, you will find that these old boxes can be used for archive sotrage of disks that contain valuable data that you can't get rid of for some reason. One example of this is my compilation of income tax data for each year that usually includes two disks — one for spreadsheet information and one for word processing data.

I use the white Avery file folder labels to label the edge of the box as MS-DOS Masters, WordStar Masters, Microsoft Word Masters, and so on. I use the same labels on the outside back of the box to record the CONTENTS of the box. Examples of these labels include MS-DOS version 3.20, MS-DOS version 3.21, WordStar 4.0, etc. I also keep a three-ring binder with all directory listings of each disk so that I can quickly locate just about any file on these masters.

Another use for an empty diskette box is to store blank disk labels. Keep them next to your system so they will be handy for immediate use. Nothing is more frustrating than to have a bunch of disks with no labels and then trying to determine if they contain valuable information.

This organization process I have described is not particularly unique, and it's only one way that it can be done — there are obviously many variations that may be better for you. The purpose of this description is to give you some ideas that work, and if you can't think of a better approach, perhaps this concept may also work well for you.

In Summary

The ideas that I have discussed here were intended to help you with some of the less obvious aspects in setting up a new computer system. For obvious reasons, you will not find any of this information in your computer manuals since it is not directly related to actual system operation. I hope this article gives you some ideas about setting up a comfortable and productive place to use your system. It is only intended to help give you some ideas that I have found useful over the years, and I think you will find some of these ideas useful in setting up a system customized to your particular needs.

All items mentioned in this article are available in a variety of places. Computer furniture can be found in most office supply stores, as well as many furniture stores and home improvement centers. Since there is considerable price variation and competition in this area, you will want to shop around to compare price, quality, and function. You should be able to find a good-quality computer table (about $200), a chair (about $70), and a light ($20-80) for a minimum investment of about $300. If you are interested in absolute top-quality solid wood furniture, you can spend as much as $2,000 for a setup, but an inexpensive work station works just as well. In the event that you do not have any suppliers of these items near your home, I have listed various vendors at the end of this article. You can find one or more of the items discussed here in their catalogs, but their prices may be considerably higher than local sources.

In The Future

This series consists of 15 articles that will help you set up and use your computer system. One article will be published each month, but if you can't wait, all articles will be compiled and published by HUG in a "book" that will be available later this year. Since many of you may be wondering what topics will be specifically discussed in this column, I have included a list of the subjects in Figure 8.

1. Setting Up Your Computer System
   - Computer furniture and a disk library
2. Powering-Up the System
   - Booting the system, basic DOS commands and file names
3. How to Use Subdirectories
   - Creating and Sailling the Seven Subdirectories
4. The DOS Command Line
   - Its limitations, commands, and how to use it
5. Important DOS Commands You Must Know
   - Includes FORMAT, DISKCOPY, COPY, and others
6. Connecting Peripherals to Your Computer
   - Includes Printers, Modems, and types of connectors
7. Using Batch Files and CONFIG.SYS
   - With practical examples of batch and configuration files
8. Using Input/Output (I/O) Redirection
   - How to use this DOS feature
9. Understanding Video Hardware
   - Video adapters (CGA, MDA, EGA, and VGA) and CRT monitors
10. Adding More Memory to Your Computer
    - ROM, RAM, expanded and extended memory
11. How to Select Applications Programs
    - Special emphasis on word processors and spreadsheets
12. How to Select Utility Programs
    - Disk organizers, undelete pro-

grams, other utilities
13. Selecting and Setting Up a Hard Disk
    - Getting what you need and setting it up
14. Other Useful DOS Commands
    - Using SEARCH, SUBST, JOIN, and other commands
15. Maintaining Your Computer System
    - How to maintain your computer and printer

Figure 8
POWERING-UP Article Titles

This list only includes the general subject for each article with a brief description of what is included in each. Many of these articles include practical examples of commands and other suggestions that will find immediate use in your system.

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

Where To Obtain Catalogs

Global Computer Supplies
45 So. Service Road
Plainview, NY 11803
Heath/Zenith Computer Centers
Heath Company
Hilltop Road
St. Joseph, MI 49085

Inmac
Mail Stop 5060
2465 Augustine Drive
P.O. Box 58031
Santa Clara, CA 95052-9977

Lyben Computer Systems
1050 E. Maple Road
Troy, MI 48083 *

FINAL Z-100 SOFTWARE CLOSE OUT

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

PART NUMBER	DESCRIPTION	LIST PRICE	SALE PRICE
MS-463-1	Z-Basic (16 bit) interpreter	$175.00	$10.00
MS-463-7	Multiplan	$195.00	$10.00
MS-253-1	BASIC-80 (8-bit)	$175.00	$10.00
CD-463-2	Condor File Manager	$299.00	$10.00
PK-100-4	All 4 listed above	$819.00	$38.00

*** 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. $244.00 Dual Drive Unit $389.00

COLOR GRAPHICS UPGRADE. All memory chips (16 pieces 150 ns) required to update Z-100 Series computer for color. Installation instructions included. Order Memory Kit #100-64-16 ..... $45.00

MEMORY UPGRADE. All memory chips (9 pieces 150 ns) required to upgrade from 128K to 192K RAM. Installation instructions included. Order Memory Kit #100-64-9 ..... $25.50

ZMF100A by FBE Research. A modification package which allows 256K chips to be used on the old-style motherboard to reach 768K. Simple assembly with no soldering or trace cutting. Compatible with Easy PC and Gemini Emulator. $60.00 alone or $289.00 with 27 256K RAM chips included.

UCI MEMORY UPGRADE PAL CHIP SET. For Z-100's with the newer motherboard (181-4918 or greater). This set permits installation of 256K RAM chips on the motherboard. PAL set $64.00 or $293.00 with 27 256K chips required for 768K of memory on the motherboard.

SmartWatch by FBE Research. If you don't have a clock for your Z-100, get this one. More details under Z-150 upgrade listings $38.00

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 $188.00 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 $288.00 With RAM installed and fully tested, 512K $432.00, One MEG $576.00, Two MEG $864.00 Add $35.00 for EasyDrive RAM Drive Software if desired.

UCI EASY-I/O. S-100 board that provides IBM PC communications port compatibility with your EasyPC. Easy I/O-1, One Serial Port $91.00. Easy I/O-2, Two Serial Ports, One Game Port, Clock-Calendar $125.00

UCI EasyWin. Winchester Drive Systems at reasonable prices. Complete Hard Disk Systems for mounting inside your Z-100. Systems complete with Seagate Drives, 21 MEG $578.00, 31 MEG $598.00 System without Drive and Controller $239.00

CDR Z-100 SPEED MODULE. Run your Z-100 Computer at 7.5 MHz. Installs easily with no soldering. Externally switchable between Speed and Normal mode. Payload $44.00

••• PANASONIC AND HEWLETT PACKARD PRINTERS •••

KPX - 1080i 120 cps 10" $219.00
KPX - 1090i 120 cps 10" 29cps NLQ $272.00
KPX - 1092i 120 cps 10" 33 cps NLQ $338.00
KPX-1592 180 cps 16.5" 38 cps NLQ $421.00
KPX-1595 240 cps 16.5" $462.00
Hewlett Packard Laser Jet Series II printer with toner $1775.00

*** SEAGATE HARD DISK DRIVES ***

ST-225 20 MEG Winchester Hard Disk $259.00
With Western Digital Controller & Cables $289.00
ST-238 30 MEG, Requires RLL type controller $289.00
With RLL Controller & Cables $315.00
ST-4038 30 MEG High Speed for Z-200 ..... $479.00
ST-4053 40 MEG High Speed for Z-200 ..... $619.00
ST-4096 80 MEG High Speed with Software $919.00
ST-251 40 MEG High Speed Z-150/200 ..... $439.00
ALL DRIVES FORMATED AND TESTED.

*** HALF HEIGHT FLOPPY *** DISK DRIVES

MITSUBISHI M501 5.25" 48 TPI DS/DD 320K/360K $105.00
MITSUBISHI M504 5.25" 96 TPI DS/DD 360K/1.2 MEG . $152.00
MITSUBISHI MF353 3.5" in 5.25" frame 720K $109.00
TOSHIBA ND04 5.25" 48 TPI DS/DD 320/360K $89.50
TOSHIBA NDO4 with grey faceplate instead of black ... $89.50
TOSHIBA NDO8 1.2 MEG for Z-200 or AT Computer ... $119.00

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

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

Single 5.25" unit for Half Height Drives $68.00
Dual 5.25" unit, for Half Height Drives $92.00
For WINCHESTER Drive, with fan $145.00

••• MITSUBISHI COLOR MONITORS •••

MODEL	MODE	DOT SIZE	RESOLUTION	LIST	PAYLOAD
XC1409C	CGA	0.4 MM	640x200	$519.00	$296.00
XC1410C	EGA	0.4 MM	640x350	$650.00	$419.00
XC1430C	EGA	0.31 MM	640x350	$739.00	$489.00
AUM 1371	EGA	HI-RES	920x580	$849.00	$519.00

*** UPGRADE ACCESSORIES FOR Z-148/150 *** SERIES COMPUTERS

VMM 150 Video Card Eliminator for Z-150/160 computers. Allows use of EGA color card $45.00

SmartWatch from FBE Research. Installs in ROM Socket on CPU Board in Zenith computer series Z-100/148/150/158/160. This 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 $38.00

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. For Z-150/160 only $164.00

Winchester Hard Disk Drive Internal Set-up. Includes Winchester drive, controller/interface card, cables and all hardware. With 20 MEG (formated) drive $289.00 30 MEG $315.00 May be installed in Z-148 using an Expansion Card sold below.

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

Winchester Hard Disk Drive External Set-up. Includes Winchester drive, controller/interface card, power supply and case with fan.
With 20 MEG (formated) drive $434.00
With 30 MEG (formated) drive $460.00

SECOND INTERNAL DISK DRIVE with instructions and hardware $105.00

ANCHOR INTERNAL MODEM. Fully Hayes compatible 1200/300 baud with powerful Com software included $95.00
Same but 2400/1200/300 BAUD $182.00
External 2400/1200/300 BAUD $198.00

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 8.0% sales tax. We accept purchase orders from schools, government and approved accounts.

XENIX

Part One

Matt Elwood
1670 N. Laddie Court
Beavercreek, OH 45432

Introduction And History

This article, which is first in a series about XENIX, will briefly cover the history of XENIX and compare it to MS-DOS and OS/2. The article will also provide information about basic XENIX concepts and operations. Each article will provide a greater understanding of XENIX as a "new" operating system.

XENIX is a multitasking operating system whose roots go back to UNIX. XENIX is the 80286/80386 version of UNIX. Because XENIX can support the complete capabilities of the CPU, there are separate versions for each unit. Because the 80386 is backward compatible with the 80286, it will work with the 80286 version, but not as fast as the 32-bit 80386. Microsoft XENIX, the one I'm reviewing, is only 80286-based, whereas SCO (Santa Cruz Operation) XENIX provides both 80286 and 80386-based versions. The 32-bit 80386 version provides capabilities approaching minicomputer capabilities in a microcomputer.

UNIX, from which XENIX was developed, has been around for some time. It predates MS-DOS and falls in the CPM time frame. With UNIX as a base, functions in MS-DOS, like the directory structure, come from UNIX. Other functions or commands like input/output redirection (using the < and > signs to redirect input or output into a disk file or device) came from UNIX. Also, the batch file idea comes from UNIX's script file. They even use the same commands, such as 'shift', which shifts the batch parameters, and 'if', which is a comparison statement.

What is shift and if? Explain these terms. When UNIX was first developed in Assembly Language in 1969 by Ken Thompson for Bell Labs in Murray Hill, New Jersey. Computers as small as they are today would not have been dreamt about. Computers were large and required large amounts of power and air conditioning. The personal computer revolution had not started yet, although it was starting with the introduction of the first "pocket" calculators. The development of transistors and integrated circuits that were necessary for today's computers was started and Bell Labs was a leader in this field. Only the government, universities and big companies owned the computer of the day. Their systems were big and provided, in many cases, fewer capabilities than today's minicomputers and the new 80386-based microcomputers. UNIX was one of the many operating systems under development. It was to develop standards which could be used by different computer systems, unlike the unique operating systems developed by major computer companies, like IBM. Although it was complex, most of the other systems were much more complex.

After a start in Assembly Language, Thompson started developing UNIX in a language called B. Dennis Ritchie joined Thompson and together they developed C. Using C as the language, they rewrote UNIX. They liked C and decided to write a compiler for UNIX which meant the system they developed could recompile itself. UNIX was kept private in Bell Laboratories until 1976 because of government regulations.

In 1976, the first public version of UNIX was released to universities. There have been many updates since that first release with the current approved Bell version being five. The original release version was six. After two more years, version seven was released and it was available for anyone. The development was then moved to Berkeley on a VAX. There, the VAX version (3 BSD) was developed. Later, versions 4.1 and 4.2 BSD were also developed. In this development stage, programs, such as vi and termio were developed. These versions all grouped together were called System III.

This was fine until the home computers started popping up beginning with the Heathkit H-8 and the Altair models. These

first home computers started the current computer revolution. This was when Microsoft was formed by the men who first developed BASIC. When IBM started the development of the PC, two operating systems were available. CPM was the most common system and could have been the operating system of the PC, but for the dealings with Digital Research, if you believe the stories that have surfaced over the years. Microsoft was working on MS-DOS when IBM approached for an operating system. From this, MS-DOS became the major operating system today. Microsoft was also developing XENIX for microcomputers, but the first computers did not have the capability necessary for it to work efficiently.

The first decision to develop XENIX was made in 1980 when Microsoft licensed UNIX source code from AT&T to be used for PDP-11s (the same as the H-11 systems). This was the first small 16-bit microcomputer system. No floppies then, only paper tape! I even remember seeing a paper tape reader/writer at one of the early HUGCONs.

Later, SCO, seeing the success Microsoft had with the PDP-11s, developed XENIX for the 8088-based IBM PC. This was where SCO and Microsoft became co-developers. This first version was slow and had limited capabilities because of the machines it operated on.

In 1983, AT&T, the first developers of UNIX/XENIX, contracted Intel to develop a version of XENIX for the new 8086/8088 series microprocessors. Intel then contracted the development work to other contractors, like Interactive Systems. These companies would develop XENIX into its current System V Release 3.0 for the 80386 microprocessor. While developing XENIX for the 80386, Interactive decided to work with Phoenix Technology to develop VP/ix, an MS-DOS shell for XENIX. To do this, Phoenix then contracted Microsoft. Microsoft in turn incorporated VP/ix into its XENIX.

Just recently, Microsoft and SCO merged their XENIXes to make one big XENIX with XENIX System 5 Release 2.2. SCO and Interactive later went on to develop versions of XENIX for both the 80286 and 80386. Several other companies have developed versions, or subsets of XENIX for microcomputers.

All this is just to say that in 80x86-based XENIX there is barely any competition, but with growth in the UNIX/XENIX area, more should appear. The two main companies, Microsoft and SCO, are now working together and this should help establish XENIX as a major contender in the operation system market. That is the history of XENIX, and now we will talk about some basic XENIX concepts.

The word used to describe XENIX is large. You need at least a 15 Meg hard drive for XENIX, and actually, a lot more if you plan to use it. Also, you need a lot of memory for XENIX just gobbles it up. Just the XENIX kernel, buffers, and boot take up 250k. In my 1 Megabyte system, I only have 401k free. XENIX, unlike MS-DOS, will use all available memory. The more RAM that is available, the better it runs.

XENIX runs on a very different basis than MS-DOS. For one thing, it is multiuser and multitasking. In another example, XENIX uses file systems to refer to disk drives. For example, hd00 is the main (boot) hard disk. The floppy disks are different. You use the command 'mount' to remap the disk drive to a directory specified by you. Remember HDOS? It used the same command and was structured like UNIX. It was device independent and required device drivers be loaded if they were to be used. (My dad had HDOS on the first H-89 we worked on.) The floppy drive names are very long and most sound the same. This can get kind of confusing, especially in the close names. Also, you have to type 'haltsys' when you are finished because XENIX keeps lots of files open. Even if you let the computer stand idle, the hard disk accesses a lot.

Microsoft just released OS/2, their multitasking operating system. This doesn't compare to XENIX in speed. OS/2 was developed for IBM to take advantage of the 80286 CPU. It is better than MS-DOS, but still leaves much to be desired. As this expensive system is released, all of the expectations that were put out about it are not being lived up to. This first version is for programmers and developers and not end users.

In a series of tests conducted by Neal Nelson and Associates comparing OS/2 and XENIX, XENIX out performed OS/2 in 13 out of 17 tests. OS/2 failed when it came to disk I/O operations, with up to 7 times longer to perform similar operations. This gets worse when more tasks are added. With 15 tasks, OS/2 takes around 300 seconds to complete a task, while XENIX takes only 40 in the test by Neal Nelson and Associates. It was pointed out in the article that OS/2 is a 16-bit operating system for the 80286, while the XENIX used the 386 32-bit version. Both tests were run on an IBM PS/2 Model 80, a 32-bit machine. In an article in Government Computer News, 22 Jan 88, Zenith was quoted as saying that many of the Z-386 machines are being shipped with XENIX. It was also pointed out that XENIX is a multiuser environment, while OS/2 is multitasking.

XENIX handles devices in a different way, too. The devices are represented by files in a special directory called /dev. The device names include file systems for disk drives, tty for serial ports and lpr for line printers. In XENIX, you need directories to handle the massive amount of files. XENIX can handle a large amount of directories and in big systems, you might have directories reaching down 10 levels (e.g.

```
usr0/bldg12/floor2/xp/jeffh/finance
/data/1987/rpts/january).
```

So you can get extremely complex. XENIX also permits locking files so that you can control your files in the multiuser world.

Some nice features in XENIX are also a mail system, UUCP (XENIX to XENIX communications), C with the development library, and a wide range of text processing commands. The mail system is nice, being originally developed by Berkeley. It is a little hard to learn at first, but has interactive help.

XENIX includes two word processors. EX is a line editor and is usable, but it is not like the commercial editors most people are use to. The text editor, 'vi' (visual editor), is in my opinion, a very bad word processing program, but you have to make do with it. It has two modes, command and typing. It is hard to tell which mode you are in, and since you use letter commands in the command mode, if you type one in while in typing mode, it will be in your file. Also, there are two commands to get into typing mode, 'a', for append to add or append to text and 'i' to insert text above the current line, so if you select one, you can't do the others. These editors will be covered in greater detail in a future article.

To make up for the bad text processor, there is a nice spell checker and 'diction', a program that checks your text and tells you where you aren't clear. To go with diction, there is 'style' that gives you 5

readability grades and tells you various information about your writing. As an example, this article has a readability index of XX. This will also be covered in a future article. There is also a text formatter, nroff and troff, so if you have a laser printer, you can do different type sizes and symbols. As you look into these formats, you must use "dot" commands. If you use WordStar, you will notice the dot commands are similar, if not the same as nroff and troff dot commands. Again, something that moved from UNIX to the microcomputer world early on.

In the development system, you get a macro assembler very much like the one in MS-DOS. You can even cross-assemble to MS-DOS. It also includes a full version of C and link editor. It also contains 'make', which does batch files for compiling C programs. Also in the development system, you get 'yacc' and 'lex'. Yacc is a compiler/interpreter generator. It gets rules and transforms them into C code. You can make your own programming language in this. Lex is a lexical analyzer where you can change English into C commands.

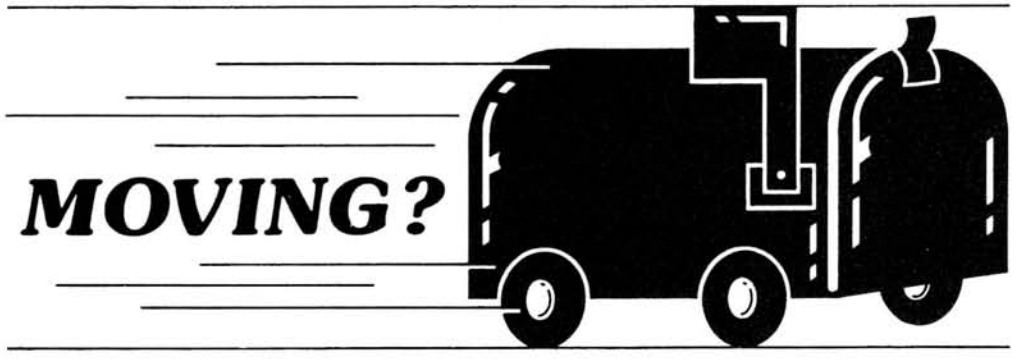
In SCO and Microsoft XENIX, you have access to DOS commands. In Microsoft, you can get text files and directories from MS-DOS disks. In SCO, you can actually run MS-DOS programs. You can have the programs running on different terminals, too. You must have the VP/ix module to run these programs.

The protection for dial-in and network is very good. Each person has a password and a login name. The password and the login name log them into the system and put them in the correct directory. Another form of security is on separate files, as I said earlier. When you list the directory in the long form, you see 10 dashes and letters on the very left side of the screen. The first bit is a file type bit, which tells what type of file it is. The other bits are permission bits. We will talk about these in a later article.

XENIX communicates with its users by shells. The shell processes commands, I/O, and script files. XENIX provides three standard shells and two special shells. The first shell is the standard shell. Its prompt is a '#' and it just does all of the regular commands. The normal user would use this shell. The next shell is the C shell. This is similar to the standard shell. Its prompt is 'n%', where n is the number of commands you have executed. A special command in the C shell is history, a command where you can list the last 20 commands you have executed and repeat any one again. The final normal shell is the visual shell. This is menu based for the beginning user. It has commands to edit, checks the mail, and does everything you can do in regular shell, but is menu-driven. This is especially good for a first time user. The last two shells are the restricted shell and the uucp shell. The restricted shell is for guest users who have less privileges. The uucp shell is for XENIX to XENIX communication.

This has been a very brief overview of XENIX. I will be back over the next several months to provide the readers of REMark more of an understanding of the operating system, XENIX. This system is available now. It is a multiuser system and versions are available for the powerful Z-386 computer systems. Applications are becoming available and I will mention those that I have had the opportunity to use.

XENIX Systems 5 Time Share System
OS-63-80 $429
XENIX Systems 5 Development System
OS-63-81 $499
XENIX Systems 5 Text Processor
OS-63-82 $149
Heath Company
Benton Harbor, MI 49022



Don't Miss A Single Issue!
Let us know 3-4* weeks before you move!

*** If you call the change is effective immediately!**

Classified Ads

REACH THOUSANDS OF DEDICATED HEATH/ZENITH COMPUTER OWNERS!!

Classified Ad Rates

There is a minimum of $5.00 for 10 words or less. Each additional word is $.40. Payment for all classified advertising must be in advance.

Display Classified is available at $35.00 per column-inch. Minimum one inch. Display Classified is commissionable to recognized agencies.

How To Count Words

Count one word each for initials, standard abbreviations, whole numbers, name, address, city, state, zip, area code and telephone number. Dimensions (such as 6 × 9) are counted as one word. Box or department numbers are counted as one word each. All classified ads (not display) are set in the same size type. The first several words, depending on the ad, are set in all caps.

Continuity Discounts

Run your ad at least 3 times during the year; and receive a discount off the regular price.

Run 3 insertions — 3% off, 4 insertions — 4% off, 5 insertions — 5% off, . . . 12 insertions — 12% off.

Issue & Closing Date

Issue Date	Closing Date	Issue Date	Closing Date
January	November 15	July	May 15
February	December 15	August	June 15
March	January 15	September	July 15
April	February 15	October	August 15
May	March 15	November	September 15
June	April 15	December	October 15

REMark Magazine has the right to refuse any ad for any reason.

Classified Order Blank

First	10	words	at	a
cost	of	$5.00	each	insertion.
11-$5.40	12-$5.80	13-$6.20	14-$6.60	15-$7.00
16-$7.40	17-$7.80	18-$8.20	19-$8.60	20-$9.00
21-$9.40	22-$9.80	23-$10.20	24-$10.60	25-$11.00
26-$11.40	27-$11.80	28-$12.20	29-$12.60	30-$13.00
31-$13.40	32-$13.80	33-$14.20	34-$14.60	35-$15.00
36-$15.40	37-$15.80	38-$16.20	39-$16.60	40-$17.00
41-$17.40	42-$17.80	43-$18.20	44-$18.60	45-$19.00
46-$19.40	47-$19.80	48-$20.20	49-$20.60	50-$21.00

Enclosed is a check or money order of $_____ for _____ words.
(Minimum order: 10 words for $5.00. Each additional word $.40.)
Please insert this advertisement in the_____issue.

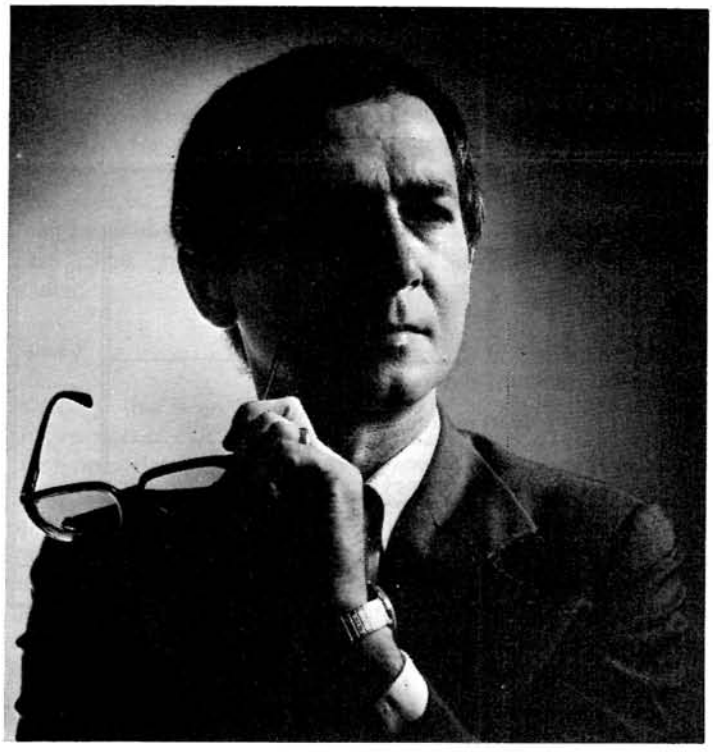
Signature_____

(Please type or print)

Name_____Phone_____
Company_____
Address_____
City_____State_____Zip_____

MAILING ADDRESS: Make check or money order payable to: Rupley's Advertising Services, Dept. — REM, 240 Ward Avenue, St. Joseph, MI 49085.

THIS COULD BE YOUR AD! For as low as $5.00, you could advertise personal items in this space. And for as low as $35.00, you could advertise commercial items. People read these ads, you did!



Mainstream Computing

Joseph Katz
10 South Edisto Avenue
Columbia, SC 29205

At last I have a Z-183 laptop computer all my own. I've wanted one for some time, but I couldn't get my hands on one until now. Now that Zenith is about to introduce a new AT compatible laptop computer, I'll be able to use and write about its XT compatible laptop computer.

Nevertheless there's a lot to say about this excellent machine, which of course is no less excellent just because Zenith is pushing on to a new model. In a sense the Z-183 is an even better machine right now because it's a better buy right now. If you even think you can use a laptop computer, do some Z-183 shopping right away. The prices ought to interest you.

As for what there is to say about the Z-183, look in this issue of *REMark* for the beginning of a new series on Zenith's laptops. I think I can help you get the most out of them. If you have any laptop hints, tips, tricks, or products you want to share, please do write me about them. I'll be grateful.

The new WordStar 2000 Plus

I'm not much interested in office word processing programs for my own use. What I and many other writers do is compose at the keyboard. So for us a word processing program is a writing instrument, not a transcribing tool. What we want is speed, power, and more of each than anyone else would ever believe. And I guess we're never quite satisfied. When the latest version of WordStar's office sibling arrived, therefore, I didn't expect to become personally involved. But that's what happened. I'm impressed. And I'm impressed because MicroPro has produced both a writing instrument and a transcribing tool.

In fact MicroPro's WordStar 2000 Plus Release 3 has star quality. It just might be the best office word processing system now available for microcomputers. What it doesn't do is make coffee. What it does do is almost everything else. *Anditdoesitfast.* It is notably faster than WordStar 2000 Plus Release 2.

Because the very first release of WordStar 2000 Plus gave this package a reputation for slowness that's hard to shake, MicroPro evidently took great pains to speed up this new version to the point at which no one could complain. Here are some speed claims from MicroPro about key differences between Releases 3 and its predecessor release: 370% faster in locating text at the end of a file; 220% faster in scrolling down to the end of a file with the down arrow key; 1,300% faster in moving a block from the top to the bottom of the file; 1,060% faster in moving the cursor to the end of the file. I haven't done my own relative time trials, but that's because I really don't care if MicroPro is fudging a few percentage points. This WordStar 2000 Plus Release 3 feels fast enough to me for almost any office environment.

Where WordStar 2000 Plus Release 3 really distinguishes itself is by providing a good integrated environment, not only for all the features we now expect in a word processing system but also for all the features we will expect from now on because this program has them.

Of course it has a spell checker and

thesaurus, both of which are very good. Of course it has MailMerge—not just "a mailmerge feature," but the current descendant of the one-and-only original MailMerge. Then it goes further. In addition to MailMerge it has a built-in mailing list management system. If what you want to do is send out documents illustrated with charts, graphs, clip art, or other illustrations, it has a specially modified version of Inset Systems' great Inset program. Inset, which on its own is winning well-deserved praise as a way to integrate graphics into text produced with word processing programs, has a lovely showcase in WordStar 2000 Plus Release 3. If you need to index or prepare a table of contents, you can do it. If you need to search for a document on your hard disk, you can do it. If you need to convert a document from another word processing program's format, you can do it.

And so, on and on, go the features of WordStar 2000 Plus Release 3. In addition to TelMerge, an integrated telecommunications program, it provides a set of "companion programs" that have won distinction on their own. Just which companion programs you get depends on the "edition" of WordStar 2000 Plus Release 3 you buy. In the "Personal Edition" you get Telemarketing Resources' PC Outline (my favorite outlining program and always on my list of software recommendations for the serious writer or student), TimeWare's ShowText (used to prepare overhead transparencies and other display materials), and Athena Software's Fill-a-Form (what you need when your work involves completion of preprinted forms such as those from Federal Express). In the "Legal Edition" you get Fill-a-Form and JURISoft's CompareRite (compares two documents and produces a third which highlights the differences in its antecedents) and CiteRite ("checks legal precedents and places them into pleadings with extreme accuracy" according to MicroPro, but it sounds like magic to me).

Look: there's no honest way to even skim the surface of what WordStar 2000 Plus Release 3 provides. It even has a sweetheart of a driver for PostScript printers—right in the tradition of MicroPro's renowned genius for supporting every printer possible. But I'm getting caught up in enumerating features again. WordStar 2000 Plus Release 3 has even more than I could have imagined putting in a word processing system. If you need this kind of word processing, this program probably is the one you need. Take a look at it.

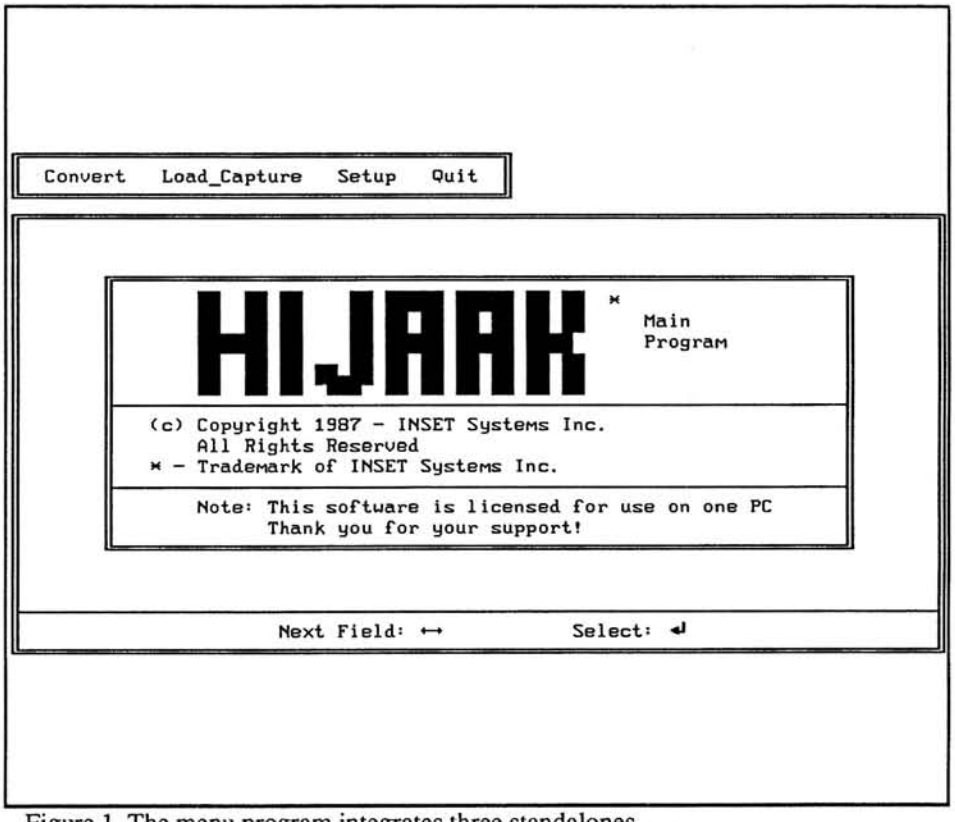


Figure 1. The menu program integrates three standalones.

Inset Systems' Hijaak

Even though Hijaak is young enough to be at Version 1.0A, it is now high on my list of programs to use in capturing screens for inclusion as illustrations in desktop publishing. Nope, even though you've seen me recommend this kind of program before, I'm not at all fickle. This kind of application is still new enough for there to be room for a variety of good products, each of which takes its own approach to a complex set of problems. And, besides, I am a fascinated observer of a situation in which several capable companies are competing to produce the perfect screen capture program. I'm interested in working with the best of them because we all benefit from their developments. Inset Systems' new Hijaak program fits in with the best.

You have a pretty good notion about the quality of Hijaak from MicroPro's selection of Inset itself for inclusion with WordStar 2000 Plus Release 3. What Inset does is integrate text and graphics screens into word processing text. What Hijaak does is capture the text and graphics screens and convert them into a form usable by a variety of important programs—not only Inset itself, but also a variety of others including PageMaker and Ventura Publisher.

The way to think about Hijaak's utility is by the file formats with which it can work: Amiga (.IFF), CompuServe (.GIF), HP LaserJet (.HPC), Inset/Hijaak (.PIX), Lotus (.PIC), Macintosh (.MAC), PC Paintbrush (.PCX), PostScript (.PSC), Tagged Image File Format (.TIF files produced by scanners), and text (.TXT). The file extensions I've included in parentheses are those that Hijaak's Convert utility needs in order to understand the file's format. It only converts from Lotus's graphics format and only converts to PostScript code, but it can convert from or to all the other formats. You therefore have access now to all the Macpaint files, for example, on which you can lay your hot little hands. Or, at least, I do with Hijaak.

Hijaak has many noteworthy features, but there are three you should know about right away. First, it can be configured to capture a screen from just about any program—including the tough ones such as Microsoft Windows and Ventura Publisher. Second, it can convert CompuServe GIF (Graphics Interchange Format) files—which means you can download the graphics treasures, produced by a host of different computer systems, available on CompuServe's active PIC Forum. Third, it can be operated simply from a menu (see Figures 1-2) or quickly by invoking the standalone programs that comprise Hijaak and are integrated by the menu. Hijaak is a package worth your attention if you need to capture or convert graphics.

