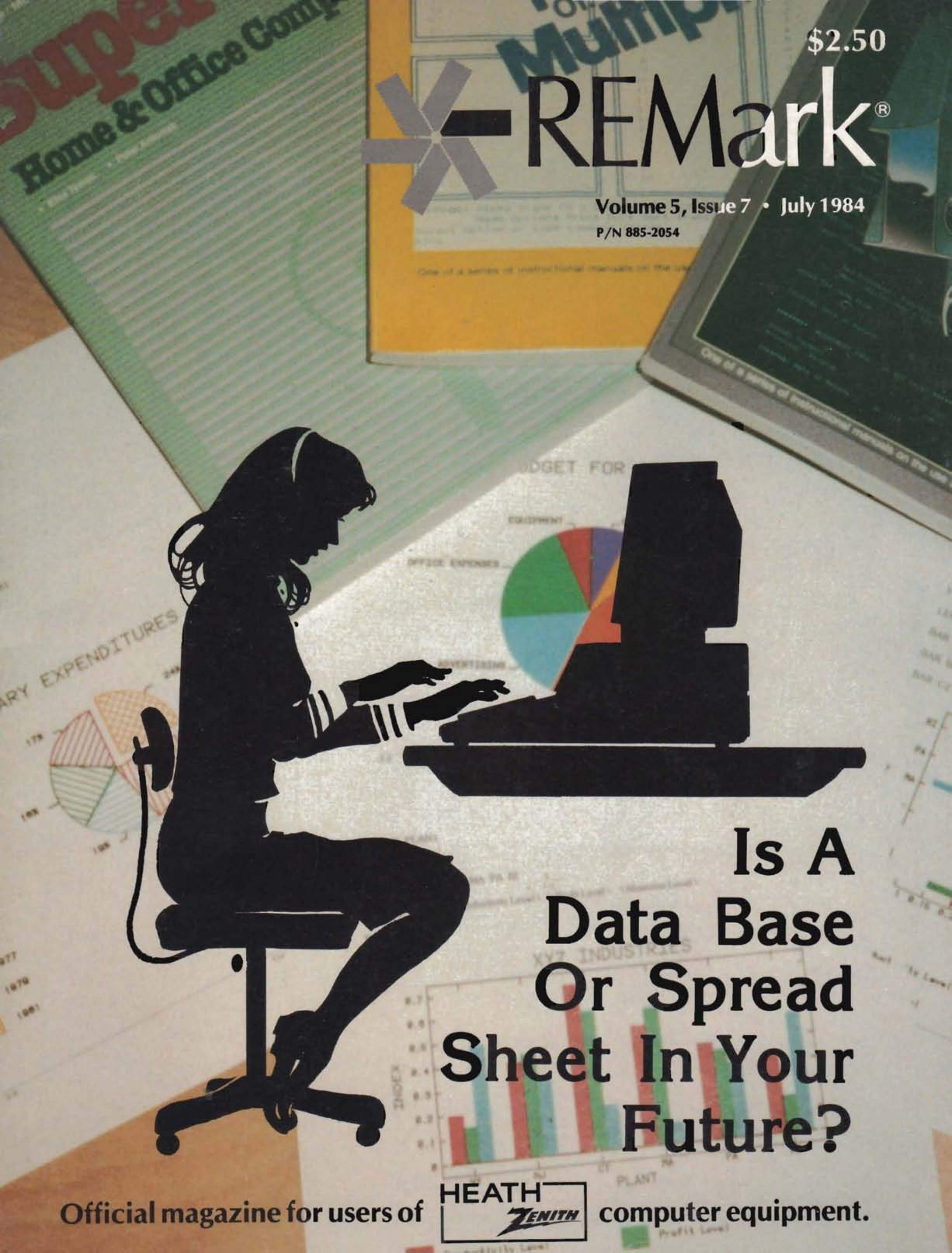
$2.50

REMark®

Volume 5, Issue 7 • July 1984
P/N 885-2054

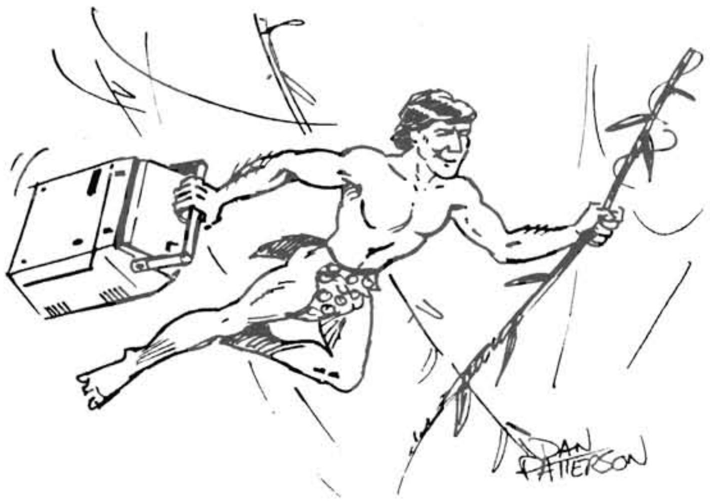


Is A Data Base Or Spread Sheet In Your Future?

Official magazine for users of HEATH ZENITH computer equipment.

SWING INTO ACTION

with the new Z-160 portable PC from Zenith!



You say you want portability and PC compatibility?
Zenith quality and performance?
You say you want a good price, and you want it now?
Then say Software Wizardry!

The Z-160 is the new portable personal computer from Zenith, and Software Wizardry has them, at a price that will make you give a jungle yell!

The Z-160 is Zenith's hot new PC addition to the Z family, and as usual, they do it better—text and full color graphics without funky add-in cards, up to 320K RAM on the main board, and MS-DOS included!

Don't stampede through the jungle with the Big Blue Elephants; just take a look at these features!

- Fully IBM^tm PC hardware and software compatible
  - will run Flight Simulator, considered an acid test
- More standard features
  - 9-inch high resolution amber monitor
  - detachable low profile keyboard
  - four open slots
  - extensive self diagnostics
  - RGB color and monochrome monitor interfaces
  - two serial RS-232 ports
  - Centronics type parallel printer port

  - 128K RAM, expandable to 320K on the main board
  - Total memory expandable to 640K

- Extra features
  - smooth scroll on monitor
  - booting from either drive
  - superior configuration program
  - four selectable display typestyles
  - MS-DOS 2.1 operating system included standard

DID SOMEONE SAY MEMORY?

They say in the jungle that elephants never forget; well, your Zenith won't forget either with RAM chips from Software Wizardry.

Guaranteed first quality, to work in your Z-100, Z-150, or Z-160

ZENITH | data systems
AUTHORIZED SALES & SERVICE

Zenith and Software Wizardry are synonymous with quality, capability, and support. Whether the top of the line Z-100 dual processor or the new PC systems, we know Zenith; so grab a vine, and swing our way!

Software Wizardry
THE MAGIC TOUCH

1106 First Capitol
St. Charles, MO 63301
(314) 946-1968

Software Wizardry, Inc. also has over 450 other items of interest to Heath/Zenith users as well. Call or write for our FREE price list. Dealer Inquiries and HUG group purchases welcome!

MasterCard VISA

Heath/Zenith Users' Group

Staff

Manager	Bob Ellerton (616) 982-3867
Software Engineer	Pat Swayne (616) 982-3463
Bulletin Board and Software Developer	Jim Buszkiewicz (616) 982-3463
Software Coordinator	Nancy Strunk (616) 982-3838
Secretary	Margaret Bacon (616) 982-3463

REMark

Editor	Walt Gillespie (616) 982-3789
Editorial and Advertising Assistant	Donna Melland (616) 982-3794
Graphics and Layout Assistant	Greg Martin (616) 982-3463
Printers	Imperial Printing St. Joseph, MI

	U.S. Domestic	Canada & Mexico	International
Initial	$20	$22*	$30*
Renewal	$17	$19*	$24*

*U.S. Funds.

Membership in England, France, Germany, Belgium, Holland, Sweden and Switzerland is acquired through the local distributor at the prevailing rate.

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

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

REMark®

Volume 5, Issue 7 • July 1984

on the stack

DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES

PRICE BREAKTHROUGH

Super Sale on New Disk Drives

Starting at $~~169.00~~! NEW LOW PRICE

Tandon — Siemens — Remex — MPI — Teac — Shugart — Tabor

40 or 80 Tracks — Single or Dual Head — New 3½'' Drivette™
Our Disk Drives are Capable of Single **and** Dual Density Operation

The **NEWEST** Technology Capable of Cperating on Most Popular Computers

Drive a Hard Bargain!!™ For your TRS/80, Color Computer, IBM, Apple, Franklin
5 M.B.-20 M.B. Complete Systems...... from $999.95

Diskette Breakthrough — 10 Pack in Library Case — $~~18.95~~ NEW LOW PRICE

SAVE!! PLEASE CALL FOR OUR MOST CURRENT PRICE REDUCTIONS.

TOLL FREE ORDERING
1-800-343-8841

GENERAL AND TECHNICAL
1-617-872-9090

Disk Drives (0123) TRS/80-IBM-Apple - TI Franklin-Max/80-LNW ..... ◄
Model I/III/IV Upgrade (Disk Drives - Memory)........................ ◄
Printers — Daisywheel/Dot Matrix .... ◄
Percom Double Density Controller (Model I) .... ◄
Color Computer Printer Interfaces .... ◄
Disk Drive Operating Systems .... ◄
Repair Services Now Offered — FAST Turn-a-Round........................ ◄
Apple/Franklin Compatible Add-On Drives with Case & Cable.......... ◄
Diskettes in Library Cases .... ◄

CALL TOLL FREE FOR NEW PRICES

DISK DRIVE CASES AND POWER SUPPLIES .... starting at $59.95
Printer **Buffers** 8K to 512K .... starting at $143.95
Holmes Model I/III Speed-up Mod .... starting at $90.00
Cables — Printer/Disk Drive .... starting at $23.00

Warranty on Disk Drives — 6 Months to 1 Year

SOFTWARE SUPPORT, INC.

One Edgell Road, Framingham, MA 01701 (617) 872-9090
Hours: Mon. thru Fri. 9:30 am to 5:30 (E.S.T.) Sat. 10 am to 4:30 pm

DEALER INQUIRIES INVITED.

TERMS:
M.C./Visa/Amex and personal checks accepted at no extra charge.
C.O.D., please add $3.00.
Shipping: Please call for amount.
Not responsible for typographical errors.

CANADA
MICRO R.G.S. INC.
751, CARRE VICTORIA, SUITE 403
MONTREAL, QUEBEC, CANADA, H2Y 2J3
Regular **Tel. (514) 845-1534**
Canadian Toll Free 800-361-5155

Service! Service!

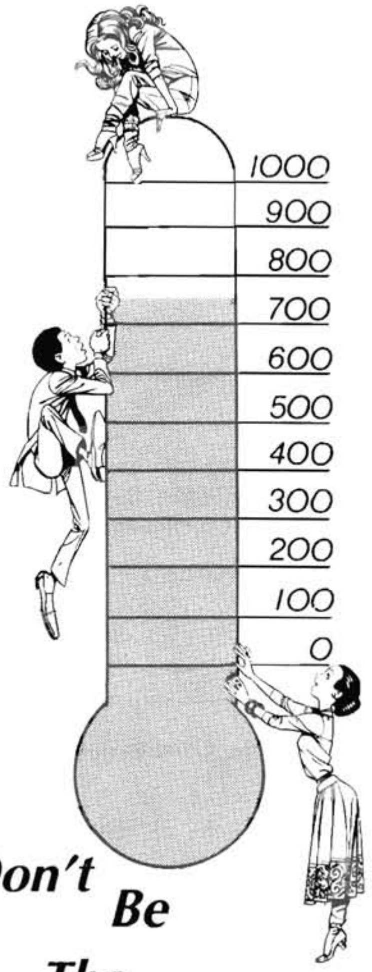
All in stock products are shipped within 24 hours of order. Repair/Warranty service is performed within 24 hours of receipt unless otherwise noted. We accept C.O.D., foreign and APO orders. School and D&B corporate P.O.s accepted.

TRS/80 Registered Trademark Tandy Corp. IBM-PC Registered IBM Corp. Apple Registered Trademark Apple Computer Corp. Franklin Registered Trademark Franklin Corp. Max/80 Registered Trademark Lobo Int.

DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES

Be There!

International Heath/Zenith Users' Group Conference
Heath/Zenith Users' Group
Saint Charles, Illinois • July 27, 28 & 29, 1984



Don't Be The One Left Out, Send Your Registration Form In Now!!

INTERNATIONAL HUG CONFERENCE

Official Conference Registration Form

Pheasant Run Convention-Resort Hotel

July 27, 28 and 29

Name(s) ______________________________

Address ______________________________

Company ______________________________

City ____________ State ____________ Zip ______

Enclosed is $22.00 per individual to attend the International HUG Conference to be held the weekend of July 27, 28 and 29, 1984. Please send tickets along with information regarding hotel reservations and transportation.

Amount Enclosed: __________ Number attending: __________

For our information:

Which Heath/Zenith computer do you now operate? ______________

Are you a Non-User-Attendee? ☐Yes ☐No

Are you a Heath/Zenith related vendor? ☐Yes ☐No

If yes, do you want exhibit space during the Conference? ☐Yes ☐No

Special Notice to Vendors:

Vendor Information Packages will be made available to Heath/Zenith Related Vendors who are planning to exhibit their products while at the Conference. You must contact us prior to May 1, 1984.

For your information:

The $22.00 you are paying for your reservation to the International HUG Conference entitles you to all functions of the Conference. This includes Saturday breakfast, buffet lunch and hors d'oeuvres in the evening. The Prize Drawing will be held during the Saturday evening Cocktail Party. You must be present to win. Vendors and $22.00 ticket holders will be eligible for prizes. ALL prizes will be awarded at that time.

Visitor tickets, for those of you simply attending the seminars and looking at the exhibits, are available for $10.00. Visitor Tickets do not include meals or eligibility to the Prize Drawings.

Send your registration form or a suitable copy to:

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

Registration(s) must be postmarked no later than July 15, 1984. Cancellations will not be accepted after this date.

BUGGIN' HUG

Some Corrections To "Computer Graphics On the H/Z-100" (May Issue)

Dear HUG,

I have just received the May 1984 issue of REMark and thoroughly enjoyed the H/Z-100 articles.

Randy Meyers' article on page 10 is an excellent description of Computer Graphics on the H/Z-100, and the assembly language listing is an almost verbatim listing of the routine that I had come up with while implementing figFORTH on my Z-100 to run under Z-DOS.

However, for the benefit of other HUGgies that may have come away a bit confused about the article or found difficulty with the running of the listed code, I would like to offer the following corrections.