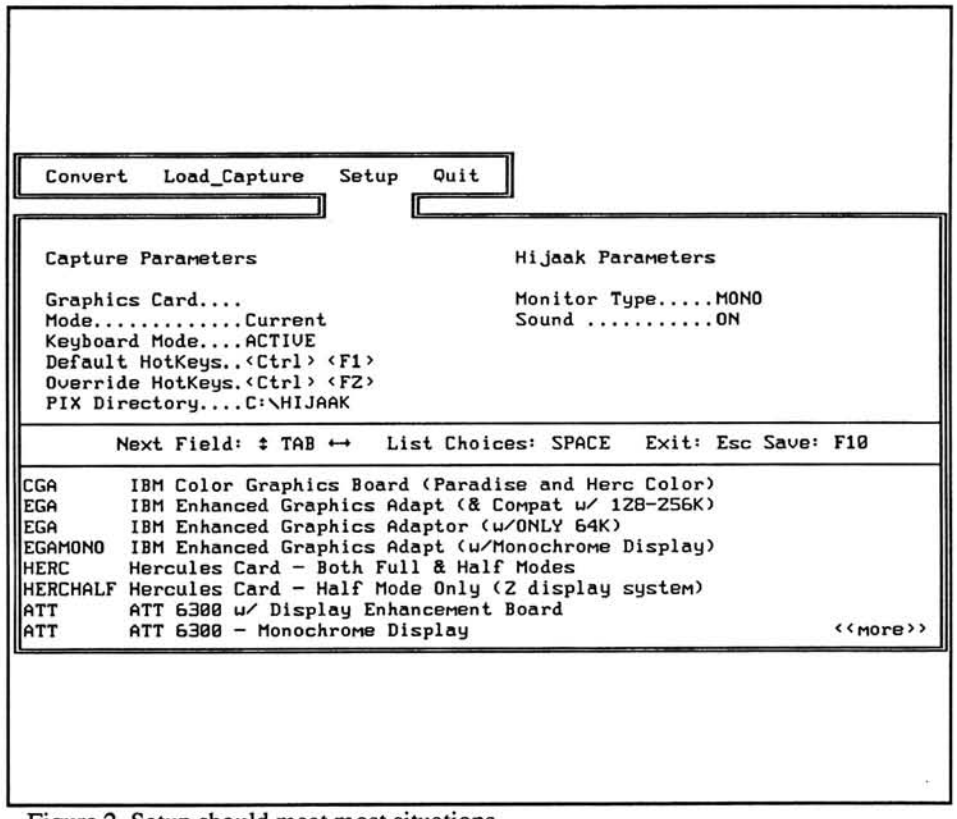


Figure 2. Setup should meet most situations.

Northgate's Omni Key/102 Keyboard

Northgate Computer Systems has discontinued its XT style 84-key CT/84 keyboard I praised last time. Also discontinued is Northgate's extended AT style 101-key CT/101 keyboard. Replacing both is Northgate's Omni Key/102 keyboard, which can be used with both XT and AT compatible microcomputers.

The Omni Key/102 is one of the most versatile keyboards available for IBM compatible computers. One of two switches on the underside of the keyboard allows it to be configured for either the XT or the AT machines. The other switch allows the CTRL and CAPS LOCK keys to be swapped: you get an extra set of key caps to make the swap official if you decide you like the result.

I made the swap, and made it official, because it results in the CTRL key being where my left pinkie expects. Unfortunately my left pinkie also expects to find ALT where CAPS LOCK winds up, so whenever I use a program like XyWrite III Plus (which makes heavy use of the ALT key) I find myself shifting into uppercase letters instead. Maybe I'll adapt in time.

But it took me no time at all to adapt to the availability of the one hundred and second key, which gives this keyboard its model number. Look at the photograph in Figure 3 and you'll see that there is an "=" key on the numeric keypad. (But ignore the dual legends on the CAPS LOCK and CTRL keys: someone at Northgate exercised artistic freedom to make a point.) That "=" key turns the numeric keypad into a true calculator keypad, and the dedicated arrow keys to the left of the numeric keypad make it a truly usable calculator keypad. Press the NUM LOCK key on the Omni Key/102 and you can do numeric data entry at your own top speed.

I also took immediately to the unusual placement of the F11 and F12 keys on the Omni Key/102. Look at Figure 1 again and you'll see them off to the left of the keyboard, in the top row, above the bank of standard function keys. Well Northgate is absolutely right: the two additional function keys *are* additions and ought to be tucked off into the corner. That way they're accessible when required by some rare program that makes use of them. At other times, because so few programs do use the F11 and F12 keys, they don't interfere with fingers that remember the standard locale of F1-F10.

My one gripe about the Omni Key/102 is the displacement—for me—of the ALT key. I really want it where my left pinkie expects it. Other than that, I like the Omni Key/102. You ought to grab one if your work involves the kind of number manipulation that makes you long for a real calculator keypad on your computer.

Mainstream Mailbag

"What I've lost is confidence in Zenith's compatibility with IBM," wrote Tim James in a long letter recently. Mr. James, an engineer in South Deerfield, Massachusetts, has a Z-248 with the following: Seagate 4038 hard disk; NEC Multisync monitor; Vega Deluxe EGA board; Mouse Systems Bus Mouse; Zenith's MS-DOS 3.1; and a Cheetah Combo/80 board with 1 MB of 80 ns DRAM, of which 128 KB is used to backfill the 512 KB of base RAM to the 640 KB directly addressable by conventional programs. "I've been experiencing memory parity failures often (even while writing this), so I disabled VDISK.SYS and the problem seemed to go away. Does that imply the Combo/80 may be bad or is it the driver Vdisk? What I've lost is confidence in Zenith's compatibility with IBM. Everytime a lockup occurs I'm asking 'Is it the Z-248 or just a software bug?'"

Mr. James notes that I wrote about the Cheetah Combo/70 with a similar setup in the September 1987 "Mainstream

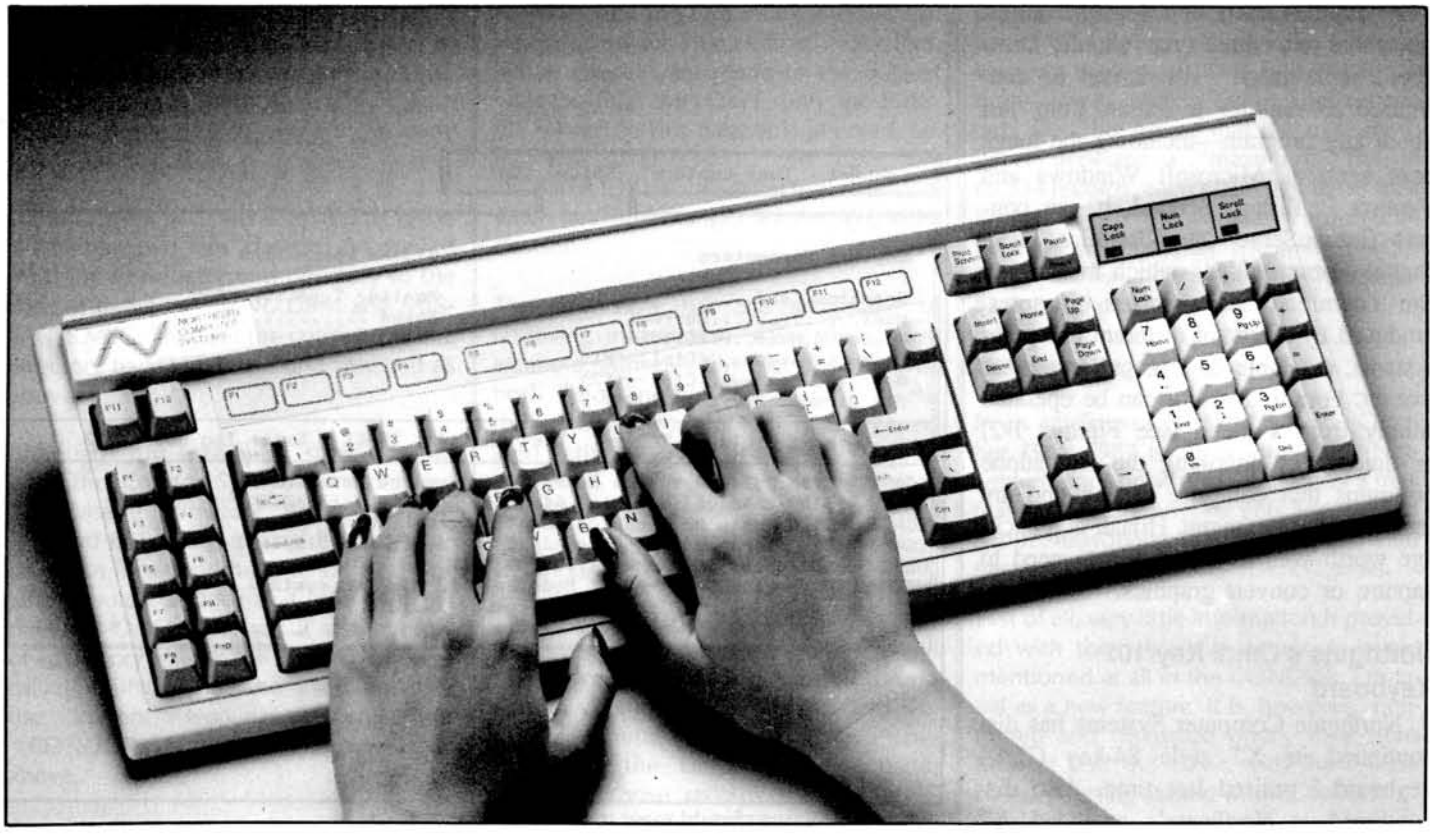


Figure 3. Northgate's Omni Key/102.

Computing" and thought I might be able to help because Zenith's technical support couldn't. Of course I'm glad to try.

Dear Mr. James:

You have every reason to sound frustrated. I sympathize. As an engineer you know that long distance diagnosis of a computer problem is hard to do and not necessarily accurate. Memory parity errors can be especially tough. But let's see what we can do anyway.

By now you should have put a fresh copy of VDISK.SYS on your hard disk—with the "verify" switch to make sure the copy is a true duplicate of the original on your MS-DOS distribution disk. (The command is: `COPY A:VDISK.SYS C:/V.`)

My real bet from what you say is that there's a bad RAM chip in your Cheetah memory board. I bet, too, that it's in the "HIGH" segment of Bank 1 or in Banks 2 or 3. And I bet, furthermore, that it's one of the chips in the row marked "PH" (the bottom row) on the board. In a moment I'll explain why I think so.

Power down the computer. Pull the Cheetah and set its switches to 256KB less RAM than is actually on the board—but don't pull any RAM chips. Then see how the computer works, VDISK and all. (When the Z-248 complains about the discrepancy between the information in Setup and the actual RAM, hit the ESC key and go on.) If your problems have disappeared, the bum chip is in "HIGH" Bank 3 and, as I've said, probably is the bottom chip in the bank. If the problems haven't gone away, tell the board it has less RAM by another 256KB and try again. Keep on that way until the problems disappear. The bank right before then is the one with the bad chip. Replace it with an 80 ns chip.

The reason why I think your problem is a bad RAM chip on the Combo/80 is that your errors take place only when the Vdisk driver is in operation. That's why I don't think the trouble is in the "LOW" segment of Bank 1, which is where your backfilled RAM is located. The reason why I think your bad chip probably is one of the bottom chips is because those are the parity chips on a Cheetah: the ninth chip in each bank is used for parity checking.

Nope, I can't be absolutely sure that my diagnosis is right. As I began by saying, long distance diagnosis is hard.

Fortunately, your equipment—all of it—is absolutely first class, from absolutely first class manufacturers. If you need more help than I've given you here, give Cheetah International a call and enlist its aid: 1-800-CHEETAH is the phone number. I'm sure you'll be pointed in the right direction.

No, I can't explain the ratings difference between your system (Norton's SI showed 7.8) and mine (SI shows 9.2). You understand that those ratings are not awfully significant. But, yes, your system is close enough to mine so it ought to score about the same as mine with the same benchmarks. I don't know the Vega Deluxe EGA board, EGA boards can slow things down, and I don't remember what board was in my computer at the time I tested things. See what happens when you've cleared up the RAM chip problem.

Are you *sure*, by the way, that the jumper on your Z-248 really is set for 0 wait states? Try it the opposite of the way it's now set and see what happens. The wait state is Zenith's way to let

you slow down the machine so you can use slow third party boards. Remember that the 248 at 8 MHz with 0 wait states still is no slouch.

Take heart. You have friends.

See you later.

Products

WordStar 2000 Release 3 $495
MicroPro International
San Rafael, CA 94903
415/499-1200

Omni Key/102 Keyboard. $99
Northgate Computer Systems
2905 Northwest Blvd., Suite 250
Plymouth, MN 55441
612/553-0111

Hijaak. $99
Inset Systems
12 Mill Plain Road
Danbury, CT 06811
203/794-0396

*

ENABLE 7

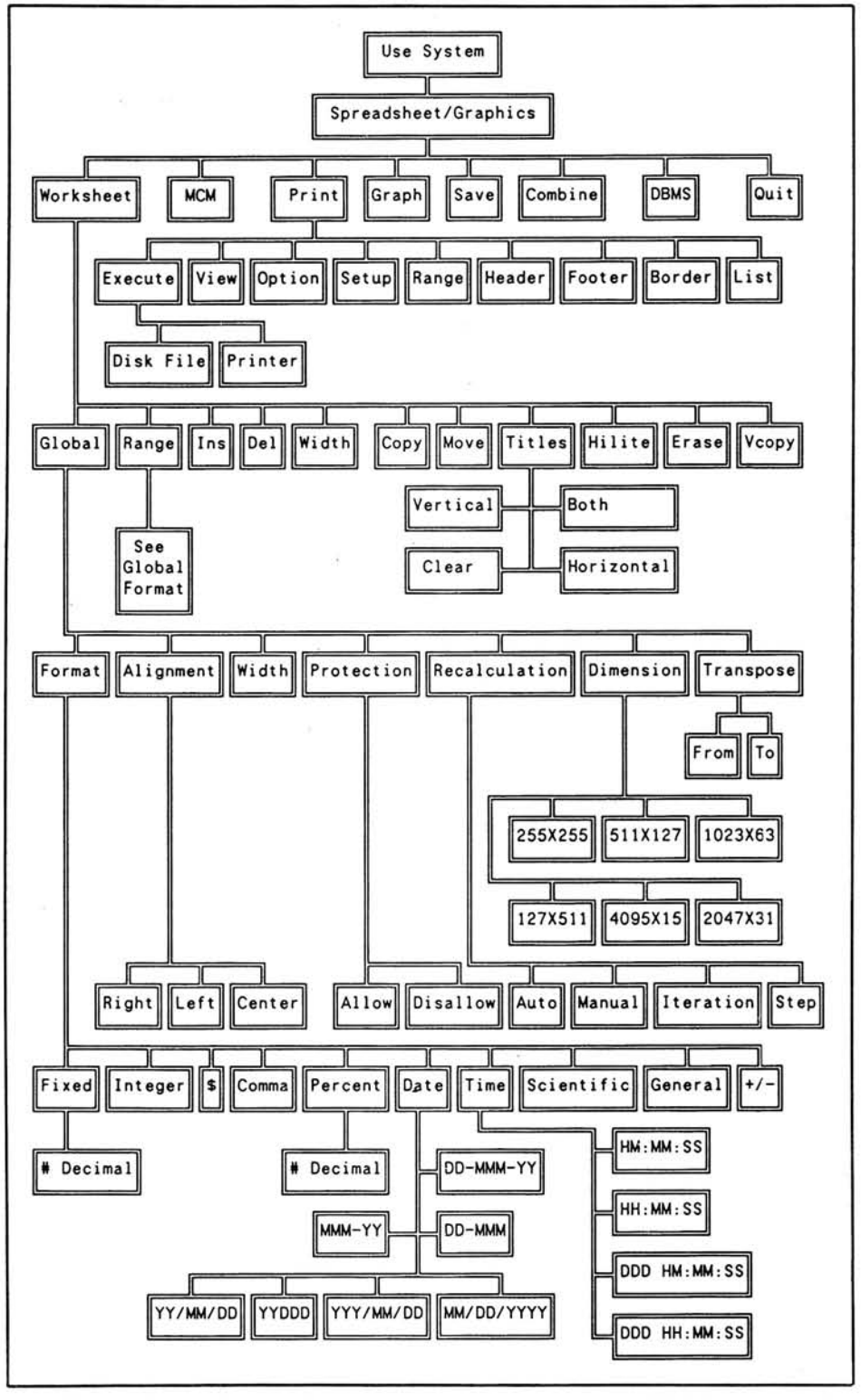
A Tutorial

George P. Elwood
1670 N. Laddie Court
Beavercreek, OH 45432

In this, the seventh in a series on ENABLE, an intermediate discussion on database management will be the topic. Of all of the modules in ENABLE, I like the database the best. It is easy to build entire databases including input screens and reports in a very short time. In this article I will continue to develop a sample order form database. This program can be used to order Z-248's for Department of Defense organizations, at least those using the Air Force procedures and forms. This article is based on the Z-100 version 2.0. Version 1.15 that is on the Z-248 contract does not have all of the capabilities described in this article while the commercial version 2.0 is the same. I will make note of those features not available in the DoD version 1.15. I will also continue to point out the different key strokes required in the Z-100 and PC versions.

ENABLE has been selected by PC Magazine as Editor's Choice in March 1988 which continues to point to the outstanding product that the Software Group has developed. At HUGCON, the Software Group was presented an award for the development of ENABLE for the Z-100. They continue to develop enhancements to the product to make it even better. There are several product improvements awaiting the Z-100 version if continued interest is shown in the product. The Software Group will only continue to improve our Z-100 version if we buy and write letting them know we care. As possibly the last major software product for the Z-100, we must support the vendors who support us. I like my Z-100 and use it over the H-386 sitting next to it and so I make my desires known.

If you plan on using ENABLE, you must make the effort to learn the system. The key strokes are different than other programs, but once learned, you will be able to do everything and more than you could do with your former product. I have been asked about using KeyMap to remap the keys so that ENABLE will act like WordStar. I firmly believe that be-



cause of the common key strokes throughout the various modules, that remapping the keys will make it harder to use the entire package as it was designed. In one inquiry, the individual was having problems with the remapped keys. Since I do not use this program, I could not answer his questions. If you have used a key remapper with ENABLE please let me know how it works and what problems you encountered and how you solved them. I will pass these tips on in future articles.

In the first part of the database article, the basic database for ordering Z-248 computer systems and other products on the DoD contract was presented along with the input screen. The order system will require another database. This next database will hold the actual orders for the computer hardware and software. It will use both the order database that was established in the first article on database and the new database. From this database will come the printed forms required when ordering.

To enter the database module from the main menu, press (U)se System, (D)BMS/ Graphics, (D)esign, (D)ata Definition. Type in "ORDERZ" as the name for this new database and a <RETURN> when prompted if this is a new database. On the next screen type in "Z-248 order database" for the data definition. Remember this is used if you print out the database definition. Again, remember that printed definitions are not available in Version 1.15. In the next block, which is the default input block, type in "ORDERZ." If you plan on having this form reside on another disk or in another directory, this information must be placed in this block along with the name so ENABLE knows where to find the form. The last block on this screen is the default report. For this database, type in "ORDERZ.RPT". This report will be built using the word processor later in the article. A <RETURN> now will bring up the basic help screen before the actual database is built. A second <RETURN> will bring you to the database definition screen which we used in the first dbms article.

The first field in the ORDERZ database is CLIN, or contract line item. This field is an external field that is brought into the database from the ORDER database. The data is not physically moved or copied into the ORDERZ database but the information is available from the ORDER database which saves space. This is pointed out when you look at the middle of the screen where the field data is displayed. When this database definition is completed, the Total Fields are eight while the real fields are three and in the same token the Total Length is 61 spaces while the Real Length is 13. This graphically shows the savings in using this type structure. Type in CLIN for the name of the field and select (D)etail description. Type in (N)o for the next question, copy the definition from another. The field is not indexed, so another (N)o. This field is required so (Y)es is the next response. The source of information is (A)nother database. When this is selected, another screen is presented with more required responses.

database, type in "CLIN". The third question is for the index field in the "other" database. This field must be indexed so ENABLE can get the correct information from the other database. Again type in "CLIN". The last question is the linking field in the new database to the other database. Type in "CLINX" for this response. You now can enter the columnar heading and error message if desired. ENABLE will now copy the definition from the other database to ORDERZ, CLIN field. This procedure is used when you want to use data that is in another database. If you need to use the data in a report, there is another method that can be used. This will be covered in a later article.

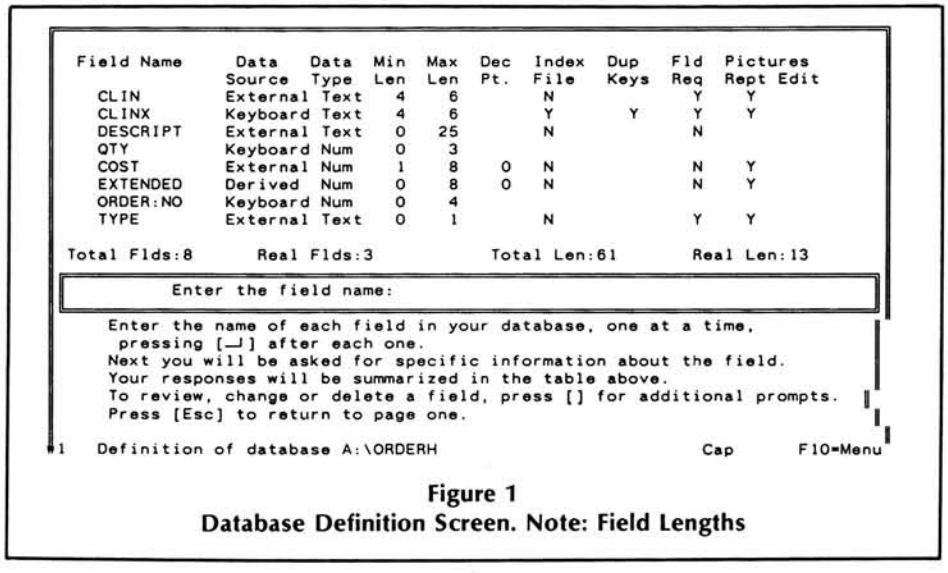


Figure 1
Database Definition Screen. Note: Field Lengths

The first question is the name of the database with the information required. Type in "ORDER". The second question requests the lookup field in the other

The next field is called "CLINX". This is the field that will be used to enter the CLIN or contract line item. "CLINX" is typed in with a <RETURN>. This field re-

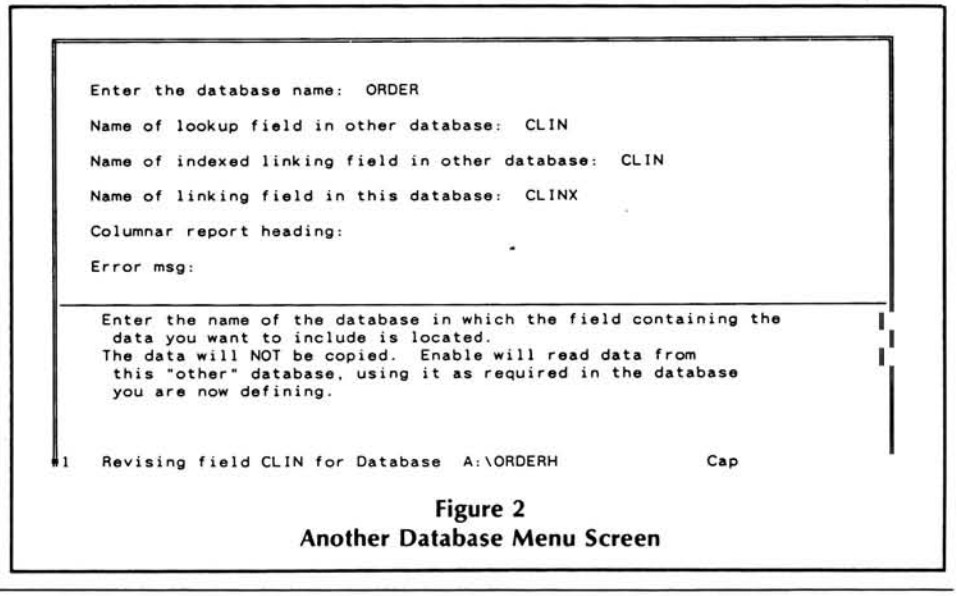


Figure 2
Another Database Menu Screen

quires a (D)etailed definition. We will not copy the definition from another database and it will be indexed. I selected "CLINO" as the name for the index name. If you have several indexes on the disk, remember that each one must be different. If you use the same name, i.e., LNAMEX, for all of the databases, no data will be indexed as the file will contain erroneous data. I make it a point to name the index file with the first letter of the database, the field name and end it with an "X". You can choose whatever works for you but they all must be different. Even though the field is indexed, duplicate CLINs can be entered, so select (Y)es for duplicates permitted. Because this field is required, select (Y)es for the question if required. ENABLE will not permit you to skip over this entry if YES is selected. The last question on this screen requests the source of the data. Select (K)eyboard as the response.

The next screen requests additional data for the input. The CLIN is a number/letter combination, so select (T)ext as the type. (A)nything is selected as the specific type of data. This field has (N)o minimum or maximum value. It also has (N)o acceptable data. The minimum field length is 4 (0003) and a maximum field length of 6 (0001AB). This completes this screen.

The next screen is the last in the detailed data definition. The first question is the report picture. You can provide the picture of the data in the report form if it is specified here. I input NNNNAA as the response to this question. This means that the first four characters in the field are number and the last two are capital letters. The second question requests the report heading. If nothing is placed here, the columnar heading will be "CLINX". If you want something different, this is where that is specified. The last question is the error message. Use this to display a message of your choice on the status line during data input if the parameters as specified are not input. I use this line when I write programs for other people, especially date input which requires a "YY/MM/DD" form. This completes the data definition of CLINX.

The next field is "DESCRIPT" and consists of textual data relating to the CLIN. Again, like CLIN, I choose to get this data from the ORDER database. This field is (D)etailed, the definition is (N)ot copied from another database, (N)ot indexed, (N)ot required, and is from (A)nother database. The next screen requests the information on how to find the correct data and is the same one we saw in the CLIN definition. The information comes from the "ORDER" database in the field called "DESCRIPT". The two databases are linked using CLIN as the indexed field in both the other and this database.

The next field is "QTY". This is a (Q)uick definition. The field is an (I)nteger with a length of "3". This is all of the information that is required for a quick definition.

The next field is "COST" and the information is located in the ORDER database. The same steps that were described for the field "DESCRIPT" are used except that the field in the other database is "COST".

The sixth field in this database is "EXTENDED". This field is different than those we have seen before in that it is derived from other fields in the database. The field is (D)etailed, (N)ot copied from another database, (N)ot indexed, (N)ot required, and is (D)erived. The next screen requires the formula to get the answer. The formula is "QTY*COST". Anything can be used in this block including the "@" functions described in the last article on ENABLE's spreadsheet. Either numbers or field names can be used in the formulas. You also may concatenate two or more fields to form another field. The result in this example is a (N)umber with (0) decimal places, and a length of 8. It has a report picture of $NN,NNN and uses the default columnar heading.

The next field is "ORDER:NO". This field is a (Q)uick definition, (I)nteger, with a length of 4. In this example, this number is used as part of the complete order number. In the Air Force base supply ordering system, the last four digits are unique to the order and are assigned by the requester.

The last field in this order database is TYPE. I added this field to show hardware or software items, each of which have to be listed separately. This field is (D)etailed, (N)ot copy protected, (Y)es is required, and comes from (A)nother database. The order database is ORDER and TYPE is the lookup field in the other database. CLIN is the indexed linking field in the other database while CLINX is the linking field in this database. Now press F10 and (S)ave to save this database to the disk. Press F10 (Q)uit and you are back to the ENABLE Main menu.

The linking of fields in other databases makes ENABLE flexible when it comes to design of new databases. From the short examples above, sharing of data has been used by linking fields between the source of the field and the end user. As has been pointed out, the source field is not copied into the new database but is retrieved from the sources when required. This saves space in the database file on the disk. Remember that ENABLE has two files with each database, the first is the definitions (XX.$BF) where all of the parameters are stored and the database data in the XX.DBF file. The database source does not have to reside on the same disk as the new database but it must be available on disk in the machine. When you define the source database you must indicate the location of the file including directories if necessary.

This completes the second database required for the order system. To make this input friendly, an input screen will be used. Again from the main menu select (U)se System, (D)atabase/Graphics, and (I)nput Form. The name of this input form will be "ORDERZ" and again you will be prompted to indicate "New File", "List Dir", or "Re-Enter". The database is "ORDERZ". If you are designing your own input form and you do not remember the name of the database, you can place a "?" in the block and all databases that are on the work disk/directory will be displayed. This procedure is the same throughout ENABLE. The next screen prompts you for an input form description, which is optional and is not displayed or printed anyplace. If you have a color monitor you can now specify the screen color and character color on the screen. You can make additional color selections later during screen development. Next comes a screen with basic instructional information and after a <RETURN> you are presented a blank screen and are in the word processor. All of the functions in the word processor are now available so bolding, italics, and underlining can be used. As indicated in the first article, ENABLE uses a "Put-it-here" capability for input screen development. This makes screen development very easy when compared to dBase II.

Continuing with this screen, type in "Order Number" in the upper part of the screen. Leave a space and then press SHIFT/F9 to indicate a field location. If you do not know the name of the fields, put a "?" in the prompt block and press <RETURN>. All fields that are in the

database will be presented. In the ORDERZ database is the field "ORDER:NO" which should be highlighted by moving the cursor over it with the arrow keys and a <RETURN> entered. From the next screen, select (Y)es for special processing. The first prompt on the next screen asks for screen color which means that you can change the color of the background screen from that selected when coming into the input form menus. If selected, this would change the input block on the screen to the designated color. Character color for the input can also be changed with the next prompt. If you are brave, try painting here. By selecting a second and third color, a "new" color can be obtained. Remember your color wheel and do this on a practice file first. The next prompt permits you to require double keying entries to ensure accuracy. The last prompt on this screen will be selected as this will copy the order number to the next input during data entry to save key strokes. The first prompt on the next screen permits you to protect the field from updates. The next prompt, if selected, will not permit the cursor to move from the field until <RETURN> has been pressed. Normally ENABLE will move to the next field when the input data block has been filled. The last prompt is for default data. This permits you to specify data that would normally be typed in the block, like a city. Again this is a method to save key strokes during data entry. The first prompt on the last screen in the special processing menu is special message for invalid data. This is similar to the Error Message capability provided for during database development. The next prompt permits you to specify unique movements within the input form based on a "GO TO field IF" option. You would use this to skip many fields if a certain condition is met again saving key strokes during data entry. The last option is for a macro procedure. ENABLE has a great macro capability in all modules and this is one way to invoke one of these procedures. Macro procedures will be covered in the advanced sections of this series.

Moving down the screen, type in "Contract Line Number", a space and SHIFT/F9. "CLINX" is the name of this field. Next type in "Item Description" a space, and SHIFT/F9 "DESCRIPT". QTY, COST and Total Cost are also input in a similar method. The Total Cost field is EXTENDED and from the database definition is the result of QTY * COST. Once all of the data is entered, press F10, (F)orm Design Options, (S)ave and <RETURN> to save your work. F10, (F)orm Design Options, (Q)uit will return you to the main menu. Note that all of the word processing menus are available with an additional step through the top line menu.

Now that both database definitions are completed and the required information has been placed in ORDER, the actual ordering process can begin. From the main menu select (U)se System, (D)BMS/Graphics, and (I)nteract. This now displays ENABLE's DBMS basic menu. Like all of ENABLE, the options can be selected by typing in the first letter or number or by moving the highlighted area with the cursor to the required selection and pressing <RETURN> or <ENTER>.

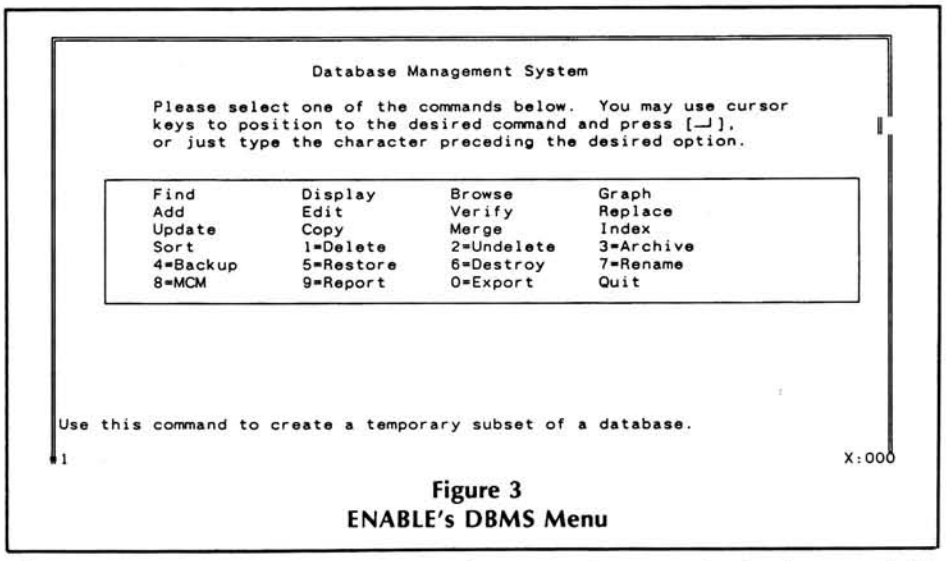


Figure 3
ENABLE's DBMS Menu

(A)dd is selected to input data to this new database. If you had checked the directory before this step, there would not have been any .DBF file, only the .$BF file. After selecting ADD, ENABLE takes you to another screen where you select additional options for the entry of data. NOTE: This is the same procedure used in the first part of the database article using the (B)uild routine. The first prompt is for the name of the database. Again, you can just put a "?" in this block and a <RETURN>. The first database to fill in is ORDER. This contains all of the actual information on the items and is input from the contract. Select ORDER from the listed databases. The next prompt is for the input form. Again from the first lesson, "ORDER.$IF" is selected. ENABLE will now display the input screen and indicate on the 25th line (STATUS LINE) the RECORDs in the database, highest Record selected, Record Selected, and the current record displayed. When you start, all of the information should be "0". As you enter each record, the numbers will increment by one. When all of the data is entered, press "F10" (S)ave and F10 (Q)uit. This will ensure that the data is saved. It is a good idea to save data frequently during entry to save what you have completed in case of trouble.

To start inputting data for the actual order, from the ADD screen, select ORDERZ as the database and ORDERZ as the input screen. The first field is the ORDER:NO. This will be the order number that is printed on the order form. This number will not have to be repeated as "automatic carry over" was selected during the input form design. You must have a list of items on the contract in order to continue. From the list, select your first item and put the CLIN number in the next field. ENABLE will then go to the ORDER database and provide the item description and cost. When you enter the quantity, the total cost will be calculated (Remember "COST* QTY"). Continue through your order with an F10 (S)ave F10 (Q)uit when you are done. All of these key strokes are the same for both the Z-100 and the PC. This completes the basic database development of the order system. I have included with this article a copy of the two databases described in the last two articles on disk. The set includes all of the input screens, menus, macros and the report form to make this program work. To start the program, all you have to do is type SHIFT/F0 F10 O (CTRL/F10 O for the PC) and the opening menu will be displayed. The only difference is that I have modified the database to include hardware and software on the three Zenith/DoD contracts, the Z-248, Z-184 lapheld and the TEMPEST Z-200T. You must identify the contract with a

Z,z,L,l, or T,t during order entry. The actual development of these functions, which make ENABLE a great package, will be covered in both the advance series of articles and follow-up article which will provide additional capabilities.

Before I leave the database discussion, I will go over some of the capabilities of the dbms menu. From this menu, 24 database functions can be performed. These commands, when used with the macros, provide a very complete and powerful set of tools to do just about anything you could think of to a database. Within each command are more options which provide even greater flexibility.

The first command is "FIND". This command permits you to find records which meet specified parameters. The first prompt is for the name of the database. The second prompt is for an index. In a window in the bottom third of the screen, ENABLE will list all indexes that are available for the database. You can also specify index search at this time, i.e. CLIN="0001A". An index file is created for each indexed field. This index file is used when requested. Index is like a sort except that the records are not physically rearranged. If you have sorted a lot, you know how long it takes to complete the process. Index is faster and you can have multiple indexes in a database. The results come out the same. If you do not have an index you wish to use, a <RETURN> will move you on to the last prompt, the Where Clause. Again ENABLE provides a list of all fields in the database. Also displayed are the operators that can be used in conjunction with the Where Clause. These operators are "< > = <= >= <> GT LT EQ LE GE NE + - * / ** () & &-" and they are the same for all of the menu choices in the database menu. The Where Clause can have multiple statements on this line. The Where Clause is limited to 145 characters with the line scrolling to the left. A multiple Where Clause would read CLIN="0001A" and contract="T". You must identify the field type as number without the quotation marks or string with quotation marks. Wild Cards can also be used in a Where Clause. A "?" can be used if you are not sure of one character or the "$" for an unknown number of characters. The "?" could be used in a date if you were not sure of the day by typing in "88/04/??". ENABLE would then select all records from the fourth month of 1988. Similarly, if you were not sure of the exact spelling, you would place type "B-$" to select records that relate to bomber aircraft, i.e., B-1, B-52, B-52H, B-47 while the FB-111 would not be selected. When you have made your selection, ENABLE will look through all records in the database. You can watch this as the numbers on the status line increment up. For every record that matches the Where Clause, one number will be added to the selected counter. All records that meet the Where Clause will be put into a select set database which you can review or use with other menu choices. These records are identified by the database name and .SS extension.

Index. If you are using a new database, all of the available indexes are listed. If you are using a select set, ENABLE passes over this response. The Where Clause is next and it is similar to FIND. Again the operators and fields are listed for your assistance. The last question is Fields. If you only want a few fields, you can specify the fields at this time by simply typing in the name of the field separated with a coma. You can also provide a format for the display by using "{$}" to specify dollars, "{n}" to specify the number of digits in the field, "{n.n}" to specify decimal places, and "{<}", "{>}" , or "{^}" to specify left, right, or centered alignment. Note, these display formats can also be set during the development of the database. If you simply press <RETURN>, all fields will be displayed on the screen, 78 columns per screen. TAB and SHIFT/HELP (TAB and SHIFT/TAB on the PC) can be used to move right or left on the screen along with the normal cur-

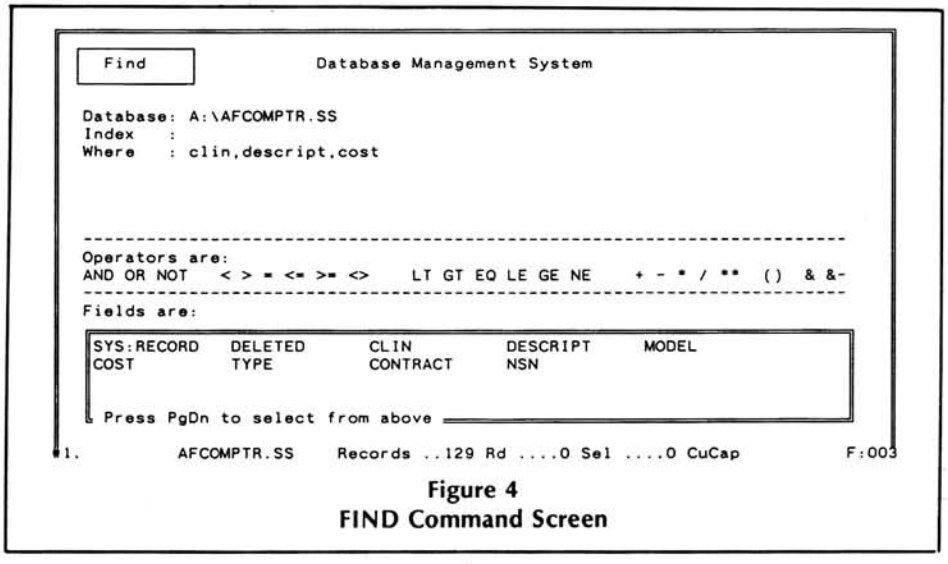


Figure 4
FIND Command Screen

To view the results of the FIND, use DISPLAY. You can also use this command to mark records for deletion, archival or moving to other modules. Again, from the dbms menu, select (D)isplay. If you are using an entire database, type in the name or a "?". If you are using the select set as defined in the find above, the name is carried forward. The next prompt is for

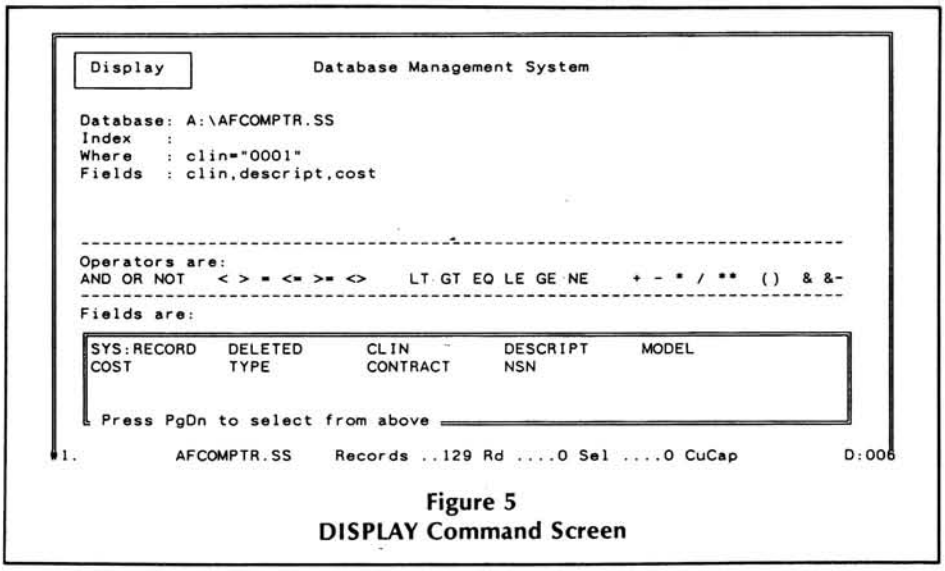


Figure 5
DISPLAY Command Screen

sor movements. SHIFT/3 or SHIFT/9 (PgDn or PgUp on the PC) can be used to move up or down in the display. To mark fields for further operations, position the cursor on the first line and press F7, the same as word processing marking text. The line will be displayed in reverse video. You can continue marking as many of the records as necessary.

I have now scratched the surface in ENABLE's database. There are more functions to cover and they will be detailed in the next dbms article in three months. The dbms functions are easy to use and I will spend more time developing these capabilities. Coming up will be payroll systems and thoughts in designing these systems.

*

Capital Heath/Zenith Users' Group

Announces Major User Conference

The Capital Heath/Zenith Users' Group (CHUG) has announced its Seventh Annual International Microcomputer Conference and Exposition, CHUGCON '88. The conference and exposition will be held at the Hyatt Regency Hotel in Crystal City, Virginia on Friday July 29 and Saturday July 30.

CHUGCON is a successful, well attended conference, allowing microcomputer users and hobbyists to see the products currently being discussed in trade journals and user group meetings. Tutorial sessions, scheduled throughout the conference, assist both new and experienced users in understanding the strengths and weaknesses of the latest products.

Over 120 booth areas are expected to be filled for the exposition. Attendees will have continuous access to the vendor area from 9:00 AM to 5:00 PM on Friday and Saturday. A banquet will be held Saturday evening, starting at 7:00 PM. Vendors will be permitted to sell their products from the booths. Attendees can expect to receive significant discounts from standard store prices.

One of the more "fun" aspects of CHUGCON is the hourly door prize drawings. The same advanced products that draw crowds have been contributed, by their vendors, to be given away. Every registered attendee has the chance of walking home with free hardware or software. A champagne tour, as well as other sight-seeing activities, have been planned for convention attendees.

Entry to the conference and exposition will cost $5.00 at the door. However, many local stores and computer related businesses will be distributing passes to their customers. CHUG is actively soliciting the involvement of local business people in "spreading the word".

Special assistance has been arranged for the hearing impaired. Advance registration forms will allow the conference management to assure sufficient aides for attendees.

A computer bulletin board system is available to answer inquiries, or to request free conference passes. It may be reached by calling (202) 797-7236 (metro DC) or (703) 339-9856.

For further information, contact CHUGCON at:

P.O. Box 16406
Arlington, VA 22215
(703) 339-9857

Fixes For ZPC

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

There are two problems with ZPC Version 2.1, one related to the use of the Ctrl-minus key combination by programs such as Microsoft Word, and the other to template editing of input lines (using DOS functions 0AH and 3FH).

The first problem was discovered while trying to use Microsoft Word's non-required ("soft") hyphen feature. On a PC, this is achieved by using Ctrl-minus, but with ZPC version 2.1, neither Ctrl-minus nor F0-minus works. This problem was fixed by changing KEY.ACM to allow the F0-minus combination to be used. While I was at it, I also enabled F0-] (to make it similar to F0-[), and F0-^, F0-\out64\ and F0-\out92\ (to round out the set).

The second problem is actually two smaller problems. The first of these is that DOS function 0AH doesn't work properly at all, and the second is that function 3FH acts incorrectly when the user's buffer is too small to hold the data. To illustrate the first of these, try the following in "native" mode on either a PC or an H/Z-100:

```
A>@edlin junk.jnk[RETURN]@
New file
*@i[RETURN]@
        1:*@line 1[RETURN]@
        2:*@[CTRL-Z]@<^Z>@[RETURN]@
*@1[RETURN]@
        1:*line 1
        1:*@[F3]@<line 1>@A[RETURN]@
*@[F3]@<1>@[RETURN]@
        1:*line 1A
        1:*@[F3]@<line 1A>@[RETURN]@
*@q[RETURN]@
Abort edit (Y/N)?@y@
A>@[F3]@<edlin junk.jnk>
\np
```

Now try a similar sequence under ZPC 2.1 in PC mode:

```
A>@edlin junk.jnk[RETURN]@
New file
*@i[RETURN]@
        1:*@line 1[RETURN]@
        2:*@[CTRL-Z]@<^Z>@[RETURN]@
*@1[RETURN]@
        1:*line 1
        1:*@[F3]@<1>@st line[RETURN]                    !!!!!@
*@[F3]@<1st line>@[CTRL-X]1[RETURN]                     !!!!!@
        1:*1st line
        1:@[F3]@<1>@[RETURN]                            !!!!!@
*@q[RETURN]@
Abort edit (Y/N)?@y@
A>@[F3]@<q>                                            @!!!!!@
```

Every single time F3 (or any of the other DOS template editing keys) is used, the results are unexpected.

The reason for this was discovered by reading file DOS.ACM. The code in this file includes a template editing routine READLN, which is called under two conditions: (1) when DOS function 0AH is invoked; and (2) when DOS function 3FH is invoked for stdin, i.e., with BX = 0 (in both cases, in PC mode only). This routine uses a template buffer TEMPLT for two purposes: (a) as a template during line input, to support the DOS function-key editing commands; and (b) as a "memory" for the last line input. While this simulates the operation of DOS function 3FH (approximately), it does not do so properly for function 0AH.

DOS function 0AH uses the original contents of the user's buffer as a template during input, and it does not "remember" the contents of the template between calls. Thus, a program may have several different templates, switching among them by the choice of DS:DX at the time of each DOS call. EDLIN obviously makes use of this feature, and many other programs also use function 0AH for line input from the keyboard. Among them are COMMAND.COM, any program which can be used under DOS 1.x, and many others. For C programmers, the "cgets" function in Turbo C is basically a DOS function 0AH call.

To fix up ZPC so it handles these function 0AH calls correctly, we must modify DOS.ACM to copy the contents of the user's buffer into TEMPLT before processing keyboard input (and not after, since we don't need "memory"). The program uses TEMPLT for the template, and the user's buffer for storage of the new line as it is being assembled (these roles could be reversed, but this would require more extensive changes).

While we're making this change, there are a couple of others we should also make. First, DOS allows the buffer for function 0AH to be up to 255 bytes long, so the size of TEMPLT should be increased to 255 bytes. (Note that COMMAND.COM's template buffer is only 128 bytes long, but other programs can and do use longer

buffers.) Second, DOS does at least two checks on the user's template to "validate" it. The first character in the user's buffer is "length" (from 1 to 255); the second is "count" (from 0 to "length"-1); and the next "count" characters are text, followed by a carriage return character, for a potential total of "length"+2 bytes. DOS checks that "count" is less than "length", and that there is a carriage return after the "count" bytes of text. If either of these checks fail, the template is considered to be empty ("count"=0). The READLN routine should perform both of these checks for proper simulation of DOS function 0AH.

These changes also affect the simulation of DOS function 3FH ("handle" or "Xenix" input; in C, the "fgets" function). In fact, except for cases where the user's buffer is too short to hold the entire input line, the original version of ZPC simulates this function correctly, while a version modified as above does not. DOS function 3FH, when reading from the keyboard, behaves exactly as if DOS kept its own private template buffer (with "length"=128), invoked function 0AH internally, and then transferred the text contents of its private buffer to the user buffer (followed by a CR-LF pair). It ignores the original contents of the user buffer, and "remembers" the template from the last function 3FH call, no matter how long ago.

To fix ZPC in such a way that all programs work properly (including those which intermix function 0AH and 3FH calls), the XREAD routine in DOS.ACM must be modified. A new template buffer is needed, which XREAD will use when it in turn calls READLN. This buffer must be separate from TEMPLT so that its contents will not be overwritten during intervening calls to function 0AH. After the input of a line is finished, XREAD copies the text from this private buffer into the user's buffer, including the trailing carriage return and an extra trailing line feed. The contents of the private template buffer are saved for the next call of XREAD.

There is yet one more change needed to XREAD. When the user supplies a sufficiently large buffer to handle the input line (a length of 129 will cope with the worst possible case), DOS function 3FH works as described above. However, when the user buffer is shorter than the input line, it behaves differently. DOS refuses to write beyond the end of the user's buffer, and simply transfers as many characters as it can. If any are left (even if only the trailing line feed is left), the next call to function 3FH does not cause a new physical read from the keyboard, but simply returns the remainder of the previous input line, or as much of it as there is room for. In the extreme case of a user buffer of length 1 byte and an input line of 127 characters followed by a carriage return, it takes 129 consecutive calls to the 3FH function to exhaust the input before the next line can be typed in. The original code in XREAD does not behave this way; it will blithely overwrite the end of the user's buffer and destroy whatever follows it in memory. The new version of DOS.ACM described here corrects this problem as well.

Below are the changes I have made to KEY.ACM and DOS.ACM, described in exactly the same way as in file ZPC.DOC on the ZPC Upgrade disk.

Changes to KEY.ACM (Version 2.1)

This section describes changes made to KEY.ACM to allow the user to type F0-<-> for Ctrl-<->, F0-] for Ctrl-], F0-^ (or F0-6) for Ctrl-^, F0-@ (or F0-2) for Ctrl-@, and F0-\ for Ctrl-\.

This change enables Microsoft Word users to use the non-required hyphen feature (Ctrl-minus), and lets Ctrl-[and Ctrl-] to be used in a parallel fashion.

Each modified section is listed including 3 unchanged lines before, and 3 unchanged lines after, each changed section of code.

```
        AND     AL,0BFH                 ;ELSE, UNSHIFT CODE
NUMLCK: MOV     BX,OFFSET KEYTBL        ;ASSUME NORMAL KEY TABLE
        CMP     AL,96H                  ;CONTROL KEY?

        *** CHANGED 22 NOVEMBER 1987 - RLF
;               TARGETS OF JUMPS CHANGED TO NOTALTJ
        JZ      NOTALTJ                 ;IF SO, SKIP NEXT CHECKS
        CMP     AL,95H                  ;ALT KEY?
        JZ      NOTALTJ                 ;IF SO, SKIP NEXT CHECKS
        IF      LEVEL1
        TEST    KBFLG,4                 ;TEST FOR CONTROL KEY
        ENDIF

*****************************************************************

        PUSH    CS
        POP     DS
        JZ      NSCTL                   ;CONTROL IS NOT SET
;       *** CHANGED 22 NOVEMBER 1987 - RLF
;               ONE LINE REPLACED TO ALLOW F0 TO BE USED WITH
;               ], -, @, ^ AND \ KEYS (AS WELL AS [).

        CMP     AL,2DH                  ;CONTROL-MINUS?
        JZ      CTLM
        CMP     AL,5FH                  ;OR CONTROL-UNDERLINE?
        JNZ     NOTCTLM
CTLM:   MOV     AX,0C1FH                ;YES, SPECIAL CODE
        JMP     SHORT CTLBX
NOTCTLM:CMP     AL,40H                  ;CONTROL-@?
        JZ      CTLA
        CMP     AL,32H                  ;OR CONTROL-2?
        JNZ     NOTCTLA
CTLA:   MOV     AX,0300H                ;SPECIAL CODE
        JMP     SHORT CTLBX
NOTCTLA:CMP     AL,5EH                  ;CONTROL-^?
        JZ      CTLU
        CMP     AL,36H                  ;OR CONTROL-6?
        JNZ     NOTCTLU
CTLU:   MOV     AX,061EH                ;YES, SPECIAL CODE
        JMP     SHORT CTLBX
NOTCTLU:CMP     AL,5DH                  ;CONTROL-]?
        JNZ     NOTCTLR
        MOV     AX,1B1DH                ;YES, SPECIAL CODE
        JMP     SHORT CTLBX
NOTCTLR:CMP     AL,5CH                  ;CONTROL-\?
        JNZ     NOTCTLS
        MOV     AX,2B1CH                ;YES, SPECIAL CODE
        JMP     SHORT CTLBX
NOTCTLS:CMP     AL,5BH                  ;CONTROL-[?
```

```
        JNZ     NOTCTLB                 ;NO
        MOV     AX,1A1BH                ;ELSE, SPECIAL CODE
        JMP     SHORT CTLBX

;***************************************************************

        PUSH    AX
        MOV     AX,0                    ;MAKE A ZERO CODE
        RET                             ;RETURN WITH IT

;       *** CHANGED 22 NOVEMBER 1987
;               JUMP ADDED FOR OUT-OF-RANGE ''JC NOTALT   ABOVE

NOTALTJ:JMP     SHORT NOTALT

NOTCBK: CMP     AL,80H                  ;NORMAL CHARACTER?
        JC      NSCTL                   ;YES, NO SPECIAL CONTROL CODES
        MOV     BX,OFFSET KEYTBL1       ;ELSE, USE SPECIAL TABLE

;***************************************************************

Changes to DOS.ACM (Version 2.1)

This section describes changes made to DOS.ACM to correct the
behaviour of the template editing routines.

Each modified section is listed including 3 unchanged lines
before, and 3 unchanged lines after, each changed section of
code. In several cases, the 3 lines before and after two changes
overlap, and these are treated like one big change.

;***************************************************************

        DB      'LPT2       ','AUX '
        DB      'COM1       ','PRN '
        DB      'COM2       ', AUX '

;       *** CHANGED 21 NOVEMBER, 1987 - RLF
;               XREAD BUFFER ADDED, LENGTH OF TEMPLT INCREASED

TEMPLT  DB      255 DUP (0)             ;TEMPLATE BUFFER
XBUFL   DB      128                     ;MAX. LENGTH OF XREAD BUFFER
XBUFC   DB      0                       ;COUNT FOR XREAD BUFFER
XBUFT   DB      128 DUP (0)             ;XREAD TEXT BUFFER
XBUFP   DB      0                       ;POINTER INTO TEXT BUFFER

INSFLG  DB      0                       ;INSERT MODE FLAG

;       INT 21H PROCESSOR

;***************************************************************

XREAD:  PUSH    CX
        PUSH    SI                      ;SAVE REGISTERS

;       *** CHANGED 21 NOVEMBER, 1987 - RLF
;               6 LINES IN ORIGINAL DELETED

        PUSH    ES                      ;SAVE ES
        PUSH    DS
        POP     ES                      ;ES = DS
        PUSH    DI

;       *** CHANGED 21 NOVEMBER, 1987 - RLF
;               4 LINES IN ORIGINAL CHANGED TO:

        PUSH    DS                      ;SAVE USER S DS, DX
        PUSH    DX
        PUSH    CS
        POP     DS                      ;SET DS:DX TO ADDR. OF XBUFL
        MOV     DX,OFFSET XBUFL
        TEST    XBUFP,0FFH              ;CHECK FOR INCOMPLETELY READ BUFFER
        JNZ     STORBF                  ;IF SO, DON T DO A READ THIS TIME
        CALL    READLN                  ;READ LINE FROM CONSOLE
STORBF: MOV     BL,XBUFP                ;OFFSET OF FIRST CHAR TO MOVE
        XOR     BH,BH
        MOV     AL,XBUFC                ;SET AX = REMAINING COUNT
        XOR     AH,AH
        SUB     AX,BX
        INC     AX                      ;INCLUDING CR
        INC     AX                      ;AND LF
        ADD     BX,OFFSET XBUFT
        MOV     SI,BX                   ;SI = ADDRESS OF FIRST CHAR
        CMP     AX,CX                   ;IS THERE ENOUGH ROOM?
        JA      NOROOM
        DEC     AX                      ;YES - HANDLE LF SEPARATELY
        MOV     CX,AX
        MOV     XBUFP,0                 ;CLEAR BUFFER POINTER
        JMP     SHORT XCOPY
NOROOM: ADD     XBUFP,CL                ;NO - BUMP BUFFER POINTER
XCOPY:  POP     DX                      ;GET POINTER TO USER S BUFFER
        MOV     DI,DX
        PUSH    CX                      ;SAVE NO. OF CHARS TO MOVE DOWN
        JCXZ    NOCOPY

        PUSHF
        CLD
        REP     MOVSB                   ;MOVE TEXT DOWN
        POPF

;       *** CHANGED 21 NOVEMBER, 1987 - RLF
;               4 LINES ADDED

NOCOPY: POP     AX                      ;AX = NO. OF CHARS MOVED SO FAR
        POP     DS                      ;RESTORE USER S DS
        TEST    CS:XBUFP,0FFH           ;ROOM FOR LF?
        JNZ     XRSTR

        MOV     BYTE PTR [DI],0AH       ;ADD LF
        INC     AX                      ;COUNT IT
        PUSH    AX
        MOV     CL,0AH
        CALL    COUT                    ;PRINT LF
        POP     AX
```

```
;       *** CHANGED 21 NOVEMBER 1987 - RLF
;               LABEL ADDED

XRSTR:  POP     DI                      ;RESTORE REGISTERS
        POP     ES
        POP     SI
        POP     CX

*****************************************************************

        PUSH    ES
        PUSH    CS
        POP     ES                      ;PUT ES HERE

;       *** CHANGED 21 NOVEMBER, 1987 - RLF
;               7 LINES IN ORIGINAL CHANGED TO:

        PUSH    CX
        PUSH    AX
        MOV     DI,OFFSET TEMPLT        ;POINT TO TEMPLATE
        MOV     BX,DX                   ;POINT TO USER S TEMPLATE
        MOV     CH,M                    ;GET LENGTH
        INC     BX
        MOV     CL,M                    ;GET USER S COUNT
        CMP     CH,CL                   ;IS COUNT < LENGTH?
        JNA     TEMBAD                  ;IF NOT, USER S TEMPLATE IS INVALID
        XOR     CH,CH                   ;ELSE, SET CX = COUNT
        INC     BX
        MOV     SI,BX                   ;SAVE POINTER TO USER S TEXT
        ADD     BX,CX                   ;CHECK FOR CR AT END
        CMP     M,ØDH
        JE      CROK
TEMBAD: MOV     CX,Ø                    ;IF NONE, TEMPLATE IS INVALID
        JMP     SETTEM
CROK:   PUSH    CX                    ;ELSE COPY USER S TEMPLATE
        PUSH    DI
        PUSHF
        CLD
        REP     MOVSB
        POPF
        POP     DI
        POP     CX
SETTEM: MOV     SI,DI
        ADD     SI,CX                   ;SI POINTS TO END OF TEMPLATE CHARS
        MOV     CS:TEMCNT,CL            ;SAVE TEMPLATE COUNT

READØ:  MOV     CS:INSFLG,Ø             ;CLEAR INSERT FLAG
READ1:  MOV     AL,CS:COLUMN            ;GET COLUMN NO.
        MOV     CS:STRTCOL,AL           ;SET STARTING COLUMN NO.

*****************************************************************

        POP     BX
        MOV     M,CH                     ;PUT IN COUNT
        MOV     CS:TEMCNT,CH            ;PUT IT HERE, TOO

;       *** CHANGED 21 NOVEMBER, 1987 - RLF
;               9 LINES DELETED FROM ORIGINAL VERSION

        MOV     CL,ØDH
        CALL    COUT                    ;PRINT CR
READX:  POP     AX
```

*

A Potpourri Of Stuff For The H/Z-100 Computer

Wojtek Bok
6390 Mary Jane Crescent
Gloucester, Ontario
CANADA K1C 3C2

Over the last four years, I have played around with the H/Z-100 computer. During that time, I have developed several routines to make my life easier. Below are some of those routines.

Easy Copying Of Files With One Floppy Drive And A Hard Drive

Do you have only one 5-1/4 inch disk drive and a hard disk? Have you tried copying a 5-1/4 inch disk to another 5-1/4 inch disk? Usually you type in: COPY A:*.* B:. Then you spend all your time switching between the two 5-1/4 inch disks.

Even worse is if the source disk is divided into sub-directories. Then you must create the sub-directories on the destination disk and switch between them as you copy. A most complicated way of doing things.

The TREECOPY utility that comes with the Programmers Utility Pack does help somewhat, but you must still switch between disk A: and disk B:. If you copy the source disk to the hard drive, and then copy it to the destination disk, you must still delete the files from your hard drive. Deleting sub-directories and their files is a tedious job.

There is an easier way of doing all this. Using a batch file and the BACKUP and RESTORE utilities, you can automate the entire process. It will even format the destination disk for you. You need MS-DOS 2 or later, because of the batch commands used.

The syntax for using this file is:

```
CP drive switch
```

The drive is the source and destinaiton disk drive. The switch indicates whether the destination disk is to be formatted. Since I always verify my formatting, the switch I use is /V. As long as a valid switch or combination of switches is used, the formatting will be done. Some examples are:

CP A: /V/N	Copies drive A: to drive A:. Will format the destination disk with: verify (/V) no prompts (/N)
CP C:	Copies C: to C:. No formatting

Below is a list of this file.

Line	Statements
1	ECHO OFF
2	IF "%1"=="" GOTO CONT
3	ECHO You are copying all the files on the disk in drive %1 to
4	ECHO a new disk in drive %1
5	IF "%2"=="" GOTO CONT
6	ECHO The new disk will be formatted with switches %2
7	:CONT
8	ECHO Insert the source disk into drive %1
9	PAUSE
10	H:
11	CD \COPIER
12	E:\BIN\BACKUP %1*.* H:COPIER \V\NXG
13	ECHO Finished phase one
14	ECHO Insert the destination disk into drive %1
15	IF "%2"=="" GOTO CONT2
16	ECHO It will now be formatted
17	PAUSE
18	E:\BIN\FORMAT %1 %2
19	:CONT2
20	ECHO Now ready to copy files to destination disk
21	PAUSE
22	E:\BIN\RESTORE H:COPIER %1*.* \0\V
23	ECHO Finished copy process -- Housekeeping
24	DEL H:\COPIER\COPIER.*
25	CD \
26	GOTO FINI
27	:BAD
28	ECHO You must enter a drive name ie: CP A:
29	:FINI

Lines 1 to 9 — These lines give you an indication of what you are going to do. I always put this sort of thing at the beginning of my batch files. Line FIVE tests to see if you are going to format the destination disk.

Lines 10 to 11 — I use partition H: on my hard disk for copying with sub-directory copier. When you do this, you must have at least the size of the source disk free on the hard disk. For instance, if you are copying an 8 inch disk, you must have 1.25 Mbytes free.

Line 12 — My utility files are kept on drive E: in the sub-directory \BIN. This can only be used with MS-DOS 3. If you are using MS-DOS 2, the PATH must be set to wherever you keep BACKUP, RESTORE, and FORMAT. The %1*.* tells BACKUP to back-up all the files on disk %1. %1 is the first parameter passed to the batch file. H:COPIER is the file that the source disk files will be put into. The switches are:

/V Verify all disk writing
/N Do not format destination disk
/G Back-up all sub-directories

The switch /N is used because you may not want to format. The formatting will be done later.

Lines 13 to 15 — Informs you that the back-up is complete. Line 15 also checks to see if formatting of the destination disk is needed.

Lines 16 to 18 — Formats the destination disk, if needed.

Lines 19 to 21 — Informs you that the batch file is ready to copy the files onto the destination disk.

Line 22 — Restores the files and sub-directories onto the destination disk. The switches are:

/O Overwrite existing files with the same name
/V Verify all disk writing.

Line 23 — A message telling you the work is done.

Line 24 — Deletes the back-up file on the hard drive. This keeps the hard drive from becoming cluttered.

Line 25 — Resets back to the root directory on drive H:.

Line 26 — Redirects the batch file to the end of the file.

Line 27 to 28 — This is the destination of line two. You end up here if the parameter %1 was left blank.

Line 29 — The target for line 26.

This file makes copying disks quite easy. It does take a little time because of the verifying, but this assures accurate copies.

Getting Around The ZBASIC Keyboard Pre-Processor

If you have done any programming in ZBASIC, you may have noticed that certain keys are not recognized. Also, some keys cannot be used in their shifted mode. The keypad is the same as the number keys above the main keyboard. Yet, when you look into the Z-100 manual, all the keys have distinct values. It seems that ZBASIC catches the key code and modifies it.

There is a way to beat this. In the H/Z-100 Technical Manual, port address F4 (hex) is shown as the keyboard buffer exit. This port holds the value of the next key in line in the buffer. As you type, if the computer is busy, the keyboard buffer fills up in a FIFO (first-in, first-out) order. Then, as the program is ready to get the next keystroke, it puts the next character in line at the port. The buffer can hold a maximum of 15 characters.

If you just use INP(&HF4), the buffer will not be updated, and the same character will always be read, so you must use this in conjunction with the BASIC keyboard routines.

```
10 A$=INPUT$(1)
20 A%=INP(&HF4)
30 A$=CHR$(A%)
```

This short routine will wait for a keyboard stroke, and then read in the character at the keyboard buffer. The ZBASIC function will update the buffer to the next character. You must keep the INP() as close to the ZBASIC keyboard function as possible, otherwise, strange results sometimes occur. The best place is on the same line, but I have broken this up for readability.

Using this in programs opens up the entire keyboard for your use. If you use the BASCOM compiler, you had to rely on the ON KEY() GOSUB to use the function keys. This is no longer needed. Plus, you get all the shifted function keys, too. The entire keypad is enabled as separate from the main keyboard, and you can use shifted cursor keys.

Below is a table of the different keys and their codes. Note that the INS LINE, ESC, and BREAK keys are not recognized by ZBASIC at all, but they can be used with BASCOM. All values are in base ten (decimal).

KEY	Normal	Shifted
BACKSPACE	8	8
ESC	27	27
DELETE	127	127
HELP	149	213
F0	150	214
F1	151	215
F2	152	216
F3	153	217
F4	154	218
F5	155	219
F6	156	220
F7	157	221
F8	158	222
F9	159	223
F10	160	224
F11	161	225
F12	162	226
INS CHR	163	227
INS LINE	164	228
BREAK	170	234

The table below is a representation of the keypad. Beside the key (in brackets) is the ASCII number that BASIC returns. Below that is the shifted value from the keyboard port, and below is the normal (unshifted) value.

```
/-------------------------------------\
|(HOME) 11| ( < ) 29| ( > ) 28| ( ^ ) 30|
|   233   |   232   |   31    |   229   |
|   169   |   168   |   167   |   165   |
|---------|---------|---------|---------|
| ( 7 ) 55| ( 8 ) 56| ( 9 ) 57| ( v ) 31|
|   247   |   248   |   249   |   230   |
|   183   |   184   |   185   |   186   |
|---------|---------|---------|---------|
| ( 4 ) 52| ( 5 ) 53| ( 6 ) 54| ( - ) 45|
|   244   |   245   |   246   |   237   |
|   180   |   181   |   182   |   173   |
|---------|---------|---------|---------|
| ( 1 ) 49| ( 2 ) 50| ( 3 ) 51|(ENTER)  |
|   241   |   242   |   243   |      13 |
|   177   |   178   |   179   |         |
|-------------------|---------|   205   |
| ( 0 )        48   | ( . ) 46|         |
|       240         |   238   |   141   |
|       176         |   174   |         |
\-------------------------------------/

   (key) ZBASIC ASCII
     shifted
     normal
```

Through the use of the keyboard port, the entire keyboard becomes open to you, and you are not held back by ZBASIC. When using the function keys from the ZBASIC interpreter, you must set them to a single character string; otherwise, if the function key holds "LIST ", five characters would be sent to the keyboard buffer, and the function key would operate five times.

A simple routine to do this is:

```
10 FOR LP%=1 TO 12
20 KEY LP%, *''
30 NEXT
```

You must set the key to something, or else no keystroke will be sent at all. This only applies to the interpreter. The compiler ignores all key settings, except for ON KEY() which is not needed with this routine.

Using The Second Page Of Display On Z-100s With 64K VRAM

If you have 64K Video RAM chips installed on your video board, you can use the second page of display. This is very effective for help screens, and two screen games/programs. Two routines are needed. One tells the main CPU which page to write to, and the second tells the video CPU which page to display. This allows updating of one page while the other is displayed, then switches to the first page. It only takes one scan to change pages (about 1/30 second).

Since you have to keep track of which page you are reading or displaying, two variables are used: PAGE.DISPLAY% and PAGE.WRITE%. These are set to 0 for the front page, and 1 for the back page.

```
1000   routine sets the CPU write page
1010 IF PAGE.WRITE%=0 THEN PAGE.
     WRITE%=1 ELSE PAGE.WRITE%=0
1020 LATCH%=INP(&HDA)
1030 OUT &HDA,LATCH% XOR &H80
1040 RETURN

2000   routine sets the display page
2010 IF PAGE.DISPLAY%=0 THEN PAGE.
     DISPLAY%=1 ELSE PAGE.DISPLAY%=0
2020 OUT &HDC,&HC
2030 LATCH%=INP(&HDD):OUT &HDD,
     LATCH% XOR &H8
2040 RETURN

3000   error trap
3010 IF PAGE.DISPLAY%=1 THEN GOSUB
     2000  display to front
3020 IF PAGE.WRITE%=1 THEN GOSUB
     1000  write to front
3030 ...
```

Line 2030 must be kept intact. The display routine only allows the main CPU access to the video registers one cycle out of three. If the routine is on separate lines, or separated by other statements, the value of the latch will not equal the value of the variable LATCH%. Then when you reset the XOR'd latch with the OUT statement, you could send the video CPU to the wrong place. So keep the INP and OUT as close as possible.

To use this, print to the front page, then GOSUB 1000. Print to the back page, then GOSUB 2000. The screen should instantly change to whatever you have written there. Remember to reset to the front page before ending the program.

```
****************** WARNING ******************
*                                           *
*  Never allow the screen to scroll when you  *
*  are using both pages and NEVER exit the    *
*  program if you are on the back page.       *
*                                           *
****************** WARNING ******************
```

The Z-100 video only uses part of the video memory available. The rest is ignored. However, if the screen scrolls while you are on the back page, the internal display routine will update the screen partially from the normal area, and partially from the unused area. The effect is unreadable garbage on the screen and the position of the cursor is unpredictable. About the only way out is to reboot the computer. The same thing will happen if you exit the program while on the back page.

You MUST set an ON ERROR GOTO within your program, and then test which page you are on, and set the display back to the front page before exiting the program.

Having said all that, here are the routines:

All Those Batch Files

I use about fifty batch files. Some need extra parameters, some don't. This got very confusing. I would run a batch file and it would not work right. So I type it to the screen to see what it needs, then run it again. Very time consuming. I also would forget what batch files I had. First I created a sub-directory named BATCH, and put all my batch files there. But after a while there were too many.

A solution to this was to make a series of menu's. These are easy to make. They contain the batch file, any parameters it needs, and a short description of what the file does, and what the parameters are used for.

Any word processor will do. A sample is shown on the next page.

```
/-------------------------------------------------------------\
|              Word Processing Menu                           |
|                                                             |
|     WW name        Runs WatchWord. Name is the sub-directory|
|                    where the files are. You must have a disk|
|                    in drive A:                              |
|                                                             |
|     PT name        Runs PeachText. Same as above.           |
|                                                             |
|     MENU           Goes to the main menu.                   |
\-------------------------------------------------------------/
```

I named this file WORD.MNU. Then, when I type WP (for word processing) from the main menu, a batch file named WP.BAT clears the screen and types WORD.MNU to the screen. This is faster than ECHOing all the lines.

```
ECHO OFF
CLS
TYPE E:\MENU\WORD.MNU
```

Each menu file contains descriptions of similar batch files. Thus, all the word processing batch files are shown through the WORD.MNU, and the programming files in the PROGRAM.MNU and so on. The CP.BAT file shown earlier is in the UTILITY.MNU.