Page 11, right column

```
row := ( ( Y_coordinate / 9 ) * 16 ) +
       ( Y_coordinate mod 9 )

char_byte    := X_coordinate / 8

bit_address  := X_coordinate mod 8
```

Page 13, right column

The third line under the label "do_green" should read as follows:

```
test   cl,green_bit     ; Green bit set?
```

Also, Randy's colors for variable "Value" give the mapping:

0 =	000 =	Black
1 =	001 =	Blue
2 =	010 =	Red
3 =	011 =	Magenta
4 =	100 =	Green
5 =	101 =	Cyan
6 =	110 =	Yellow
7 =	111 =	White

There is nothing in error with this color assignment, however, I would wager that most H/Z-100 owners are more familiar with the Z-BASIC color assignment which maps thusly:

0 =	000 =	Black
1 =	001 =	Blue
2 =	010 =	Green
3 =	011 =	Cyan
4 =	100 =	Red
5 =	101 =	Magenta
6 =	110 =	Yellow
7 =	111 =	White

If other HUGgies wish to use the Z-BASIC color assignment, make the following changes to the assembly listing:

Page 12, right column

```
red__bit   equu   04h  ; Value for red plane enable.
green_bit  equ    02h  ; Value for green plane enable.
```

Thanks again for another winning REMark edition. I'm looking forward to the next issue.

Chuck Rogalo
P. O. Box 1020
APO San Francisco 96555

Patches Needed For Magic Wand

Dear HUG,

I've seen several letters in Buggin' HUG in REMark concerning the inability to use all of the features of Peachtext 5000 with dot matrix printers. I have a similar problem. I have Magic Wand, also a Peachtree product, and a C.Itoh Model 8510 printer. The printer is capable of incremental vertical and horizontal feed and proportional printing. However, Magic Wand does not support these features. Has anybody made any patches, or could anyone give some hints as to where to patch Magic Wand to support the escape codes to utilize the features of the Itoh (or Epson) printer? Thank you.

Peter E. Walberg
3352 East Orchard0 Drive
Decatur, IL 62521

Correction to "A ZBASIC Program" in Buggin' HUG, May Issue

Dear HUG,

Here is a correction to a Buggin' HUG letter that was published in the May 1984 issue, Volume 5, Issue 5. On page 8, the letter titled "A ZBASIC Program", line 400 should read a follows:

```
IF POINT((RM+9),(189-D))=4 THEN BEEP:GOSUB 440
```

Mark Van Sickle
Heath Co.

Help Needed From Someone Using the Z-100 Business Graphics Pkg.

Dear HUG,

I found a bug in the BASIC program on the Demonstration diskette for the H/Z-100 microcomputer. On stmnt. 360 it reads:

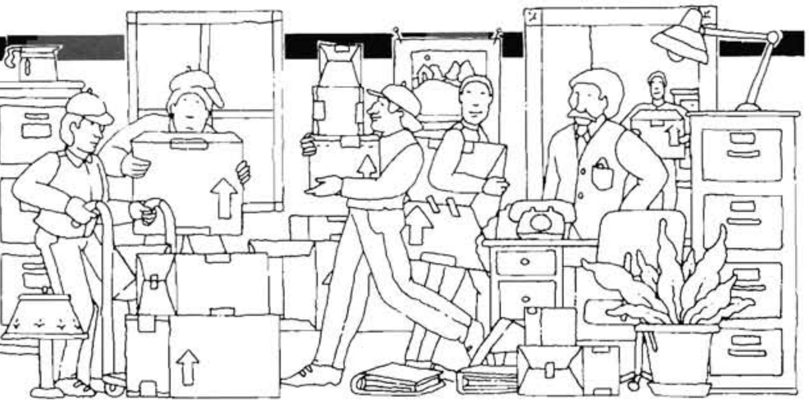
```
360 CLS:GOTO 360:END 'abnormal end
```

This obviously forces the program to do an infinite loop. However, by correcting this error, I found myself inside the huge Business Graphics program. Since I have not bought a printer, I was unable to debug the program. The error message I get refers to stmnt. 125. I hope someone who uses the Z-100 Business Graphics could be of some help in this problem. I am the only HUG member in my area. I would very much welcome any HUG members in Southern Africa. My contact address is 3 Yeatman Road, North End, Bulawayo, Zimbabwe.

Thank you and greetings to many HUG members all over the world.

Sobantu Ndimande
3 Yeatman Rd.
North End
Bulawayo, Zimbabwe

Vectored to 64 ☞



Installing Your Business Computer

D. C. Shoemaker
HQ US European Command
Box 897
APO NY 09128

Your computer is coming. After looking at everything the marketplace has to offer, you've decided that it's time to add a computer to your office. You've read multitudes of articles on what's important to look for in computer selection, the kinds of software available, what sort of customer support and after-sales service to check on, and a hundred other details. You know about purchase versus lease decisions. You've done all your research, made your choices, and the system will be delivered next week. In one sense, the most difficult part of your office automation plan is behind you. But in another sense, you're only half way to your goal. The important question now is what to do to get your money's worth from all that hardware and software you've laid out your hard-earned sheckels to lease (you did lease your system with a no-cost option to purchase, didn't you?). We're going to discuss things to consider when you're making your plans to install and set up your computer so that it will be an efficient part of your office. This is a subject you don't often see much about. It isn't glamorous or exciting, but how you install your computer system can mean the difference between success and failure in your office automation project.

We're going to discuss installation and operations from the point of view of the beginner. The only assumptions are that you have a computer and an office to put it in. Much of what this article will cover may seem trivial or obvious at first, but each point represents someone's past error or problem, often an expensive one. Even if you think it's a trivial point, you might benefit from considering the discussion. If we don't cover your exact problem, we may come close enough to give you some insight into your particular situation that will save you money, exasperation, or both.

The old cliches are often all too true. "An ounce of prevention is worth a pound of cure" definitely applies in this business. After the computer system is installed, it can be very expensive to make corrections. Some problems can be cured only with great difficulty after the fact, often requiring a complete re- installation. Do it right the first time, and have a plan to work from. The pointers that follow will help you prepare for a successful, effective, and efficient installation.

Office environments are made up of many components. Let's list some of the most important: light, sound or noise, dirt and dust (including smoke and food particles), electrical power, space (and the existing office furniture that occupies that space), operating procedures, and most important of all, your employees. This list is far from all-inclusive; it's meant to help focus your attention on the things that can have a major impact on how your office automation project turns out. For now, we'll just look at these.

Office lighting is probably one of the most misunderstood and ignored parts of the office environment. Lighting conditions will have a profound impact on how your office system as a whole operates, and this includes your computer. A properly lighted office will usually require no major changes to accomodate a computer and its users'. Unfortunately, most offices have terrible lighting. The worst problem is glare, both from the overhead lights and from windows. Glare makes it difficult enough to read a paper, but on the glass of the computer terminal's screen, it's intolerable. Most new computer terminals have non-glare screens, but even that won't help much if the source of the glare is very strong. The computer's users will have a difficult time just reading the screen, and the long term effects are eye-strain and undue fatigue. Any steps you can take to reduce glare will benefit the users' in the long term.

The first step in coping with glare is to position the computer terminal to avoid as much excess light as possible. This may mean that the window view you thought would be so nice as a restful break from staring at the screen might not be a good idea. The worst place to put the terminal is with the user's back to a window. It will probably work all right if the terminal is placed across the office from the window, provided the screen is facing away from the window. A small partition can also help cut down on the light from other windows.

The other source of glare is the office lighting system itself. Many older offices have harsh fluorescent lighting that casts strong light on everything. I've found that if the room lights cast pronounced shadows on papers on the desk, the light is probably going to cause glare problems on the screen. Your computer's users must have enough light to see well what they're reading from, but avoid the reflections. "Egg-shell" diffusion covers that fit over fluorescent light fixtures offer a low-cost fix to the problem. Many office supply vendors carry terminal glare shields that act as a short hood over the screen, which can be a big help. There's a lot of personal preference involved, but many people find that soft ceiling lights and directional "goose-neck" lamps for the desks can make a good compromise that doesn't cost too much.

Sound is the next problem to examine. No office is particularly quiet. One consideration is what to do about the background office noise that makes it difficult for the computer user to concentrate. These problems are the same ones that impact on other office workers, including traffic noise, voices, doors opening and closing, fans and air conditioning, and other office machines. The other consideration is what to do about the sound produced by the computer system itself. The problems are interrelated, and when we solve the problem of the noise associated with the computer, we've taken a big step toward solving the general noise problem.

What's so noisy about a computer? A great deal depends on the computer, but some things are universal. The two main sources of noise are the power supply fans in the computer, the terminal, the printer, or all three, and the noise of the printer. Some fans are quiet, and some make a fair amount of noise. Wang, Zenith, IBM, and many others all have fans that run all the time the system is on, and the noise gets to you after a while. In a quiet room it can be irritating over a period of hours. In a larger room, like a sales display area or a carpeted office, you might not notice it so much. What did the computer at the sales display sound like? You probably didn't notice, and that's our clue to solving the office problem. The display area probably was carpeted, and likely had sound-absorbing partitions strategically located about the area. You can carpet the office floor. A good carpet can actually be easier to clean, and absorbs sound like crazy. Use acoustical tile for the ceiling. There are many patterns to choose from, not just the perforated off-white tile of years ago. This will also give you a chance to re-do the colors of your office. They may well need it. There are many good office interior decorator/consultants around. They may not know much about installing computers (some do), but they can help with things like colors and tones that can make a big difference in the work environment, but that is beyond the scope of this article.

The worst noise levels generally come from the printer, unless you're fortunate enough (or have enough of a fortune) to afford a laser or jet-ink printer. The sounds made by impact printers, where some moving part strikes the paper, can be overwhelming. Daisy-wheel printers often come with sound-proof boxes as options. Get one for each of your printers. If you have a dot-matrix printer for graphics, get a box for it, too. But see that it has a large, quiet fan for cooling. The larger the fan, the slower it must turn to provide cooling air, and the slower the fan, the quieter. It's those small, high-speed fans that create the most noise.

If you have to have a printer that runs a good part of the day, producing extensive reports that use continuous forms, think about locating it in another room. You'll disturb fewer people, and the paper dust associated with large, fast printers that use continuous forms will be kept more or less in one place.

The next problem is dirt and dust. No matter how clean you think your office is, installing a computer will demonstrate that you're wrong. All you'll have to do to see this for yourself is examine the computer's filters after just a few days' use. Computers are sensitive to a dirty environment, some computers more so than others. The most sensitive part of the system is the floppy disk drive subsystem, and the disks themselves are the most sensitive part of the subsystem. Disks store data in the form of magnetic patterns that are extremely small, and the least speck of dust can obscure or obliterate the data at that point. Worse, the dust can physically damage the disk surface. Unlike people, who can work around small blots on a piece of paper, the computer is a literal-minded beast, and it has to be able to read all of a disk data file. If it can't, you're in trouble. Such problems can't be prevented altogether, which is why you're constantly cautioned to make back-up copies of any disk data files you think you may want to use again someday. However, there are a few simple things you can do to hold the dirt problem to a minimum.

First of all, smoke is a serious pollutant, not only to people but to your computer's disks. If you must smoke, don't do it around your computer. Go to some other room. If your computer users' smoke, it will increase the possibility of disk damage several hundred per cent. Long term effects on other computer components are just as severe. A clean machine runs better than a dirty one, and the computer attracts dust and smoke particles from the air just as any other electronic device.

If your office is in a dusty environment, such as a factory or other dust-producing activity, consider air conditioning. If you have air conditioning but factory workers must frequently enter the office to do their business, thereby letting in dust from outside, think about a separate room for the computer. Do anything you can to reduce the amount of dust in the air.

Eating around the computer is an automatic taboo. The same goes for drinking. One good coffee spill, especially if the coffee has sugar in it, spells disaster. Many keyboards will simply short circuit and stop working. Others simply become sticky. You may end up having to replace the keyboard, one of the few good arguments for separate keyboards. Crumbs from the morning Danish can also cause problems, and not just because they attract bugs. And there's always the danger of spilling something on a disk. That's instant disaster.

This is probably a good place to say that when something bad happens to one of your disks, it's often possible to recover most, if not all, the data. It's a tricky job, and one that's not usually covered in the operator's manual. The services of a computer professional (or your next door neighbor's thirteen year old computer whiz) will be required, and this takes time. And time is money if you're in business. So do it right the first time. No eating, drinking, or smoking in the same room as the computer. That's cheap insurance.

Clean power for your computer is just as important as a clean environment. In fact, to your computer, power is a major part of the system environment. There are a lot of things that can go wrong. Not enough power is sometimes a problem, but fairly rare in these days when a computer system draws about as much current as a couple of light bulbs. But as you begin to add peripheral equipment such as printers and hard disk subsystems, the power requirements grow.

The best approach is to provide your system with its own line, to which nothing else is connected. An electrician can do the wiring for your office without much difficulty. A dedicated power line means that no one in the next room has connected a high-current device to your circuit, one which will cause surges when it's turned on and off. Large coffee makers, the blow dryers found in washrooms, duplicating machines and the like can all cause transients, surges and other electrical "noise" that can affect many computers. Some computers are more resistant to such noise than others, but the cleaner the line, the better.

Another advantage to a dedicated line is that it's less likely to be turned off accidentally. That can cause massive damage to a running computer. Not only will the computer lose the data you were working on when the power failed, but there's a possibility of disk damage whenever there's a power failure. When some computers die, they send a spray of meaningless data throughout the system, and this sometimes can cause the disk drive to write some random data to the disk. If this happens in the area of the disk that stores the directory, that may make the disk useless. For that reason, no matter where the computer is plugged in, be certain that it can't be turned off accidentally, as by someone tripping over the cord or throwing the wrong switch.

This shouldn't preclude you from taking the necessary precautions to be sure you can turn off the computer system in a hurry if you have to. Fire is a good example. If something really serious happens to your system and it catches fire, the ability to hit a "panic switch" and kill the power will reduce the damage greatly. You may well lose some data, and some of the computer will be damaged by the fire, but once the power's off, you can extinguish the fire with anything available, even water, with safety. Water won't hurt the computer once the power is off. The system won't rust, dissolve, stain or anything else. It can be dried off. But if the power's on when you spray the water, the resulting massive short-circuits will finish what the fire started (or make worse whatever started the fire.)

Keep in mind that while your office power seems clean from your point of view, it's possible that your next door neighbor is doing something on the power circuit that you share in common. If this something is a small arc welding machine (don't laugh, it's quite common in industrial park environments), you could have noise on your line that you might not suspect, coming from a block or more away. Any good electrician can check your line for stray noise, and can install filters if there's no other way to obtain clean power. In extreme cases, there are devices available that will completely isolate your system from any source of surges or noise. They're expensive, but effective.

It may already have occurred to you that an on-site demonstration of the system you select might not be a bad idea. This is another good reason for leasing the system before you buy it.