This really eases the amount of commands you have to remember, and which disk/partition/sub-directory you put those programs in. Also, at the end of each batch file, put in the command to retype the menu, so you always have a menu showing on the screen. If you include a command to show the time, you will know when the last time someone used the computer.

In addition, I always put what the batch file does at the beginning, as well as a check for needed parameters.

Spicing Up Batch Files

You can add color to your batch files and to menus by embedding <ESC>mFB in the text. The <ESC> stands for the escape character (decimal 27). F and B are foreground, background. The colors are: 0 BLACK, 1 BLUE, 2 RED, 3 MAGENTA, 4 GREEN, 5 CYAN, 6 YELLOW, and 7 WHITE. You must use both the foreground and background. So, to write a line in cyan with the word 'switch' in red:

```
<ESC>m50Use the <ESC>m20switch<ESC>m50
to set the parameter
```

The color escape sequence is a non-printable character set. So even though you can see it when you are typing it, when it is printed on the screen, it will take up zero space. So the above example will show up as:

```
Use the switch to set the parameter
```

Note that the first <ESC> in the first example will line up with the word 'use' in the second example. The <ESC> character can be added by pressing the F8 key while using either EDLIN or COPY CON filename.

You can also add graphics. The H/Z-100 Technical Manual shows the characters that are available in the ALTCHAR.SYS file that comes with Z-DOS/MS-DOS. Just make sure that this file is in the root directory of the disk/partition that you are booting from. It will be automatically loaded. To turn on the graphics use <ESC>F, and to turn them off <ESC>G.

There are many escape sequences that the H/Z-100 allows. Anything from cursor addressing to that loveable key click. The only one you should NOT use is '<ESC>} , as this will disable the keyboard. To reset the keyboard use '<ESC>{'.

By the way, the cursor addressing uses printable characters, not just numbers. So, to reach the upper left corner, type '<ESC>Y'. The 'Y' is followed by two spaces. Row 1, column 1 is '<ESC>Y!!', and so on down the ASCII set. If you use '<ESC>11', you will put the cursor at row 18, column 18.

If your word processor allows adding control characters, you can add a beep. Just include CTRL-G (decimal 7). Again, this will not take up any space.

Changing The Keyboard

When I program, I use the number keypad a lot for entering numbers. I also use the special symbols above the numbers on the main keyboard. This involves a lot of shifting. Eventually, I got tired of this and looked for a solution. The FONT.EXE utility, that comes with the operating system, has a feature that allows for re-mapping the keyboard. So away I went.

Run FONT and choose the keyboard from the menu. You are shown two lists of numbers which run from 00H to FFH. Each number is shown twice. The first number is the actual key pressed. The second number is sent to the keyboard buffer. It is possible to change the second number.

For instance, if you press the 1 key on the main keyboard, a 31H (Hexadecimal) would be noted by the keyboard CPU. It looks up 31H in a table held in RAM and sends the number there to the buffer. This would normally be 31H. If you changed it to 32H, then every time you pressed 1 you would get 2.

This is not very useful, but if you changed it to 21H and changed the number at 21H to 31H, then the 1 key would produce ! and the shifted 1 key (!) would produce 1. This is what I have done.

Below is a table with all the changed values. All the numbers are in hexadecimal.

Change	to	Change	to
21	31	32	40
23	33	33	23
24	34	34	24
25	35	35	25
26	37	36	5E
28	39	37	26
29	30	38	2A
2A	38	39	28
30	29	40	32
31	21	5E	36

This table will only affect the number keys above the main keyboard. It will NOT affect the number keypad. It has different and distinct key codes.

This changing of values is very time consuming. Luckily, FONT provides a very easy way to change fonts and key mapping. After making all the changes, save them using FONT's save option. Then, you can type:

```
FONT filename
```

Filename is the file that contains the changed values. I called the file that inverts the number key NUM_SHFT.KEY, and the number restore file NUM_NORM.KEY.

FONT only saves the fonts/keys that are changed. So, to change back, you must first change the keys, run font, and manually change the values back to the originals. Then save the changes. Now when you want the numbers inverted, type FONT NUM_SHFT.KEY, and for normal numbers FONT NUM_NORM .KEY. This method also works for different character fonts.

Of course, I made a batch file to all this for me.

```
1 ECHO OFF
2 IF "%1"== "R" GOTO UPPER
3 IF "%1" == "r" GOTO UPPER
4 IF "'%1"  == "N" GOTO LOWER
5 IF "'%1"  == "n" GOTO LOWER
6 ECHO Type   NUM R for reversed number and symbols
7 ECHO        NUM N for normal operation
8 GOTO FINI
9 :UPPER
10 ECHO You are reversing the numbers and symbols
11 PAUSE
12 FONT E:\NUM_SHFT.KEY
13 ECHO Numbers and symbols reversed
14 GOTO FINI
15 :LOWER
16 ECHO You are putting the numbers and symbols back to normal
17 PAUSE
18 FONT E:\NUM_NORM.KEY
19 ECHO Numbers and symbols back to normal
20 :FINI
```

If you want the reversed numbers loaded every time you boot, make the changes to the ALTCHAR.SYS file.

Resident Program Loading

There are some resident programs that resent being loaded twice. Yet from a batch file point of view, you will try to load them multiple times. One of these is the IBM emulator program ZPC. If you run it a second time, it will complain. However, you do want to load it once.

One way to get around this, is to create a file in the RAM disk, and then check for that file in the resident program batch file. Since, when you reboot the computer, the contents of the RAM disk are lost, along with the resident programs. The absence of the file will load the resident program. Below is an example of loading a resident program called TEST.

```
ECHO OFF
IF EXIST I:TEST.FLG GOTO CONT
ECHO t > I:TEST.FLG
E:\RESIDENT\TEST
:CONT
.....
```

The ..... stands for whatever needs to be done with the loaded resident program. If you name the flags, then you can use a bunch of them.

Speeding Up Batch Files

If you have a long batch file with many commands and goto's, its operation can become time consuming. An easy way to speed things up is to copy the batch file to a RAM disk and then run it from there. Because the RAM disk is very fast, the MS-DOS command processor will run at the maximum speed possible.

The overhead of copying it to the RAM disk is quite small when compared to the overall time savings. Set your PATH command to look into the RAM disk first. If the batch file is there, it will execute, if not, little time is wasted as the RAM disk is very fast.

Checking For Three Banks Of VRAM

The H/Z-100 computer can have one, two, or three banks of video RAM. When writing programs that rely on color, it is important to know if the three banks are there. A fast and simple routine to check for this is shown below.

```
10  FOR A%=7 TO 0 STEP -1
20    PSET (0,0),A%
30    IF POINT(0,0)<>A% THEN FLAG%=1
40  NEXT
50  IF FLAG%=1 THEN BEEP:PRINT "NOT a c
olor computer":END
60  ....
50  IF FLAG%=1 THEN C1%=0:C2%=7:C3%=7:C
4%=7:C5%=7:C6%=7:C7%=7:ELSE C1%=1:C2%=2
:C3%=3:C4%=4:C5%=5:C6%=6:C7%=7
60  ....
```

Lines 10 to 40 write a single pixel to the VRAM, and then check that the same color is there. If not, a flag is set. The STEP -1 is used because it does not leave behind a pixel in the upper left corner of the screen. Line 50 depends on whether or not you have made provisions for monochrome. The first line will end the program with a message. The second line 50 will set variables for use in the program. Simply use COLOR C3%.

If you want both foreground and background colors, C1% has been set to 0, so use COLOR C1%,C7%. If color is present, the colors will be blue on white, if there is no color, it will be black on white, still an eye catcher. To get really fancy with colors, just test C1%. If C%1=0, then use a monochrome combination; otherwise, use colors.

This routine is automatic at the beginning of the program, and gets away from needing a question/answer routine for color.

Have fun experimenting with the ideas shown here, and good computing! *

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

HUG Price List

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
ASCIRITY	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	30.00
HDOS PROGRAMMERS HELPER	885-8017	HDOS	UTILITY	16.00
HOME FINANCE	885-1070	HDOS	BUSINESS	18.00
HUG DISK DUPLICATION UTILITIES	885-1217-[37]	CPM	UTILITY	20.00
HUG SOFTWARE CATALOG	885-4500	VARIOUS	PRODUCTS THRU 1982	9.75
HUGMAN & MOVIE ANIMATION	885-1124	HDOS	ENTERTAINMENT	20.00
INFO. SYSTEM AND TEL. & MAIL SYSTEM	885-1108-[37]	HDOS	DBMS	30.00
LOGBOOK	885-1107-[37]	HDOS	AMATEUR RADIO	30.00
MAPLE	885-8005	HDOS	COMMUNICATION	35.00
MAPLE	885-8012-[37]	CPM	COMMUNICATION	35.00
MICRONET CONNECTION	885-1122-[37]	HDOS	COMMUNICATION	20.00
MISCELLANEOUS UTILITIES	885-1089-[37]	HDOS	UTILITY	20.00
MORSE CODE TRANSCEIVER	885-8016	HDOS	AMATEUR RADIO	20.00
MORSE CODE TRANSCEIVER	885-8031-[37]	CPM	AMATEUR RADIO	20.00
PAGE EDITOR	885-1079-[37]	HDOS	UTILITY	25.00
PROGRAMS FOR PRINTERS	885-1082	HDOS	UTILITY	20.00
REMARK VOL 1 ISSUES 1-13	885-4001	N/A	1978 TO DECEMBER 1980	20.00
RUNOFF	885-1025	HDOS	TEXT PROCESSOR	35.00
SCICALC	885-8027	HDOS	UTILITY	20.00
SMALL BUSINESS PACKAGE	885-1071-[37]	HDOS	BUSINESS	75.00
SMALL-C COMPILER	885-1134	HDOS	LANGUAGE	30.00
SOFT SECTOR SUPPORT PACKAGE	885-1127-[37]	HDOS	UTILITY	20.00
STUDENT'S STATISTICS PACKAGE	885-8021	HDOS	EDUCATION	20.00
SUBMIT (Z80 ONLY)	885-8006	HDOS	UTILITY	20.00
TERM & HTOC	885-1207-[37]	CPM	COMMUNICATION & UTILITY	20.00
TINY BASIC COMPILER	885-1132-[37]	HDOS	LANGUAGE	25.00
TINY PASCAL	885-1086-[37]	HDOS	LANGUAGE	20.00
UDUMP	885-8004	HDOS	UTILITY	35.00
UTILITIES	885-1212-[37]	CPM	UTILITY	20.00
UTILITIES BY PS	885-1126	HDOS	UTILITY	20.00
VARIETY PACKAGE	885-1135-[37]	HDOS	UTILITY & GAMES	20.00
VOLUME I	885-1008	N/A	SOFTWARE LISTINGS	9.00
VOLUME II	885-1013	N/A	SOFTWARE LISTINGS	12.00
VOLUME III	885-1015	N/A	SOFTWARE LISTINGS	9.00
VOLUME IV	885-1037	N/A	SOFTWARE LISTINGS	12.00
WATZMAN ROM SOURCE & DOC	885-1221-[37]	CPM	H19 FIRMWARE	30.00
WATZMAN ROM	885-4600	N/A	H19 FIRMWARE	45.00
WHEW UTILITIES	885-1120-[37]	HDOS	UTILITY	20.00
XMET ROBOT X-ASSEMBLER	885-1229-[37]	CPM	UTILITY	20.00
Z80 ASSEMBLER	885-1078-[37]	HDOS	UTILITY	25.00
Z80 DEBUGGING TOOL (ALDT)	885-1116	HDOS	UTILITY	20.00
H8 — H/Z-89/90 — H/Z-100 (Not PC)				
ADVENTURE	885-1222-[37]	CPM	GAME	10.00
BASIC-E	885-1215-[37]	CPM	LANGUAGE	20.00
CASSINO GAMES	885-1227-[37]	CPM	GAME	20.00
CHEAPCALC	885-1233-[37]	CPM	SPREADSHEET	20.00
CHECKOFF	885-8011-[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
DUNGEONS & DRAGONS	885-1209-[37]	CPM	GAMES	20.00
FAST ACTION GAMES	885-1228-[37]	CPM	GAME	20.00
FUN DISK I	885-1236-[37]	CPM	GAMES	20.00
FUN DISK II	885-1248-[37]	CPM	GAMES	35.00
GAMES DISK	885-1206-[37]	CPM	GAMES	20.00
GRADE	885-8036-[37]	CPM	GRADE BOOK	20.00
HRUN	885-1223-[37]	CPM	HDOS EMULATOR	40.00
HUG FILE MANAGER & UTILITIES	885-1246-[37]	CPM	UTILITY	20.00
HUG SOFTWARE CATALOG UPDATE #1	885-4501	VARIOUS	PRODUCTS 1983 THRU 1985	9.75
KEYMAP CPM-80	885-1230-[37]	CPM	UTILITY	20.00
MBASIC PAYROLL	885-1218-[37]	CPM	BUSINESS	60.00
MICRONET CONNECTION	885-1224-[37]	CPM	COMMUNICATION	16.00
NAVPROGSEVEN	885-1219-[37]	CPM	FLIGHT UTILITY	20.00
REMARK VOL 3 ISSUES 24-35	885-4003	N/A	1982	20.00
REMARK VOL 4 ISSUES 36-47	885-4004	N/A	1983	20.00
REMARK VOL 5 ISSUES 48-59	885-4005	N/A	1984	25.00
REMARK VOL 6 ISSUES 60-71	885-4006	N/A	1985	25.00
REMARK VOL 7 ISSUES 72-83	885-4007	N/A	1986	25.00

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
SEA BATTLE	885-1211-[37]	CPM	GAME	20.00
UTILITIES BY PS	885-1226-[37]	CPM	UTILITY	20.00
UTILITIES	885-1237-[37]	CPM	UTILITY	20.00
X-REFERENCE UTILITIES FOR MBASIC	885-1231-[37]	CPM	UTILITY	20.00
ZTERM	885-3003-[37]	CPM	COMMUNICATION	20.00
H/Z-100 (Not PC) Only				
ACCOUNTING SYSTEM	885-8048-37	MSDOS	BUSINESS	20.00
CALC	885-8043-37	MSDOS	UTILITY	20.00
CARDCAT	885-3021-37	MSDOS	BUSINESS	20.00
CHEAPCALC	885-3006-37	MSDOS	SPREADSHEET	20.00
CHECKBOOK MANAGER	885-3013-37	MSDOS	BUSINESS	20.00
CP/EMULATOR	885-3007-37	MSDOS	CPM EMULATOR	20.00
DBZ	885-8034-37	MSDOS	DBMS	25.00
ETCHDUMP	885-3005-37	MSDOS	UTILITY	20.00
EZPLOT	885-3023-37	MSDOS	PRINTER PLOTTING UTILITY	20.00
GAMES CONTEST PACKAGE	885-3017-37	MSDOS	GAMES	25.00
GAMES PACKAGE II	885-3044-37	MSDOS	GAMES	25.00
GRAPHICS	885-3031-37	MSDOS	ENTERTAINMENT	20.00
HELPSCREEN	885-3039-37	MSDOS	UTILITY	20.00
HUG BACKGROUND PRINT SPOOLER	885-1247-37	CPM	UTILITY	20.00
KEYMAC	885-3046-37	MSDOS	UTILITY	20.00
KEYMAP	885-3010-37	MSDOS	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
TREE-ID	885-3036-37	MSDOS	TREE IDENTIFIER	20.00
USEFUL PROGRAMS I	885-3022-37	MSDOS	UTILITIES	30.00
UTILITIES	885-3008-37	MSDOS	UTILITY	20.00
Z100 WORDSTAR CONNECTION	885-3047-37	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 And PC Compatibles				
ADVENTURE	885-3016-37	MSDOS	GAME	10.00
ASSEMBLY LANGUAGE UTILITIES	885-8046-37	MSDOS	UTILITY	20.00
BOTH SIDES PRINTER UTILITY	885-3048-37	MSDOS	UTILITY	20.00
DEBUG SUPPORT UTILITIES	885-3038-37	MSDOS	UTILITY	20.00
DPATH	885-8039-37	MSDOS	UTILITY	20.00
HADES	885-3040-37	MSDOS	UTILITY	40.00
HELP	885-8040-37	MSDOS	CAI	20.00
HEPCAT	885-3045-37	MSDOS	UTILITY	35.00
HUG BACKGROUND PRINT SPOOLER	885-3029-37	MSDOS	UTILITY	20.00
HUG EDITOR	885-3012-37	MSDOS	TEXT PROCESSOR	20.00
HUG MENU SYSTEM	885-3020-37	MSDOS	UTILITY	20.00
HUG SOFTWARE CATALOG UPDATE #1	885-4501	VARIOUS	PROD 1983 THRU 1985	9.75
HUGMCP	885-3033-37	MSDOS	COMMUNICATION	40.00
HUGPBBS SOURCE LISTING	885-3028-37	MSDOS	COMMUNICATION	60.00
HUGPBBS	885-3027-37	MSDOS	COMMUNICATION	40.00
ICT 8080 TO 8088 TRANSLATOR	885-3024-37	MSDOS	UTILITY	20.00
MATT	885-8045-37	MSDOS	MATRIX UTILITY	20.00
MISCELLANEOUS UTILITIES	885-3025-37	MSDOS	UTILITIES	20.00
REMARK VOL 5 ISSUES 48-59	885-4005	N/A	1984	25.00
REMARK VOL 6 ISSUES 60-71	885-4006	N/A	1985	25.00
REMARK VOL 7 ISSUES 72-83	885-4007	N/A	1986	25.00
REMARK VOL 8 ISSUES 84-95	885-4008	N/A	1987	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
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
PS's PC UTILITIES	885-6011-37	MSDOS	UTILITIES	20.00
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
ULTRA RTTY	885-6012-37	MSDOS	AMATEUR RADIO	20.00

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

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

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

ORDERING INFORMATION

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

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

10 – Very Good
9 – Good
8 – Average

TABLE C
Product Rating

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

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

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

ORDERING INFORMATION

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

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

NOTES

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

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

EZPLOT II
HUG P/N 885-3049-37
H/Z-100 Version $25
HUG P/N 885-6013-37
PC Compatible Version . . $25

Originally released in 1985, EZPLOT is a user friendly high resolution graphics function plotting program for engineers, scientists, and just about anyone who would like to have their printer plot curves from data on disk. A complete abstract of the original EZPLOT can be found in the HUG Software Catalog Update #1. This new version of EZPLOT maintains all of its original features, plus the following summary of enhancements.

1. The I/O interface has been completely rewritten resulting in a more flexible, more tolerant, and easier to use process.
2. EZPLOT now plots in user selectable colors.
3. EZPLOT plots in high resolution. The H/Z-100 version will plot in 640 × 400 resolution, even on machines that only have 1 bank of 32k video-RAM chips. The PC version supports both CGA (640 × 200), EGA (640 × 350), and Hercules Graphics resolutions.
4. In addition to standard functions and x-y path functions, EZPLOT will also plot discrete points.
5. EZPLOT now supports logarithmic scales, as well as linear scales.
6. EZPLOT can now plot as many as six functions.
7. EZPLOT now supports most modern dot matrix printers, including the new 24-pin models.
8. EZPLOT now offers the selection of various sizes and orientations of printed output.
9. EZPLOT provides the capability of creating, saving and retrieving templates.
10. All graphics and menu routines have been rewritten to maximize speed.

Printers now supported by EZPLOT II include the folloiwng:

1. Any Epson/IBM Command Set Compatible 9/18 pin Dot Matrix Printer, including:
   - Epson FX, MX, or RX series
   - IBM Dot Matrix Printers
   - Panasonic 1080 & 1090 series
   - Okidata Microline 92
   - Texas Instruments TI-850 Series
   - ALPS Dot Matrix Series
2. Star Gemini 10X/15X
3. C.Itoh Prowriter
4. NEC 8023A
5. Any Epson LQ-1500 compatible (24-pin), including:
   - Epson LQ-800
   - NEC P-6
   - ALPS 200/300 (24 pin option)

If your printer is not listed, it is possible that your printer is compatible with one that is listed, especially if your printer is relatively new. However, if your printer is not compatible with one listed, then EZPLOT II can be installed to use your resident graphics screendump software.

Original owners of EZPLOT can update their disk to the newer version by returning the original disk along with a check or money for $10, made out to HUG to: The Heath Users' Group, P.O. Box 217,

Benton Harbor, MI 49022. HUG P/N 885-3023-37 will be upgraded to 885-3049-37, and HUG P/N 885-6003 will be upgraded to 885-6013-37, depending upon which original disk product is returned.

HUG P/N 885-4601 FlipFast Guide To GW-Basic $20

This book is the FlipFast Guide to Zenith GW-Basic by William Barden Jr. This 408 page tome tells it all. Whether you're a long-time user of BASIC (even old-timers can't remember everything), or just getting started, this quick reference manual is the only book you'll ever need. Here are some of the features you'll find between its covers:

- **Names** of all commands, statements, and functions
- **Syntax** with available options
- **Command usage** with concise explanations
- **Examples** with "real-world" explanations
- **Requirements** for general usage that minimize programming errors and simplify debugging
- **Application notes** that give specific information about command functions
- **Hints, tips, and bugs** that can save you hours of programming time
- **Error messages** listed and explained for each command, where applicable

Plus

- How to use GW-BASIC
- How to write better and faster programs
- Tips on using variables
- Complete reference appendix

The retail price of this manual is $21.95. Why spend more? Save $1.95 by ordering HUG P/N 885-4601 today!

HUG P/N 885-4602 FlipFast Guide To MS-DOS $23

This book is the FlipFast Guide to MS-DOS by William M. Adney. Bill, as you may already know, also writes two monthly columns for REMark Magazine, "On The Leading Edge", and "Powering Up". Designed to be a quick reference manual, this book is for beginners and 'seasoned' users alike. As editor of this magazine, and assembly language programmer, here are some of the reasons I use it as my only reference guide:

- **Names** of all commands or subcommands
- **Syntax** with available options
- **Applications** concisely explaining typical command usage
- **Examples** with "real-world" explanations
- **Requirements** for general usage that minimize system errors and speed computer operations
- **Error messages** are explained for each command, where applicable

Plus

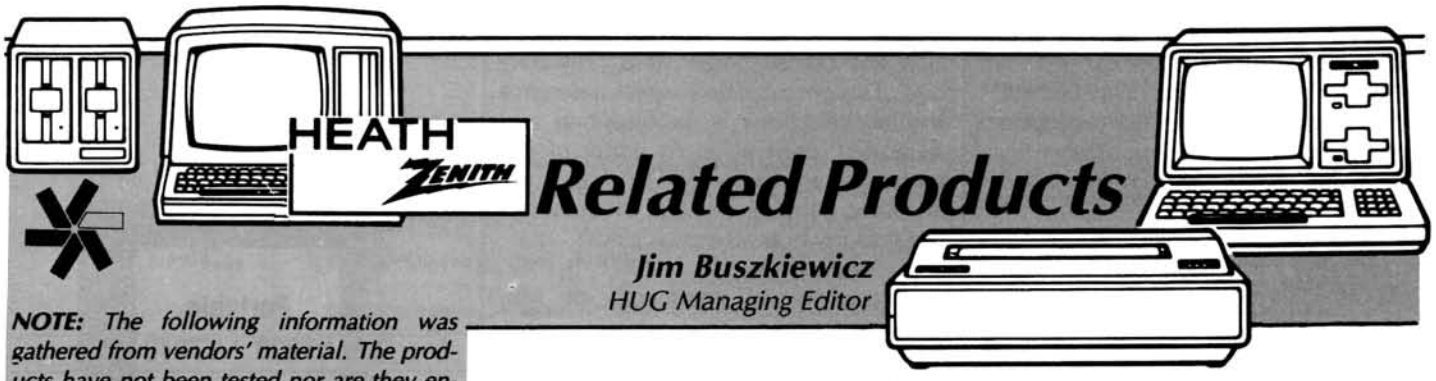
- How to use MS-DOS
- Complete batch file & hard-disk reference
- Programmer's command reference
- DOS Technical reference
- Config file reference, glossary and more!

The retail price of this manual is $24.95. Why spend more? Save $1.95 by ordering HUG P/N 885-4602 today.

HUG P/N 885-4603 FlipFast Guide Combo Offer $41

Save Even More by ordering both FlipFast guides!! When purchased together you can save $4.90 off the retail price of both guides. Order HUG P/N 885-4603 today!

*



Related Products

Jim Buszkiewicz
HUG Managing Editor

***NOTE:** The following information was gathered from vendors' material. The products have not been tested nor are they endorsed by HUG. We are not responsible for errors in descriptions or prices.*

Headquartered in Ann Arbor, Michigan, **Condor Computer Corporation** is a software development firm specializing in relational database management systems. A recent Dataquest survey ranked Condor third in the size of its installed user base, which now totals over 200,000 users. Virtually every imaginable application has been developed using Condor, including those for government, small business, non-profit organizations, sports, accounting, religious organizations, schools, and banks.

Condor's most recent release, Condor 3 Version 2.20, is a custom application development package for novice and intermediate users. It has garnered enthusiastic reviews in the microcomputer press.

Condor Computer has extended its special $95 update price on the popular Condor 3 Relational Database Management System (release 2.20) through June 30, 1988, for HUG MEMBERS ONLY!! The offer had originally been scheduled to expire on May 31, 1988.

Condor 3 is a relational, programmable database management system offering novice computer users a complete application development capability. It features a "command-by-example" user interface; a "what you see is what you get" (WYSIWYG) screen painting facility for creating data entry forms, custom menus and reports; uniform command syntax linking the application language, and report writer; relational functions, and "set-level processing" for simplified application development.

The new update includes dozens of enhancements, among them an average speed increase of 400%, improved subdirectory support and expanded PRINT and LIST commands. In addition, Condor has totally rewritten and repackaged the documentation. "Getting the new documentation with the update is essentially like getting an entire new Condor 3 system for only $95," Condor President Larry Hauptman commented.

"Build Your Own: Custom Relational Database Applications" has just been published by Condor Computer Corporation. The self-teaching course is an introduction to the concepts and practical aspects of developing applications with relational database software.

"This course is for anyone who wants to know what's involved in database management," commented Condor Mr. Hauptman. "Because the Condor 3 software is included, it offers a valuable hands-on learning experience for novices exploring database managment."

It consists of a paperback book with 12 easy-to-follow lessons, review questions and answers, and a glossary of microcomputer terms. Also included is a complete working copy of the Condor 3 Relational Database software, with a capacity of 50 records. "Build Your Own" was developed from Condor 3 course materials currently in use by hundreds of educational institutions in their database courses.

"Condor 3 is an ideal vehicle for teaching the concepts of relational database management," said Hauptman. "First, because its application language is English; and second, because its consistent logic makes sense to first-time users. The result is they can concentrate on the principles of database management instead of having to master a difficult product."

Build Your Own $72.50
Condor Computer Corp.
Eduation Division
1490 Eisenhower Place
Ann Arbor, MI 48108
(313) 971-8880

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

HUG Club Correction

In the April issue of REMark, Page 44 in the *Local HUG Club Information Update* column, the phone numbers for the St. Louis HUG Contact Person and the Bulletin Board were reversed. The paragraph should have read:

New Contact Person for **St. Louis HUG** is Shirley Rubenstein. Phone: **(314) 946-2639**. They meet the 3rd Thursday each month at 7:00 pm at the HEC. New BB# **(314) 291-1850**.

On The Leading Edge

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

Apple And Microsoft, Baker's Dozen, Uninterruptible Power Supplies

Once in a while, I run into an interesting problem that has, or potentially can have, a serious impact on my computer system. One such problem is a power interruption, since that usually causes data loss for all information in memory. The power here in Grand Prairie seems to be subject to a lot of what I call "power blips". A power blip is a very short loss of power — usually 2-5 seconds — that is more than adequate to cause complete data loss in memory, not to mention its potential bad effects on the hard disk and other hardware.

Of course, it is quite frustrating to lose data that has not been saved to disk, and that is a particular problem for me because of all the writing that I do. I depend on my computer system a lot, and when its reliability suffers for any reason, it has a direct effect on my ability to meet various deadlines. Fortunately, I save data to disk frequently, so that is not often as serious as it might otherwise be. In order to understand the situation, let's see why the problem occurs, what causes it, and how to solve it.

Losing Things

When a computer system is powered-off, ALL data in memory is completely lost. That is the reason why you must be absolutely certain to copy all data from a RAM (or memory) disk to a floppy or hard disk before you turn off the power to your system. You can create a RAM disk in memory by using the VDISK device driver that is supplied with all current DOS versions. But there is another part of this discussion that you also need to know.

Virtually all computer software, such as a word processor or spreadsheet, performs all of its functions in memory. When you start an application, two things happen. First, the program is read from a disk into the computer's memory. Most people know that. When you load a data file for processing by that application, it is also important to know that part or all of the DATA is also read into the computer's memory. How much of the data file read into memory is dependent on the specific software and the size of the file. For example, assume you are editing a large file with a word processor, such as WordStar or Microsoft Word.

For a very large file, the program will only load PART of the file into memory, and the remainder of the file will be stored on disk in a special file. As you edit or page through the file, the program will keep track of where you are and read or write the appropriate data from or to the disk in a special temporary file. The important point to understand is that whatever changes you make to the file are made in memory only. These changes are only saved to disk when you tell the program to do so. If the power is interrupted (or turned off) before the memory contents are saved to disk, all changes to the file in memory will usually be lost. Some manufacturers have recognized this problem and provide a special "autosave" software feature that helps prevent data loss, but most software does not have this feature yet.

In short, it is imperative to save data to disk periodically to prevent it from being lost due to a power interruption. WordStar has the CTRL-KS command for this purpose, and Word uses the (T)ransfer (S)ave command. Although I wrote about this same problem several years ago, I think it is worthwhile to mention it again for our new members.

Causes And Effects

I was writing an article a couple of summers ago on my Z-100 using WordStar when the power "blipped" six times in one afternoon. Although each blip lasted less than five seconds, the last one was disastrous for me because the SuperBlock on my hard disk was destroyed. The SuperBlock is a special reserved area that DOS uses to identify partitions and in-

cludes other critical information for the system.

Although correct positioning of the "Format Enable" jumper on the Z-217 hard disk controller card is supposed to prevent this from happening, it did not in this case. The old adage — if it can go wrong, it will — certainly applied in this case. And so I lost EVERYTHING on my 26 megabyte hard disk because DOS could not find it as a result of the SuperBlock being clobbered. Fortunately, that was just after the first of the month, and I had just taken a complete backup of all partitions on the hard disk. I only lost about 20 minutes' worth of work, since I had also kept a floppy disk in the system for backup of that article. But the problem cost me an entire day's time because I had to PREP, FORMAT, and reload all seven partitions on the Z-100.

My solution was to get an Uninterruptible Power Supply (UPS), since I was generally familiar with the equipment because of my mainframe computer experience. After some research, I bought a Minuteman 500 UPS from Para Systems that I mentioned a couple of years ago. The UPS provides a 500 watt backup supply that has a specified switching time of four milliseconds. And it solved my power problem with the Z-100. When I transferred my production writing to my '248, of course I connected that UPS to that system.

When I was recently trying to "send" an article to HUG on CompuServe, we were having thunderstorms in Grand Prairie along with the inevitable power interruptions that usually occur. I spent nearly two hours trying to transfer the article to CompuServe, but power interruptions blew me away each time, and the UPS did NOT save me in this case. You can probably imagine how frustrating this was.

My first thought was that something was wrong with the UPS, since I had owned it for a couple of years. Since Para Systems is a local manufacturer, I contacted them directly for help. As it turns out, there was no problem with the Minuteman 500 at all — it was still well within the specified four millisecond maximum switching time for that unit. Therefore, the problem was clearly due to my changing to the '248 computer. But before I get to that, it is important to have some idea of how most UPSs work.

Uninterruptible Power Supplies

When you get a UPS, there are two important things to consider. The first is its power rating, which is technically specified in Volt-Amperes (VA), although some UPS manufacturers also list or define the specification in watts. For example, the Minuteman 500 can support up to 500 watts. Although my Z-100 system had a measured requirement of about 250 watts, my conservative engineering background led me to choose a UPS that had a 100% reserve capacity in case I decided to upgrade the system, although a 300 watt UPS is adequate for nearly all microcomputers and their CRTs.

The general purpose of a UPS is to give you reserve power so that you and your computer can ignore short power losses of up to five minutes or more. If the power is interrupted for long periods of time, the UPS gives you plenty of time to save your data to disk and perform a normal power-off. Although a UPS is usually not intended to be used for extended power support (e.g., an hour or more), the amount of time that a given UPS will provide power is strictly dependent on the hardware connected to it. For example, a specific UPS with a 300 watt capacity might provide up to 20 minutes of power for a given hardware configuration, including a computer and CRT. For that same system, a 600 watt UPS might provide nearly double that (e.g., 40 minutes), and that is one reason to choose a larger unit. The critical point is that the length of time is dependent on the UPS capacity, as well as the exact hardware connected to it.

There are generally two types of UPS units that you can buy. The first is usually referred to as a "straight through" unit, and this technique is generally used for mainframes. In this type of unit, the plug is connected to the wall socket, and electricity is always routed directly through the batteries so that the computer always gets power from the batteries. The advantage of this type of unit is that there is no switching time which is critical in a mainframe computer. Even a short loss of power measured in a fraction of a millisecond can cause serious problems in a mainframe. The disadvantage is that this approach is expensive because more electronics must be used.

This technique requires that the AC power from a wall socket be converted to DC and fed to the batteries. On the output side of the batteries, the DC must then be converted back to AC for the computer. A unit that converts AC wall socket power to DC (or vice-versa) is called an inverter. In the straight through type of unit, there are two inverters, and both must be built to handle the full UPS rating. There is a less expensive way to accomplish this.

The second technique requires that an electronic circuit monitor the voltage at the AC wall socket, and if the voltage falls below a specified value (e.g., 100 volts), the UPS "picks up" the load by transferring the load from the AC socket to its internal batteries. This transfer of the load takes a certain amount of time that is measured in milliseconds. The transfer or switching time is absolutely critical as I found out the hard way. Most of the UPS units for microcomputers have a transfer time of from one to eight milliseconds. Faster units are generally more expensive, but still cheaper than straight through units.

The UPS that has the transfer feature only requires one inverter to change the DC power from the batteries to AC power for the computer. Batteries are kept up to full charge by a trickle circuit, and a monitoring/transfer circuit continuously monitors AC power. When the AC line voltage drops below a specified level, the UPS switches to internal battery power.

The better UPS units also provide additional types for filters for various kinds of interference, as well as surge and overvoltage protection.

Researching The Problem

I spent considerable time trying to find out about this problem. I knew for a fact that the Minuteman 500 with its four millisecond switching time would work with my '100, so I decided to investigate why it did not work with the '248. And I found out some very interesting things.

It is quite apparent that I am not the only one who has discovered the problem, and it is not limited to my computer alone. I have found people who have had similar problems with 4 millisecond UPS units connected to various Zenith computer models, including the Z-150 series, the Z-200, and the Z-386. I even found one ZDS dealer who also learned about this problem the hard way. Interestingly enough, I found that the same problem occurs in various IBM computers like the PC, XT, and even some AT models.

My informal survey included a wide range of systems. Three were '248 systems, three were '386 systems, and most of the remainder were part of the Z-150 series with a couple of IBM systems mixed in for a total of 14 systems. Although I know a lot of people with various kinds of PC compatible systems, it was not easy to find those who have tried a UPS with their specific system(s). I found that, without exception, those who had tried a 4 millisecond UPS had found that these "slower" transfer times were not adequate. In short, all 4 millisecond units failed, and there were various brands involved.

The Details

Perhaps the most interesting of these failures was the Dallas Heathkit store. They previously had a '248 which was used for their Point of Sale (POS) system, and this had been changed to a '386. What is interesting about this is the fact that they tried to use the PM-450 UPS which is advertised in the current Heath catalog for power backup. They tried two different PM-450 units, and both failed to support either POS terminal.

I also know of another organization that tried the PM-450 on both machines with the same results — failure. Based on my own experience with a different brand of UPS, I don't think the brand has anything to do with the problem. The 4 millisecond transfer time is just too slow for some systems.