The other main resource under your control is the office space and the furnishings that occupy that space. Clearly, you'll need a place to put the computer, but beyond that, you should consider what you're going to put it on. This can have a big effect on how well your employees use the system. Any typist can tell you that there's an optimum height for the keyboard, and there are some chairs that are better than others for use when typing. There is a large variety of computer furniture to choose from. Don't try to get by with something that was all right to use with your old Underwood typewriter. Get a desk with adjustable height to insure that the keyboard is in the right position relative to the user. If your computer has a separate keyboard, be sure there's a good place to put it. I favor all-in-one systems because they're more damage-resistant and because when the keyboard is in the correct position, the rest of the computer is, too. If you have a separate keyboard, be sure it can't get knocked off onto the floor, be sure no one's going to trip over the keyboard cable, and be certain the rest of the computer (the CPU box and the monitor) will be located properly.

If you expect your users to sit in front of the computer for hours each day, give them decent chairs to sit in, with good back support and easy adjustments. Frequent adjustments to the user's seating position will help ease fatigue. Most computer-related fatigue complaints come from the positioning of the equipment, not from the actual use of the equipment.

Disk storage is another matter to consider. Most small systems use 5 1/4" floppy disks, which are surprisingly reliable and have a usefully long life (several years, if treated with care.) Always store disks upright, like records. Don't put anything on top of a disk laying on a worksurface. Keep them warm and dry, at the same temperature and humidity as the computer. If your environment has a humidity over 80 per cent, you can expect disk troubles from a peculiar characteristic of Mylar, the base material of the disk. Mylar can "swell" and bulge out of flat under certain circumstances. This phenomenon is sometimes called the "magic egg", and when it happens to your disk, you've got a serious problem. The disk will be unreadable due to the relocation of the magnetic pattern on its surface. You may be able to recover some of the data, but probably not all of it. Still another reason for a liberal back-up policy. Disks are cheap, but the time needed to re-create and re-enter the data on the disk is expensive.

While we're thinking about office furniture, we should consider another natural hazard often overlooked until it's too late. Static electricity can cause your computer to forget everything it was doing, including how to save your workfile and gracefully exit to the operating system. When this happens, you've lost possibly several hours of work, depending on what happens to your disk operating system and the disk directory. There are lots of anti-static precautions that you can take. First, use anti-static carpeting if possible. If not, there are anti-static sprays that can reduce the problem for short periods (up to a few days.) There are anti-static mats that can be placed at the computer workstation. And there are some environmental considerations. For instance, most mainframe computer installations keep the humidity at the manufacturer's recommended levels. If you have air conditioning, this should be easy.

Once the computer is installed, there are a few other precautions you should take. The first and most elementary is to insist, on penalty of death, that users cover the equipment when it's shut down for the day. The use of inexpensive, waterproof covers has saved many a system from damage when fire sprinkler systems trigger when no one is around. In some cases, a fire anywhere in a building will set off the sprinklers throughout the building, with predictable results. More cheap insurance.

Many office procedures call for leaving the computer on 24 hours a day. There are good and bad aspects to this policy. A computer, like most other electrical devices, is most likely to fail when power is being turned on. Hence leaving the computer going tends to increase the time between power-up failures. However, if the power fails for a moment during the night, the computer will be powered up with no one present to take any corrective action that might be required. And in case of fire, the system will be on when the sprinklers come on, with the almost inevitable short-circuits. Many offices process routine business transactions and print reports at night, so there's clearly some benefit to the risk. Whether the benefits outweigh the risks is a question for you alone to answer.

A related question is what to do with the computer during the day. On a day-to-day basis, it's definitely better to leave the system on all day. This is especially true if your computer has one or more hard disk (Winchester) drives. These are especially delicate, and there's a greater risk of failure when you turn them off. Winchesters also take a certain amount of time to "spin up" when you turn your system on. Turn the computer on in the morning when you're ready to use it, then leave it on. If possible, make the screen blank, to reduce the "burning" of an image into the phosphor of the tube, but that's not critical. Most 5 1/4" disk drives turn themselves off when not in use, but most 8" drives do not. In the case of the 8" drives, you may want to open the door and withdraw the disk an inch or so. This will reduce wear on the disk itself, and has the side benefit of having the drive "off-line" if the system crashes for any reason. It's possible to have your dealer install a simple modification to your 8" drives to make them turn themselves off when not in use, like the 5 1/4" drives. This is a worthwhile modification.

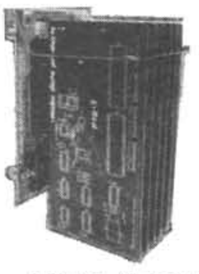
A word about computer use and computer-related strain might be useful at this point. There have been hundreds of studies, prompted by thousands of complaints, about the hazards of using computers. The most widely publicized complaints have dealt with the hazards of low-level radiation from the terminal's cathode- ray tube (CRT). The heart of the complaints is that staring into the CRT is harmful to the eyes. Symptoms run from tired eyes to headaches to backaches to all sorts of relatively exotic complaints. The vast majority of complainers know that something's wrong, but can't explain just what, so they assume it's the radiation from the computer.

There are all sorts of reasons why people complain about their workplace, but in this case, few actually relate to the computer. The radiation level is so low as to be undetectable, so the real problem is far more likely to originate from position. Tired eyes are a symptom of poor lighting. Headaches can be due to lighting or to faulty posture. Backaches are most likely to be caused by posture. All these things can be remedied by careful attention to installation. But there is another factor you should be aware of.

Stress in the workplace is probably the most common cause of complaints, and this is often especially true in relation to computers. The secretary who spends six or seven hours a day typing at the word processor is under a lot of strain. So is the data entry clerk and the person who does your financial analysis and modeling using Super-Calc or Lotus 1-2-3. The psychological strains of the job are far more difficult to cope with than the physical ones relating to the installation. I can't help you with these, other than to point out that they do exist. It's a problem that's often overlooked.

If you plan ahead and address these basic considerations, the installation of your computer system will have a much greater probability of success. Your employees will be far more responsive to the new addition to the office, and they'll be more productive users' of the system that you spent so much money to get for them. After all, if your employees aren't using your computer effectively, does it matter how good the system itself is?

Heath/Zenith Users' Group



H89/Z90's CAN NOW DEAL WITH A FULL DECK!

THE ORIGINAL ALL-IN-ONE. ACCESSORY BUS EXPANDER.

MH89+3 doubles expansion capacity. Allow for 6 right-hand type cards instead of the usual 3. Room at last to run those neat accessory boards you've seen advertised!

Piggyback motherboard installs internally with a screwdriver in just minutes – with no modifications! 3 slots exactly duplicate the originals. The 3 added slots occupy unused addresses and eliminate previous conflicts. 100% compatible with all accessory boards!

No overheating problems! Simple design draws little power. Leaves plenty of overhead for the minimal load of most accessories. Full technical information provided.

The best news about this "No-hassle" design is the price — **ONLY $150.** About 1/3 the price of other solutions!

Price includes assembled and tested MH89+3 expander, complete instructions and one (1) year warranty. CA residents add 6% tax. USA include $5 shipping. Foreign add $10. Telephone and COD orders accepted.

mako data products
1441-B N. RED GUM, ANAHEIM, CA 92806
PHONE (714) 632-8583

PRIMERS FOR THE BEGINNER

GETTING STARTED WITH CP/M AND MBASIC
WITH PARTICULAR REFERENCE TO RANDOM FILES

Featuring a complete "menu driven" ready-to-run disk mail list program, program explanations, and complete tutorials, this package forms the perfect introduction to MBASIC programming under CP/M, and includes useful information for those new to the CP/M operating system (ZBASIC also supported). Included is a 56 page manual and a disk containing sample programs. Specify Heath/Zenith Computer model number, disk size and format—hard- or soft-sectored, single- or double-density, 5¼" or 8", and CP/M or ZDOS. $25.00

GETTING STARTED WITH MS-DOS AND BASIC
(AS ABOVE, EXCEPT NO CP/M)

For Z-150/160, MS-DOS and GW-BASIC. 65 page manual and disk included. $25.00

GETTING STARTED WITH HDOS & ASSEMBLY LANGUAGE PROGRAMMING

A 36 page tutorial covering aspects of Assembly Language programming under HDOS. Provides significant information for HDOS which other manuals lack. $15.00

PLEASE SEND CHECK OR MONEY ORDER TO:

WILLIAM N. CAMPBELL, M.D.
855 Smithbridge Road
Glen Mills, PA 19342
(215) 459-3218

USE ALL YOUR SPECIAL FUNCTION KEYS

With WORDSTAR™

WSKEY™: Now you can take the mystery out of WordStar with SKILL DATA's WordStar enhancement, which implements all twenty-one of the H/Z89-19 function/pad keys or all twenty three Z-100 labeled key commands.
Function key commands are labeled by a twenty-fifth line banner, which can be toggled on and off by you during your session.

With dBASE II™, ZIP™, SuperCalc™

DBKEY™, ZPKEY™, and SCKEY™: Just type your favorite SKILL DATA key command. All function key commands are labeled by a twenty-fifth line banner, which can be toggled on and off by you during your session. Pad keys also function and send multiple key inputs with a single stroke. All previous command key sequences are still available for the old and painful ways.

Just $29.95 for each program (plus shipping charges shown below).
H/Z89-19 diskettes are 5.25" 10HS, require CP/M 2.2.03.
Z-100 diskettes are 5.25" SS; specify CP/M-85 or Z-DOS.

Available in many Heathkit Stores and Quikdata Inc., Sheboygan, WI...or complete and mail this order form with your check or money order today!

Mail to SKILL DATA, P.O. Box 1943, Olympia, WA 98507.

Yes, I want to use all my special function keys! Send me:

☐ WSKEY—keys to WordStar: ☐ for H/Z89-19 ☐ for Z-100	$29.95
☐ SCKEY—Keys to SuperCalc (for CP/M only)	$29.95
☐ DBKEY and ZPKEY—keys to dBASE II: ☐ for H/Z89 ☐ for Z-100	$29.95

For Z-100 diskettes, specify operating system: ☐ CP/M-85 ☐ Z-DOS

Name ______________________
Street ______________________
City ______________ State ________
Zip ______________ Phone ______________
Visa/MC Card # ______________ Expires ________

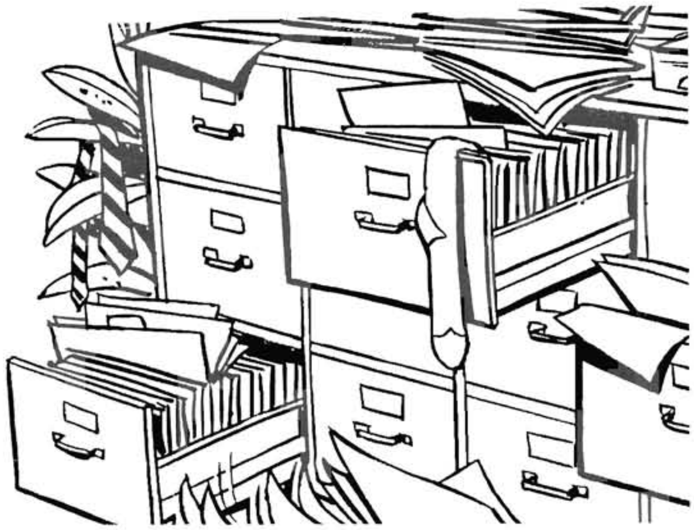
All orders shipped by first class mail. Include $3.00 per order for shipping and handling. $5.00 for non-U.S. orders. Payment by check, money order, or Visa/MasterCard. Allow 2-4 weeks for delivery. 206/352-0669 (evenings).

SKILL DATA, P.O. Box 1943, Olympia, WA 98507

The Latest Generation of Applications Development Software

Q-PRO 4

Dale Grundon
11456 Links Drive
Reston, VA 22090



It does not seem to be that long ago that we were loading Benton Harbor BASIC data base management programs from a cassette storage system. These programs were limited in the amount of data that could be manipulated and if you wanted to revise the software, it often would have been easier to rewrite the entire program. This part of our "ancient history" has moved a long way during the past five years.

Today, data base programming is considerably different. Software to deal with a data base can easily be revised to support changes to an existing data base and the tools available for creating a new set of data are numerous.

Q-PRO 4 falls into the category of the latest technique for programming control and manipulation of microcomputer data. Quik-n-easi Products, Inc. considers their system to fall into the "fourth generation" of applications development systems. It is an integrated system that allows you to handle the entire application with one package. All of the essentials are available to permit formatted data entry, data editing, data base management, and formatted output. Easily developed menu driven runtime programs can be oriented to the least experienced operator with the use of plain English error and help messages along with other simplified features.

The version of Q-PRO 4 that was provided for this review was in generic MS-DOS and I operated it on my Z-120 system. The program is also available for CP/M and an expanded version can be purchased for use with MP/M. A SETUP program is provided to configure the software for your particular terminal. A dual disk drive system with 64K is necessary to run any of the applications that you may develop.

With this package there are no restrictions on record size or the number of fields. A file could be up to eight megabytes in length and up to 255 files may be open at any one time. The files developed with this system can be set up to handle index sequential, random, and sequential. There are even provisions to handle sequential files generated by other data base systems. A data base set up with Q-Pro 4 could be designed in several ways such as hierarchal or relational.

Development of a data base system with Q-Pro 4 relies on a series of executable binary and command files. These files are generally interlinked to allow your operation to easily flow from one function to the next. The file identification generator, a run- time program, and a report generator. Each of these programs uses extensive menu selection to assist in the design of all of the criteria associated with the information system that you are designing.

Format Builder

The Format Builder file is the starting point for all applications development. It's provisions support construction of the format background or screen menus that will be invoked by the application, defining the fields within the data base, and an editor for writing the description of procedures and tables to support the programming portion of the application. Other features directly available from the Format Builder are the ability to merge previously written program segments direct to a new program you are writing, resequencing of the steps assigned to a program, listing the program to your printer, and chaining to the run-time program.

Using the background or screen portion of the builder is extremely easy. As you enter the building process you begin with a blank screen, and by using the cursor movement keys you may locate each item that is to appear on your final screen layout. There are limited graphics available consisting of either horizontal or vertical lines and the standard "T" shaped line graphics that are available with Heath-/Zenith terminals. Screen building uses the full features of the keyboard including most of the special function keys. After you have designed and saved a screen, it is very simple to recall it for modifications should you need to make any additions or deletions.

After the screen has been developed, which in most cases would be a display of the major fields for each record of your data base, each of the fields may be defined. When you enter the field definition phase, the screen that you developed for your data base will appear on your terminal. By placing the cursor at a location where the field data would be entered by a user, you will immediately assign that portion of the screen or menu to a field. At this point a selection menu will be provided to you from the File Builder. This menu lets you name the field, the length of the field, type of data to be entered, set-up for right or left justification, fill characters to use if the data that is later entered is shorter than the field length, and several other elements.

This definition phase is then continued for each of the fields that would exist for that data base. The combination of the screen building and file definition becomes a simple process. This technique definitely results in a well planned screen menu that will later make data entry by even a novice operator very easy.

The File Item Description

After a file has been built with the File Builder, the main menu lets you chain directly to a File Item Description Generator. This utility program will build and maintain descriptions of file records. Since this information is maintained separately in a different disk file, it provides a program-independent method to maintain data file formats.

References to file fields using the item description provide the advantage to make changes to a file format without regard to a particular program. This is because all references to a file field are symbolic or field name related.

It is through this item description that Q-Pro 4 gains it's ability to allow an unlimited number of fields within a record. Another key advantage is the ability to overlap data fields. This technique permits varying of a record format within a data file.

An example of this overlapping would be a data record that has common elements such as the name and address of a customer. When that customer has had one of your products shipped, the other data fields in the record could pertain to invoicing and shipping data. But once that customer had paid for the shipment, that invoicing data could be transposed by overlapping into payment related data, since the shipping and invoice information is no longer essential.

Procedures

After the file items have been described, it's back to the file builder. At this point of program design, you have finished much of what is usually the most tedious work that is associated with constructing a data management system. You are now ready to develop the program that will permit the actual entry and manipulation of the data.

At first glance, a review of the procedure statements for Q-Pro 4 appear to be similar to the system commands and the structure of many of the more recent and popular data base management programs. However, this system has several unique methods for handling procedures, as well as some handy built-in add-ons.

Basically, a Q-Pro 4 program is a collection or series of procedures. Each of these procedures is executed either as a procedure associated with a field, as a standard system procedure, or as a procedure called from another procedure.

To start off, all of the procedures are assigned a paragraph number. This technique permits calling of one procedure from within another simply by referring to the desired paragraph. By writing all of your procedures within the space of the 23 lines available on the screen, I found that it is much easier to maintain some mental concept of the overall program that is being developed because your are looking at it in blocks. The editor that is provided for entering the procedures automatically keeps track of the paragraph number assigned. It is quite simple to review all of the paragraphs by advancing through each one in sequence.

The command statements that are available include those for file handling, procedure control transfer, screen control and cursor movement, data movement and formatting, operating system interface, and extensive error handling. Statements dealing with arithmetic go beyond the routine to include a CALC command for performing calculations on data.

One particular handy command statement that is available will perform table searches. This table search capability looks similar to a two element matrix. It can be quite handy for validating data entry, could be used to set up field lengths within a record, or for defining default values. A good example of the table search application would be for a program menu option selection. The user enters a number associated with an option displayed on the screen. The procedure then would use the table search to first validate the numerical value entered and, if valid, use the second portion of the table element to jump to the selected procedure. A much simpler technique than the old . . "if user types a 1 do the first thing or if the user types a 2 do the second".

Q-Pro 4 has an excellent set of registers that can be used for temporary storage by the programmer. These registers consist of a string register that may contain up to 255 characters, a numeric register, and a boolean flag register. An error register is available and will return error codes that may be interpreted by the program. There is one other register that will return a numeric value relating to which key was used on the keyboard to give control to the program.

The Report Generator

Producing billing records, mail labels, inventory lists, and other reports is the main purpose of a having a data management system and the report generator provided with Q-Pro 4 is excellent. The best overall description of the Report Generator is that it can produce your report in any sequence and in any combination of the data fields in the file.

Some of the features and capabilities of the Report Generator include simultaneous output from up to six files, sorting on any field with either ascending or descending order, calculation of a print field, output to the disk or to the printer or to the screen, and the ability to update or delete records during print time.

Like the other main programs with Q-Pro 4, the Report Generator's major functions are selected from a master menu. Beyond the master menu, other menus are provided to help in the selection for development of the final report. The report technique that you develop may be saved for future recall, as well as directly printed as a hard copy.

One of the most difficult segments within most data management systems is setting up format for producing a report. With this system, you have a screen editor available, so that what you see is what you get. The complete report can be formatted right on the screen and even provides horizontal scroll of the information for reports that will be wider than the 80 characters available on the screen. In fact, a report can be formatted to a width of 300 columns. Each of the report headings can be set up and displayed and below the headings, the designated data field is indicated to the correct length you initially described for that field. Using this screen editor makes it very easy to directly visualize what the final report is going to look like and if you don't like some aspect of your initial set up, it's very easy to shift things around.

I don't know how many times I have been running a program that will work on information contained within another file and forgotten some particular aspect that I need to know before I can progress. One of the other option selections from the main menu is a listing of the fields that are within a data file. The list also includes the type of data that has been designated for the field along with the length. If you gave a field a descriptive name when you originally created the file, that description is also shown.

When running the Report Generator or most of the other programs within this system, many of those little 'gotchas' that occur in other software have been covered. For example, whenever you have made a change to something within the program you have developed and make your move to exit the Format Builder or Report Generator, you will be immediately asked by the system if you want to save the changes that you just made. If you had done an update on your own, the system knows that fact and will not bother you with the question. tion.

The Documentation

When you are dealing with a program with as many possibilities as there are available within a data management system, it takes quite a bit to explain how to use the system. Q-Pro 4 is no exception and the manual provided gives as many of the highlights and details as might be possible. However, my problem in learning the system from the manual was that I often wanted to shout - "Adam Green where are you?". The author of the manual managed to include all of the most essential information, but it is very easy to get lost in the process.

Starting with the self-teaching guide at the beginning will move you along toward learning to set up a data file, but it side steps its step-by-step- process at one point and I found it difficult to get back on track. At this point I realized that it would be better to take the book away from the computer and just read the manual.

This too has its problems. When you first pick up a book, it's good to first look at the contents and the index. Unfortunately, this manual is lacking in a good table of contents and the index leaves much to be desired. This detracts later on after you have read the book thoroughly and need to refer to a particular bit of information. Again, I believe the author has provided all of the significant information to use Q-Pro 4 to its fullest capability, but since the manual is more like a text book, it is difficult to look up information when you actually get to the point of needing it. There is a good chapter containing short explanations about the system commands, but the hints and suggestions to implement them are just too scattered around.

If you get involved with this software, my best suggestion is to make a good set of notes while you read the manual. After you have gone completely through the manual, then go back and run through the tutorial. Keep the supplied command reference card handy, as it will help divide the commands into a logical sequence as you need to look them up.

Supporting Elements

Unlike some of the other data applications software that is currently available, Q-PRO 4 has a wide variety of supporting programs either built-in or provided as a separate program. These include a SETUP program to configure the terminal features to operate with your system. You may select from a wide range of pre-configured terminal definitions that are already provided or if you want to redefine a particular key function, you can do that. Any configuration may be saved as a disk file for future recall.

The index files in Q-PRO 4 are based on a tree structure to produce file operations with minimal disk and memory overhead. It is possible that the manner that data is entered into the index files may eventually result in an internal tree structure that is not optimum. To overcome this situation, the author has provided a reorganization utility that will accept the index file as input and produce an optimized file as the output.

Other utilities are provided to verify that an index file is still good, to regenerate invalid file identification structure that result after major changes to a structure, to convert procedure files that you created with a text editor to Q-PRO 4 format, and a program to convert SDF files to indexed files.

Conclusions

If you are currently using one of the more recent data base management systems, I would not suggest that you immediately drop that system and convert over a massive amount of material to Q- PRO 4. The first thing that you are not going to like is that you cannot operate in a "command mode" and run a trial sequence of commands or procedures. Q-PRO 4 forces you to write your procedures and then run them as a sequence. This may not be the method that you have become used to using, but I must admit that it does make you plan your application in advance and get many of the bugs out before you actually do the first trial run.

If you are not currently using a data management system and are looking for a good method to start-up, then you certainly need to put Q-PRO 4 on your list of potential systems. It's format builder and report generator are excellent. They will both free your time for developing the procedures that you will need to operate on your data. Unfortunately, unlike some of the older application systems, there are not a lot of examples around yet to assist you in writing your applications. But with even a minimum knowledge of MBASIC or ZBASIC data file techniques or a book on data structures, you will easily get rolling into some handy Q-PRO 4 procedures.

Like any programming system, the more you use it the easier it will be to develop other applications. Q-Pro 4 works right with you in this aspect. With the onboard feature to directly make a hard copy of all of the programmed elements of your applications, documentation of your past applications becomes a pleasure. When you are working on a new application, both the printed documentation combined with the ability to merge those existing procedures all help to make your task quicker and easier.

Q-PRO 4 is available from:

Quik-n-easi Products Inc.
136 Granite Hill Court
Langhorne, PA 19047
(215) 968-5966

The ILLUSTRATOR

The **ILLUSTRATOR** is a full-featured graphics drawing program for use with the Z-100 pixel graphics (w/wo color) or the H89/H19 IMAGINATOR pixel graphics option. No need for a lightpen.

Features Include:

• Turtle Graphics	• Area Fill	• 64 Colors
• Rubber Banding	• Box Fill	• Color Define
• Line Drawing	• Circle Fill	• Color Pattern
• Box Drawing	• Diamond Fill	• Screen Save
• Circle Drawing	• Copy Area	• Screen Restore
• Ellipse Drawing	• Erase Area	• Area Save
• Diamond Drawing	• Text Mode	• Area Restore
• Screen Print	• Invert Mode	• Compacted Files
• Dot Cursor Control	• Help Display	• more. . .

ONLY $89.95!

HDOS version for H89, H8, requires IMAGINATOR
ZDOS version for Z-100, color memory optional
Supports many dot graphics printers. Call for info.

NEWLINE SOFTWARE
P.O. Box 402, Littleton, MA 01460 (617) 486-8535

NAME ______________________ CHECK ONE
STREET ______________________ ☐ H89, H8/H19, HDOS
CITY ______________________
STATE ______________ ZIP __________ ☐ Z-100, ZDOS
Send me ______ "The ILLUSTRATOR" program(s) at $89.95 each.
Check one: ☐ payment enclosed ☐ send COD (add $4.00)
Send order to:
NEWLINE SOFTWARE, P.O. BOX 402, LITTLETON, MA 01460
Foreign orders: add $3.00 Airmail, $10.00 for non-U.S. checks

HDOS is a trademark of Heath Company
ZDOS, Z-100 are trademarks of Zenith Data Systems, Inc.
IMAGINATOR is a trademark of Cleveland Codonics, Inc.

A Program To Transfer ASCII Files Into MultiPlan

Ken Simmons
Charles W. Williams, Inc.
801 North Pitt Stree, Suite 117-118
Alexandria, VA 22314

This article presents and explains a ZBASIC program to translate an ASCII sequential data file into a form suitable for input to MultiPlan. Such an ASCII file could come from Condor or some other data base or file manager, or it could come from a word processor like WordStar, or from a BASIC program, or most any other source.

MultiPlan has computation capability far beyond that of the typical file manager or data base manager. Unfortunately MultiPlan does not import an ASCII file created by other software. I have frequently wanted to transfer data from Condor files to MultiPlan for the more complex computations. When this issue first arose, I could find no convenient way to do this. I searched Zenith Z-100 documentation, called local "experts", and finally called Benton Harbor looking for software to translate a conventional sequential ASCII data file into the "SYLK" form required by the MultiPlan manual (see Appendix A3 of that manual). These searches were fruitless so I wrote the BASIC program presented here. See flow chart in Figure 1 and listing in Figure 2.

This article explains what the program does, how to use it, and why it works. It assumes that the reader has familiarity with use of MultiPlan and at least minimal competence in the use of ZDOS and other software. In program design, simplicity and ease of use are intended to be in balance with versatility.

What the program does:

It takes in a sequential ASCII data file and converts it into a form suitable for importing into MultiPlan. This form is the "Symbolic Link" (SYLK) form described in Appendix A3 of the MultiPlan manual.

How to use the program:

- Create a sequential ASCII data file from a file manager or data base manager (I use a Condor relational data base manager).
- Place MPSYLK.BAS, shown in Figure 1, on the same disk as your ASCII data file.
- Run MPSYLK.BAS with your ASCII data file as input, producing an output "SYLK" file.
- Load MultiPlan and shift into the "Symbolic" mode.
- "Transfer Load" your new "SYLK" file into MultiPlan.

You now have your ASCII file loaded into MultiPlan; insert headings and perform calculations as needed. Suggestion: shift out of "Symbolic" mode and back into "Normal" mode immediately upon loading your new SYLK file into MultiPlan. This will insure saving your new file in the normal form.

Detailed Steps in Using the Program

Figures 3 through 11 depict the flow of events in using the program. Figures 3-6 and 8-11 are "Shift F12" screen dumps on the Z-100. Figure 7 derives from the TYPE command. These figures are described as follows:

The distribution package of Condor relational data base manager software contains three disks. Disk #3 contains example programs, including a "PRODUCTS" file. Figures 3 and 4 represent a modified version of this "PRODUCTS" file. Figure 3 is the format of the file, and Figure 4 depicts all the data in the file.

Assume that the business operator using this file desires to produce a "What-if" sequence of dollar sales in the year 1990, based upon different assumptions as to growth rates and prices. Condor is not well suited to rapid calculation of a variety of possibilities, but MultiPlan is. So the business operator may want to transfer into MultiPlan only those fields necessary for product identification and for calculation. Thus Project No., Model No., and Market Type would not be required. Dollar sales would not be necessary, but since it is already present in this example we will go ahead and transfer it. Thus on the basis of this rationale we decide to transfer Product No., Description, Unit Price, Unit Sales, and Dollar Sales from our Condor data base into MultiPlan.

Figure 5 depicts the next two steps. The "PROJECT" Condor command creates a RESULT file composed of just those fields named in the command, in this case the Product No., Description, Unit Price, Unit Sales, and Dollar Sales. The Condor "WRITE" command translates this RESULT file into ASCII format and writes it to the disk under the name "PRODFILE". At the lower portion of the figure is the file typed out to show its contents.

Figure 6 shows the screen content upon running MPSYLK.BAS. This figure also shows the entries into MPSYLK.BAS used in this example. The first four entries are required. We are translating PRODFILE into the "SYLK" format so PRODFILE is the name of the input, inserted as the first required entry; we assign the name PRODMP to the output SYLK file, assigned as the second required entry; and the file contains 2 alphanumeric and 3 numeric fields, inserted as the third and fourth required entries. Although it is not necessary to enter additional values in response to the prompts, we have done so in our example in order to show the capability.

Figure 7 shows the completed "SYLK" file. The meaning of the symbols is described in some detail in the next section of this article.

Figure 8 shows the MultiPlan screen for selection of the "Symbolic" mode of transfer. Upon completion of running MPSYLK.BAS, a MultiPlan disk was inserted into drive A, and the new PRODMP file placed on the disk in drive B. MultiPlan is set to the "Symbolic" mode and "B:PRODMP" Transfer-Loaded into memory as shown in Figure 9.

Figure 10 shows the MultiPlan screen upon successful loading of the new file. Note the correspondence between it and the data in Figures 5, 6, and 7. Figure 11 shows an unnecessary step, but one which finally places the new MultiPlan file into the normal format for MultiPlan storage.

Detailed Explanation of the SYLK Coding System

Shown to the right is a blow-up of the first portion of Figure 7, the "SYLK" file created by the program for entry into MultiPlan. An explanation of the coding follows it.