In the particular case of the 300 watt PM-450, it is advertised as being able to support "fully loaded Heath/Zenith, IBM and other computers" in the catalog. My survey of field testing of this unit indicates that there are at least some Heath/Zenith computers that the PM-450 UPS will not support. In addition, Para Systems has manufactured backup power supplies for some years, and their experience is also consistent with my survey. They were not at all surprised when I reported that the 4 millisecond Minuteman 500 was not able to successfully support my '248 system. And subsequent testing of my old UPS indicated that it was still within design specifications.

There is a reasonable explanation for this apparent paradox. Although I am sure that Heath tested this particular UPS before it was placed in the catalog, it apparently does not work in the field like it did in a controlled test. As it turns out, the transfer time of a UPS is also directly related to the load on it. Consider two '151s for example — one has 256K of memory and two floppy drives; and the other has 640K of memory, two floppy drives and a 30 MB hard disk. While a four millisecond UPS might handle the first one, it probably would not handle the second because of the increased power requirements.

That is consistent with my survey. All systems had a hard disk and most had additional memory in addition to two floppy drives. It is also consistent to note that a 4 millisecond unit uniformly failed to support these systems, and the brand of the UPS was not relevant. While a four millisecond UPS might support some systems, it is unlikely that it will support hardware configurations with a lot of add-ons, like more memory and a hard disk. Since the fault does not appear as a result of the brand of UPS, my next logical step was to find out some details of the power supply used in these systems.

Computer Power Supplies

I contacted ZDS to find out some of the details of the computer power supply design, particularly for the '248 and '386 systems. I was particularly interested in the design specification for the MAXIMUM "hold-up" (NOT typical) time when the power supply is under full load given that AC power is lost. That number, which should be specified in milliseconds, would tell me what kind of UPS to buy. In contrast, a TYPICAL time is a measured value that may or may not have any relationship to the design specification. A "typical" time may be measured for a 50% load, and it is nearly impossible to use that number for decision-making purposes unless you know the EXACT testing configuration. For my money, I want to know the maximum switching time of a UPS, not the typical time.

In answer to my question, a ZDS spokesman told me that the power supply had been redesigned, but he was not able to give me any details as to exactly what the changes were. Unfortunately, he was also unable to tell me when the design change was made or when it was actually implemented in the production models. While I suspect that the "hold-up" time for the new power supply was probably increased, it was not possible to verify that. Currently available computers in the field, including my own, indicate that the hold-up time has not been changed yet, so it is best to assume that whatever design changes were made did not affect it. It is also important to remember that this problem is not specifically limited to Heath/Zenith systems, and it occurs with some IBM systems, as well. Since this information is not readily available, what is the solution?

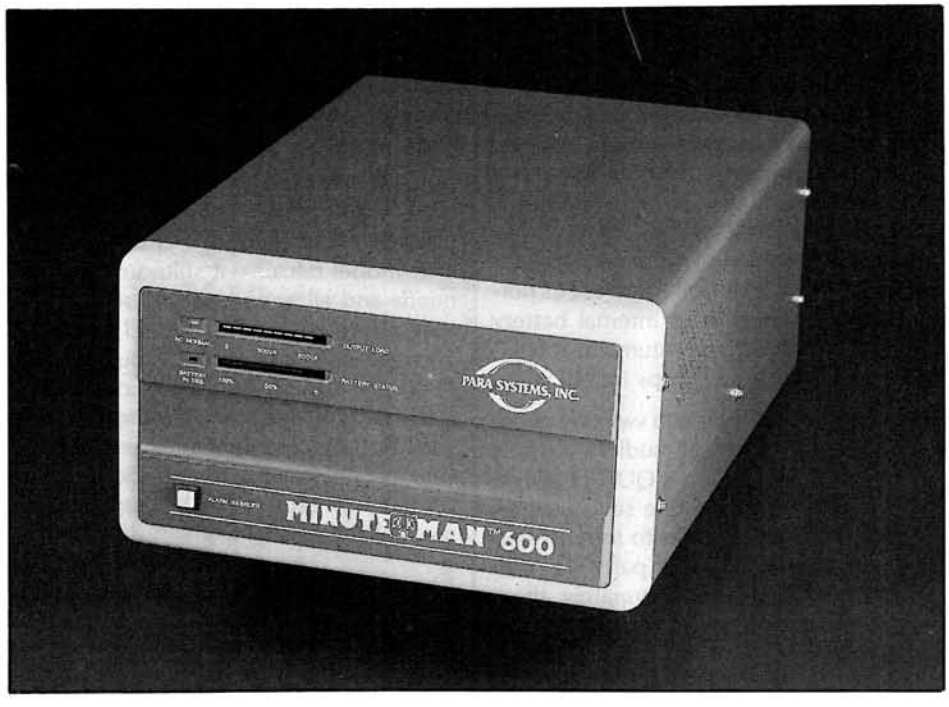


Photo 1
Minuteman 600 UPS

The Solution

My objective was to find a minimum 500 watt UPS with the fastest transfer time of 1 millisecond. Para Systems has the Minuteman 600 which fit my needs perfectly. In addition, it has built-in surge suppression, as well as EMI (electromagnetic interference) and RFI (radio frequency interference) filtering which I also required. The Minuteman 600 is an attractive looking unit which is shown in Photo 1 below.

You can probably see two "lines" in Photo 1 which are the multi-segment LEDs for "Output Load" and "Battery Status". During normal AC operation, the Output Load segments give an approximate indication of how much of the available capacity is being used. For my system, two segments are lit during operation indicating that I am using about one-third of the available load capacity. When the UPS switches to internal power, the Output Load display is turned off, and the "Battery Status" gives you a visual display of approximately how much battery power remains in terms of percentage.

A green LED is used for the "AC Normal" indicator to show that power is coming from the AC line. The red "Battery in Use" light blinks when the batteries are being charged and displays solid red when the unit has switched to internal battery power.

The transfer from AC to internal battery power occurs when the AC line voltage drops to 102 volts, and the unit returns to the AC line at 108 volts. Transfer time in both cases is a maximum of 1 millisecond. This is a reasonable setting that helps prevent "brownout" problems, as well as a complete loss of AC power. The Minuteman 600 also provides overvoltage protection if the AC line voltage exceeds normal limits. It transfers to internal battery power at 135 volts and returns to the AC line when the voltage drops to 129 volts.

When the Minuteman 600 switches to internal battery power, an audible alarm is sounded, and this alarm is QUITE audible. I have been on the phone several times when the UPS transferred to internal battery power, and the other party was also able to hear the alarm. Fortunately, there is an "Alarm Silencer" button which you will see in the lower left-hand corner of Photo 1.

One final note for engineers in the audience — I recognize that AC power is typically specified in VA (Volt-Amperes) for a UPS, but some manufacturers, including Para Systems, use a wattage specification because it is less confusing for most people. I agree. Since power supply information for microcomputer components is commonly listed in the manuals in watts, I have not attempted to discuss AC power in technical terms of phase angle, voltage lead or lag, effective power or apparent power and their relationship to watts. While VA or KVA ratings certainly are appropriate for a mainframe system, most microcomputer components list power supply ratings in watts. For example, my '248 has a 200 watt power supply (Owner's Manual, page 5.5), and the NEC MultiSync draws 76 watts. It is easy to figure out that a 300 watt UPS will provide sufficient backup power because these are the maximum ratings of the power supply. For those users who are not familiar with those terms, there is an easier way to choose a UPS.

Choosing A UPS

For most users, a 300 watt UPS should be adequate to support current microcomputer models (e.g., a '248 or a '386) with a CRT. If you also intend to provide backup power for a printer or other peripherals, or if you want to provide backup power for a longer time, you will probably need a larger unit.

For example, the Minuteman 600 can provide 20 minutes of reserve power for a 300 watt load or 8 minutes of backup at the full 600 watt load assuming that the batteries are fully charged when the power loss occurs. Para Systems also has a smaller unit — the Minuteman 300 — that supplies 20 minutes at 150 watts or 8 minutes at 300 watts. Both have a transfer time of 1 millisecond. I chose the 600 watt model because it suited my current needs and all anticipated future requirements, but the 300 watt model should be adequate for most users. As you might expect, the larger capacity model costs more. If you need even more capacity, Para Systems also makes larger units, as well as additional battery packs that can be used to extend the backup time for the smaller units.

I have been told that the PM-800 (600 watt rating) and PM-1200 (1000 watt rating) advertised in the current Heath catalog also work reliably with the '248 and '386 systems, but I have not personally verified this. Both of these units have a typical (not maximum) 2 millisecond transfer time which is apparently adequate for these systems. Although the PM-450 unit with its 4 millisecond time may work fine on some systems, I do NOT recommend it because I was unable to find any PC compatible system in my local area that could be successfully backed up with ANY 4 millisecond UPS. While I only located 14 systems out of the 14 million or so compatibles that have been sold, I do not recommend equipment that does not generally support my experience. It is interesting to note that the Para Systems' UPS units are not only faster than those listed in the Heath catalog, they are less expensive too.

If you need a UPS, your best bet is to discuss your specific needs with a dealer in that equipment. If you decide to try a "slower" unit, I suggest that you get some kind of return guarantee in the event that it does not work with your system.

In summary, both the Minuteman 600 and Minuteman 300 are highly recommended as being suitable for all Heath and Zenith computer systems. Both feature a 1 millisecond MAXIMUM transfer time, sine wave output, surge protection, EMI and RFI filtering, and overvoltage protection. These features are packaged in an attractive case, and both are available at a very competitive price. If you need UPS protection for your data and your sanity, both are highly recommended.

The Baker's Dozen Utilities

A dozen is 12. A baker's dozen is 13. The Buttonware Baker's Dozen is 14+. More than 14 special utility programs on a single disk — each one unique — and one very useful for very special jobs.

That is the beginning paragraph in the Baker's Dozen manual that gives you an idea of what this software package includes. Depending on how you count the utilities, you can find more than 14 different ones, but let's take a look at each one of the programs.

BTTNCALC uses a spreadsheet format that provides one display page with 20 rows and 6 columns. It has an incredible number of features and can be used as a calculator, although the manual suggests that it is a mini-spreadsheet. This spreadsheet has many of the features of the "big guys" in addition to the generally accepted copy, move, and delete functions. It has standard built- in functions that can calculate trigonometric values, factorials,

combinations, permutations, business-related values (e.g., NPV and IRR) plus decimal to hex conversion. The arithmetic operators include a caret (^) for exponents, and logical comparisons and operations can also be performed. This little spreadsheet follows the usual conventions for formulas and is fast enough to use for casual calculations when you don't want to use your usual spreadsheet.

CALENDAR can be run as a memory-resident or non-resident program that is easily customized to your needs. It has a separate table that you can note your special dates, such as Fred's birthday, and the calendar can handle a wide variety of requirements, such as Chinese and Jewish years.

DISKUTIL is my favorite program in the package because it can do so many things. For example, it can change the Date, Time, and Attributes of a file. I used that feature recently when I copied a number of files from a CP/M disk using the Zenith RDCPM command and found the date for each file was the current system Date. Since most of these files had dates as part of the text (e.g., letters), it was a simple matter to correct the DOS Date of the file with this feature.

This program also has an editing feature that allows you to perform byte-level editing on a disk. You can read and/or modify a file, the disk directory or the File Allocation Tables (FATs). The program also provides information about a disk, such as shown in Figure 1 that was taken for drive C in my '248 system.

As an added feature, DISKUTIL can also help you restore deleted files. Because of that feature, you may find this package is worth the price even if you never need or use any of the other features or programs.

FILECOMP compares two text files similar to the DOS COMP command, but it is more helpful than the Zenith FC command. FILECOMP displays information about which lines in the files match and which ones do not.

GKEY will be of primary interest to programmers because it displays hex and ASCII values for the characters. It can also display the keyboard scan code for a key, as well as the actual value sent to the INT 9 key values sent when a key is pressed or released (i.e., the close and open code).

```
DISK INFORMATION and STATISTICS for DRIVE C

F8 media descriptor      512 bytes   / sector       17 sectors / track
16 bits / fat entry        4 sectors / cluster       5 heads
                        2048 bytes   / cluster    34918 total_sectors

     data type           # of sectors        begins at sector
------------------       -------------       ----------------
     boot sector(s)           1                      0000h
   2 fat tables              35 / table              0001h
 512 entry dir table         32                      0047h
     file data               34812                   0067h

   0 marked BAD clusters          0 bytes ¦     0 clusts in    37 subdir(s)
   0 RESERVED clusters            0 bytes ¦  7740 clusts in   930 files
7782 clusters used          15937536 bytes ¦     0 lost clusters
 921 clusters available      1886208 bytes ¦ -----
8703 clusters on drive      17823744 total ¦  7740 clusters used

last cluster used =  7935     cross coupling noted in    0 clusters

                                          P->Printer F->File Esc->exit
```

Figure 1
DISKUTIL Disk Information

LOCATE helps you find a file on a disk, and in that respect, it is like the SEARCH command that is available in the Zenith MS-DOS. But LOCATE goes one step further — you can also use it to find a specific group of characters in any file on a disk.

P90 is a program that prints files at 90 degrees (i.e., sideways) to accommodate a wide spreadsheet on continuous paper so you are not limited to the 8-1/2" width.

PC__SORT is a filter like the DOS SORT program, but it is much more powerful. PC__SORT can sort on up to four fields in ascending or descending order for each. Sorting can be defined as case sensitive or insensitive.

PRN__FILE is an interesting idea. You may want to create a file with some data, but your program will only print the file. PRN__FILE will take the printer output of a program and redirect it to a file so that you can edit it or whatever.

RDIR is one program that was forgotten by the DOS designers. Have you ever wanted to delete a subdirectory and found out the hard way that it must be empty before the RMDIR command will work? Or found that there was a Read-Only file or something else that you could not delete? RDIR deletes all files in the subdirectory (with a prompt), as well as deleting the subdirectory name, too. This is a particularly useful program for maintaining a hard disk.

SET__SCREEN is used to set a border and screen color using the ANSI device driver similar to the PROMPT command. This is useful when you exit some programs, and they leave strange color combinations on the screen.

SNAPSHOT allows you to take "pictures" of any non-graphics screen for later use in a file. The data for Figure 1 showing the DISKUTIL disk information display was taken with this utility and copied into a WordStar file for this article.

SWCOM12 and SWLPT12 allow you to easily switch your computer's output from a serial port (i.e., COM1 to COM2) or parallel port (i.e., LPT1 to LPT2) in the event you are fortunate enough to have two ports.

The Baker's Dozen — a highly useful set of 14 programs that provide a number of functions that are helpful to just about any computer user. These utilities run on a PC compatible, which includes any current model Heath or Zenith computer. Memory requirements should not be a problem, but BTTNCALC and DISKUTIL do require at least 256K. Perhaps the capability with the most general interest is part of DISKUTIL that can be used to recover erased files. It is an excellent value for the price and is highly recommended.

A special note of thanks to Andy Brunskill (Seattle, WA) for bringing this program to my attention. He is so enthusiastic about this set of utilities that he not only took the time to write me about the Baker's Dozen, he even wrote a letter to Buttonware asking them to send me a review

copy. His thoughtfulness is much appreciated — Thanks Andy.

A Special Note

I know a lot of you have special programs, utilities or special hardware that are your favorites, and you might want to consider writing about them. This can include just about any software or hardware that is currently available. If you don't have the time or inclination to write an article for REMark, let me know about it. I will be writing about various utilities and some interesting hardware over the next few months, but I am always looking for new and different things to write about. Help HUG help you by exchanging your information with other members.

Apple And Pepsi

To its dismay, Microsoft has learned that an Apple and Pepsi do not mix very well. As you may have heard, Apple has filed a lawsuit against Microsoft which alleges a copyright infringement by the new Windows 2.0. This is one of those so-called "look and feel" deals where Apple seems to believe that the Windows interface somehow infringes on its copyright for the Macintosh interface. But this particular case has a peculiar twist that I have not seen before.

It seems that Microsoft and Apple signed an agreement back in 1985 that essentially said that Microsoft could use the same kind of display technology for Windows that was used in the MacIntosh. One interesting point is that this agreement provided for royalty-free use of the technology which apparently included the icons used in the display. For example, a trash can icon is used to illustrate that you want to delete a file. Regardless of what is involved, it appears that Apple and its attorneys feel that they have a reasonably good chance of winning.

As you can imagine, there has been a lot of press coverage of this particular situation, and there have been a variety of views as to why Apple filed the lawsuit in the first place. One opinion is that Apple feels the lawsuit is required to protect their legitimate business interest in the MacIntosh interface. Another opinion is that Apple is having some difficult times, and this lawsuit is a "last ditch" measure. Or perhaps Apple is really trying to stop, or at least inhibit, the development of the Presentation Manager interface for OS/2. That leads to speculation that Apple might really be trying to open the lid on the IBM type of market, since IBM is heavily committed to OS/2. By the way, Presentation Manager is simply a Windows-like interface that "manages" the applications. Regardless of what motivated Apple to file this lawsuit, there is one implication that I feel is critical, but has been overlooked, and we will take a look at that next month.

Powering Down

I've been experimenting with some new ideas and techniques for my articles, and while I won't say exactly what they are, I trust that they will be immediately obvious within the next couple of months. If things work out as planned, I think you will find this column is even more helpful.

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

Products Discussed

Hardware

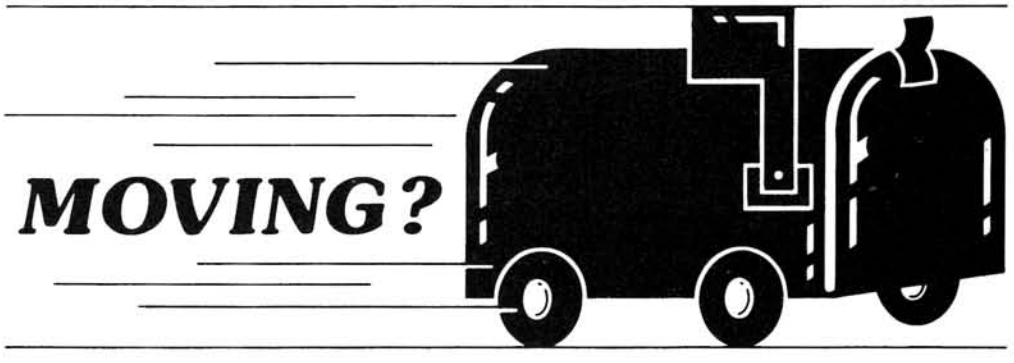
Uninterruptible Power Supply
300 watt (PM-450) $599.00
600 watt (PM-800) 999.00
1000 watt (PM-1200) 1399.00
Heath/Zenith Computer Centers
Heath Company Parts Department
Hilltop Road
St. Joseph, MI 49085
(800) 253-0570
(Heath Catalog orders only)

MINUTEMAN 600 UPS
(600 watt) $899.00
MINUTEMAN 300 UPS
(300 watt) 549.00
Para Systems, Inc.
1455 LeMay Drive
Carrollton, TX 75007
(800) 238-7272 (M/C, VISA, & COD orders)
(214) 446-7363 (Information)

Software

Baker's Dozen $59.95
ButtonWare, Inc.
P.O. Box 5786
Bellevue, WA 98006
(800) J-BUTTON (orders only)

*



Please let us know 3-4 weeks in advance, so you won't miss a single issue of REMark!

How to Get the Most from a Zenith Laptop Computer

by Joseph Katz

Part 1: Enable the Z-183's Drive B; Train the internal batteries for longevity; Save power with the LCD; Save power with the hard disk; Speed up the floppy diskette drives; Use a RAM disk; Install a disk cache.

This series of articles on Zenith's laptop computers provides a collection of tips, tricks, and products that can increase the computers' portability and enhance their utility. You probably won't need everything here, because some of it is useful only in special situations, but much of it ought to be of value and interest to anyone who uses a Zenith laptop computer.

How to enable the free Drive B that's in every Z-183

The MSDOS Version 3.21 *User's Guide* has a section entitled "If You Have Only One Floppy Disk Drive" of interest to users of the Z-183 laptop computer. It has one floppy diskette drive and one hard disk drive. The manual says that the command `COPY A:LETTER.DOC B:LETTER.DOC` should work.

But if you try copying to Drive B with an off-the-shelf Z-183 it will hang up. So if you want to make floppy diskette backups (which you should want to do frequently), you'll need to do it the hard way: copy the files from the source floppy diskette to the hard disk; then swap the source diskette for a blank floppy diskette; then format the blank floppy diskette to make it the backup destination; then copy the files from the hard disk to the destination floppy; and then—at last—erase those now-useless files from the hard disk to release their space.

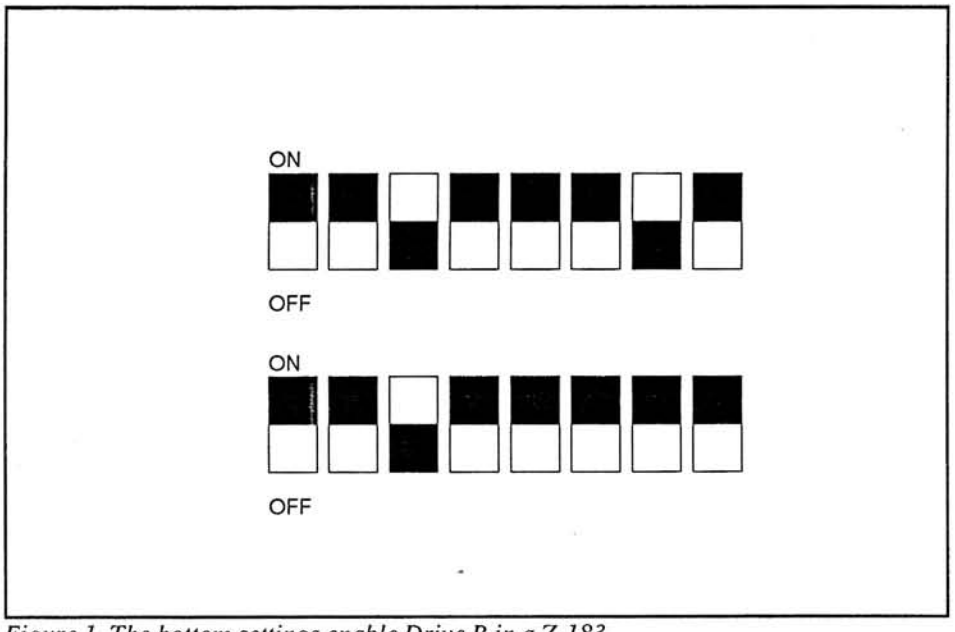


Figure 1. The bottom settings enable Drive B in a Z-183.

It's indeed much simpler to use the MSDOS commands that employ two drives, espe-

Address all correspondence to Dr. Joseph Katz, 103 South Edisto Avenue, Columbia, South Carolina 29205.

cially DISKCOPY A: B: and let the Z-183 do most of the work. You'll have to swap disks one or two times, but that's less pesky than the folderol involved in employing the hard disk as a staging area for what should be the simple task of backing up floppy diskettes. With a Z-183, however, you first must enable Drive B.

You can enable Drive B yourself in a few seconds. Make sure the power on your Z-183 is off and its case is snapped shut. Then turn the computer bottomside up. In the middle of the underside of the computer's case is a rectangular rubber plug. Pry it out with your fingernails and you'll see a DIP switch with eight switches. It will be set as in the top row of switches in Figure 1. Slide Switch 7 from OFF to ON as in the bottom row of switches in Figure 1. Now replace the rectangular plug securely, turn the computer rightside up, power it on, and enter the command DIR B: to see what happens. What happens is that you'll be prompted to PUT DISK B IN DRIVE A:. PRESS ANY KEY WHEN READY.

What you've just done is enable the physical "Drive 0" (which is the way MSDOS and IBM compatible computers identify the first floppy diskette drive in an IBM compatible computer) as two "logical" drives, Drive A and Drive B. Now you can use Diskcopy, Copy, and other commands so your Z-183 will operate as if it has two floppy diskette drives. It does, but they are "logical" (which means "conceptual") and not "physical" (which means "tangible").

The reason why you had to enable Drive B is that the Z-183 is factory preset as a dual-floppy-diskette machine. Your off-the-shelf Z-183 therefore will hang if you try to access Drive B, because the computer has been misinformed and told to seek a second physical floppy diskette drive. It does . . . until the end of time or until you reboot, whichever comes first. Setting Switch 7 to ON configures the machine for one physical floppy diskette drive. MSDOS then understands that when you try to access Drive B you mean a logical instead of a physical floppy diskette drive, and everything works as it should.

Train the internal batteries for longevity

The internal power in your Z-180 laptop computer comes from a 12-volt "NICAD" ("Nickel Cadmium") battery pack. In the Z-183 it's either the standard model ZA-180-45 or the optional HA-180-40: the former is rated at 2.4 amps per hour; the latter supports a power draw of 4 amps per hour. Your Z-180 laptop computer draws .7 amps. Under ideal conditions you'll get a maximum of about 3 hours of battery-powered computing from the standard battery and a maximum of about 5 hours from the optional battery.

How you treat the battery pack in your Z-180 laptop computer goes a long way to determining how close your own conditions are to being "ideal." There's often much you can't control, and you don't want to be a slave to your computer anyway, but there are some things you can do easily to get the most out of each battery charge. The rest of this article explains different things you can do. This section concentrates on the training and feeding of the battery pack.

About the best thing you can do is forget any notion about keeping the internal batteries fully charged at all times. Don't. You can't do it and you mustn't try. If you're the prudent kind of person who tops off your automobile's gasoline supply whenever it drops below your own level for happy motoring, you might have an especially tough time believing that a similar attempt at prudence will weaken and eventually kill the NICADs in your Z-180 laptop computer.

The reason is that NICAD batteries have a kind of "memory"—not RAM nor ROM nor any other silicon memory device but a chemical kind of remembrance. The NICADs remember and adapt quickly to the usage patterns you set for them. So if you establish the pattern of topping off the batteries by recharging the battery pack after every computing session, in a short time all you will be able to get from it is power for what has been your average session. You won't get anything near the ideal maximum. The NICADs remember. For that same reason you don't want the battery pack to get lazy and expect to be always charged to full capacity. In time the battery pack won't work at all.

So, as your *Owner's Manual* says, don't leave the Z-180 laptop computer plugged into the AC adaptor when the batteries are fully charged and the computer is not in use. You'll destroy the ability of the batteries to take or maintain a charge. When the computer is not in use and you don't want the batteries recharged, unplug the AC adaptor from the electrical outlet or the computer.

Nor do you want to recharge the batteries until they're fully discharged. Don't top off the batteries. Don't recharge them until the computer won't do a thing on battery power. Then recharge the batteries to full capacity—about eight to twelve hours, according to the *Owner's Manual*.

If you've just unpacked a brand new Z-180 laptop computer from its shipping carton, invest some time in training the batteries to their ideal life cycle of full discharge followed by full charge. With the AC adaptor unplugged, turn on the computer and let it completely discharge. You want it to become a paperweight. Then turn off the computer and plug in the AC adaptor. With the computer off, let the batteries charge for about eight to twelve hours. Then repeat the cycle of discharge and recharge two more times.

At that point your computer is ready for use. From then on, run the batteries until they drop before you recharge them, then recharge them completely.

Save power with the LCD

You can squeeze more time between battery recharges if you're crafty about how you set that beautiful LCD ("Liquid Crystal Display") whenever you operate your Z-180 laptop computer on its internal battery pack. Make it temporarily less beautiful.

The LCD draws power. A rule of thumb is that the longer it's on and the sharper it looks the more power it's drawing. You won't care when power is coming from the AC adaptor or other external source. When you're running on batteries, however, it pays to be miserly about the cost of supporting your proud beauty. What you need to do is both cut the time during which the LCD draws maximum power and limit the amount of power the LCD draws during that time.

To limit the LCD's maximum power consumption, set the CONTRAST and BRIGHTNESS slider switches above the keyboard so they're as near their MIN positions as your eyes can tolerate.

To limit the amount of time during which the LCD draws maximum power, use the MSDOS Mode command to adjust down the timeout value of the computer's screen blanker. You'll notice that the display on the Z-180 laptop computer will dim after two minutes of keyboard inactivity. Depending on how you use the computer, you might be able to conserve power by resetting the screen blanker's timeout value to one minute: the command is MODE EL 1. The EL switch for the Mode command is calibrated in *minutes* from 0 to 12. The value 0 keeps the display's backlight on all the time, which is what you don't want while operating on the battery pack. But it might be just what you *do* want while you're working on external power.

Here's a related tip. When you're ready to begin typing again you need to unblank the

screen. You may have noticed that the screen relights when you press any key except the FN key. You also may have noticed that most of the keys you press to relight the screen are passed through to your program. It's annoying to have to erase those keystrokes input into a program each time you only want to unblank the screen. To avoid that annoyance try one of these keys: CTRL, SHIFT (left or right), or ALT (left or right). Few programs take any action when one of those keys is pressed alone, without some other key pressed simultaneously, so they're relatively safe bets when all you want to do is unblank the screen without passing a keystroke to the program.

Save power with the hard disk

That hard disk drive in a Z-183 laptop computer is a joy, but it does consume more power than the floppy diskette drive. You can demonstrate their relative appetites the next time you're draining the battery pack to prepare it for a full charge. Soon after the LOW POWER warning light and beeps begin, there's not enough power to use the hard disk drive. But you can still use the floppy diskette drive, and you can do so for quite some time.

What you ought to think about, therefore, is using floppy diskettes instead of the hard disk drive whenever possible for the work you do on battery power. Keep your programs on the hard disk drive but use a floppy diskette for your data and work off the floppy diskette drive when you're running on batteries. Then, when you're back on AC power, copy the data files from the floppy diskette back to the hard disk drive. It's not such a silly idea as you might think at first. It will extend the time you can use the battery pack before it needs recharging.

When you're working on battery power and using a floppy diskette for your work, you'll usually come out ahead if you use an undocumented option in the MSDOS Version 3.21 Mode command to set the hard disk motor of the Z-183 so it powers down immediately after each hard disk access. (A "hard disk access" is what the computer does when it must read from or write to the hard disk.)

Of course you shouldn't try anything so self defeating as sorting a big database on battery power. Avoid all such disk-intensive operations until your computer is plugged into a reliable source of external power. Most other times, as when you're writing with a word processing program or using a spreadsheet program on battery power, you might as well have the hard disk motor power down as soon as possible. The command is MODE MOTOR 1. The Motor option is calibrated in *seconds* from 1 to 1200. The value 0 keeps the hard disk motor on all the time, which is what you don't want while operating on the battery pack.

Of course your word processing, spreadsheet, or other program will turn the hard disk motor on automatically whenever it absolutely has to access the hard disk on a Z-183, as it will if it's the kind of program that stores parts of itself in overlays. (See the section below entitled "Make programs fit in RAM.") There'll be a moment's delay before each hard disk access, and you will become irritated if your program requires numerous hard disk accesses to carry on the sort of business you require from it. But the longer operating time you'll get from the battery pack between recharges ought to compensate for the annoyance, and you might even be able to reconfigure your program so it fits entirely in RAM and doesn't depend on overlays.

Speed up floppy diskette drives

You'll save some battery power by resetting the floppy diskette drive to start sooner than it does as shipped by the factory. Then, when your program needs to read from or write to a floppy diskette, there won't be a momentary delay before each floppy diskette operation. You'll enjoy the instantaneous response.

You'll have to do a little programming, though. But don't fret. Zenith has supplied the only programming tool you'll need: DEBUG.COM, on the MSDOS distribution disk that came with your Z-180 series laptop computer. And I've provided a script you can make so that Debug will create the program for you: it's in the sidebar entitled "How to Make ZFASTER.COM". Follow those instructions precisely.

When you're finished you'll have an executible program called "ZFASTER.COM" Make sure you test ZFaster from a floppy diskette in case you did something wrong. After you run it, try accessing the floppy diskette by calling for a directory listing of Drive A. The green access light should go on immediately, the signal that Drive A has snapped into action. When you're sure the program works, copy it to your boot disk and run it once each time you boot your computer: include the command to run it in your AUTOEXEC.BAT.

(I'm indebted to Pat Swayne, Software Engineer for the Heath/Zenith User's Group, who suggested the basic idea for this speedup.)

Make programs fit in RAM

No matter which Z-180 laptop computer you own, you'll avoid the nagging heartbreak of excessive disk accesses if your application program—your word processing program, for example, or your database manager— fits entirely in RAM ("Random Access Memory").

"Disk accesses," by the way, take place whenever your computer reads from or writes to a disk, whether it's a floppy diskette or, in the Z-183, the hard disk. The reason why disk accesses are an important factor in determining how much battery-powered computing time you get is that they draw much more power than any other operation. They're important enough to deserve special consideration: look above for the sections "Save power with the hard disk" and "Speed up floppy diskette drives."

Some conventional "transient" programs, those that occupy RAM only when they are running, are so large that they can't fit entirely into RAM all at once. What they do is store parts of themselves on disk in subprograms called "overlays." Then, when they need a feature programmed into one of those overlays, they swap it into memory to replace part of the main program. Of course that swap involves at least two disk accesses—one to read in the overlay, the other to reread the feature that has been swapped out of memory to make space for the overlay.

If you're absolutely married to a program that uses overlays, see if you can create a leaner version for use on your Z-180 portable computer. Some programs, such as WordStar or XyWrite III Plus, can be reinstalled or reconfigured to make a slimmer version that will run entirely in RAM. You'll sacrifice some features: the online help in WordStar, for example. But when you really don't absolutely need those features and you really do need as much time as possible of battery-powered operation, you'll save power by shoehorning the entire program into RAM.

Use a RAM disk for program overlays

Some programs, such as WordStar, can be configured to look for their overlays in a place different from that which holds the main program. If you must have all the program's features, configure the version you use on your Z-180 laptop computer that way.

Then have the CONFIG.SYS file on your computer install a RAM disk with the VDISK.SYS driver on your MSDOS distribu-

How to Make ZFASTER.COM

Use EDLIN.COM (on your MSDOS distribution disk) or another editor to type the script at the end of this sidebar into a file named "ZFASTER.DEB." (If you employ WordStar or other word processing program, make sure you use its "non-document" mode: you don't want any formatting characters or other control codes in the file you make. Type *everything* below and type it *exactly* as it appears here, down to the spacing between characters. Proofread carefully. Your completed ZFASTER.DEB should have exactly 340 bytes in it: check the disk directory when you've finished.When your ZFASTER.DEB is perfect, copy it and DEBUG.COM to the same floppy diskette. Issue this command: `DEBUG <ZFASTER.DEB`. You'll see the script file scroll up the display. When it's finished, you should see a new file, ZFASTER.COM, in the directory. It should be precisely 294 bytes in a directory listing. If it's not, you made some kind of error in ZFASTER.DEB. (Make sure you distinguish the numeral "0" from the letter "O.")

```
a
push es
mov ax,0
mov es,ax
es:
mov byte ptr [0579],0
es:
mov byte ptr [057a],0
pop es
mov dx,11b
mov ah,9
int 21
ret
db "ZFASTER - Copyright 1988 by Joseph Katz [Version 1.0]",0d,0a
db "The diskette drives on your Zenith laptop computer",0d,0a
db "have been set for FAST access.$",0d,0a

RCX
126
n ZFASTER.COM
w
q
```

tion disk. Appendix D of your MSDOS *User's Guide* says how. You'll have to do some arithmetic and calculate the size of the RAM disk to install with VDISK.SYS: total the disk space used by the program's overlays. Make sure to copy the overlays to the RAM disk before you use the program. (If it's the program you use mostly, you might want to have the copying automated whenever you boot the computer. Put a line such as `COPY C:\WP\*.OVR E:`—or whatever is appropriate to your system—in the AUTOEXEC.BAT.)

Check your program's manual for information that might tip you off about the desirability of copying the main program file to the RAM disk too. If there's no such information, observe what happens when you run the program. Should the disk drive lights flash on at times when you're not reading from or writing to a data file, the program probably is accessing the main program file after it's through with the overlays. In that case you should copy the main program file to the RAM disk too.

Install a disk cache

When what you do involves repeated accesses of the same disk areas—as when you rely on a program that must keep parts of itself in overlays, or if you just like typing DIR repeatedly—a disk cache can save power and give you more time when computing on the battery pack.