Additional Program Characteristics

1. You may have as many columns of alphanumeric data as you wish, but they must all appear as the leftmost columns in your spreadsheet.

2. You may also have as many numeric columns as you wish; these will appear to the right of the alphanumeric data columns.

3. The ASCII file is to contain data items only; no file title and no column headings; these will be entered by the operator when MPSYLK.BAS is run.

4. The width of column 2 is adjustable to permit entry of complete descriptions of data items.

5. When you run the MPSYLK.BAS program you will find the first three prompted inputs require you to make specific entries; the remaining prompts permit default entries by simply striking RETURN.

Suggested Procedure for use with Condor:

• PROJECT selected fields of Condor file into file to be read into MULTIPLAN. See Figure 5.

• WRITE RESULT <descriptive name of your new ASCII file>[B]The name PRODFILE is used in this article as the descriptive name of the ASCII file to be read into MultiPlan. See Figure 5.

• Insert your MultiPlan working disk into the A drive in place of your Condor disk.

• Insure that ZBASIC.COM and MPSYLK.BAS are on either your MultiPlan disk or on your PRODFILE disk, then enter:

```
ZBASIC MPSYLK
```

to load and run MPSYLK.BAS.

```
ID;PMP                              |  The ID symbol indicates the first
                                    |record in the SYLK file and identi-
                                    |fies the file as a SYLK file.

F;W2 2 25                           |  The "F" appearing as the initial
                                    |symbol is equivalent to selecting
                                    |"Format" in the MultiPlan menu.
                                    |";W" selects the "Width" option,
                                    |the first "2" says to start with
                                    |column 2, the second "2" says end
                                    |with column 2, and the 25 says to
                                    |set width as 25 characters.

F;DF2G10                            |  "F" selects "Format" as before.
                                    |";D" signifies selection of the
                                    |"Default" option. "F2" selects the
                                    |Format Code "Fix" option with
                                    | "# of decimals:" as 2.
                                    |"G" selects "General" as the Align-
                                    |ment Code option.
                                    |"10" is then the default cell width.

B;Y 8 X 5                           |  "B" signifies that the active
                                    |spreadsheet is bounded by the cell
                                    |identified following the ";"--in
                                    |this case Y8 (row 8) and X5 (col 5).

C;Y1;X1;K"PROJECTION ....YEAR 1990" |  "C" indicates that the cell indi-
                                    |cated following ";" (in this case
                                    |Y1--row 1, and X1--col 1) is des-
                                    |cribed by what follows.  The ";K"
                                    |indicates that what follows is the
                                    |value to be inserted into the cell.

F;FC0D                              |  Initial "F" is Format as before.
                                    |";F" indicates selection of the
                                    |"Cell" option. "C" indicates selec-
                                    |tion of Format Code "Continuous".
                                    |the "0" has no meaning since "Fixed"
                                    |was not selected, and "D" indicates
                                    |selection of the "Default" Alignment
                                    |code.

F;X 2 ;FC0D                         |  Format as before. Row is previously
                                    |selected row (1 in this case) by de-
                                    |fault. "X 2" indicates column 2; ";F"
                                    |again indicates "Cell" selection,
                                    |"C" is Format Code "Continuous", and
                                    |"D"  is again "Default"  Alignment
F;X 3 ;FC0D                         |code.  The next four lines of code
F;X 4 ;FC0D                         |simply extend the "Continuous" selec-
F;X 5 ;FC0D                         |tion into the columns indicated --
F;X 6 ;FC0D                         |columns 3 through 6.

C;Y2;X 1 ;K"ID NO."                 |  As before, this specifies that the
C;Y2;X 2 ;K"PRODUCT DESCRIPTION"    |alphanumeric value "ID NO." be in-
C;X 3 ;K"UN.PRICE"                  |serted into cell at row 2 col 1,
F;FD0C                              |"PRODUCT DESCRIPTION" into cell at
C;X 4 ;K"UN.SALES"                  |row 2 col 2, etc. Alphanumeric
F;FD0C                              |values are enclosed in double quotes,
C;X 5 ;K"DOL.SALES"                 |and numeric values are not.
F;FD0C                              |

C;Y 3 ;X 1 ;K"103"                  |  The ID number of the first item,
                                    |#103, is entered in alphanumeric
                                    |form into row 3 col 1.
                                    |
C;Y 3 ;X 2 ;K"15# greenbar paper"   |  The descriptor of item #103 is
                                    |entered into row 3 col 2.

C;X 3 ;Y 3 ;K 48                    |  The numeric values for item #103
C;X 4 ;Y 3 ;K 59                    |are entered into row 3 columns
C;X 5 ;Y 3 ;K 2832                  |3, 4, and 5.

                                    |  Remaining entries follow this same
                                    |pattern of data entry.

E                                   |  End of SYLK file.
```

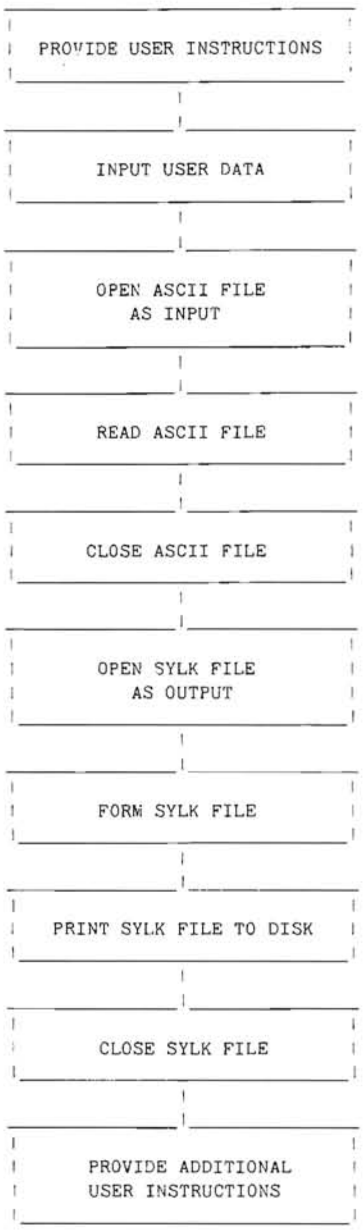


Figure 1

```
10 REM PROGRAM TO TRANSER ASCI FILES INTO MULTIPLAN: "MPSYLK" ON DIRECTORY
20 DIM A$(50,50), X(50,50)
30 CLS
40 PRINT "****************************************************************"
50 PRINT "*                                                              *"
60 PRINT "*                     MULTIPLAN SYLK                           *"
70 PRINT "*                       CONVERSION                             *"
80 PRINT "*                ( MPSYLK.BAS ON DIRECTORY )                   *"
90 PRINT "*                                                              *"
100 PRINT "****************************************************************"
110 PRINT
120 PRINT "THIS PROGRAM TRANSFORMS AN ASCII FILE FROM CONDOR, WORDSTAR, OR"
130 PRINT "OTHER SOURCE INTO A FILE IMPORTABLE INTO MULTIPLAN (MP)."
140 PRINT
150 PRINT "PRESS RETURN TO CONTINUE."
160 A$=INKEY$:IF A$="" THEN 160
170 CLS
180 PRINT "EACH FIELD OF AN ASCII RECORD BECOMES A COLUMN IN MP."
190 PRINT "YOU WILL ENTER THE NUMBER OF FIELDS (MULTIPLAN COLUMNS) YOU WISH"
200 PRINT "TO BE FORMATTED FOR ALPHANUMERIC ENTRIES AND THE NUMBER FOR "
210 PRINT "NUMERIC ENTRIES.  THE SPREADSHEET COLUMNS, FROM LEFT TO RIGHT,"
220 PRINT "WILL BEGIN WITH THE ALPHANUMERIC COLUMNS.  KEEP IN MIND THAT"
230 PRINT "ALPHANUMERIC DATA WILL APPEAR IN THE FIRST COLUMNS, STARTING"
240 PRINT "AT THE LEFT IN COLUMN 1.  THERE MAY BE AS MANY ADDITIONAL"
250 PRINT "NUMERIC FIELDS AS YOU WISH."
260 PRINT "CAUTION: IF YOU ATTEMPT TO ENTER A VALUE FROM A CONDOR"
270 PRINT "ALPHA, ALPHANUMERIC, OR JULIAN FIELD INTO ANY MP COLUMN WHICH"
280 PRINT "YOU'VE LABELLED NUMERIC, IT WILL APPEAR AS '0' ON YOUR SHEET."
290 PRINT
300 PRINT "PRESS RETURN TO CONTINUE"
310 A$=INKEY$:IF A$="" THEN 310
320 CLS
330 PRINT "ENTER DATA IN RESPONSE TO PROMPTS WHICH FOLLOW.  THE FIRST FOUR"
340 PRINT "PROMPTS REQUIRE ENTRIES -- THE NAME OF YOUR ASCII FILE, THE "
350 PRINT "NAME OF YOUR NEW MP FILE, THE NUMBER OF ALPHANUMERIC DATA"
360 PRINT "COLUMNS, AND THE NUMBER OF NUMERIC DATA COLUMNS IN YOUR ASCII FILE."
370 PRINT "THE REMAINDER OF THE PROMPTS MAY BE ANSWERED SIMPLY BE STRIKING"
380 PRINT "RETURN.  DEFAULT VALUES ARE INDICATED WITH EACH OF THESE PROMPTS."
390 PRINT
400 PRINT "PRESS RETURN TO CONTINUE"
410 A$=INKEY$:IF A$="" THEN 410
420 CLS
430 PRINT "WHEN YOU HAVE COMPLETED YOUR DATA ENTRY AND THE PROGRAM HAS RUN,"
440 PRINT "YOU WILL BE ADVISED ON HOW TO LOAD YOUR FILE INTO MULTIPLAN."
450 PRINT
460 PRINT "PRESS RETURN TO CONTINUE"
470 A$=INKEY$:IF A$="" THEN 470
480 CLS
490 REM
500 REM ********************* DATA INPUT SECTION OF PROGRAM ****************
510 REM
520 PRINT "PLEASE ENTER THE NAME OF THE ASCII FILE TO BE IMPORTED."
530 INPUT "(RESPONSE REQUIRED)";F$
540 PRINT "ENTER NAME YOU DESIRE FOR NEW MULTIPLAN FILE."
550 INPUT "(RESPONSE REQUIRED)";M$
560 INPUT "NO. OF ALPHANUMERIC FIELDS. (RESPONSE REQUIRED)";L
570 INPUT "NO. OF NUMERICAL FIELDS. (RESPONSE REQUIRED)";N
580 PRINT "ENTER TITLE YOU WISH TO APPEAR ON FIRST LINE OF SPREADSHEET:"
590 INPUT "(DEFAULT IS BLANK)";T$
600 INPUT "NO. CELLS REQ'D FOR SPREADSHEET TITLE. (DEFAULT =1)";K
610 PRINT "DESIRED WIDTH, SECOND SPREADSHEET COLUMN (DESCRIPTION COLUMN)."
620 INPUT "(DEFAULT IS 10)";W
630 IF L=0 THEN 700
640 FOR I=1 TO L
650     PRINT "LABEL FOR HEADING OF COLUMN";I
660     INPUT "(DEFAULT IS BLANK)";S$(I)
670 NEXT I
680 IF N=0 THEN 740
690 PRINT "RECALL THAT COLUMNS STARTING HERE ARE FOR NUMERIC DATA"
700 FOR I=1 TO N
710   PRINT "ENTER HEADING FOR COLUMN";(I+L)
720   INPUT "(DEFAULT = BLANK)";C$(I)
730 NEXT I
740 REM
750 REM ********** DATA INPUT COMPLETE; BEGIN INPUT OF ASCII FILE ***********
760 REM                          'THIS SECTION READS THE ASCII FILE INTO
770 OPEN "I",#1,F$               '  MEMORY USING STANDARD PROCEDURES FOR
780 FOR I=1 TO 500               '  HANDLING SEQUENTIAL FILES.
```

```
790     FOR J=1 TO L
800     IF EOF(1) THEN 890          'THE END OF FILE BRANCHING IS DONE IN
810     INPUT #1, A$(I,J)           '  THE J LOOP MERELY AS INSURANCE IN CASE
820     NEXT J                      '  A FILE ENDS IN THE MIDDLE OF A LINE.
830     FOR J=1 TO N
840        IF EOF(1) THEN 890
850        INPUT #1,X(I,J)
860     NEXT J
870     M=I                         'THE M VALUE IS THE NUMBER OF RECORDS IN
880 NEXT I                          '  THE FILE AND IS USED IN THE
890 CLOSE #1                        '  SECTION BELOW.
900 REM
910 REM ******* INPUT OF ASCII FILE COMPLETED; BEGIN FORMING SYLK FILE ********
920 REM
930 OPEN "O",#2,M$                       'OPEN, AS OUTPUT, THE FILE NAMED ABOVE
940 PRINT #2, "ID;PMP"                   'REQUIRED STARTING ITEM FOR SYLK FILE
950 PRINT #2, "F;W2 2";W                 'WIDTH OF COLUMN 2 IS 'W' - FROM L620
960 PRINT #2, "F;DG0G10"                 'DEFAULT FORMAT GENERAL; DEF. WIDTH 10
970 PRINT #2, "B;Y";(M+3);"X";(N+2)      'SHEET EXTENT: ROW (M+3) COLUMN (N+2)
980 PRINT #2,"C;Y1;X1;K";                'ENTRY FOR ROW 1 (Y1) COLUMN 1 (X1)
990 WRITE #2,T$                          '  IS T$  — INSERTED IN L590 ABOVE.
1000 PRINT #2,"F;FC0D"                   'FORMAT CONTINUOUS THROUGH CELLS
1010 FOR I=2 TO (K+1)                    '  INDICATED BY I LOOP.
1020   PRINT #2,"F;X";I;";FC0D"
1030 NEXT I
1040 FOR I=1 TO L                        'ENTRIES FOR ROW 2 COLUMNS 1 THRU L
1050    PRINT #2,"C;Y2;X";I;";K";        '  ARE THE S$(I)
1060    WRITE #2,S$(I)                   'WRITE VS PRINT IOT GET DOUBLE QUOTES
1070 NEXT I                              '  AS REQUIRED BY SYLK FORMAT
1080 FOR I=1 TO N                        'LOOP TO PRINT NUMERICAL COLUMN HEAD-
1090   PRINT #2,"C;X";(I+L);";K";        '  INGS - COLUMNS (L+1) TO (L+N)
1100   WRITE #2,C$(I)
1110 NEXT I
1120 FOR I=1 TO M                                   'LOOP TO PRINT ALPHANUMERIC
1130   FOR J=1 TO L                                 '  DATA ENTRIES IN FIRST
1140      PRINT #2,"C;Y";(I+2);";X";J;";K";         '  L COLUMNS
1150      WRITE #2,A$(I,J)
1160   NEXT J
1170   FOR J=1 TO N                                      'LOOP TO PRINT NUMERIC
1180     PRINT #2,"C;X";(J+L);";Y";(I+2);";K";X(I,J)     '  DATA ENTRIES IN
1190   NEXT J                                            '  REMAINING COLUMNS
1200 NEXT I
1210 CLOSE #2
1220 REM ****** SYLK FILE COMPLETE; PROVIDE FURTHER USER INSTRUCTIONS *******
1230 CLS
1240 PRINT "NOW TAKE THE FOLLOWING STEPS:"
1250 PRINT "  o  WHEN YOU HAVE READ THESE STEPS STRIKE 'RETURN';"
1260 PRINT "  o  THEN LOG ON TO YOUR MULTIPLAN DISK DRIVE AND LOAD MULTIPLAN;"
1270 PRINT "  o  PRESS 'TOS' TO SET UP THE 'SYMBOLIC' MODE IN MULTIPLAN"
1280 PRINT "       (SEE MULTIPLAN MANUAL, VOL II, APPENDIX A3)"
1290 PRINT "  o  LOAD YOUR NEW MULTIPLAN FILE"
1300 PRINT "  o  REMINDER: SHIFT MULTIPLAN TO 'NORMAL' MODE TO SAVE"
1310 PRINT "        YOUR NEW FILE"
1320 A$=INKEY$:IF A$="" THEN 1320
1330 SYSTEM
```

Figure 2

```
                   PRODUCTS

[PRODUCT.NO.]: ___

[DESCRIPTION]: ____________________

[PROJECT.NO]: __ [MODEL.NO]:_______ [MARKET.TYPE]:___

[UNIT.PRICE]: __________

[UNIT.SALES]: __________

[DOLLAR.SALES]: __________



Rep Mode: ins Mode (Ctl A), Abort (Ctl C), End (Ctl E, Refresh Screen (Ctl R)
```

Figure 3

```
B>>PRINT PRODUCTS BY PRODUCT.NO. DESCRIPTION PROJECT.NO MODEL.NO MARKET.TYPE UNI
T.PRICE UNIT.SALES DOLLAR.SALES

Database: PRODUCTS                 5 Records


Printing
Done.

PRODUCT.NO. DESCRIPTION         PROJECT.NO MODEL.NO MARKET.TYPE UNIT.PRICE UNIT.SALES DOLLAR.SALES

103         15# greenbar paper  A1         123      B                48.00         59     2832.00
111         18# greenbar paper  A2         124      B                44.00         45     1980.00
117         15# white paper     A2         123      B                48.00        105     5040.00
209         18# white paper     A2         124      C                44.00         47     2068.00
112         Printout Mailer     A1         123      B                24.00        175     4200.00
```

Figure 4

```
B>>PROJECT PRODUCTS BY PRODUCT.NO. DESCRIPTION UNIT.PRICE UNIT.SALES DOLLAR.SALE
S

Database: PRODUCTS                 5 Records

Busy

Total Records in Result Set =5

B>>WRITE RESULT PRODFILE [B]

Database: RESULT                   5 Records

Busy
Total Records in output file = 5
B>>


TYPE PRODFILE
"103","15# greenbar paper",48.00,59,2832.00
"111","18# greenbar paper",44.00,45,1980.00
"117","15# white paper",48.00,105,5040.00
"209","18# white paper",44.00,47,2068.00
"112","Printout Mailer",24.00,175,4200.00
```

Figure 5

```
ENTER DATA IN RESPONSE TO PROMPTS WHICH FOLLOW.  THE FIRST FOUR
PROMPTS REQUIRE ENTRIES -- THE NAME OF YOUR ASCII FILE, THE
NAME OF YOUR NEW MP FILE, THE NUMBER OF ALPHANUMERIC DATA
COLUMNS, AND THE NUMBER OF NUMERIC DATA COLUMNS IN YOUR ASCII FILE.
THE REMAINDER OF THE PROMPTS MAY BE ANSWERED SIMPLY BE STRIKING
RETURN.  DEFAULT VALUES ARE INDICATED WITH EACH OF THESE PROMPTS.

PRESS RETURN TO CONTINUE
```

```
EACH FIELD OF AN ASCII RECORD BECOMES A COLUMN IN MP.
YOU WILL ENTER THE NUMBER OF FIELDS (MULTIPLAN COLUMNS) YOU WISH
TO BE FORMATTED FOR ALPHANUMERIC ENTRIES AND THE NUMBER FOR
NUMERIC ENTRIES.  THE SPREADSHEET COLUMNS, FROM LEFT TO RIGHT,
WILL BEGIN WITH THE ALPHANUMERIC COLUMNS.  KEEP IN MIND THAT
ALPHANUMERIC DATA WILL APPEAR IN THE FIRST COLUMNS, STARTING
AT THE LEFT IN COLUMN 1.  THERE MAY BE AS MANY ADDITIONAL
NUMERIC FIELDS AS YOU WISH.
aution; if you attempt to enter a value from a condor
lpha, ALPHANUMERIC, OR JULIAN FIELD INTO ANY MP COLUMN WHICH
YOU'VE LABELLED NUMERIC, IT WILL APPEAR AS '0' ON YOUR SHEET.
PRESS RETURN TO CONTINUE


TYPE PRODMP
ID;PMP
F;W2 2 25
F;DGOG10
B;Y 8 X 5
C;Y1;X1;K"PROJECTION OF PRODUCT SALES TO THE YEAR 1990"
F;FCOD
F;X 2 ;FCOD
F;X 3 ;FCOD
F;X 4 ;FCOD
F;X 5 ;FCOD
F;X 6 ;FCOD
C;Y2;X 1 ;K"ID NO."
C;Y2;X 2 ;K"PRODUCT DESCRIPTION"
C;X 3 ;K"UN.PRICE"
C;X 4 ;K"UN.SALES"
C;X 5 ;K"DOL.SALES"
C;Y 3 ;X 1 ;K"103"
C;Y 3 ;X 2 ;K"15# greenbar paper"
C;X 3 ;Y 3 ;K 48
C;X 4 ;Y 3 ;K 59
C;X 5 ;Y 3 ;K 2832
C;Y 4 ;X 1 ;K"111"
C;Y 4 ;X 2 ;K"18# greenbar paper"
```

```
WHEN YOU HAVE COMPLETED YOUR DATA ENTRY AND THE PROGRAM HAS RUN.
YOU WILL BE ADVISED ON HOW TO LOAD YOUR FILE INTO MULTIPLAN.

PRESS RETURN TO CONTINUE




PLEASE ENTER THE NAME OF THE ASCII FILE TO BE IMPORTED.
(RESPONSE REQUIRED)? PRODFILE
ENTER NAME YOU DESIRE FOR NEW MULTIPLAN FILE.
(RESPONSE REQUIRED)? PRODMP
NO. OF ALPHANUMERIC FIELDS. (RESPONSE REQUIRED)? 2
NO. OF NUMERICAL FIELDS. (RESPONSE REQUIRED)? 3
ENTER TITLE YOU WISH TO APPEAR ON FIRST LINE OF SPREADSHEET:
(DEFAULT IS BLANK)? PROJECTION OF PRODUCT SALES TO THE YEAR 1990
NO. CELLS REQ'D FOR SPREADSHEET TITLE. (DEFAULT =1)? 5
DESIRED WIDTH, SECOND SPREADSHEET COLUMN (DESCRIPTION COLUMN).
(DEFAULT IS 10)? 25
LABEL FOR HEADING OF COLUMN 1
(DEFAULT IS BLANK)? ID NO.
LABEL FOR HEADING OF COLUMN 2
(DEFAULT IS BLANK)? PRODUCT DESCRIPTION
RECALL THAT COLUMNS STARTING HERE ARE FOR NUMERIC DATA
ENTER HEADING FOR COLUMN 3
(DEFAULT = BLANK)? UN.PRICE
ENTER HEADING FOR COLUMN 4
(DEFAULT = BLANK)? UN.SALES
ENTER HEADING FOR COLUMN 5
(DEFAULT = BLANK)? DOL.SALES


NOW TAKE THE FOLLOWING STEPS:
  o  WHEN YOU HAVE READ THESE STEPS STRIKE 'RETURN';
  o  THEN LOG ONTO YOUR MULTIPLAN DISK DRIVE AND LOAD MULTIPLAN;
  o  PRESS 'TOS' TO SET UP THE 'SYMBOLIC' MODE IN MULTIPLAN
       (SEE MULTIPLAN MANUAL, VOL II, APPENDIX A3)
  o  LOAD YOUR NEW MULTIPLAN FILE
  o  REMINDER: SHIP MULTIPLAN TO 'NORMAL' MODE TO SAVE
       YOUR NEW FILE
```

Figure 6

```
*************************************************************
*                                                           *
*                     MULTIPLAN SYLK                        *
*                       CONVERSION                          *
*              ( MPSYLK.BAS ON DIRECTORY )                  *
*                                                           *
*************************************************************

THIS PROGRAM TRANSFORMS AN ASCII FILE FROM CONDOR, WORDSTAR, OR
OTHER SOURCE INTO A FILE IMPORTABLE INTO MULTIPLAN (MP).

PRESS RETURN TO CONTINUE.
```

```
C;X 3 ;Y 4 ;K 44
C;X 4 ;Y 4 ;K 45
C;X 5 ;Y 4 ;K 1980
C;Y 5 ;X 1 ;K"117"
C;Y 5 ;X 2 ;K"15# white paper"
C;X 3 ;Y 5 ;K 48
C;X 4 ;Y 5 ;K 105
C;X 5 ;Y 5 ;K 5040
C;Y 6 ;X 1 ;K"209"
C;Y 6 ;X 2 ;K"18# white paper"
C;X 3 ;Y 6 ;K 44
C;X 4 ;Y 6 ;K 47
C;X 5 ;Y 6 ;K 2068
C;Y 7 ;X 1 ;K"112"
C;Y 7 ;X 2 ;K"Printout Mailer"
C;X 3 ;Y 7 ;K 24
C;X 4 ;Y 7 ;K 175
C;X 5 ;Y 7 ;K 4200

B:
```

Figure 7

```
#1          1          2          3          4          5          6
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15
 16
 17
 18
 19
TRANSFER OPTIONS mode: Normal Symbolic Other    setup:

Select option
R1C1                                 100% Free        Multiplan: TEMP
```

Figure 8

Figure 9

```
#1          1          2          3          4          5          6
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
TRANSFER LOAD filename: B:PRODMP

Enter a filename, or use direction keys to view directory
R1C1                               100% Free        Multiplan: TEMP
```

Figure 10

```
#1      1              2              3        4        5          6
 1 PROJECTION OF PRODUCT SALES TO THE YEAR 1990
 2 ID NO.     PRODUCT DESCRIPTION     UN.PRICE  UN.SALES DOL.SALES
 3 103        15# greenbar paper            48        59      2832
 4 111        18# greenbar paper            44        45      1980
 5 117        15# white paper               48       105      5040
 6 209        18# white paper               44        47      2068
 7 112        Printout Mailer               24       175      4200
 8
 9
10
11
12
13
14
15
16
17
18
19
COMMAND: Alpha Blank Copy Delete Edit Format Goto Help Insert Lock Move
         Name Options Print Quit Sort Transfer Value Window Xternal
Select option or type command letter
R1C1     "PROJECTION OF PRODUCT SALES   99% Free        Multiplan: PRODMP
```

Figure 11

```
#1      1              2              3        4        5          6
 1 PROJECTION OF PRODUCT SALES TO THE YEAR 1990
 2 ID NO.     PRODUCT DESCRIPTION     UN.PRICE  UN.SALES DOL.SALES
 3 103        15# greenbar paper            48        59      2832
 4 111        18# greenbar paper            44        45      1980
 5 117        15# white paper               48       105      5040
 6 209        18# white paper               44        47      2068
 7 112        Printout Mailer               24       175      4200
 8
 9
10
11
12
13
14
15
16
17
18
19
TRANSFER SAVE filename: B:READYMP

Enter a filename
R1C1     "PROJECTION OF PRODUCT SALES   99% Free        Multiplan: PRODMP
```

GREAT BUYS ON FINE ZENITH SOFTWARE!!

...FOR COMMUNICATIONS

Buy COM-PAK! COM-PAK is our two most popular communication products, **ZED-COM** and **REM-CON,** packaged together with complete documentation in an attractive hard cover linen binder.

With **ZED-COMM,** using your Z-100 to communicate with bulletin boards, time-sharing services, and other micros has never been so easy. With single keystroke commands, help screens, a wide range of set options, variable baud rates and a host of other valuable functions, **ZED-COMM** just can't be beat.

Wish you could use your Z-100 from across town? With **REM-CON** you can do it! REM-COM is just that, a REMote CONsole. Using any remote dumb ASCII terminal attached to your Z-100 via telephone lines and a smart modem, REM-CON disables your main console and allows you to run almost any program, just as if you were sitting in front of your machine. It's great!

COM-PAK is normally a $94.90 value.
(Available for the Z-100 under ZDOS.) ──────► **NOW ONLY $69.95!**

...FOR EDUCATION

Buy EDU-PAK! Learn how to program the quick-and-easy way be using our Video Professor self-tutorial computer-based instruction courses, the courses that use your computer to do the teaching, not a textbook! EDU-PAK includes our 3 most popular Video*Professor courses, Assembler I, MBasic (ZBasic) I, and Structured Programming Techniques. All Video*Professor programs include at least 10 thorough lessons with self-check quizzes.

ASSEMBLER I teaches you the basics of micro-assembler programming using the 8080 instruction set (8088 under ZDOS).

MBASIC (ZBASIC) I gives you a solid foundation for writing quality programs in the Basic language.

STRUCTURED PROGRAMMING TECHNIQUES covers the principles of good programming which makes programs easy to follow, enhance, and de-bug and which conserve machine resources (CPU, memory, and disk).

EDU-PAK is normally a $119.85 value ──────► **NOW ONLY $72.00!**
(Available for the H89 under HDOS and CP/M-80 and for the Z100 under ZDOS and CP/M-85. Ask about our Video*Professor courses on other popular subjects!)

...FOR BUILDING YOUR PROGRAM LIBRARY

Buy ZMAG! **ZMAG** is like a magazine because it is published 8 times a year and has interesting articles, tutorials, reviews, editorials, and advertisements.

...But unlike a magazine, with each issue of **ZMAG** you receive a diskette with 10-15 good programs on it! ZMAG is a great way to build your program library because by the end of a subscription year you will have over 100 programs covering a wide variety of applications and written in several different languages.

That's not all! ZMAG subscribers receive 8 free consulting calls to us on any computing subject, a 10% discount on Husker name brand software, and special offers not available to other customers. That's value added!

Add up what you get for a ZMAG subscription and you'll find it is the best offer available anywhere!