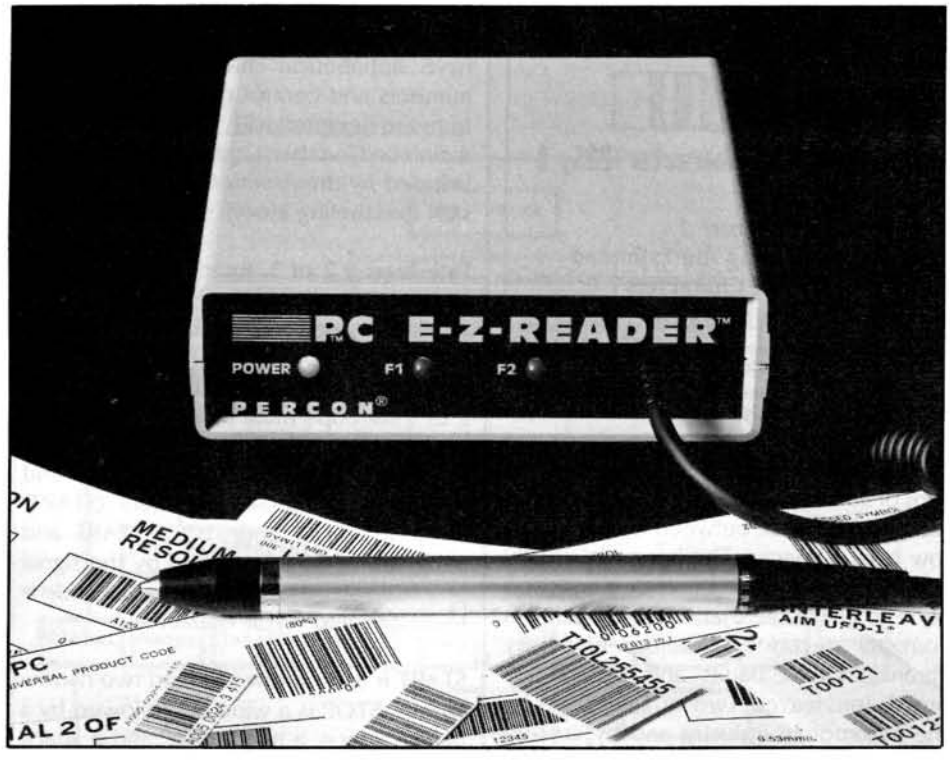
The idea behind a cache is that repeatedly-accessed data—such as are stored in a program's overlays—are identified by the cache program and stored in RAM for later use. RAM accesses are faster and use less power than disk accesses, so not only do things go faster but also they use less power.

ZCACHE.SYS on your MSDOS distribution disk is a cache program made especially for your Zenith computer. You'll find instructions in Appendix D of your MSDOS *User's Guide*. Ignore the `/e` and `/a` switches mentioned in that appendix. An XT compatible such as the Z-180 laptop computer can't have EMS memory and your off-the-shelf Z-180 laptop computer does not have Extended Memory. Unless your computer is a floppy-only Z-181 you probably ought to follow the advice in the *User's Guide* about ignoring the `/f` switch. If you do own a Z-181 you should be able to use the `/f` switch to good purpose if you keep your programs on one floppy diskette (Drive A) and your data files on the second floppy diskette (Drive B).

*

Low Cost,

J. Richard Hines
Oholiab Technology
P.O. Box 851731
Richardson, TX 75081



High Speed Data Entry: Bar Codes

If you own your own business or have a friend who does, you may have already thought about building a computer-based inventory management system. Using bar codes to identify products will simplify your system.

What Is Bar Code?

Bar code is digital data in bars and spaces, instead of ones and zeros or highs and lows. A wide bar or wide space represents a one and a narrow bar or space represents a zero. For example, four bars and four spaces can represent any of the 256 extended ASCII characters inside your PC. Figure 1 shows the extended ASCII character 0 (character 00110000_2 or 48_{10}) and β (character 10100001_2 or 225_{10}) in bar code.

No one would try to use only four bars and four spaces to represent all 256 extended ASCII characters, however. Without the dashed lines in Figure 1, it is impossible to tell the difference between zero (00110000) and one (00110000) or between alpha (10100000) and beta (10100001). One possible solution would be to use four bars and four spaces to represent the one hundred twenty-eight standard ASCII characters (the first one hundred twenty-eight extended ASCII characters). The first four bars and three spaces could represent the character and the fourth space could be an "end of character" marker which can be either a wide space or a narrow space. This code, too, has problems since it does not have any error detection. Most practical codes use more bits or represent fewer characters so there is information for error detection imbedded in the code.

Reading Bar Code

To read bar code, you need a light-emitting diode/photodiode pair. Most low-cost bar code readers place the LED/diode pair inside a pencil-sized cylinder called a wand. When the wand is dragged across the bar code, light from the LED is reflected by the spaces, but not the bars. The reflected light is detected by the photodiode. (This is shown in Figure 2 below.) Information about the presence or absence of reflected light is sent from the wand to additional signal processing circuitry. The additional circuitry translates this information into standard ASCII characters which are sent to your computer.

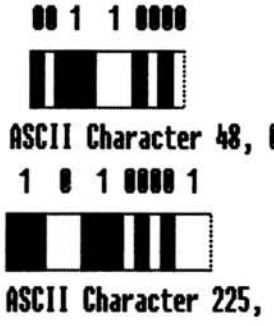


Figure 1
Representing the Extended ASCII Characters ß and 0 with Bars and Spaces

Kinds Of Bar Code

Even with very good LEDs and diodes and a lot of additional circuitry, it is hard to detect the difference between wide and narrow bars or spaces. The bar code usually contains some sort of error correction code to guarantee that the information is correctly decoded. Codabar (used by libraries, blood banks and Federal Express), Interleaved Two of Five (used by the automotive industry and by custom applications), UPC or Uniform Product Code (used by retailers) and Code 39 (used by the automotibe industry and by the military) are the most common codes with error correction. Each has its advantages and disadvantages.

Codabar. Codabar allows you to write twenty different characters in bar code: the decimal numbers (0 through 9), six control characters (-, $, :, /, ., and +) and four start or stop characters (a/DC1, b/DC2, c/DC3, and d/DC4). Codabar bar code must be at least three characters long (a start character, one digit or control character, and a stop character) and cannot be longer than 64 characters (a start character, 62 numbers or control characters, and a stop character). The start and stop characters are usually not transmitted from the signal processing hardware to the computer when the bar code is translated into ASCII characters.

Each Codabar character requires four bars and four spaces: the first four bars and three spaces represent the character and the last space respresents "end of character". The character consists of carefully chosen combinations of two or three wide bars or spaces and four or five narrow bars or spaces. This simplifies error detection. Figure 3A shows the Codabar code for the number twelve thousand, three hundred forty-five.

Codabar works well so it will continue to be used in existing systems. However, newer bar codes either require fewer bars and spaces for the same information or have alphabetical characters, as well as numbers and control codes, so new systems are designed with those codes. A variation on Codabar, Codabar/ABC was developed by the American Blood Commssion for labeling blood.

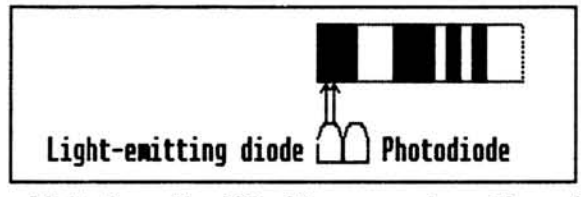
Interleaved 2 of 5. Interleaved 2 of 5 allows you to write twelve different characters in bar code: the decimal numbers (0 through 9), a start character (START) and a stop character (STOP). Interleaved 2 of 5 bar code must be an even number of characters, at least four characters long (START, two digits, and STOP) and not longer than 34 characters (START, thirty-two digits, and STOP). START and STOP are never transmitted by the signal processing hardware when the bar code is translated into ASCII characters.

START is two narrow bars and two narrow spaces, STOP is a wide bar followed by a narrow space, a narrow bar, and a space which can be either narrow or wide. The pairs of Interleaved 2 of 5 digits require five bars and five spaces: the bars represent the first digit and the spaces the second digit. This simplifies error detection.

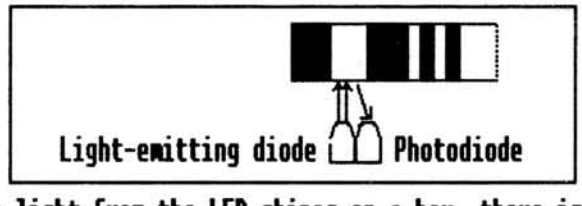
Interleaved 2 of 5 contains almost twice as much information per bar as codabar, so it is often used in custom industrial applications where information density (characters/bar) is important. It is used in few standard applications, however, and does not allow letters, as well as digits, so it is probably not a good first choice unless you need high information density. Figure 3B shows the Interleaved Two of Five code for the number twelve thousand, three hundred forty-five. (It is written 012,345 since Two of Five cannot represent an odd number of bits.)

UPC (Uniform Product Code). Uniform Product Code is a complicated bar code system used on products sold in retail stores. Packages for all grocery and drug items must carry a UPC bar code. UPC A code requires eleven digits to identify the product, UPC E code requires only seven digits to identify the product. (There is an algorithm which translates any UPC E code into UPC A code, but not all UPC A codes can be translated into UPC E codes.) UPC code is complicated and requires a large database, which will take a lot of time to enter. If you are interested in UPC, it is best to buy a turnkey system from a vendor with UPC experience.

Code 39. Code 39 (sometimes called Code 3 of 9) allows you to write 44 different characters in bar code: the decimal numbers (0 through 9), the capitol (A through Z), seven control characters (-, $, :, /, ., and +) a space (SPACE) and a start/stop character (START/STOP). Code 39 bar code must be at least three characters long (START/STOP, a digit, letter, space or control character,



When light from the LED shines on a bar, there is no reflected light.



When light from the LED shines on a bar, there is reflected light which is detected by the photodiode

Figure 2
An Ideal Bar Code Reader

a12345a

Figure 3A
Codabar/ABC

012345

Figure 3B
Code 2 of 5

12345

Figure 3C
Code 39

12345

Figure 3D
Code 39

START/STOP) and not longer than 82 characters (START/STOP, eighty digits, letters, spaces or control characters, START/STOP). START/STOP is never transmitted by the signal processing hardware when the bar code is translated into ASCII characters.

Each Code 39 character requires five bars and five spaces: the first five bars and four spaces represent the character and the last space represents "end of character". The character consists of carefully chosen combinations of three wide bars or spaces and six narrow bars or spaces. This simplifies error detection.

Code 39 is alphanumberic, so it is desireable in any application where a human is going to look at the ASCII translation of the bar code. Also, it is required for many government contracts. It is a good choice for your custom system, unless you need high information density. Figure 3C shows the Code 39 code for the number twelve thousand, three hundred forty-five.

Buying A Bar Code Reader

There are five or six manufacturers of low-cost ($300-$600) bar code readers who advertise in various computer magazines. All manufacturers make about the same claims for their units, but not all work well, so the best policy is to try it before you buy it. Bar codes printed on a 120 dpi (dots per inch) dot matrix printer, like a Citizen 120D or a Panasonic 1081i are a good test of the reader. Reading a copy of a copy is also a good test of the reader. If you can't try before you buy, get recommendations from someone who is currently using a bar code reader. If they like their reader, buy from that manufacturer. If they don't like their reader, buy from a different manufacturer.

I recommend the Percon E-Z-Reader™ shown in Figure 4: It works well with bar codes printed on my Panasonic 1081 and will read Codabar, Interleaved 2 of 5, UPC and Code 39 bar codes. The manual that comes with the E-Z-Reader is surprisingly good, too: it is easy to read, complete, has good pictures and contains a lot of helpful hints.

RS-232C SERIAL VERSION

KEYBOARD COMPATIBLE VERSION

Figure 4

Percon, like most bar code reader manufacturers, make two kinds of bar code readers: a reader which is placed in parallel with your keyboard (data is treated like keyboard input and dumped into the keyboard buffer) and a reader which is connected to an RS-232 port on the back of your PC (data is treated like moderm input). You can choose the model which suits your needs. The keyboard input model is best, if you are simply adding bar code to an existing inventory control system: data entered from the bar code reader is treated as data entered on the keyboard so you don't have to modify your program. I would not use the RS-232 model unless I had a PC or terminal whose keyboard did not use a DIN connector.

Buying Bar Code Writing Software

It is possible to write your own bar code writing software. However, it may not work too well at first. Two features you want in your bar code are a text translation of the code (usually printed below the code) and comments (usually written above the code). These two features simplify your inventory control system a lot. When the bar code reader does not work, the operator can read the direct translation of the bar code and enter the code into the computer with the keyboard. When the computer is down and the operator must find an item manually, the comments (usually called HRI or human-readable-interface) make this easy to do. Direct translation and HRI are shown in Figure 4.

If you want software that works right away, buy bar code writing software from the company that makes your reader. It is reasonable to assume that the two work well together. Percon's Writebar™ software works very well with the Percon E-Z-Reader. Most of the bar codes in this article were printed with Writebar. Writebar can be used in batch mode to print labels manually entered by the user, or it will print labels directly from data in a dBASE-compatible file. This makes it very easy to use.

Some companies advertise code writing software that can be called from BASIC, Pascal, or C programs. If the software works and you have the time to write the code, it might be better than Writebar. Try to evaluate this software before you buy it, however. None of the software vendors will be willing to supply code for me to test for this article, which makes me suspect it does not work well.

Summary

Bar code is so simple that you should be using it now! If you have questions, consult Percon or another bar code company and get started today.

Software For Inventory Management

There are many good programs you can use to write very good custom inventory control software. If you are building a simple inventory management system and want to write as little code as possible, there are several database programs that will do almost everything you want to do.

VP Info from Paperback Software is a nice, low-cost package. It is similar to dBASE II and dBASE III. It even reads dBASE files, and VP Info files can be read by dBASE. You will probably need twenty hours of study to become familiar with VP Info. (To do anything fancy in VP Info, you will need to be familiar with VP Info commands, since it is not menu driven.) VP Info is a compiler, so it is considerably faster than dBASE or R:BASE.

R:BASE V from Microrim is a nice, expensive package. It is very different from dBASE or VP Info. It must translate dBASE files to read them and R:BASE files cannot be read by dBASE unless they are converted to dBASE format. You will probably need an hour of study to become familiar with R:BASE System V. (To do something fancy in R:BASE, you simply follow the fill-in-the-blank menus.) A companion program, R:BASE RUNTIME allows you to make "compiled" copies of R:BASE programs which run without a copy of R:BASE V. However, the cost per copy is still higher than the cost of VP Info.

VP Info is the best way to quickly create a simple inventory management system if you have more time than money, or if you want to work with dBASE-compatible files. R:BASE System V is the best way to create a simple database if you have more money than time, or if you want to work with R:BASE files.

If you are building a complicated inventory management system and have to write a lot of code, there are several packages that will do everything you want to do and more.

dBC III allows you to write code in Lattice C which will manipulate databases created by VP Planner or other dBASE-compatible database programs.

R:BASE Program Interface allows you to write code in Microsoft MS C, MS Pascal or MS Fortran, which will manipulate databases created by R:BASE System V. However, it is very expensive to program with the R:BASE Program Interface. You must purchase both MS C and R:BASE V to use the R:BASE Program Interface.

Turbo BASIC Database Toolbox and **Turbo Pascal Database Toolbox** allow you to write code in Turbo BASIC or Turbo Pascal, which will create and manipulate B-tree databases. If you do not know C, this is probably your best choice, but you will have to do much more work. It is the least expensive way to write the software. However, your files will be difficult to transport to a standard database program.

dBC and Lattice C are the best ways to write software for a complicated management system, if you get dBASE-compatible files from your database program or already use Lattice C. **R:BASE Program Interface** and Microsoft C are the best ways to write software for a complicated inventory managment system, if you use R:BASE V or already use MS C. The **Turbo Database Toolbox** is the best way to write software for a complicated inventory management system, if you program in either BASIC or Pascal.

These tools really make it easy to write inventory management software for your PC. If you are interested in learning more about writing inventory management software in C, you should read Al Steven's C *Development Tools for the IBM PC* (Brady: 1986). About half this book deals with file management and databases.

*

Percon E-Z-Reader™
Peripheral Connections, Inc.
2190 W. 11th
Eugene, OR 97402
(503) 344-1189

R:BASE System V
R:BASE Runtime
R:BASE Program Interface
Microrim
P.O. Box 97022
3925 159th Avenue NE
Redmond, WA 98073-9722
(206) 885-2000

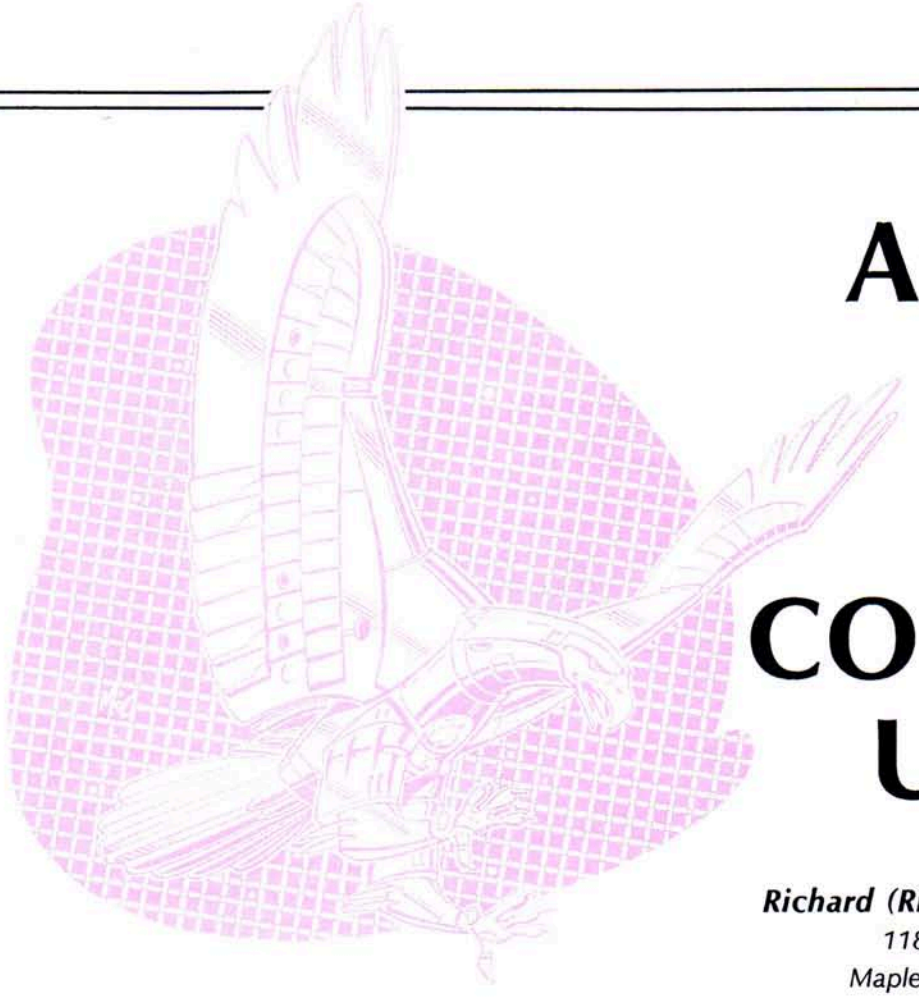
Turbo BASIC
Turbo BASIC Database Toolbox
Turbo Pascal
Turbo Pascal Database Toolbox
Borland International
4585 Scotts Valley Drive
Scotts Valley, CA 95066
(408) 438-8400

VP Info
Paperback Software
2830 Ninth Street
Berkeley, CA 94710
(415) 644-2116

dBC III and Lattice C
Lattice Corporation
P.O. Box 3072
Glen Ellyn, IL 60138
(312) 858-7950

MS C, MS Pascal, and MS Fortran
Microsoft Corporation
10700 Northrup Way
Belleview, WA 98004
(206) 882-8080

*



Another CONDOR 3 Update

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

Introductory Comments

The purpose of this article is to give you, the Heath/Zenith CONDOR 3 users, an update on the latest version of CONDOR 3 (V2.20), the Relational Database Management System from Condor Computer Corporation. This version was released back in the Fall of 1987. Just prior to release, I had the opportunity to do a quick Beta Test on this version and was quite pleased with the materials and quality of the product. However, I must apologize for not getting this article out sooner.

This is my third article on Condor Computer Corporation's products. Back in Volume 6, Issue 11 of REMark (November 1985), I wrote an article on the then latest release of CONDOR 3 (V2.11.12). Following that, I wrote an article on Condor's "New Generation Editor" in REMark Volume 8, Issue 10 (October 1987). Now I would like to continue with my articles on Condor Computer Corporation products.

Version 2.20

With the release of Version 2.20 of CONDOR 3, a number of enhancements were made as well as some new features. Also the documentation was enhanced. In the next few pages, I would like to discuss the new features and enhancements in some detail.

Path Specification

The first feature is the "Path" capability. In previous releases, it was possible to have the Condor System files and the datasets on two different drives (hard or floppy) and in sub-directories as long as the current working directory for each drive was set to the directory of the Condor System and to the directory of the datasets respectively. However, it was not possible to have the Condor System files and the datasets on the same drive in different directories. There just was no way to specify directory paths, all one could do was specify a drive letter. With Version 2.20, having the Condor System files and datasets on the same device but in different directories is now possible.

With Version 2.20 of CONDOR, one can now use the **SET MASTER** command to specify the directory that contains the Condor System files. Since this is somewhat confusing, let me use an example to show how one specifies different directories.

Assume the following: the hard disk drive that will be used is drive C:, the Condor System files will be in directory \CONDOR3, and the dataset files will be in directory \DATABASE. First, set the current working directory to \CONDOR3 using the MS-DOS/PC-DOS **CD** command. Then load CONDOR 3 using the command **DBMS C C**. After CONDOR 3 is loaded, then enter the following two commands:

```
SET Master C:\CONDOR3\
INSTALL
```

Please note that CONDOR 3 requires that the user enter a trailing backslash (\) following the Condor System directory name. Omission of this backslash, results in an error termination of the command. Exit CONDOR 3. Using the MS-DOS/PC-DOS **PATH** command, specify CONDOR3 as a directory to search for the **DBMS** command.

Next there are two situations depending on whether the user chooses to use the CONDOR 3 Menu System or not. If the

Menu System is not to be used, then set the current working directory to \DATABASE. Then enter **DBMS C C**. This will load CONDOR 3 which is in another directory. The user is now ready to use CONDOR 3.

If the CONDOR 3 Menu System is to be used, then two files need to be copied from the Condor System directory to the dataset directory: MENU.DBM and MESSAGES.MSG. Then set the current working directory to \DATABASE and proceed as above.

Although my example showed that both the Condor System directory and the dataset directory were both in the hard disk's root (main) directory, this does not, however, have to be this way. One could decide that the dataset directory DATABASE was a subdirectory of directory CONDOR3. Then before calling CONDOR 3, the user must set the current working directory to \CONDOR3\DATABASE and proceed as above.

Multiple Index Files

CONDOR 3 now allows the user to generate multiple index files for the same dataset. Just as with the SET MASTER command, I will use an example to better describe this concept. Assume that the user has a dataset with the title of EMPLOYEE.

The first step in generating multiple index files is to create an alternate title (name) for dataset EMPLOYEE. Remember that the Data Dictionary contains information about each of the datasets on the user's disk and the Data Dictionary itself is a dataset. Because of this last fact, the user can enter a new record into the DATA.DIC file via the **ENTER** command.

Specify a new title, say WORKER, and enter the word EMPLOYEE for the Data, Definition, and Form fields. Leave all other fields blank and exit the ENTER command. The user can now type LIST WORKER and upon exit, type LIST EMPLOYEE and one will see that the contents of both are the same. They should be; only an alternate title was created.

Generate index files for both titles via the **INDEX** command and specify field name EMPNO as the key for EMPLOYEE and NAME as the key for WORKER. Now the user has two index files using different keys referring to the same dataset. Remember that the two index files are not related; they are two distinct index files. When changes are made to the dataset with the title EMPLOYEE, only the index file related to this dataset is affected, i.e. file EMPLOYEE.IDX. In this case, file WORKER.IDX is not affected; likewise, when making changes to the WORKER dataset.

The user now has the advantage of retrieving information, using the above example, either by EMPNO or by NAME, both fields in the original dataset. Although only two index files were generated in this example, any number can be generated. I am not aware of any limit at this time.

UNDELETE

The **UNDELETE** command allows the user to recover deleted records in certain cases. First of all, records deleted from a dataset via the EMPTY, DEFINE, and DESTROY commands can NOT be recovered with the **UNDELETE** command. However, those records that were deleted using the DELETE and UPDATE commands can be recovered.

You experienced users may recall that when a record is deleted using the DELETE or the UPDATE command, the status of that record is changed from "active" to "inactive." The "inactive" record stays in the dataset until the dataset is sorted. At that time, the record is actually removed from the dataset. The **UNDELETE** command just changes the "inactive" status back to "active." However, if the file is sorted, all "inactive" records are removed and **UNDELETE** can NOT recover any deleted records in this case. So be cautious when using the DELETE, UPDATE, and SORT commands.

Printing

The **PRINT** and **LIST** commands can now support up to 255 character wide reports. This was limited before to 80 characters, and everything over 80 characters was wrapped around to the next line. Now with the 255 characters, one can use compressed print on a dot matrix printer with a wide carriage and get better than 3 times the information on a line. Reports look neater, more professional.

File Size

In previous versions of CONDOR 3, as well as this version, CONDOR 3 will support up to 65,534 records, each with up to 1024 bytes. However, in previous versions, there was a 4MB (Mega-Bytes or more simply, Million Bytes) limit on the data file. This meant that the user could never generate a "maximum" file of 65,534 records, each with 1024 bytes for a total of over 67MB of data. Version 2.20 removes this limitation. The limiting factor for the file size now is the disk storage capacity of the user's system.

Network Support

In an article on CONDOR 3 in the October 19th, 1987 issue of InfoWorld, Networking support was mentioned as a new feature. Before anyone gets too excited about Networking, I need to explain what this really means to CONDOR 3 users.

First of all, very little information is provided with the release materials. It is not mentioned at all in the CONDOR 3 manual as a new feature. It is, however, mentioned and discussed briefly in a small booklet that did come with the release called "Getting Started with CONDOR 3."

What Networking Support really means here is that the CONDOR 3 System files can be shared simultaneously among many users, provided that each user is licensed to use CONDOR 3. Thus, using network terminology, the CONDOR 3 System files are **Shareable/Read-Only** files. These files would reside on the Network File Server device.

However, the CONDOR 3 data files (datasets) that the users create, update, etc., must be dedicated to a single user at a time. In fact, Condor Computer Corporation recommends that users maintain their datasets/bases in their home directories (i.e. on their own microcomputer systems). If necessary, these data files could also reside on the Network File Server device, but accessed by only one user at a time. Again using network terminology, these data files are **Non-Shareable/Read-Write** files.

CONDOR 3 is installed on a network in single user mode. This means that CONDOR 3 does not provide file and record locking. Thus, only one user at a time should access CONDOR 3 data files. If simultaneous access is attempted, the integrity of the data is in jeopardy. As stated in the "Getting Started" booklet, CONDOR 3 can be installed on the following: PCNET, NOVELL, and 3COM3+ Networks.

As I stated earlier, Condor Computer recommends that the users maintain their data on their own systems and only share the CONDOR 3 System files residing on the Network File Server device (which is a different disk from the one used on their systems). Using this setup then is similar to running CONDOR 3 in a dual floppy environment. The **SET MASTER** command would not have to be used in this case. This command was discussed earlier.

Speed

Although this is an important feature of this version of CONDOR 3, I didn't have the type of databases and the hardware to really see any difference in execution speed. First of all, I used a Zenith Z160 with dual floppy drives (no hard disk). Secondly, my databases are not very large (just a few hundred records) and I don't use many of the relational type commands.

The InfoWorld article that I referred to above, stated that this version of CONDOR was 4 to 15 times faster than the last version. The release materials contained basically the same statement. Although I wasn't able to really see any increase in speed in my use of CONDOR, knowing that work was done in speeding up CONDOR is gratifying.

Documentation

Now let's shift gears and talk about documentation. The documentation that came with the release of V2.20 is in itself a significant enhancement. A product can be the best in its category but unless there is documentation to help users get started and provide an excellent reference source, the product is not worth much in my opinion.

Condor Computer Corporation completely rewrote the Manual for the V2.20 release. The Manual contains much more information and comes in a hard-covered 3-ring binder. The 3-ring binder as most of you know by now, is rapidly disappearing and being replaced with soft-covered textbook-like manuals or spiral-like bound manuals.

The Manual consists of three main sections. The first is the Tutorial section and the largest section. It covers about 80% or so of the CONDOR 3 commands. This section is designed to teach a person most of the CONDOR 3 Relational Database Management System in a step by step process with lots of examples and exercises. The Tutorial also provides the experienced user an excellent reference of examples and techniques.

The second section is the Reference Guide. This section covers each of the CONDOR 3 commands except for the Report Writer. The explanations are quite good and there are ample examples to help the user understand the material.

The third section covers the Report Writer in detail. The section actually consists of two parts. The first part is a Tutorial and the second part is the Report Writer Reference Guide. The entire section provides the user with a very good explanation and description of the Report Writer.

Following the three main sections is a small Error Messages section. This section covers error messages in general then command specific.

Although not part of the CONDOR V2.20 release materials, available to the users at an additional cost is the "Hands on CONDOR 3" book, third edition. This is a 'Self-Instruction Guide.' For new users and occasional users, this is an excellent guide to most of the commands of CONDOR 3.

In some ways, this Self-Instruction Guide is a duplication of what is covered in the Tutorial section of the CONDOR 3 manual. However, in other ways it is different and actually supplements the Tutorial. In both the Guide and the Tutorial, examples are given to set up datasets, add data, change data, join datasets, etc. Each uses different dataset examples.

Since "Hands on CONDOR 3" is a self-instruction book, it does contain review sections after each chapter and after a group of chapters. Each review is a set of questions to answer directly or perform exercises first, then answer the questions. These reviews are designed to see what the user has learned in the current chapter or group of chapters, and to help the user with additional small exercises. Answers to the questions are provided at the end of the book.

Just as with the Tutorial, many illustrations are provided in the Self-Instruction Guide to help the user visually see what is happening. In the Reference Guide section of the CONDOR 3 manual, I counted 51 commands. Around 35 or so are covered in the Self-Instruction Guide, and a little over 40 are covered in the Tutorial.

For new users, I would strongly recommend the "Hands on CONDOR 3" book to supplement the Tutorial. The new user should go through all the chapters, exercises, and reviews of this book, and then go through the Tutorial. I realize, as I said before, that there will be some duplication, but it's worth it. For experienced users, the Tutorial should be used as a reference guide for examples and techniques for all the commands covered especially for those the user is not familiar with.

Closing Comments

This concludes my comments and discussion of the enhancements and new features that went into CONDOR 3 V2.20. In the same InfoWorld article that I mentioned before, the president of Condor Computer Corporation was quoted as saying: "CONDOR 3 support will continue indefinitely. This is definitely not the last update of CONDOR 3." As new updates are released, I will try to summarize and discuss the enhancements and new features.

In addition, reference was made back in an August 1987 issue of InfoWorld that Condor Computer Corporation announced CONDOR 4 which will be a major new database package. At the time of this writing, CONDOR 4 was not available. More on that when it does become available.

Continue to use and enjoy CONDOR 3 as I am doing.

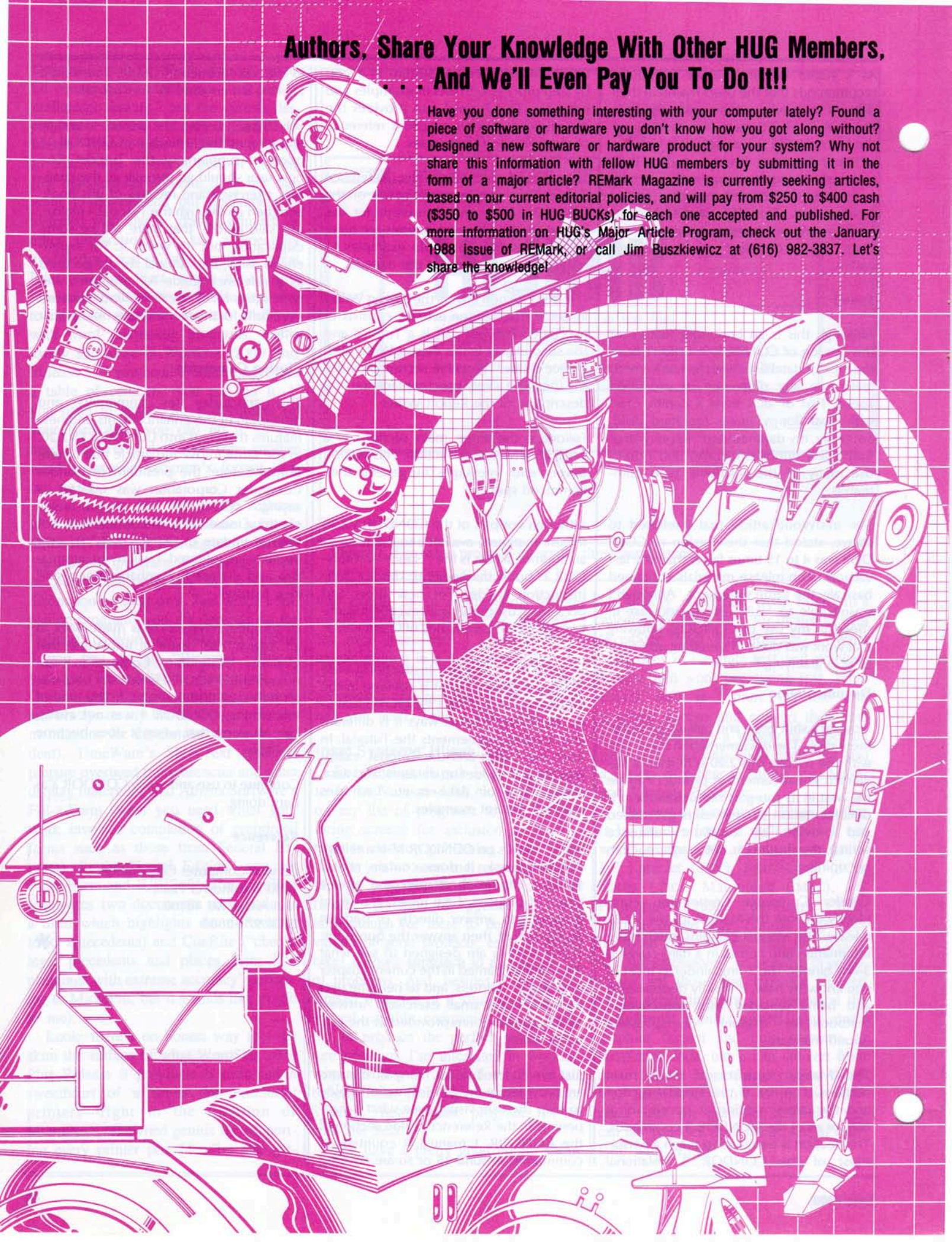
Reference

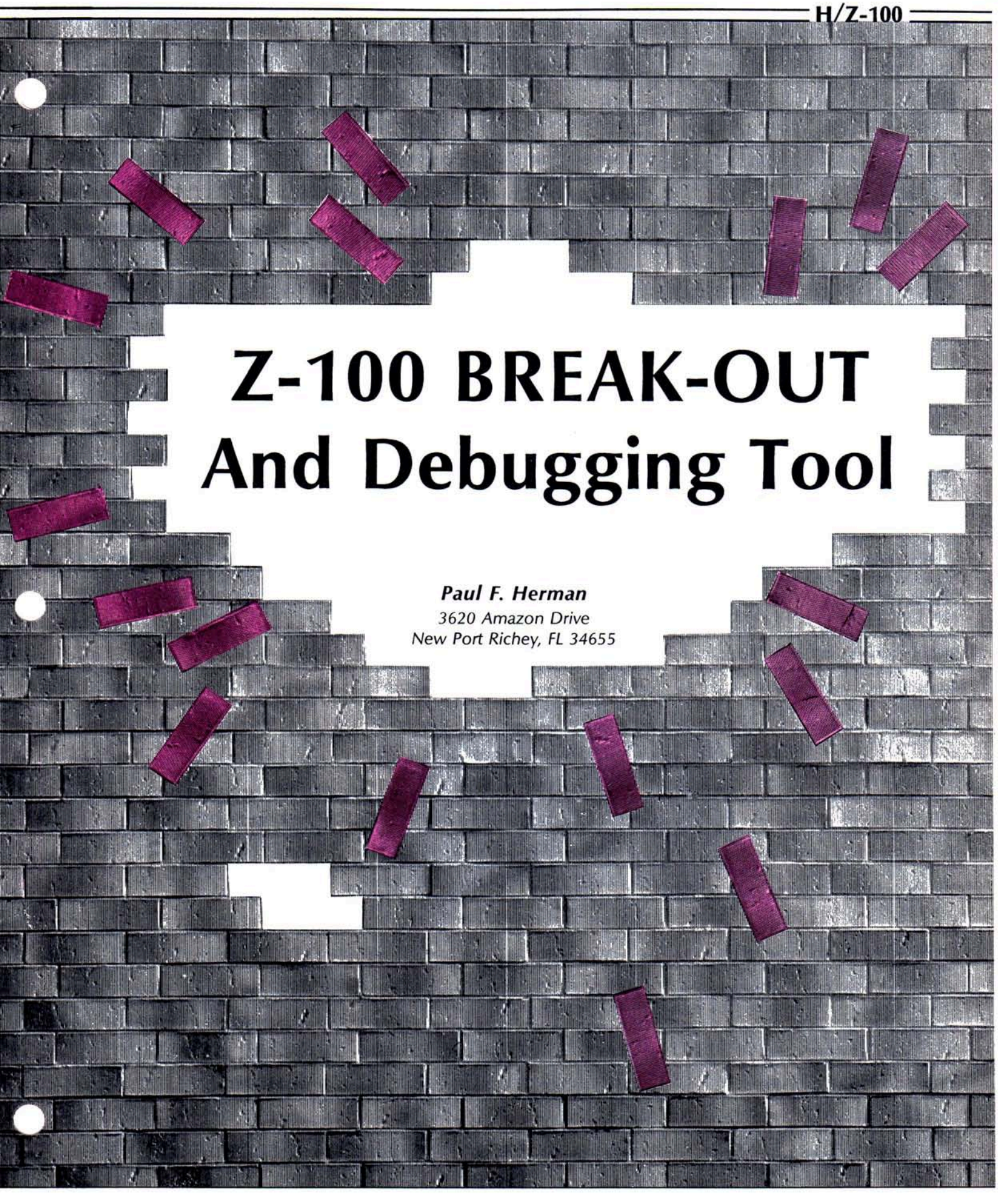
Condor Computer Corporation
1490 Eisenhower Place
Ann Arbor, MI 48108
(313) 971-8880

*

Authors, Share Your Knowledge With Other HUG Members, . . . And We'll Even Pay You To Do It!!