──────► **OUR CONTINUED LOW PRICE IS ONLY $147.00!**

...and it is payable in two installments if you wish! Pay $75 now and we'll bill the remaining $72 in 60 days!
(Available for the H89 under CP/M-80 and for the Z-100 under ZDOS)

Fine Products From:

HUSKER
Systems of Nebraska

A Division Of NewAmerica Technologies Corporation

Current Date ____________ Payment Method: Cash_____ VISA _____ MasterCard _____

Card Number ____________ Expires_______ ZMAG Easy Pay?______ Send Brochures______

QUANTITY	ITEM (Specify Oper. System)	UNIT PRICE	TOTAL
*In Nebraska Add 5 % Sales Tax.		TOTAL*	

NOTE:
1) Please specify operating system.
2) H89 diskettes are hard sector. Z100 diskettes are soft sector.

To get our 100+ page master catalog of discount hardware and software for only $2.00 check here. □

ORDER TODAY!
Send Orders To:
Husker Systems of Nebraska
6657 Redick Avenue
Omaha, NE 68152
Ph. (402) 572-6290
Or Call Toll-Free:
(800) 835-7427 extension 277

Name/Company ____________ *Address* ____________

Telephone ____________ *City* ____________ *State* ______ *Zip* ______

Paying the GOTO Ransom

Dennis C. White
P. O. Box 81108
Davis-Monthan AFB, AZ 85707

One of these days the Constitution of the United States will guarantee everyone wanting to write structured BASIC code the right to do so. But until that day, those of us possessing a BASIC which does not support structured constructs will have to improvise.

The first microcomputer BASIC I had access to was Benton Harbor BASIC. It's not a bad BASIC implementation, but it does not contain structured constructs like WHILE/WEND, REPEAT/UNTIL, and DO/CASE. What it does contain is GOTO.

After all the eloquent treatises on their uses and abuses, after all the heated arguments over their appropriate or inappropriate place in the computer society, let me say one thing. I don't like GOTOs, and I won't use GOTOs.

If you feel like me, but also feel your BASIC is keeping you hostage with a loaded GOTO to your head, read on. Together we might be able to come up with the ransom to set you free.

The September 1981 issue of BYTE contains a Programming Quickie which shows a REPEAT/UNTIL structure made from a FOR/NEXT loop. Using that as a basis, one can construct two other structured programming constructs which make GOTOs obsolete.

The code which I will use appeared in BYTE like this:

```
10 FOR D=0 TO 1
20 loop-body
30 D = condition
40 NEXT D
```

This construct behaves as a REPEAT/UNTIL loop. The routine in lines 10 through 40 will REPEAT UNTIL the variable "D" evaluates to the value of 1. Line 30 must be written so the variable "D" evaluates to 1 when the programmer wants the loop to terminate. To do that, a programmer may directly assign the variable "D" the value of 1 any time he or she wants. There is a much more useful procedure, however.

All BASICs have the ability to evaluate the results of logical operations. If I type "PRINT 0=0", Microsoft BASIC-80 on my Heathkit H-89 types "-1". Likewise, if I type "PRINT 0<>0", BASIC types "0".

Here, MBASIC evaluates the expression "0=0", and returns a parameter (-1) to say the expression is equal or true. Similarly, it evaluates the expression "0<>0" and signifies the falsity of the expression by returning a zero.

By altering the code in the sample routine to fit the way your BASIC evaluates true and false expressions, you too will have a REPEAT/UNTIL construct. Then we will alter that basic routine to create two more programming tools.

Begin by typing in the following program. Then RUN the program. Make a note of the program lines printed on your screen. They are the customized REPEAT/UNTIL structure based on how your BASIC evaluates logical expressions.

```
10  PRINT CHR$(27)+"E"
20  TRUE=(0=0):FALSE=(0<>0)
30  IF TRUE < FALSE THEN C$="STEP -1"
40  PRINT "10 ";"FOR X =";FALSE;"TO ";TRUE;C$
50  PRINT "20 loop-body"
60  PRINT "30 X = condition"
70  PRINT "40 NEXT X"
80  END
```

Line 20, the loop-body can be anything -- a subroutine, an input, or a print. There are no restrictions here. Normally, the routine would be a repetitive operation performed until some pre-defined condition occurs. However, the number of iterations before that condition occurs is probably not known or else a FOR/NEXT loop could have been employed. A little later, I'll tell you how you can cause this structure to behave as a repeating DO/CASE routine.

Line 30 is the most critical line because it sets the stage for the routine to terminate. If it is not precisely written, routine performance will be unpredictable. The object here is to assign the variable "A" your BASIC's "true" value only when you are ready for the routine to terminate.

Let's say you want a routine to INPUT a character until the operator enters only a CHR$(13) carriage return. Your routine might look like this:

```
10 REM REPEAT/UNTIL LEN(N$)=0       10 REM REPEAT/UNTIL LEN(N$)=0
20 FOR A=0 TO 1                     20 FOR A=0 TO -1 STEP -1
30 INPUT N$                         30 INPUT N$
40 A=(LEN(N$)=0)                    40 A=(LEN(N$)=0)
50 NEXT A                           50 NEXT A
```

In this little routine, you cause the variable "A" to step from zero to either 1 or -1, depending on how your BASIC works. However, "A" will never become 1 or -1 unless the condition in line 30 evaluates to a true condition. When that occurs, the variable "A" is assigned the terminating value of 1 or -1 and the loop ends upon executing line 40.

For purposes of illustration, let's say my computer evaluates a true statement as a "-1" and a false statement as a "0". Line 30 assigns the variable "A" the value which results when the computer evaluates the expression (LEN(N$)=0). If the length of N$ is zero, then "A" assumes the value of -1 and the FOR/NEXT loop terminates. If, however, the length of N$ is greater than zero, the variable "A" assumes the value of zero and the loop continues.

The next variation of the REPEAT/UNTIL construct involves adding one external routine line to imitate a WHILE/WEND routine. Let's assume you have a file maintenance routine you want executed only if you have not reached the End Of File. You want to DO the routine WHILE NOT End Of File (EOF).

```
 10 REM DO/WHILE NOT END OF FILE (EOF)
 20 IF NOT EOF THEN GOSUB 100
 30 END
100 FOR A=0 TO 1
110 INPUT #1,N
120 PRINT #2,N
130 A=(EOF):REM I'm assuming EOF returns "1" if true
140 NEXT A
150 RETURN
```

In this example, line 20 tests the condition to see if there is an End Of File condition. If there is not such a condition, program execution jumps to line 100 and executes through line 140 WHILE NOT EOF (traditionally, the EOF function returns a "0" or a "-1" as a flag to indicate file condition). When the End Of File condition occurs, the subroutine terminates and RETURNS to the calling line.

The only difference between this routine and the ones previously presented is line 20 which tests the condition and provides a subroutine jump. Line 20 provides the "WHILE NOT EOF" part of the construct and the subroutine beginning at line 100 provides the "DO" procedure.

Earlier I said one could use this construct as a repeating DO/CASE structure. All that entails is adding a little more code to the loop-body. Here is a routine which continuously jumps to one of three routines depending on the CASE of a variable "B".

```
10 REM DO/CASE FOR B IS LESS THAN, EQUAL TO,
   OR GREATER THAN 0
20 FOR A=0 TO 1
30 INPUT B
40 IF B<0 THEN GOSUB 100
50 IF B=0 THEN GOSUB 200
60 IF B>0 THEN GOSUB 300
70 A=(0<>0)
80 NEXT A
```

This routine will take one and only one action depending on the CASE of the variable "B". Notice that this routine will execute endlessly since the expression "(0<>0)" will never evaluate true. Of course, with a little input processing, lines 40 through 60 could have been replaced with "ON B GOSUB".

You will have to evaluate your own programming style and decide how and when you will apply these tips to avoid GOTOs. Perhaps you will decide to tolerate GOTOs in subroutines and shun them in the main program. On the other hand, you may feel the improvements in your ability to read your programs and trace their execution are worth the little extra efforts to plan your work to incorporate these new techniques.

Using a FOR/NEXT loop, we have been able to create three structured programming constructs. If you feel, like I have, that your BASIC has been holding you hostage and forcing you to use GOTOs against your will, now you can write structured code without GOTOs.

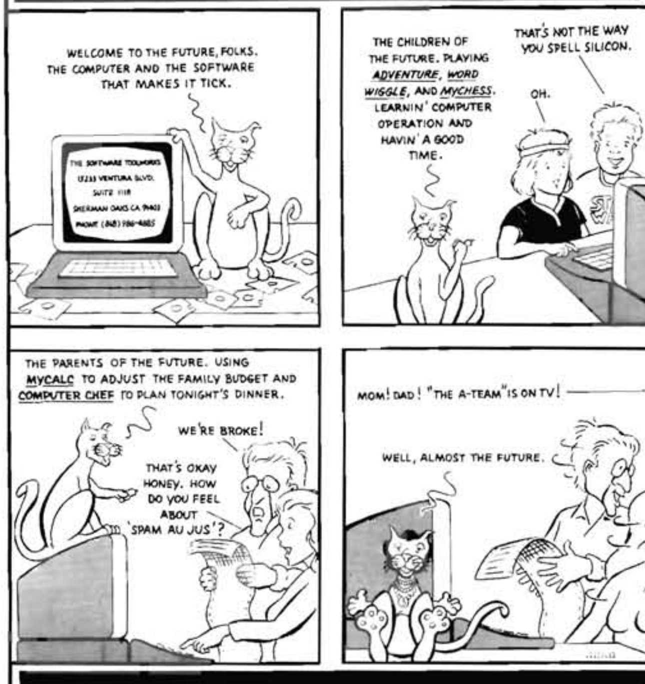
Check out the

New HUG Products

See page 34

The Software Toolworks® presents:

THE FUTURE



WELCOME TO THE FUTURE, FOLKS. THE COMPUTER AND THE SOFTWARE THAT MAKES IT TICK.
THE SOFTWARE TOOLWORKS 15233 VENTURA BLVD. SUITE 1118 SHERMAN OAKS CA 91403 PHONE (818) 986-4885
THE CHILDREN OF THE FUTURE. PLAYING ADVENTURE, WORD WIGGLE, AND MYCHESS. LEARNIN' COMPUTER OPERATION AND HAVIN' A GOOD TIME.
THAT'S NOT THE WAY YOU SPELL SILICON.
OH.
THE PARENTS OF THE FUTURE. USING MYCALC TO ADJUST THE FAMILY BUDGET AND COMPUTER CHEF TO PLAN TONIGHT'S DINNER.
WE'RE BROKE!
THAT'S OKAY HONEY. HOW DO YOU FEEL ABOUT 'SPAM AU JUS'?
MOM! DAD! "THE A-TEAM" IS ON TV!
WELL, ALMOST THE FUTURE.

Languages, games, productivity software, computer cookbooks and more are formatted for IBM PC, PCjr, most CP/M systems and Heath/Zenith HDOS. They range in price from $19.95 to $59.95. For a free 41 product catalog contact The Software Toolworks, 15233 Ventura Blvd., Sherman Oaks, CA 91403, (818) 986-4885.

CP/M is a registered trademark of Digital Research.

LEARN TO PROGRAM H89/H19 GRAPHICS IN MBASIC

BASIC GRAPHICS PRIMER is a computer based course that will teach you fundamental MBASIC programming techniques to control the H19 terminal or H89 computer graphics. Seven lessons will teach you many forms of graphic plots, data plotting, multiple sequential and random access files used for graphics, basic figure animation techniques, and simple computer graphics game design. Each lesson consists of simple illustrative program segments with complete explanations of the programming techniques involved, questions to reinforce the lesson material, and one or more example programs. Most H19/H89 terminal commands are covered in the lessons, including use of the special function keys. **BASIC GRAPHICS PRIMER** is a "HOW TO" course with plenty of examples. If you want to learn how to take full advantage of the H19/H89 graphics features in your programs, **BASIC GRAPHICS PRIMER** is a must. For CP/M only. Requires a printer.

ONLY $49.95!

CP/M version for H89, H8, H19 requires CP/M and MBASIC

NewLine Software

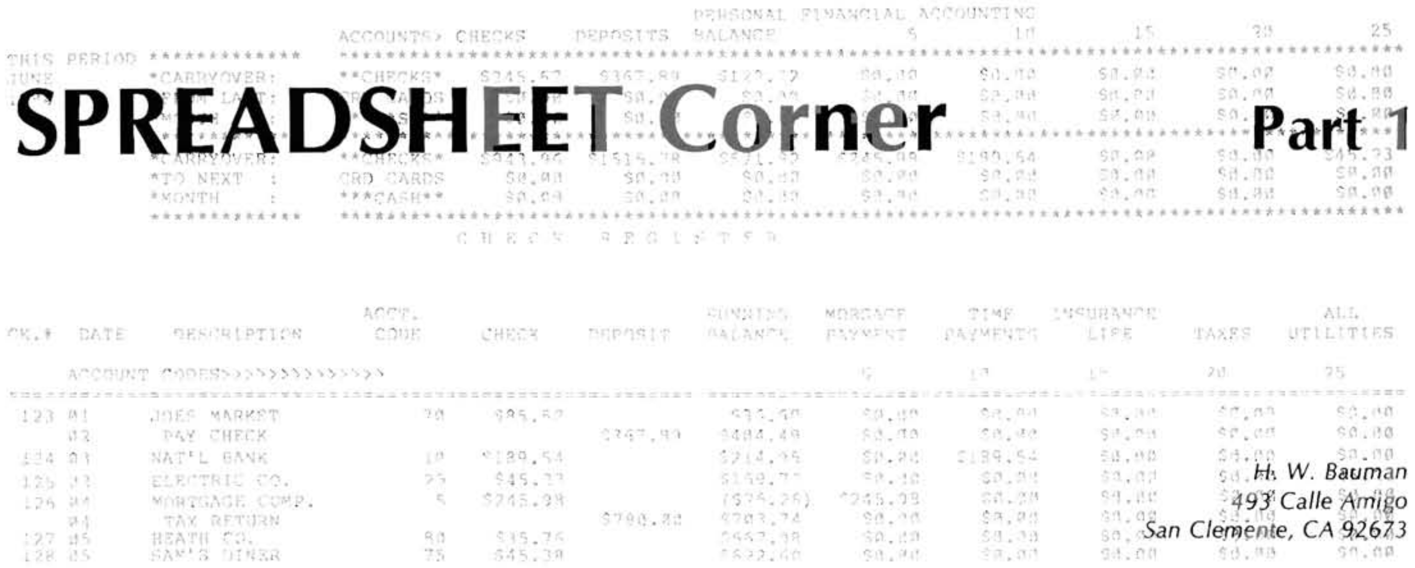
P.O. Box 402, Littleton, MA 01460 (617) 486-8535

NAME ______________________ Send me ____ BASIC
STREET ______________________ GRAPHICS PRIMER
CITY ______________________ program(s) for CP/M
STATE ____________ ZIP ________ at $49.95 each.

Check one: ☐ payment enclosed ☐ send COD (add $3.00)
Send order to:

NEWLINE SOFTWARE, P.O. BOX 402, LITTLETON, MA 01460
Foreign orders: add $3.00 Airmail, $10.00 for non-U.S. checks

CP/M is a trademark of Digital Research, Inc.
MBASIC is a trademark of Microsoft, Inc.