Have you done something interesting with your computer lately? Found a piece of software or hardware you don't know how you got along without? Designed a new software or hardware product for your system? Why not share this information with fellow HUG members by submitting it in the form of a major article? REMark Magazine is currently seeking articles, based on our current editorial policies, and will pay from $250 to $400 cash ($350 to $500 in HUG BUCKs) for each one accepted and published. For more information on HUG's Major Article Program, check out the January 1988 issue of REMark, or call Jim Buszkiewicz at (616) 982-3837. Let's share the knowledge!





Z-100 BREAK-OUT And Debugging Tool

Paul F. Herman
3620 Amazon Drive
New Port Richey, FL 34655

How many times have you tried to use a new program and the computer just locks up, forcing you to re-boot? Or perhaps you are writing your own program, and during the debugging process you have to re-boot frequently. Or you forget to switch to ZPC's PC mode before running a PC program and get to watch endless 'Wild Interrupt' messages dance across the screen.

The program presented in this article will allow you to regain control and return to the DOS prompt in many of these instances. It can also be used as a debugging tool to help you determine why the program 'hung' or caused a wild interrupt in the first place.

The Need For A Break-Out Utility

There are many times when a program will cause the computer to lock up so that you must re-boot to continue. Some software bugs may cause such a condition, and the problem is particularly noticeable to programmers during program development. One thing that is important to understand is that as long as no hardware problem exists, the computer always continues to process instructions. Most commonly, the program has entered an 'endless loop' from which it cannot escape. From your view point, the computer appears locked up, but the CPU just thinks this is part of the program. And an endless loop doesn't necessarily have to be caused by a program bug. For instance, when a program needs to output to the printer it uses a polling loop to determine when the printer is ready for a character. If you have the printer off-line, most programs will continue to poll the printer status forever. Of course, this problem should be corrected by putting the printer on-line, but most computer 'hangs' are not as easily resolved. As a matter of fact, the only solution many times is to re-boot the machine.

The people who write operating systems recognize this problem, and have provided us with a solution in many instances. They provided a break-out routine with which we are all familiar — it is activated by hitting Control-C. The only problem with Control-C is that it only works under certain conditions. Normally, MS-DOS checks for Control-C during console or I/O activity. In version 2 and above, you may tell MS-DOS to check for Control-C every time a program calls a DOS function. This is done by typing BREAK ON at the DOS prompt. For programmers (and others) this added BREAK ON feature was a real help. Control-C is now an excellent way to abort a normally running program. The only problem left is that most truly 'hung' programs aren't actively making DOS function calls. Therefore, MS-DOS can't recognize the Control-C's you are madly typing at the console. Here is where our BREAKER utility comes in handy. BREAKER will recognize its 'hot key' at just about any time, and give you back control of the computer.

What The BREAKER Utility Does

When BREAKER is activated, it displays its sign-on message and tells you the memory location (code segment and instruction pointer) where the program was interrupted. BREAKER also displays the contents of the CPU registers at the time of the interrupt. At this point, you are allowed several options. You may return immediately to the DOS prompt. Or you may continue with the original program from the point where it was interrupted. While BREAKER has control, you may also view the contents of memory. This may help you to discover where a problem has occurred or what caused the error condition.

Use of BREAKER is not limited to emergency conditions. You can also use the program's break-out feature during the normal course of a program. We'll describe some of these 'normal' uses of BREAKER later in the article. And we'll also describe some of the conditions which may interfere with BREAKER's ability to break-out of a program cleanly.

Overview Of How The Program Works

The BREAKER program is a memory-resident utility. That is to say, it loads itself into memory when invoked, and then stays resident in memory until you re-boot the computer. Once loaded, the program is activated by hitting the 'hot key'. Our program listing uses SHIFT HELP as the hot key.

When BREAKER is loaded, it does more than just terminate and stay resident. It 'hooks' into two interrupt vectors. The first of these is interrupt 46H, which is the interrupt generated by the 8259 interrupt controller whenever a key is struck. The interrupt routine normally addressed by this vector is in the Z-100 BIOS, and is used to process keyboard input. Every time a key is struck, BREAKER receives control just long enough to determine and save the user program's location and register contents. Then control is passed on to the normal BIOS interrupt routine. Processing of this interrupt to save register contents is done by BREAKER at all times — even when it is not activated. It may be interesting to note that interrupt 46H is also used for light pen and vertical sync interrupts.

The other interrupt used by BREAKER is interrupt 50H. This is a software interrupt generated by the BIOS, which is intended to give user programs a chance to process a keyboard character. BREAKER checks each keyboard character to see if it is the 'hot key'.

When the 'hot key' is detected, BREAKER takes control. The sequence of events leading up to activation goes something like this . . .

1. Key struck.
2. Keyboard generates hardware interrupt to 8259 interrupt controller.
3. 8259 interrupt controller generates hardware interrupt 46H.
4. BREAKER intercepts INT 46H and saves all CPU registers.
5. BREAKER then passes control to the BIOS for processing.
6. The BIOS saves most of the registers on its stack.
7. The BIOS then checks to see if the interrupt was caused by the light pen or a vertical sync pulse.
8. If not, then the BIOS checks for keyboard input and gets the character.
9. The BIOS issues software interrupt 50H.
10. BREAKER intercepts INT 50H and checks for its 'hot key'.
11. If this is not the 'hot key', BREAKER returns control to the BIOS.
12. Otherwise, BREAKER takes command.

The BREAKER program sets up its own stack and then saves any register it intends to use. A command is issued to the 8259 interrupt controller telling it the interrupt has been serviced. This is necessary so that BREAKER can receive its own keyboard input. The sign-on message is displayed. Then, the values of the CPU registers previously saved by the INT 46H routine are displayed. The program will now allow you to view contents of any address in memory, if you like. See the section 'How To Use BREAKER' later in this article. After you are done looking,

you have two options for exiting the BREAKER program . . .

CONTINUE — The program will restore all registers and return to the BIOS just as if nothing happened. The interrupted program will continue just where it left off. If BREAKER was activated during a 'hung' condition, the computer will still be hung when you continue.

ABORT — The program simply issues an MS-DOS function call 4CH (Terminate Process Function). This will cause any opened files to be closed, and then control is returned to the user program's 'parent' program. The parent program will usually (but not necessarily) be DOS. As a clarification, if you invoked the currently running program from the DOS prompt, and that program was aborted with BREAKER, you will return to the DOS prompt. However, if the program you invoked in turn called another program during which BREAKER was aborted, you will return to the originally invoked program. In this case, you can then quit the original program in a normal fashion, or you could activate BREAKER again and abort to the DOS prompt. Since DOS function call 4CH is only available in MS-DOS 2 and above, BREAKER cannot be used with Z-DOS.

One question may be puzzling to the programmers out there who are following this discussion. If the BIOS saves all the registers before generating interrupt 50H, why don't we just pull the CPU registers off the BIOS stack instead of hooking into interrupt 46H? Two reasons . . . First, the BIOS doesn't push ALL the registers. The user program's stack location and code location are not available on the BIOS stack at the time of the interrupt. The BIOS does save the user's stack, but the location of this information may be changed from one version of DOS to the next. Secondly, and most troublesome; one of the popular desktop utility programs hooks into the BIOS routine by changing the BIOS interrupt call. From a programmer's viewpoint, rewriting the BIOS is considered to be dirty pool. The result is that the offset to the user registers on the BIOS stack changes, depending on whether this desktop utility is loaded.

Notes About The Program Listing

The header to the program listing tells how to assemble the source listing into an executeable COM program. In order to do this you will need a text editor, the Microsoft assembler and linker, and the Microsoft EXE2BIN utility.

HOTKEY — You may select any key code you like for the 'hot key' (sometimes referred to as the activation key). We have chosen SHIFT HELP, which is key scan code 0D5H. See your Z-100 User's Manual for a list of the key scan codes for each key combination.

EXTERNAL ADDRESS DEFINITIONS — This portion of the listing is simply defining entry points used by BREAKER which already exist in the BIOS and monitor ROM. The assembler uses this information for addressing purposes. No code is generated for this section.

START OF CODE SEGMENT — The first executeable instruction is a jump to the installation routine, which is at the end of the program. This technique is common practice in memory-resident programs in order to make the resident portion of the program as small as possible.

DATA AREA — All of the data space and variables required by the resident portion of the program are in this area. Data needed only by the installation routine is put at the end of the program, so it will not take up space in the resident portion.

The first two variables in the data area deserve special mention, since they serve a rather general purpose which should be taken into account for any memory-resident program. The first of these is the program ID string. The program uses this ID to determine if it is already installed when invoked. If so, the program will exit with the message 'BREAKER is already installed'. This prevents the user from loading several copies of BREAKER into memory. Not that it would cause any harm — it just wastes memory. The second variable we want to mention is the FLAG byte. Whenever BREAKER takes control, it sets the FLAG to a one to indicate a busy state. This prevents a re-entrant condition in the case where the user hits the 'hot key' while BREAKER is still active. In addition to preventing the re-entrant problem, we use the FLAG byte for another purpose. The interrupt 46H routine will not save the values of the CPU registers when the FLAG shows BREAKER is active. This is necessary because the BREAKER program requires keyboard input itself. If the interrupt 46H routine saved the registers every time, the saved user program registers would be overwritten by BREAKER's register values.

8259 KEYBOARD INTERRUPT HANDLER — This is our interrupt 46H routine. The contents of the user CPU registers are saved every time this interrupt is issued, unless BREAKER is active. Notice that the user's code segment, instruction pointer, and flags are popped from the stack, and then the stack pointer is adjusted back to its original position.

START OF BREAKER INTERRUPT ROUTINE — This is the entry point for our INT 50H routine, which is actually the main BREAKER program. Notice that before taking control, the program checks to see if the key struck was the 'hot key', and then checks the FLAG byte to make sure it is not already active. The BIOS stack is saved and BREAKER sets up its own stack. Then, all registers used by BREAKER are saved on its stack, and the data segment is set to be the same as the code segment so variables can be referenced without a segment override. After all this is done, interrupts are allowed, and the 8259 interrupt controller is allowed to process interrupts again.

BREAK2 — The program displays its sign-on message and the values of the CPU registers previously saved by the INT 46H routine. The user is prompted for what action he would like to take — Continue, Abort, or View Memory.

BREAK3 — This code acts as a command dispatcher and routes control to the appropriate routine. If the user types something other than a C, A, or V, then a beep is sounded.

CONT — This is the exit code for the Continue option. All registers used by BREAKER are set to original values, the BIOS stack is restored, and control is passed back to the BIOS keyboard input routine. Notice that we changed the FLAG byte to show that BREAKER is no longer active.

ABORT — This is the exit code for the Abort option, which is really quite simple. The FLAG byte is cleared, and then a DOS function 4CH is called. This function will close any open files, and return control back to the program calling the interrupted program. (The current program's parent). Usually, this will cause an exit to the DOS prompt (See discussion earlier in the article). DOS function 4CH gives us the option of setting a return code when terminating a process. We have elected to use a return code of 1, which will signify an error condition to many programs. This

was done because the interrupted program will be more likely to exit back to DOS if it senses an error condition.

REGS — This subroutine displays the user program's CPU register contents on the screen. All register values were previously saved in variables, so all the routine has to do is get the values and display them.

VIEW — The subroutine which allows the user to view the contents of memory. Each line on the screen contains the hex bytes and ASCII characters of 16 bytes of memory. See the section 'How To Use BREAKER' later in the article for a better description of what this routine does.

GETADD — This subroutine is called by VIEW to get a memory address for viewing. The address is entered in segment: offset form.

GETW — This routine searches for a numeric value in the string pointed to by register SI, and then returns the value in register BX.

OUTCH — The MTR_SCRT smart terminal routine is used for outputting characters to the console. All registers in use are saved before calling the MTR-100 monitor routine. Actually, the MTR_SCRT routine does not effect all of these registers, but it is good programming practice to save anything that matters whenever calling an external routine which may be changed.

NOGOOD, SPACE, CRLF — All of these routines are used for console output. Their use should be self-explanatory.

OUTS — This subroutine displays a zero terminated ASCII string on the console.

OUTW, OUTB — These routines take a word or a byte and display it on the console as a 2 or 4 digit hexadecimal number.

INKEY — This routine uses the BIOS_CONFUNC function to input a character from the keyboard. All alphabetic characters are converted to upper case. Notice that we are only recognizing ASCII key scan codes for input. This means that you may not be able to use the keypad numbers for input in all programs. This was done to keep the program as simple as possible. Support for keypad entry of numbers and hyphen would be easy to add. But if you do, remember that the keypad can be expanded, in alternate mode, or both.

INS — This routine inputs a line-edited string from the keyboard. Backspacing is supported. Notice that only numbers, letters, and punctuation expected by BREAKER are permitted. The input string is right filled with spaces unless input is aborted at the first character, which causes the original buffer contents to be left unchanged.

RESEND — This label indicates the end of the resident part of the program. Anything after this label will not remain resident in memory after installation.

SETUP — This code is executed only once when BREAKER is first invoked. The first thing SETUP does is check the current interrupt 50H vector to see if the program is already resident in memory. This is determined by looking for the I.D. string at the start of the program data area. If BREAKER is already loaded into memory, the routine simply exits with an appropriate message. If the program is not currently installed the program disables interrupts and proceeds to patch into the interrupt 46H and 50H vectors. A message is displayed telling you that BREAKER is installed, and finally, the program uses DOS interrupt 27H to terminate and stay resident.

How To Use BREAKER

To load BREAKER into memory, just run the program by typing BREAKER at the DOS prompt. After loading, BREAKER is ready to be activated at any time by hitting the 'hot key'. If you find this program useful, as we do, you may want to load it during your AUTOEXEC sequence.

To activate BREAKER, simply hit the 'hot key' at any time. With very few exceptions, BREAKER will always respond with its sign-on message. The break location and registers of the program you interrupted will be displayed on the screen.

You now have a choice of whether you would like to (C)ontinue, (A)bort, or (V)iew memory. If you select 'C' to continue, the interrupted program will be allowed to resume where it left off. If you hit 'A' to abort, the interrupted program will be aborted, all open files will be closed, and you will (in most instances) be returned to the DOS prompt. In some cases, you may be returned to a previous section of the interrupted program. See the discussion under 'How The Program Works' for a more detailed discussion about which program receives control after an abort.

While BREAKER is in control, you also have the option of viewing the contents of memory by hitting 'V' at the prompt. The program will request a memory address in segment:offset form. You may enter an address followed with a RETURN, or you may accept the default address by hitting RETURN.

BREAKER will display a full screen of memory data for your viewing pleasure. The data is printed in hex bytes and ASCII. You may then select to go to the previous page, the next page, or return to the original BREAK menu.

Some Useful Things You Can Do With BREAKER

The obvious use for BREAKER is the one it was designed to be used for — break out of a hung condition or a never-ending loop of Wild Interrupt messages. If you are trying to figure out why the program crashed, try the following techniques:

1. If you broke out of a 'hung' condition, note the code segment and instruction pointer displayed by BREAKER just to the right of the sign-on message. This is the address of the instruction immediately after the one where you broke out of the program. You may want to use DEBUG or a similar utility to find out what instructions were executing at the time.
2. To determine if the program is in a loop, hit 'C' to allow the program to continue. Then activate BREAKER again. Do this several times. Does the instruction pointer indicate the same area of code that is being executed each time? Then, the 'hung' program is probably in a loop.
3. If you are debugging a program you wrote, the register contents may give you a clue as to the problem. Do they contain appropriate values for the indicated instruction counter location? You may also want to use BREAKER's view memory option to investigate variable values in your data area.
4. If you broke out of a Wild Interrupt loop, you would probably like to know which interrupt was called to generate this error. This is a little hard to determine, because when you break out with BREAKER, the CS:IP will always indicate somewhere in the BIOS Wild Interrupt routine. However, die-hard programmers should be able to find

the faulty interrupt instruction by searching the BIOS stack for the user CS:IP which was pushed by the interrupt instruction.

There are uses for BREAKER other than just trying to salvage an emergency situation. For instance . . .

1. Having BREAKER loaded gives you the ability to examine memory at any time, without having to run DEBUG or some other utility. A great help for debugging during Assembly Language programming. When you need to look at memory, simply activate BREAKER. When you are done browsing, tell BREAKER you would like to continue, and you will be returned to the program you were using.
2. You can use BREAKER to debug a program by 'breaking in' at any time while it is running. You can then examine the CPU contents or data area to see if the program is working correctly.
3. Use BREAKER to help find out how other software works. Quite often, I find myself wondering how some program does what it does. How does GW-BASIC set a pixel on the screen, for example. If I knew where the pixel setting routine was in memory, it would be fairly easy (though tedious) to determine how it works by disassembling the code. BREAKER will tell you where the code is. Simply write a program which does nothing, but set pixels. Then activate BREAKER while running the program.

Some Limitations And Cautions When Using BREAKER

There are some conditions under which BREAKER will be unavailable for activation. For instance, if a program changes the keyboard to UP/DOWN mode, BREAKER will not recognize its 'hot key'. Any program which patches into interrupt 46H or 50H, and doesn't pass control to other routines will also interfere with BREAKER activation. And any program which replaces or modifies the BIOS keyboard interrupt routine may cause problems.

If a program 'locks up' in the middle of a routine that has interrupts disabled, BREAKER will also be disabled. A good example of this is when a program 'hangs' during disk access.

There is one important caution about using BREAKER. Don't get in the habit of using BREAKER to abort a program as a normal way of exiting. Save this feature for emergency use. Even though BREAKER closes any open files, and uses the MS-DOS accepted way of terminating a program, you could run into problems with some programs. Why? Because the program may have made changes which are not properly restored by aborting with BREAKER. For example, a program may change the keyboard mode or modify the video control port. Normally, the program would take care of restoring any changes it made as a part of its clean-up when exiting. But aborting with BREAKER will not perform these housekeeping chores. I have found that GW-BASIC is one program which causes problems. After aborting GW-BASIC with BREAKER, running some programs will cause the computer to 'hang' during disk access — a problem that even BREAKER can't overcome. Another symptom of aborting from GW-BASIC is the message 'Can't Shell to BASIC' the next time you try to invoke GW-BASIC. It's perfectly safe to use BREAKER in GW-BASIC — just be sure to use the continue exit — then leave GW-BASIC using the normal SYSTEM command.

Wrapping It Up

I think you will find this BREAKER program to be a useful utility. This is especially true if you are a programmer, or one who likes to experiment with his Z-100. Many modifications and enhancements could be made to the program presented here. For example, including a mini-disassembler option would be a great idea. I welcome any comments you may have about the program which does nothing but set pixels. Then activate BREAKER while running the program.

HUG Software Engineer's Note: *HUG has a program similar to the one described in this article. The program is called BREAKOUT, and it is on HUG disk 885-3038-37, the HUG DEBUG Support Utilities. BREAKOUT is designed for use with DEBUG, and when its hot key sequence is pressed, DEBUG reacts as if a* break point had been hit. The DEBUG Support Utilities disk also contains Processor Window, a program that allows you to view registers and/or memory locations in real time while a program is running. These programs work on either Z-100 (not PC) computers or PC-compatible computers.

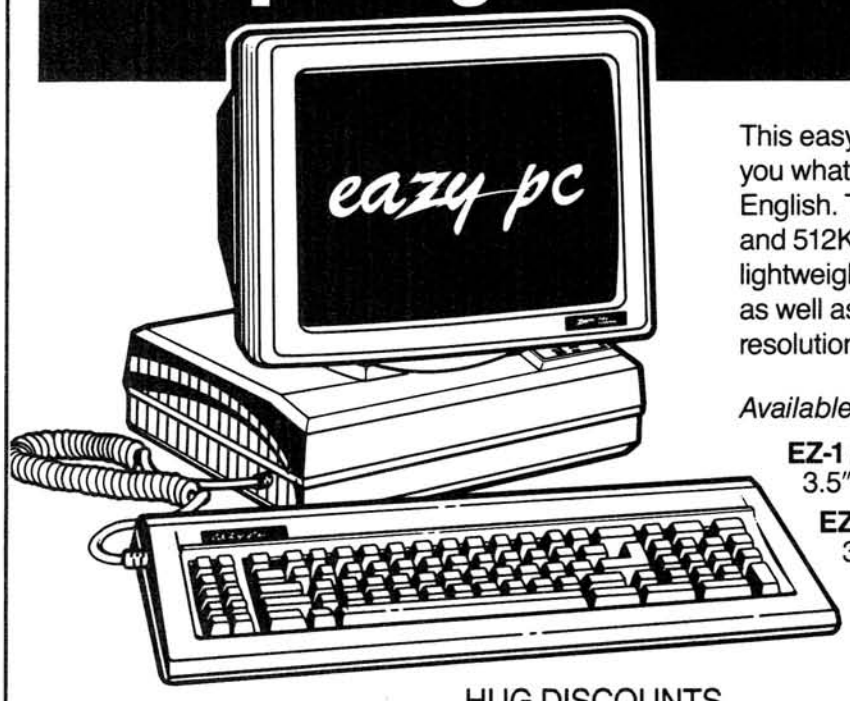
```
;
; BREAKER.ASM  A break routine and debugging tool
;
;       Paul F. Herman
;       3620 Amazon Drive
;       New Port Richey, FL 34655
;
;       to make BREAKER.COM
;               type in this text file (BREAKER.ASM)
;               MASM BREAKER;
;               LINK BREAKER;
;               EXE2BIN BREAKER BREAKER.COM
;               DEL BREAKER.EXE
;               DEL BREAKER.OBJ

HOTKEY          equ     0D5H            ; set this to any key code you like
                                        ; 0D5H is the code for SHIFT HELP

;*****************************************************************************
; EXTERNAL ADDRESS DEFINITIONS

INT_SEG segment at 0                    ; INTERRUPT PAGE
        org     4*46H                   ; Master 8259A keyboard interrupt
I8259K  equ     this word               ;
        org     4*50H                   ; BIOS generated keyboard interrupt
KBDI    equ     this word               ;
INT_SEG ends

BIOS    segment at 40H                  ; Z-100 BIOS SEGMENT
        org     51H                     ; BIOS console routine
BIOS_CONFUNC label far                  ;
BIOS    ends

MTR_SEG segment at 0FE01H               ; MTR-100 SEGMENT
        org     19H                     ; smart terminal handler
MTR_SCRT label far                      ;
MTR_SEG ends

;*****************************************************************************
; START OF CODE SEGMENT

CODE    segment public
        assume  cs:CODE, ds:CODE, es:nothing
        org     100H

START:  jmp     SETUP                   ; go to installation routine

;*****************************************************************************
; DATA AREA - All data is in the code segment

ID      db      'BREAKER'               ; I.D. to see if already installed
FLAG    db      0                       ; flag indicating BREAKER is running
STK     db      128 dup(?)              ; this is our program's stack area
STKTOP  dw      ?                       ;
SYSSP   dw      ?                       ; storage for system stack pointer
SYSSS   dw      ?                       ; and stack segment
LASTSEG dw      ?                       ; last segment
LASTOFF dw      ?                       ; and offset of memory displayed
HEXTAB  db      '0123456789ABCDEF'
CLS     db      01BH,'E',0
BKRMESS db      0DH,0AH,'Execution halted by BREAKER at ',0
REGMESS db      ' AX   BX   CX   DX   SI   DI   BP   SP   DS   ES   SS   FL'
        db      0DH,0AH,0
ACTION  db      '(C)ontinue, (A)bort, or (V)iew memory ? ',0
ACTION2 db      '(P)revious page, (N)ext page, or hit RETURN to go back to BREAK
                menu...',0
ADDMESS db      0DH,0AH,'Enter start memory address to view <'
BUFF    db      '0000:0000> ? ',0
USERAX  dw      ?                       ;
USERBX  dw      ?                       ;
USERCX  dw      ?                       ;
USERDX  dw      ?                       ;
USERSI  dw      ?                       ;
USERDI  dw      ?                       ;
USERBP  dw      ?                       ; storage for user's register values
USERSP  dw      ?                       ; at time of interrupt
USERDS  dw      ?                       ;
USERES  dw      ?                       ;
USERSS  dw      ?                       ;
USERFLG dw      ?                       ;
USERCS  dw      ?                       ;
USERIP  dw      ?                       ;
RETURN: db      0EAH                    ; far jump instruction
RETOFF  dw      ?                       ; return address for INT 50H routine
RETSEG  dw      ?                       ;
RETURN1:db      0EAH                    ; far jump instruction
RETOFF1 dw      ?                       ; return address for INT 46H routine
RETSEG1 dw      ?                       ;

;*****************************************************************************
; 8259 KEYBOARD INTERRUPT HANDLER

SAVEREG:cmp     cs:FLAG, 0              ; is BREAKER active? yes, don't change
        jne     RETURN1                 ; user register values, just return
        mov     cs:USERAX, ax           ; else, save user's AX,
        mov     cs:USERBX, bx           ; BX,
        mov     cs:USERCX, cx           ; CX,
        mov     cs:USERDX, dx           ; DX, ¦*****************************¦
        mov     cs:USERSI, si           ; SI, ¦ Notice that we don't allow  ¦
        mov     cs:USERDI, di           ; DI, ¦interrupts during any of this¦
        mov     cs:USERBP, bp           ; BP, ¦*****************************
        mov     cs:USERSP, sp           ; SP,
        mov     cs:USERDS, ds           ; DS,
        mov     cs:USERES, es           ; ES,
        mov     cs:USERSS, ss           ; SS,
        pop     cs:USERIP               ; get user's IP, CS, and flags from
        pop     cs:USERCS               ; stack and put them in storage too
        pop     cs:USERFLG              ;
        sub     sp, 6                   ; restore stack pointer
        jmp     RETURN1                 ; return so BIOS can process interrupt

;*****************************************************************************
; START OF BREAKER INTERRUPT ROUTINE

BREAK:  cmp     al, HOTKEY              ; is this the hot key?
        jne     RETURN                  ;   no, return before anybody notices
        cmp     cs:FLAG, 0              ;   yes, is BREAKER currently active?
        jne     RETURN                  ;     if it is, don't allow re-entry
```

```
        mov     cs:FLAG, 1              ; okay we're in ... show busy signal
        mov     cs:SYSSP, sp            ;
        mov     cs:SYSSS, ss            ; save BIOS stack
        mov     cs:STKTOP, cs           ;
        mov     ss, cs:STKTOP           ; set up our stack
        mov     sp, offset STKTOP       ;
        push    ax                      ; save everything in case user wants
        push    bx                      ; to continue his interrupted program
        push    cx                      ;
        push    dx                      ;
        push    si                      ;
        push    di                      ;
        push    bp                      ;
        push    ds                      ;
        push    es                      ;
        mov     ax, cs                  ;
        mov     ds, ax                  ; make DS same as our CS
        sti                             ; let interrupts thru now
        mov     al, 20H                 ; tell 8259 interrupt controller
        out     0F2H, al                ; that it can go back to work now
BREAK2: lea     si, BKRMESS             ; print sign-on message
        call    OUTS                    ;
        call    REGS                    ; print contents of CPU registers
        lea     si, ACTION              ; ask what user wants to do
        call    OUTS                    ;
BREAK3: call    INKEY                   ; get input from user
        cmp     al, 'A'                 ; should we abort to DOS?
        je      ABORT                   ;
        cmp     al, 'C'                 ; or go back to user's program?
        je      CONT                    ;
        cmp     al, 'V'                 ; or let user look at the memory?
        jne     BREAK4                  ;
        call    VIEW                    ;
        jmp     BREAK2                  ;
BREAK4: call    NOGOOD                  ; else, must not be a C, A, or V
        jmp     BREAK3                  ; beep, and try again
                                        ; EXIT BACK TO USER PROGRAM
CONT:   call    OUTCH                   ; echo 'C' to screen
        call    CRLF                    ; go to next line
        pop     es                      ; put everything back on the stack
        pop     ds                      ;
        pop     bp                      ;
        pop     di                      ;
        pop     si                      ;
        pop     dx                      ;
        pop     cx                      ;
        pop     bx                      ;
        pop     ax                      ;
        cli                             ; turn off interrupts for stack switch
        mov     sp, cs:SYSSP            ; restore stack
        mov     ss, cs:SYSSS            ;
        mov     cs:FLAG, 0              ; show we are done
        mov     ah, 1                   ; tell BIOS we used the character
        jmp     RETURN                  ; jump to original keyboard routine
                                        ; ABORT TO DOS
ABORT:  call    OUTCH                   ; echo 'A' to screen
        call    CRLF                    ; go to next line
        mov     FLAG, 0                 ; show we are done
        mov     al, 1                   ; return code of 1 to show error exit
        mov     ah, 04CH                ; try to check out of user's program
        int     21H                     ; using DOS 2 terminate process function
        jmp     BREAK2                  ; if we get to this point it means that
                                        ; DOS didn't understand the 04CH call
                                        ; This would happen if DOS is v1.x
                                        ; just jump back to BREAKER message

                                        ; DISPLAY CPU REGISTERS WHEN INTERRUPTED
REGS:   mov     ax, USERCS              ;
        call    OUTW                    ; print user's CS:IP
        mov     al, ':'                 ;
        call    OUTCH                   ;
        mov     ax, USERIP              ;
        call    OUTW                    ;
        call    CRLF                    ;
        mov     cx, 58                  ;
        call    SPACES                  ; print a blank line
        call    CRLF                    ;
        lea     si, REGMESS             ;
        call    OUTS                    ; print register heading
        lea     si, USERAX              ;
        mov     cx, 12                  ; will print contents of 12 registers
REGS1:  mov     ax, [si]                ;
        push    cx                      ;
        call    OUTW                    ; print user register contents
        call    SPACE                   ; followed by a space
        pop     cx                      ;
        add     si, 2                   ;
        loop    REGS1                   ; loop until done
        call    CRLF                    ;
        mov     cx, 58                  ;
        call    SPACES                  ; print blank line
        call    CRLF                    ;
        ret                             ;

                                        ; VIEW DATA IN MEMORY
VIEW:   lea     si, ADDMESS             ; prompt for start address
        call    OUTS                    ;
        call    GETADD                  ; and get segment:offset from user
VIEW1:  lea     si, CLS                 ; clear screen
        call    OUTS                    ;
        mov     cx, 23                  ; will display 23 lines of info
VIEW2:  push    cx                      ; save line count
        mov     es, LASTSEG             ; get segment to display
        mov     si, LASTOFF             ; get offset to display
        mov     ax, es                  ;
        call    OUTW                    ; display segment
        mov     al, ':'                 ;
        call    OUTCH                   ;
        mov     ax, si                  ;
        call    OUTW                    ; and offset
        mov     cx, 2                   ;
        call    SPACES                  ; two spaces
        mov     cx, 16                  ; display 16 hex bytes per line
VIEW3:  push    cx                      ; save byte count
        call    SPACE                   ;
        mov     bl, es:[si]             ; get a byte
        call    OUTB                    ; and display it
        inc     si                      ; next byte
```

```
        pop     cx                      ;
        loop    VIEW3                   ; loop until done with hex bytes
        mov     cx, 3                   ;
        call    SPACES                  ; three spaces
        mov     si, LASTOFF             ; get start of bytes again
        mov     cx, 16                  ; display 16 ascii bytes per line
VIEW4:  push    cx                      ; save byte count
        mov     al, es:[si]             ;
        cmp     al, ' '                 ; see if byte is printable character
        jl      VIEW5                   ;
        cmp     al, 7EH;
        jle     VIEW6                   ;
VIEW5:  mov     al, '.'                 ; if not, use a period
VIEW6:  call    OUTCH                   ; display it
        inc     si                      ; next byte
        pop     cx                      ;
        loop    VIEW4                   ; loop until done with ascii characters
        call    CRLF                    ; go to next line
        mov     LASTOFF, si             ; update our address register
        pop     cx                      ; get line count back again
        loop    VIEW2                   ; loop until done with lines
        lea     si, ACTION2             ;
        call    OUTS                    ; prompt for action to take
VIEW7:  call    INKEY                   ;
        cmp     al, 0DH                 ; RETURN key?
        je      VIEW11                  ;
        cmp     al, 'P'                 ; (P)revious memory page?
        jne     VIEW10                  ;
VIEW8:  sub     LASTOFF, 736            ; yes, decrease offset
VIEW9:  jmp     VIEW1                   ; and display another screen of data
VIEW10: cmp     al, 'N'                 ; Next memory page?
        je      VIEW9                   ; yes, display next screen
        call    NOGOOD                  ; none of the above, beep at user
        jmp     VIEW7                   ; give him another try
VIEW11: ret                             ;

                                        ; GET MEMORY ADDRESS FROM USER
GETADD: call    INS                     ; input string from keyboard
        lea     si, BUFF                ;
        call    GETW                    ; find the segment
        mov     LASTSEG, bx             ;
        inc     si                      ;
        call    GETW                    ; and offset
        mov     LASTOFF, bx             ;
        ret                             ;

                                        ; CONVERT ASCII DIGITS TO WORD
GETW:   mov     bx, 0                   ; zero result
GETW1:  mov     al, [si]                ; get character
        mov     di, offset HEXTAB       ;
        mov     cx, 16                  ;
GETW2:  cmp     al, [di]                ;
        je      GETW3                   ; check for legal hex digit
        inc     di                      ;
        loop    GETW2                   ;
        ret                             ; if not hex digit, we're done
GETW3:  sub     di, offset HEXTAB       ; find which digit
        mov     cl, 4                   ;
        shl     bx, cl                  ; shift result one nibble
        add     bx, di                  ; add new digit
        inc     si                      ;
        jmp     GETW1                   ; get next character

                                        ; OUTPUT A CHARACTER TO THE SCREEN
OUTCH:  push    bx                      ;
        push    cx                      ;
        push    si                      ; save registers
        push    bp                      ;
        push    ds                      ;
        push    es                      ;
        call    MTR_SCRT                ; print the character in AL
        pop     es                      ;
        pop     ds                      ;
        pop     bp                      ;
        pop     si                      ; restore registers
        pop     cx                      ;
        pop     bx                      ;
        ret                             ;

                                        ; MAKE A BEEP
NOGOOD: mov     al, 7                   ; load ascii bell character
        call    OUTCH                   ; output to console
        ret                             ;

                                        ; PRINT SPACES
SPACE:  mov     cx, 1                   ; entry for 1 space only
SPACES: mov     al, ' '                 ; entry for CX number of spaces
        call    OUTCH                   ;
        loop    SPACES                  ;
        ret                             ;

CRLF:   mov     al, 0DH                 ; PRINT A CRLF
        call    OUTCH                   ;
        mov     al, 0AH                 ;
        call    OUTCH                   ;
        ret                             ;

                                        ; PRINT AN ASCIIZ STRING ON THE SCREEN
OUTS:   lodsb                           ; get a character
        cmp     al, 0                   ; are we done yet?
        je      OUTS2                   ; yes, quit
        call    OUTCH                   ; else, print to screen
        jmp     OUTS                    ; loop until done
OUTS2:  ret                             ;

                                        ; PRINT A HEX 4-DIGIT WORD
OUTW:   mov     bl, ah                  ;
        push    ax                      ;
        call    OUTB                    ; print most significant nibble
        pop     bx                      ;
        call    OUTB                    ; print least significant nibble
        ret                             ;

                                        ; PRINT A HEX 2-DIGIT BYTE
OUTB:   mov     cx, 2                   ; 2 nibbles in a byte
OUTB1:   rol     bl, 1                  ; rotate for next nibble
        rol     bl, 1                   ;
        rol     bl, 1                   ;
```

```
        rol     bl, 1                   ;
        push    bx                      ;
        and     bx, 0FH                 ; isolate nibble
        mov     al, byte ptr HEXTAB[bx] ; convert to hex
        push    cx                      ;
        call    OUTCH                   ; output the character
        pop     cx                      ;
        pop     bx                      ;
        loop    OUTB1                   ; repeat until done
        ret                             ;

INKEY:  mov     ah, 1                   ; GET A CHARACTER FROM KEYBOARD
        call    BIOS_CONFUNC            ; get character
        jc      INKEY                   ; try again if none there
        cmp     al, 'a'                 ; got one
        jl      INKEY1                  ; if its a letter, make it upper case
        cmp     al, 'z'                 ;
        jg      INKEY1                  ;
        sub     al, 20H                 ;
INKEY1: ret                             ;

                                        ; GET STRING INPUT FROM KEYBOARD
INS:    lea     si, BUFF                ; get start of buffer area
INS1:   push    si                      ;
        call    INKEY                   ; get a character
        pop     si                      ;
        cmp     al, 0DH                 ; RETURN key?
        jne     INS3                    ;
INS2:   cmp     si, offset BUFF         ; yes, still at start of buffer?
        je      INS10                   ; yes, use default string
        jmp     INS9                    ; no, right fill with spaces
INS3:   cmp     al, 8DH                 ; ENTER key?
        je      INS2                    ;
        cmp     al, ':'                 ; colon?
        je      INS8                    ;
        cmp     al, '-'                 ; hyphen?
        je      INS8                    ;
        cmp     al, 8                   ; back space?
        jne     INS7                    ;
        cmp     si, offset BUFF         ; yes, are we at start of buffer
        jne     INS5                    ;
INS4:   call    NOGOOD                  ; yes, can't do it
        jmp     INS1                    ;
INS5:   mov     al, 8                   ; blank previous screen character
        call    OUTCH                   ;
        mov     al, ' '                 ;
        dec     si                      ; blank previous buffer character
        mov     [si], al                ;
        call    OUTCH                   ;
INS6:   mov     al, 8                   ; move cursor to proper position
        call    OUTCH                   ;
        jmp     INS1                    ;
INS7:   cmp     al, '0'                 ; otherwise must be a number
        jl      INS4                    ; or A thru F
        cmp     al, '9'                 ;
        jle     INS8                    ;
        cmp     al, 'A'                 ;
        jl      INS4                    ;
        cmp     al, 'F'                 ;
        jg      INS4                    ;
INS8:   cmp     si, offset BUFF+9       ; at end of buffer?
        je      INS4                    ; yes, beep
        mov     [si], al                ; otherwise, put in buffer
        call    OUTCH                   ; echo to screen
        inc     si                      ; bump pointer
        jmp     INS1                    ; and get another character
INS9:   cmp     si, offset BUFF+9       ; done, now right justify with spaces
        je      INS10                   ;
        mov     byte ptr[si], ' '       ;
        inc     si                      ;
        jmp     INS9                    ;
INS10:  ret                             ;

RESEND:                                 ; end of resident code

;*****************************************************************************
; INSTALLATION CODE (non-resident)

SETUP:  xor     ax, ax                  ; get interrupt segment
        mov     es, ax                  ;
        mov     es, es:KBDI+2           ; get segment of keyboard int routine
        lea     si, ID                  ; get offset to ID string
        mov     di, si                  ;
        mov     cx, 7                   ; ID string is 7 bytes long
        rep     cmpsb                   ; does int routine match our ID?
        jne     SETUP1                  ;
        lea     si, IMHERE              ; yes, BREAKER is already installed
        call    OUTS                    ; print message,
        int     20H                     ; and return to DOS
SETUP1: mov     es, ax                  ; get interrupt segment again
        mov     ax, es:KBDI             ; save BIOS generated keyboard interrupt
        mov     RETOFF, ax              ; vector as our return address
        mov     ax, es:KBDI+2           ;
        mov     RETSEG, ax              ;
        mov     ax, es:I8259K           ; save 8259 keyboard interrupt
        mov     RETOFF1, ax             ; vector as our other return address
        mov     ax, es:I8259K+2         ;
        mov     RETSEG1, ax             ;
        cli                             ; douse interrupts
        mov     ax, offset BREAK        ; patch into BIOS keyboard interrupt
        mov     es:KBDI, ax             ;
        mov     ax, offset SAVEREG      ; patch into 8259 keyboard interrupt
        mov     es:I8259K, ax           ;
        mov     ax, cs                  ;
        mov     es:KBDI+2, ax           ;
        mov     es:I8259K+2, ax         ;
        sti                             ; allow interrupts
        lea     si, HELLO               ; announce BREAKER is installed
        call    OUTS                    ;
        mov     dx, offset RESEND       ; get address of end of resident code
        int     27H                     ; terminate but stay resident

HELLO   db      'BREAKER v1.0 installed...  Hit Shift HELP',0DH,0AH,0DH,0AH,0
IMHERE  db      'BREAKER is already installed!',0DH,0AH,0

CODE    ends                                                          *
        end     START
```

Zenith Data Systems eaZy pc
Computing has never been easier



This easy-to-use IBM PC compatible tells you what to do and when – on-screen in plain English. The EaZy PC comes with MS-DOS and 512K RAM expandable to 640K. Plus it's lightweight and has a professional keyboard, as well as a 14" monitor that provides high resolution text and graphics.

Available in three models:

EZ-1 with one 720K 3.5" drive ..... ONLY **$799**

EZ-2 with dual 720K 3.5" drives ..... ONLY **$959**

EZ-3 with 20 MB Winchester ..... ONLY **$1359**

Also comes with our FREE Partnership Pack support package worth $500 in savings!

HUG DISCOUNTS DO NOT APPLY.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Jacksonville, 32211
9426 Arlington Expressway
904-725-4554

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

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

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

HAWAII – Honolulu (Pearl City), 96782
98-1254 Kaahumanu St.
808-487-0329

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

Skokie (Chicago), 60076
3606 W. Dempster
312-583-3920

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

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

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

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

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

Rockville, 20852
5542 Nicholson Lane
301-881-5420

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

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

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

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

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

MINNESOTA – Minneapolis (Hopkins), 55343
101 Shady Oak Rc.
612-938-8371

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Phone orders accepted.

Heath Zenith Computers & Electronics

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

HZC-402

Using DEBUG To Write Short Assembly Language Programs

Matt Elwood
1670 North Laddie Court
Beavercreek, OH 45432

You're getting sick of the key click. You want a box cursor. You want to change the colors. What do you do? One way is to make an Assembly Language program. But the assembling and the linking is not worth it for such a small program. There is another alternative: You can use DEBUG.

DEBUG?!?! You mean the cheap 13k program that comes with DOS? You probably think some program that comes with DOS is not worth much of anything and a commercial program is probably better. Well, DEBUG isn't like that. It has lots of features and is useful for editing disk sectors, modifying programs, and making small programs.

Okay, let's get started. You call up DEBUG with DEBUG. Simple enough. But if you want to load a file to modify or create, type the name after DEBUG like this:

```
A>DEBUG DEBUG.WPF
```

This would load "DEBUG.WPF" from the current drive. If the file doesn't exist, DEBUG prints: "File not Found", but still keeps the file name, if you want to save it later.

The first command we will learn is the "E" command. This is the "enter bytes" command. This is what you use to enter strings or single bytes. For example, "E105" would let you start entering data at memory location 0105 in the current empty segment selected by DEBUG. Then DEBUG displays this:

```
-E105
       02.
```

The "02" is the hexadecimal value now stored at 0105 and the dot is a prompt. You enter your new value (in hexadecimal) here. Then, to go on to the next byte, type space. Do not type return, for this brings you back to the prompt.

There are other ways that "E" can be used. You can put the values for the bytes on the command line like this:

```
-E107 03 17 47 18 96
```

This would enter the five bytes into the memory locations 0107-010B with each of the numbers. Another way is just to type in ASCII and enclose it with quotes:

```
-E14F "Please place program disk in
     drive A"
```

That would put the hex value for each of the characters in the string in memory starting at 014F. You can also mix the two:

```
-E157 "User Number : " 05 27 1B 96 F3 "."
```

The next command is the "W" command. This will save the amount of bytes specified in register CX and the number of complete segments specified in BX. You must be at the starting segment of the block you want saved. Write uses the file name specified on the command line or the file name changed by the "N" command.

The "N" command changes the current file name. You use N like this:

```
-NDEBUG2.WPF
```

If you want a listing of the registers, use the "R" command. No other arguments, just a plain old "R".

```
-R
AX=0000  BX=0000  CX=0000  DX=0000
  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0DF7  ES=0DF7  SS=0DF7  CS=0DF7
  IP=0100   NV UP EI PL NZ NA PO NC
0DF7:0100 65           DB      65
```

But if you want to change a register, use R followed by the register name.

```
-RCX
CX 0000
:4848
```

This would change CX to hex value 4848.

Another major command used is "D" for dump. This prints 80 bytes of memory starting at the address you specify after the "D".

Also, you use the "A" command to input Assembly Language instructions. We will be using this and the "U" (unassemble) commands later.

Now, let's get to the good part. Here is a step-by-step plan on how to create an Assembly Language program to shut off the key click on the Z-100.

About The Author

Matt Elwood is 13 years old and has been working on computers since he was 5 (the good ol' H-89). He's programmed in BASIC, Assembly Language, C, and Turbo Pascal. He's also in the advanced subjects in ENABLE and DOS.

```
-F 100 FFFF FF   <- Fills the segment with FFs. This gives us a
              <- clean slate to work in.

-A100          <- Starts assembly input at 0100

0E0B:0100 MOV DX,0109 <- Moves the address of the code into DX
0E0B:0103 MOV AH,09   <- Function Call 09 - Write String
0E0B:0105 INT 21      <- Interrupt 21 - Execute Function
0E0B:0107 INT 20      <- Interrupt 20 - Exit to DOS
0E0B:0109            <- Hit RETURN to stop input of assembly.

-E109 1B "x2$"       <- Escape code

-RCX               <- Make save area 9 bytes long
 CX 0000                CX = bytes to save
 :000D                   BX = 64k segments to save starting
                        at current segment.

-R                 <- Look at registers

AX=0000  BX=0000  CX=000D  DX=0000  SP=FFFE  BP=0000  SI=0000  DI=0000  DS=0E0B
ES=0E0B  SS=0E0B  CS=0E0B  IP=0100   NV UP EI PL NZ NA PO NC 0E0B:0100 BA0901

-D100              <- Dump out memory

0E0B:0100  BA 09 01 B4 09 CD 21 CD-20 1B 78 32 24 FF FF FF   :.4.M!M .x2$...
0E0B:0110  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF   ................
0E0B:0120  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF   ................
0E0B:0130  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF   ................
0E0B:0140  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF   ................
0E0B:0150  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF   ................
0E0B:0160  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF   ................
0E0B:0170  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF   ................

-NNOCLICK.COM                <- File to be saved under

-U100                      <- One last check of assembly code.
0E0B:0100 BA0901       MOV    DX,0109
0E0B:0103 B409         MOV    AH,09
0E0B:0105 CD21         INT    21
0E0B:0107 CD20         INT    20
0E0B:0109 1B7832       SBB    DI,[BX+SI+32]   <- Disassembler interprets our
0E0B:010C 24FF         AND    AL,FF           <- data as code.
            etc...

-W                         <- Save our file

-Q                         <- Quit
```

We'll go through it step-by-step. First, we used the "F" command to fill in the entire memory with FFs, therefore giving us a clean slate. Next, we entered "MOV DX,0109" to move into DX, a pointer that tells MS-DOS where the data is. Next, "MOV AH,09" tells what command will be executed. "INT 21" executes the command, and "INT 20" quits. Next, we used the "E" command to enter the data into memory. Then, we put the length of the file-to-be, D (13) bytes long. Last, we checked memory to make sure we did it right, specified the file name, and saved it.

Now we'll talk about the intermediate commands in DEBUG that you use in manipulating data or code. These are the "S"earch, "I"nput from Port, and "O"utput from port.

The "S" command will search the current segment or part of the current segment for the data you specify. For example, if you want to search for "74 FB" in the current segment, you would use:

```
-S 0 L FFFF 74 FB
```

Then, DEBUG would print the addresses of the occurrences of "74 FB". The 0 specifies the starting address and the FFFF specifies the ending address. The "L" really means nothing. So "S 0 L FFFF 74 FB" means the same as "S 0 FFFF 74 FB".

Another thing about the "S" command is that you can use "S:CS" or "S:DS" to search that segment for the data.

The "O" command is used to output a byte to an 8- or 16-bit port. You use the syntax "O value byte", where value is the hex value of the port address and byte is the byte to send.

The "I" command inputs a byte from the port. You just use "I value", where value is the hex value of the 8- or 16-bit port. DEBUG displays the value on the next line.

The last intermediate command is "L", for load file. You just use "L" for the load into the current segment at offset 100. You can also use "L address" to load the file into the address in the current segment. "L" uses the file name specified in "N".

Now we are ready to change the "NOCLICK.COM" to a new file, "BLOCKCUR.COM". We will load NOCLICK, then make an identical copy, except for changing one byte to BLOCKCUR.

```
-NNOCLICK.COM <-File name is NOCLICK.COM
-L          <- Load it
-NBLOCKCUR.COM <-New file name is
             BLOCKCUR.COM
-S 0 L FFFF "x2" <- Search for code to
                 turn off keyclick.
0E0C:010A <- DEBUG displays occurence.
-E10A "x4" <- Change that code
-W       <- Write changes to BLOCKCUR
Writing 000D bytes <- CX is retained
                   between files
-Q <- Quit
```

Here's the rundown step-by-step. First, we set the file name to NOCLICK and read it in. Then, we changed the name to BLOCKCUR. We then searched for the "x2" code for turning off the key click and changed it to "x4". Then, we saved it.

Now it's time for the advanced commands. We will cover the Compare, Go, Hex, Move, Trace Special, Trace, and advanced uses for Load and Write.

The first command is "Compare". This compares two blocks of memory. You use the syntax "C b1s b1e b2s". The "b1s" is the beginning address of block one. The "b1e" is the ending address of block one. The "b2s" is the beginning address of block two. The ending address of block two is automatically calculated. Therefore, to compare a block of memory from address 140 to 155 and 180 to 195, you would type "C 140 155 180".

The second command is "Go". This is one of the most dangerous commands in DEBUG, for it will hang up if it gets caught in a loop. The syntax for go is "G=addr1 addr2 addr3 ...". The first address must be preceded by an "=" so DEBUG can distinguish it from the rest. The first address is the starting address and the rest are the breakpoints. The breakpoints are addresses two through twelve. There can be ten breakpoints set. When DEBUG hits a breakpoint, it stops. Using the "G" command will start it again. The breakpoints just insert a CC (INT 3) command in memory, so they must be before an instruction.

The next command is hex arithmetic. The syntax for this is "H para1 para2". DEBUG adds para1 and para2 and subtracts para1 minus para2. The sum, then the difference is displayed on the next line.

The fourth command is "Move". This is used to move blocks of memory into another place. If the "from" address overlaps the "to" address, no data is lost, for DEBUG finds out which data might be overwritten and moves them first. The syntax is "M fromstart fromend tostart". Moves across segments are allowed.

The last two commands, trace and trace special are almost the same. We will talk about trace first.

Trace executes the code one instruction at a time and displays the registers, flags, and instruction. The syntax is "T=addr1 value". The "=addr1" must have the equal sign for the same reason as go. This specifies the starting place for trace. The value is the number of instructions to run. If you just type "T", DEBUG will return a display of all the registers, flags, and instructions at the current address.

Trace Special (using a P) is the same as trace, but if a CALL, INT, or LOOP is detected, DEBUG will execute all the instructions in the sequence rather than the first instruction in the routine.

The last advanced commands we will talk about are the ways to read/write sectors using L and W. Be careful, for you may mess up the directory or FAT on a disk.

The syntax you use is "L address drive startsector sectors" or "W address drive startsector sectors". Address is the starting address where the data will be put in. Drive is the number of the drive. For example A=0, B=1, C=2, D=3. Startsector is the starting sector on the disk to read or write. Sectors is the number of sectors to read in starting with startsector.

For example, "L 100 0 6 2" will read drive A:'s directory sectors into memory at the current segment at 100 hex. To display this, just type D100. You should see the file names every 20 hex. Deleted files are displayed normally, except the first byte is E5 hex. To undelete a file (be careful!) just change the E5 to a normal letter. If you have written to the disk since the delete, the FAT has already been written over and this will do no good.

Well, that's about it. Those are all the commands in DEBUG. (At least DEBUG for MS-DOS Version 2.) I covered just about everything, except for what is considered a string in DEBUG, and more unnecessary stuff.

For now, good-bye and happy DEBUGging! *

The Short, Unhappy Life Of H/Z-100 dBASE III And dBXL, An Alternative

Richard A. Tilden
10 Thurston Street
Somerville, MA 02145

This article offers some information for buyers and potential buyers (if such is possible) of Z-100 dBase III, and then describes an alternative I discovered, a generic MS-DOS clone of dBase called dBXL.

The honeymoon was very short for buyers of Heath/Zenith Z-100 Dual Processors. Although we had bought the most advanced and reliable computer on the market, within six months we were orphans -- Zenith had gone with the marketplace onto the bandwagon of total IBM compatibility, which meant that software for the Z-100 was slow in coming, if it came at all, and that most of the exciting stuff we read about in the magazines would never be available to us.

Especially frustrating since Z-100 versions of the programs would have been functionally superior to IBM PC versions, the Z-100 having, for example, superior (and simpler) graphics until very recently.

It was surprising, therefore, to find a new and exciting program available for the machine this winter when it was five years old and had already disappeared from the catalog: dBase III (AT-463-4, $695). (Naturally, the Z-100 was receiving dBase III at a time when the world had had dBase III Plus for some time, just as we received MS-DOS 2 at a time when the world had MS-DOS 3.) The product was available for a very limited period, apparently, as by the time the Fall catalog appeared, it was gone.

A Pleasant Surprise

The product we received is a strange one, as I discovered talking to the support people at Ashton-Tate. It is not only dBase III as it describes itself in the manual that comes with it but, as indeed its help files reflect, something between dBase III and dBase III Plus. It includes all of the functions which III Plus added, and several of the commands. Essentially it is dBase III Plus less the Advisor (if that's what it's called) and less commands dealing with new file types -- .vue .scr and .cat.

And here's the piece of information which I particularly wanted to share with HUG readers: if you call Ashton-Tate and act outraged at the idea of having a program whose documentation is out of step, you should be able to get, at no cost, a replacement manual! You want something they refer to as the 'Wang and Other Ports' document. It is a complete replacement for the one shipped with Zenith dBase III and, as far as I can see, accurately describes Z-100 dBase III.

In case some shipments included one manual and some the other, you can test which you have by whether it documents the History feature, a III Plus feature available in the Z-100 version.

The Bad News

Until this purchase, I bragged that at least I was above all this copy protection nonsense: I have most of the available software for the Z-100, and can freely back it up, move it around on my hard disk, etc. Somehow I felt more trusted than those unreliable IBM types, potential thieves all of them. I had, for instance, the world's only non-copy protected Lotus 1-2-3 (how Mitch must have regretted that).

Sure enough, Z-100 dBase III is copy protected (even though Ashton-Tate has since removed copy protection from its products). Worse than that, as I have discovered talking to several people at A-T, it is copy protected with 'Prolock', a facility A-T found troublesome and replaced with 'Superlock'. Worst of all, no one at A-T understands how Prolock works and the Prolock people won't tell them. To add insult to injury, I discovered that military and government Z-100 users can get a non-copy protected version not available to civilians. (I wonder what else they can get that I can't?)

In the problems I am about to describe, you should understand that my hard disk is not the standard internal one at E:, it is a non-bootable one using a special driver at

I:. The Ashton-Tate people suspect that some of my problems might not exist on the standard hard disk.

-- When dB is installed, the installation creates a hidden zero-length directory named 'VAULTP&D' in the root, and creates 24k of bad tracks on the hard disk. God knows what these are used for. The bad tracks are detected but not corrected by CHKDSK; they can be removed only by reformatting.

-- Despite this, when dB is activated after installation, it calls for its original (key) diskette in drive A! (Strangely, if the disk is not in drive A, it selects drive B and looks at it for a while before giving the message demanding the disk -- but if its desired disk is in B, it won't buy it.)

It is this second behavior that several A-T people suspect is incorrect: they believe that for a standard hard disk, dB is runnable without the key disk.

I should mention that the quality of support from A-T is cordial and well-intentioned, but ineffective. I spoke to several different people in Product Support and Customer Service: they wanted to help, they returned phone calls, they knew their product. They didn't know, and couldn't find out, about Prolock, something they had licensed from a second source. And their bottom line was they thought that their management and Zenith should agree to solve the problem by removing copy protection; they had proposed this to management and it was being studied. Unfortunately, the removal of Z-100 dBase III from the Heath catalog clearly implies what alternative decision Zenith took. Orphanned again.

If you can live with the copy protection, and the knowlege that what you has is what you keeps, then Z-100 dBase III is a powerful and useful tool. However, I have discovered a way of having the functionality and more, without the hassle, and at one-fourth the price.

An Alternative

dBXL, from WordTech Systems (P.O.Box 1747, Orinda, California 94563; (415)254-0900; list $169, discount available) is a total dBase III functional clone: an almost fully compatible language, with the details of screen presentation, etc., differing. The language restrictions of dBXL as against dBase III Plus and dBase III position it very close to Z-100 dBase III in functionality: more than III and less than III Plus. However, since dBXL promises ongoing development as a generic MS-DOS product, the probability of growth seems higher (at least nonzero).

It also contains several features not available in dBase at all: notably user-definable and - controllable windows and keyboard reconfiguration.

Its major drawbacks (I have not yet used it extensively, and cannot comment on speed or reliability) result from the fact that, being generic, it controls the screen with the least common denominator of screen commands, meaning that window drawing is rather creaky. It could be a lot smarter about screen handling without loss of generality: one hopes improvements will come.

Some Installation Tips

If you purchase dBXL, the first thing to check is that the .EXE files you receive are not dated 1/5/87 but 2/1/87 or later. WordTech corrected several MS-DOS interface problems at that time without changing the dBXL version identification. (Nobody's perfect.) With dBXL you receive a 'free update' card. If you have the 1/5 version and want to change, don't expend your free update: complain where you bought the system or to WordTech that you aren't updating, that you are seeking a program that works in the first place. Otherwise they may want that card, which is presumably worth hanging on to. Firmness worked for me.

Once you have the right dBXL, there are two parts of the installation procedure I should mention.

The manual says that you need the ANSI driver, and the Install program even mentions a Z-100 as a possible machine! If you are using that driver and have a monochrome machine, that's fine, but you have options. Although not clearly documented, it is possible to install dBXL with color and without the ANSI driver. In the Install program, simply choose 'user specified' rather than Z-100 as your terminal. You will then be asked for various escape sequences, for which you supply the native Z-100 sequences from your Z-100 manual. The sequences you supply, including color setting sequences, will be used instead of ANSI ones: hence dBXL runs without ANSI. (One of the problems with the 1/5 version was that it used a mixture of these sequences and ANSI ones.)

Install also allows specification of line drawing characters, but I have not figured out a way to get it to use Z-100 graphics mode. It allows merely the specification of single characters to use for the frames of windows, and makes no allowance for associated escape sequences. Hence you must use equals signs, hyphens and the like rather than the true lines available on the Z-100. Similarly, it allows the specification of only ten function keys. I have written WordTech about these two points, but am not sanguine about seeing them changed.

As far as the keyboard interface is concerned, dBXL allows considerable flexibility there. Install will walk you through most (unfortunately not quite all) of the dBXL control keys and let you show it which key you wish to use for each.

Shifted Arrows

I like to use my numeric keypad for program control rather than numeric entry, and hence wish it to be as rich in options as possible. I like, therefore, to have a different command associated with shift-arrow than with arrow (page up versus cursor up, for example). When powered up, the Z-100 is in 'keyboard expansion' mode, which means that keypad and function keys generate escape sequences when struck, and that arrow and shift-arrow generate the same one. The number keys generate the same thing as those on the main keyboard. If keyboard expansion is turned off (like turning off key click) with the sequence '<escape>,y,?', then each keypad and function key instead delivers a unique code larger than 127 -- a different one shifted and not shifted.

Access to these is readily possible with dBXL, as Install provides for the specification of an escape sequence meaning 'prepare to receive color commands'. It can be told to issue a Leave Keyboard Expansion command here, making the non-numeric keypad available. (Another problem with 1/5 was that it only looked at seven bits of input characters, making this usage impossible.)

If you wish to use this method when talking to dBXL, but want to have the numbers on the keypad available to a certain

Continued on Page 82

Getting Started With . . . MASM

Patrick Swayne
HUG Software Engineer

In just about every issue of REMark, and in many other computer magazines, you will find assembly language source listings of programs. These listings usually accompany articles, which usually contain statements like this: "You can create the program FOOBAR.COM mentioned in this article by typing in and assembling the listing at the end of the article." But many readers are not familiar with the process of producing a running program from an assembly listing.

In this article, I will describe the steps required to type in an assembly listing and then convert it to a program. I will only cover what is actually necessary to produce the program, and will not go into detail about the operation of MASM, LINK, etc. This article will not be a course on assembly language. I am concerned with showing you how to get programs working and not with explaining how they work.

Note: This article assumes that you are familiar with general MS-DOS operating procedures, and that you have a copy of the Microsoft assembler, MASM. MASM is available from Heath Co. and Zenith Data Systems as part of the Programmer's Utility Pack (catalog no. CB-3163-30), and it is also available from Microsoft and several independent software vendors. An early version of MASM was also supplied with Z-DOS, the original version of MS-DOS supplied for Z-100 computers. This early version of MASM can probably be used to assemble most assembly programs appearing in REMark, but it will not accept full path descriptions (including directory names) as part of file names supplied to it.

Typing in the Source Code

The source code listings that appear in REMark can be typed in using any editor (such as EDLIN that comes with MS-DOS), or with any word processor that is capable of producing plain text files that do not contain special word processing codes. Plain text files are often called "ASCII files", so you might want to look for that term in your word processor manual.

When you type source code from a magazine listing, it is a good idea to type the code exactly as it is printed. However, it is possible to reduce the amount of typing by leaving out the comments in the code. In any case, you should be familiar with what part of the listing is actual code, and what part is comments. It is easy to tell what part is what if you think of assembly code as consisting of "fields". These fields are the label field, the opcode or instruction field, the operand field, and the comment field. They are laid out in a listing as shown below

```
label       opcode      operand          comment

START:      MOV         AX,0             ;GET A ZERO
            MOV         DS,AX            ;MOVE ZERO TO DS
```

The comments within the comment field are introduced by a semicolon, making it easy to tell them apart from the actual code. Comments can also be placed by themselves on a line, again using a semicolon to introduce them, as in this example:

```
; This is a comment line.
```

There is also a COMMENT assembler instruction that may be used to introduce a block of text within the code. These comments can be left out as well, but if you are not sure if something is a comment, leave it in.

Assembly listings are sometimes printed in REMark "sideways", because some of the lines are too long for efficient layout the normal way. If you see a listing printed sideways, and it covers more than one page, you should be aware that both columns on a page are to be typed before you move on to another page.

Assembling the Program

After you have typed in a program listing, and have checked it out thoroughly, you can assemble it into a program. The procedure for assembling the program will depend on whether the finished program is to be an .EXE file or a .COM file. The article should tell you what kind of file it is to be.

The assembler, MASM.EXE, along with a program called LINK.EXE (supplied with MS-DOS), and sometimes a program called EXE2BIN.EXE (also supplied with MS-DOS) will be required to assemble the source listing into a program. Copies of these programs should be placed on your system disk either in the root directory or in a directory pointed to by your PATH description.

The assembler, MASM, does not create an executable program by itself. Instead, it creates a file in a special format called the object format. This "object file" must be further processed by the LINK program, which creates an .EXE file from the object file. If the program is to be a .COM file, the .EXE file produced by LINK must

be processed again by EXE2BIN to make the .COM file.

All of this may sound confusing, but you really do not have to remember any of it. All you have to do is to create a couple of little batch files, which will do all of the work for you. These batch files are called ASME.BAT and ASMC.BAT.

To create ASME.BAT, make the drive/directory containing MASM.EXE the default drive/directory, and type the following lines, pressing Return after each line:

```
COPY CON ASME.BAT
MASM %1,%1;
LINK %1,%1;
ERASE %1.OBJ
^Z
```

The ^Z means to type Control-Z (hold down the Ctrl key and press Z). If the system prompt does not re-appear when you press Return after typing Control-Z, press Return again. To create ASMC.BAT, type these lines:

```
COPY CON ASMC.BAT
MASM %1,%1;
LINK %1,%1;
ERASE %1.OBJ
ERASE %1.COM
EXE2BIN %1 %1.COM
ERASE %1.EXE
^Z
```

To verify that you have created the batch files, enter

```
DIR ASM?.BAT
```

A directory listing of the two files should be displayed. Now, all you have to do to assemble a source listing into an .EXE file is to type

```
ASME filename
```

where filename is the name of the file you want to assemble. The only restrictions to using this batch file are that the source file must have an .ASM extension, and that you must NOT include that extension in the command line. For example, if the program you are assembling is called TEST.ASM, you would type ASME TEST to assemble it.

You can use the same procedure with ASMC to create a .COM file. Just type

```
ASMC filename
```

where filename is the name of the file you want to assemble. With either of these batch files, the filename you specify can be a complete path name including a drive designation and/or directory name before the actual file name. The resulting .EXE or .COM file will be placed in the same drive/directory as the source file.

If you are creating a .COM file (using ASMC.BAT), you will probably see a message like this when LINK runs:

```
Warning: no stack segment
```

It is normal to see this message when you are linking a file in the .COM format, and you may ignore it.

After you have created your .COM or .EXE file, you can test it. The article containing the source listing will usually have instructions for using the program you have made.

Assembly Errors

If MASM prints error messages while it is assembling the program, it means that your source code probably contains typographical errors. You should examine your source code carefully, comparing it with the original in the magazine. If you cannot find any errors, have another person compare your code with the magazine. If you are absolutely certain that your code does not contain typographical errors, you may want to write to the author of the article to see if the magazine printed his code correctly.

More Efficient .EXE Files

If you have version 3.00 or higher of the LINK program, you can use a switch in the LINK command line (/EX) that will cause it to create a more efficient .EXE file. The size of this .EXE file will be reduced as much as 50% or more compared to the size of the file produced without the switch. To create a batch file that takes advantage of the switch, type these lines:

```
COPY CON ASMEX.BAT
MASM %1,%1;
LINK /EX %1,%1;
ERASE %1.OBJ
^Z
```

To assemble a program with this batch file, enter

```
ASMEX filename
```

where filename is the name of the program to assemble. Not all programs can be linked properly with the /EX switch, and the LINK program will indicate this with an error message. A very few programs will link ok, but they will not run properly. If you use ASMEX to make a program, be sure to test it thoroughly. ✱

Continued from Page 44

dBXL program, that program can type an Enter Keyboard Expansion command when starting, and a LKE command on completion. Don't forget to provide a separate program that merely does a LKE, for use when the first program aborts before getting a chance to do so. And remember, if you do this, that numeric keypad numbers and the function keys cannot be available at the same time (the function keys having been assigned to codes rather than escape sequences).

Of course, dBXL will work fine without this special handling. Let the keyboard remain in normal mode and keypad numbers and function keys will be available as usual. The enhanced functionality of key assignment remains, either way. It is only limited by the restriction that control key functions cannot be reassigned: meaning, for example, that one has to live with ctrl-Y for delete line rather than being able to assign it to the DelLine key. (Of course you could always use the MS-DOS Font command to cause DelLine to generate ctrl-Y.)

Super dBASE

As a final note, whether dBASE or dBXL is used, WordTech offers a dBASE/dBXL compiler called Quicksilver. I haven't tried this, but it is advertised as accepting the same language as dBXL, and is also offered as a generic MS-DOS product. For someone with large dBASE applications, or for systems developers, this royalty-free product probably offers dramatic speed and space improvements. Ashton-Tate makes a compiler too, but it is of course not available on the Z-100. The cost of Quicksilver is about the same as that of dBase III. ✱

Announcement!

HUG MEMBERS ONLY!!

The HUG-386 and HUG-386-C upgrade kits will be available shortly. Wheelin' Dealin' Jim has managed a super-fantastic deal on these two products for Heath Users' Group members who originally purchased an H-241 or H-248; ***one-thousand two-hundred dollars*** off the regular purchase price! That's right! If you originally purchased an H-241 or H-248, and you're a HUG member, you can get $1200.00 off the regular retail price of either of these two upgrade kits!

The HUG-386 and HUG-386-C are upgrade kits that let you upgrade your H-241 or H-248 series computers up to a full H-386. Now, how do you determine which upgrade kit to buy? The H-386-C includes a dual winchester/floppy controller, while the H-386 does not include any disk controller. Since the old H-241 controller is not '386 compatible, you'll probably want the "C" model if you're upgrading a '241. If you're upgrading a '248, your decision will depend on whether you need a new dual controller or not.

Here are the three ways you can order your upgrade:

Write-In Orders

- Non-HUG members ***can*** order by including payment (with the upgrade kit order) for one year's membership in the Heath Users' Group.
- All orders should be submitted to the Heath Users' Group.
- Each order must indicate the model number of the upgrade kit desired, and which computer kit it was purchased for.
- Each order must have the persons HUG ID number written on it.

Phone-In Orders

- Non-HUG members ***can*** order by first ordering a one year's membership in the Heath Users' Group.
- All orders must be phoned in to (616) 982-3838 from 8 AM to 4:30 PM EST.
- Each order must indicate the model number of the upgrade kit desired, and which computer kit it was purchased for.
- The person ordering must supply his/her current HUG ID number.

Heath/Zenith Computer Store Sales

- Non-HUG members ***can*** purchase an upgrade kit by first purchasing a HUG membership from the store.
- Orders for the upgrade kit can be taken in the normal fashion.
- Each order must have the buyer's HUG ID number on it.
- Each order should indicate which computer kit the upgrade was purchased for.

Heath/Zenith Users' Group

P.O. Box 217
Benton Harbor, MI 49022-0217

BULK RATE
U.S. Postage
PAID
Heath Users' Group

POSTMASTER: If undeliverable, please do not return.

P/N 885-2101