SPREADSHEET Corner Part 1

H. W. Bauman
493 Calle Amigo
San Clemente, CA 92673

This month REMark is introducing a new series of tutorial articles starting at the beginners' level to show the reader how to use the many, very important software programs that have been named "Spreadsheets"! They are really Financial Planning and Modeling Languages. Another name for these programs could be "Mindware". Mindware helps the computer user to come to decisions based on a "synergistic" interaction between the mind and the computer. If these two words have not scared you from reading this article, you will find the rest of the series easy to read and follow!

In 1979, two rather unknown programmers, Dan Bricklin and Bob Frankston of Software Arts, "electrified" the microcomputer world on its "rear" (maybe I should use "ear") by releasing their electronic spreadsheet program VisiCalc. This program won the admiration and minds of thousands of computer operators and non-operators (businessmen?) and helped to sell APPLE (dirty name?) computers by the thousands.

What is a "spreadsheet"? It is nothing more than an electronic piece of paper with columns, rows, and cells or blocks (where the columns and rows intersect). The format (another fancy word?) is arranged on the computer screen as a grid as shown below:

```
        A         B         C         D         E         F

1

2

3

4

5

6

7

8

9

10
```

The columns are identified by letter designations, the rows by numbers. Each position where a column and a row intersect (What again? Why not say meet?) is a coordinate (Not another fancy word.), or location, like you would find on a Thomas Street Map (Why didn't he say that in the first place?). The relationships between values in these coordinates is determined by simple instructions entered into the coordinates in the form of algebraic formulas (don't get panicky, that just means (a + b) or some other similar expression). If you can visualize that street map as a grid of images forming squares (we will call them cells, easier to spell and type) you will easily and quickly catch on to the power of a spreadsheet and how it can work for you! I now promise to stop trying to impress you with my lousy humor and fancy vocabulary.

For our first assignment, please obtain a state of Michigan map and find what "cell" the cities of Benton Harbor and Saint Joseph are in. I am SERIOUS! This will visually show you what we have just discussed. Now, wasn't that an easy assignment? The hardest thing is to get the road map. We will attempt to make all the problems "seem" that easy! The sense of accomplishment will be higher, I promise.

The spreadsheet programs are replacements for the traditional financial modeling tools -- the accountant's columnar pad, the pencil, and the calculator. In fact, the spreadsheet programs are to those "tools" what the word processor programs were to the typewriter. The spreadsheet offers dramatic improvements in the ease of building and using financial "models", also sometimes called "templates". I hope you do not mind my leanings towards the financial, but that is my field. I will try to work other fields into the future articles.

Gradually, computer programs are becoming easier to use, but book and magazine publishers believe that the demand for tutorial information that explains software will continue to grow! The world of computer novices is enormous and with the sale of every new microcomputer it increases. No matter how "user friendly" the software gets, there will always be plenty of people for who using a certain kind of software, such as an electronic spreadsheet or a DBMS (database-management-system) is a new experience. Even experienced people who know how to use complex software can use a little assistance once in awhile. I know that is true for myself. I get myself in a "rut" and cannot make further progress. For example, creating a sales forecast, writing a business plan or analyzing the effect of taxes on alternate investment strategies would be easier if frequently used words, formulas and numbers didn't have to be determined and keyed into the computer through the keyboard. To provide that ease, "SPREADSHEET Corner" will create what is called "models" or "templates" (programs). These are used in the computer along with the spreadsheet software for the needed analysis, study, etc. The template is a general model created by someone who knows a subject, tax accounting for example, and who records (saves) it on a magnetic disk. When the computer operator uses the template, it is read by the software; and the general "model", which otherwise would have to be designed and entered from the keyboard, is read into the computer's memory without effort. Because the computer's memory must be able to hold the entire spreadsheet program while the model is being used or designed, the program is not bound by the physical limitations of the visual or printed page. (This is why spreadsheet software requires a lot of RAM (working memory)! More about that later.)

But templates are not very flexible! For example, a template that computes your income tax return requires annual changes, Congress changes the laws and the rates continuously every year. As a consequence, templates have not sold well. Most templates just give users better ideas of how, as opposed to solving problems and the users end up redoing the model. Therefore, "SPREADSHEET Corner" must show you readers how to do this!

The act of building (creating) a spreadsheet model defines all the mathematical relationships in the model. Don't let that scare you. We will make it easy! Until you decide to change them; every Sum, Product, Division, Subtraction, Average, Net Present Value, etc. will remain the same. Each time you enter data (values) into a model, computations will be made with no effort on your part! All computations will be calculated without math errors. The next time you enter new data (values), the formulas will still be available to again calculate for you. How about that?

Even more important, the spreadsheet allows you to ask "WHAT IF...?" after your model has been designed. If you were to use paper, pencil, and calculator to build your model, then every change would require recalculating every relationship in the model. If the model has 100 formulas, and you change the first one, you must make 100 calculations by hand to "flow" the change through the entire model. When you use a spreadsheet, the same change requires only pressing a few keys! The program does the rest! This ability makes extensive "WHAT IF" analysis practical. Thus, I hope that you can see what makes spreadsheets the "important tools" that they are for both large and small businesses. More and more companies are expecting their employees to know how to use them to help speed and simplify financial, engineering, chemical, and architectural analysis. Your job and promotions could depend on this!

Spreadsheet programs are sometimes called planning tools, and one of their major uses is business planning. But thinking of spreadsheet programs only in this way ignores the spectacular power that they have to solve other types of problems. They can be created (configured) to solve about any problem that we use to attack with paper and pencil. In fact, the more complicated the problem, the greater the savings in time and accuracy with spreadsheets. I hope to demonstrate this to you with future "SPREADSHEET Corner" articles!

Next, I am going to assume that many readers do not have spreadsheet software. I do not want to make this article a "Product Review", but I do want to discuss some of the many spreadsheet software products that are available to help you make a choice that might best fit your possible applications, computer, and pocketbook.

The first generation of spreadsheet software products are all very similar and they usually can be purchased in an 8-bit or 16- bit version. Thus, any reader with an H-8 or H/Z-89/90 with at least 48K RAM or an H/Z-100 computer can work along with "SPREADSHEET Corner". However, I want to warn you that some of the models we will create use a lot of RAM! At this point down the "road" quite a few months from now, you will have memory problems, so be sure to have all the RAM that your pocketbook can meet. Here's a list of some first generation spreadsheet software products:

1. VisiCalc
2. SuperCalc, also known as PeachCalc
3. MultiPlan

They all provide a worksheet area (like the road map) with 254 rows and 64 columns. That means that you have 254 times 64, or approximately 16,000 cells that can contain titles, text, numbers, or formulas with nominal size columns. This software provides the ability to vary column width as needed by the user. The 8-bit versions require CP/M version 2.2, 48K RAM (better have 64K), at least one (1) floppy disk drive (two (2) are better), and a printer if you want hard copies. The 16-bit versions require MS-DOS/Z-DOS with at least 128K RAM (192K is better and PeachCalc can use up to 256K), two (2) floppy disk drives, and a printer if you desire hard copies. In the order of my preference, I would choose MultiPlan first, followed by SuperCalc/PeachCalc, and then VisiCalc.

For flexibility, Microsoft Corporation has created a family of what it calls expert programs for MultiPlan. The programs ask the user questions in English about a particular application for the program in use. MultiPlan with the "template" then creates and analyzes the "model". For example, with MultiPlan and the company's template of budget expertise, it is possible to create in minutes a model of a variable-cost accounting system that ordinarily could take man-days to design and enter by the keyboard to the computer. In the months that it first appeared on the market, it was on the list of best-selling software!

In early 1983, LOTUS Development Corporation published the popular microcomputer 16-bit LOTUS 1-2-3 software. This was the start of what I will call second generation spreadsheet software. At the Las Vegas Fall 1983 Comdex Show, the microcomputer makers and retailers "roamed" over 10 miles of aisles of booths studying new products, making deals, and comparing notes. While there was no single product that "wowed" the approximately 80,000 attendees, the largest crowds were clumped around software company's booths. They were there for one reason! Software -- the "sets" of electronic instructions that bring a computer to life -- is the greatest item that sets the course of the $5- billion-a-year microcomputer business. This is due to the wave of new software products that increase what both the novice and the expert microcomputer users' can do, but also how quickly and easily they can do it. It is also due to the increasing uniformity of computer designs themselves.

In the minds of many show-goers, honors for the software product that stold the "soft-war" show belonged to LOTUS 1-2-3! They had THE PRODUCT and the merchandising know-how. VisiCorp's VisiOn and Microsoft's Windows were close behind! There was an abundance of "integrated" software packages, such as the top-selling LOTUS 1-2-3 product. Integrated packages which "fuse" the electronic spreadsheet, database management, and graphing programs (let us not forget the valuable six lesson tutoring programs), have become the software industry's best selling product. SuperCalc3 of the San Jose-based Sorcim Corporation was unveiled at the Comdex show, but was nearly a year late compared to LOTUS 1-2-3 and in this writer's mind offered nothing better. Microsoft did not show their upgraded MultiPlan. VisiCalc displayed a slightly upgraded version with their second generation product still coming. Another, called InteCalc, from Schuchardt Software Systems is currently available on the IBM PC. Also, we observed Innovative Software Inc.'s Smart spreadsheet. These are presently not available for the H/Z-100.

My first choice for a second generation spreadsheet software product is LOTUS 1-2-3 followed by Sorcim's SuperCalc3! Both have a spreadsheet, limited word processing, limited database management, and graphics. The new spreadsheet features include two-key sort and financial functions; such as, internal rate of return and future values of ordinary annuities, to name a few. The graphic's programs produce Pie, Line, Area, Bar, Stacked Bar, X-Y and High-Low graphs. (Be sure you have a printer with this capability if you want hard copies of these graphs.) They both have color facilities if your computer will support color.

Like MultiPlan, expert templates are under development for LOTUS 1-2-3 that will be even more powerful. The LOTUS programs combine financial modeling with the ability to manage electronic files and to draw charts on the computer's video screen or printer. By special commands known as "macro-language", the expert model could guide the LOTUS 1-2-3 automatically through a complete analysis of the information in its electronic files, or database, and present the results graphically on a video screen or plotter. With enough RAM, LOTUS 1-2-3 has one of the largest worksheet grids available. It can provide a maximum of 256 columns by 2048 rows for a worksheet with over half-a-million cells. Does that "wow" you?! It does me. With the help of you readers, maybe we can find a way to use all of this power! How about it?

A third generation of spreadsheet software products have already been unveiled. LOTUS has shown their SYMPHONY integrated program. It is a menu-driven program combining word processing (looks impressive), communications, database management, spreadsheet, and graphic's functions with a window-management system. It will be available on the IBM PC soon and I hear it will be offered to LOTUS 1-2-3 users as a $200.00 update. We will discuss this later in the series when we know more about the software. Context, MBA's latest advertised software claims to extend the integration from 1-2-3 (spreadsheet/database management/graphics) to 4-5-6 (telecommunications/word processing/forms creation). In fact, 4-5-6 is a trademark of Context Management Systems. Who will go for 7-8-9 trademark? Up to what numbers can integration expand? From what I have heard about Context, it is a disappointment. When we know the facts, we will discuss it in future articles.

What does a company after inventing a product like VisiCalc do for an encore? In answer to that question, Software Arts has now released its second software product, TK!Solver. (The TK stands for Tool Kit.) TK is a microcomputer program that lets a non-programmer use interrelated equations to solve technical problems where the above discussed spreadsheet software is most commonly used by home and business analysis. TK is clearly oriented toward professionals in technical fields.

At the Comdex show, Harold C. Kinne, analyst with the Richardson, Texas-based research firm Future Computing Inc., predicted that 1984 will bring introduction of the first big-selling examples of a class of sophisticated data-handling programs called "relational data-based managers" (rDBMS). Such software promises to allow users to quickly sift and retrieve information from vast, separate collections of data. For example, using such a program with a spreadsheet program would enable the sales executive to quickly "pour" through separate data bases maintained by personnel, sales and manufacturing departments to assess the performance of a member of his sales force. The leading Relational Data-Based Management System for microcomputers, in this writer's mind, is Condor rDBMS published by Condor Computer Corporation. They have already announced an upgrade to their software. I would like to have the "SPREADSHEET Corner" readers know how to use such software. It will be a necessary tool in the future, if not already, that companies will expect their employees to know!

I will be using an H/Z-100 with 192K RAM plus a Z-205 256K RAM upgrade board, two (2) floppy disk drives, and a color monitor (Z-135). The software, as of this date, will be LOTUS 1-2-3 with Z-DOS version 1.10 for most of my work. I will also test the programs on MultiPlan and SuperCalc (PeachCalc). I will include any necessary comments, ideas and suggestions about these optional software products in the articles.

Now, lets look into the future and list some subjects that "SPREADSHEET Corner" will discuss:

1. Financial forecasting.
2. Financial planning.
3. Profit and Loss Statements.
4. Stock portfolio planning.
5. Inventory planning.
6. Income tax returns.
7. Tax analysis.
8. Rate-of-return calculations.
9. Manpower assignments.
10. Pricing, quoting, estimating and costing strategy.
11. Loan amortization.
12. Investment planning.
13. Personal financial statements.
14. Sales forecasting.
15. Lease-purchase decisions.
16. Accounts receivable-Ageing report.
17. Accounts payable-Cash flow.

This is to name just a few of the possibles and not in the order that we will study them. I hope that at least some of them will be of interest to REMark's readers. Also, "SPREADSHEET Corner" WANTS reader participation in this series! If you have a subject that you would like to have discussed, please let us know. If the author does not know how to do it, he can invite the readers to help out. I know that there is a wealth of "brains" amongst our readers. Also, if you would like to contribute to "SPREADSHEET Corner" your favorite "model", please submit it and share it with the other readers. All we ask is that you provide "lead time" for publishing and that it "fits" into the "level" of complexity that we have the readers up to. Of course, we will always welcome your suggestions, criticism, and questions! PLEASE! No phone calls. If you want an answer, please enclose a business-size, stamped self-address envelope. If the subject is of individual interest, the writer will provide an individual answer. If the subject is of general readership interest, the writer reserves the right to answer in the articles. Just remember that there are no stupid questions. Only stupid answers! So, do not be afraid to bring up your problems. We are really trying to help the novice the most!

You, the reader, are expected to know your Operating System, CP/M or MS-DOS/Z-DOS! They will not be the subject of any "SPREADSHEET Corner" articles. REMark has had many good CP/M articles. I do believe, that as the author of this series, REMark could use a series of tutorial articles, starting at the novice (beginner's) level about the MS-DOS/Z-DOS Operating Systems for the H/Z-100 computers. Why don't one of you readers offer to write such a series? You will get a lot of satisfaction out of being helpful to your fellow HUG members! This is this writer's idea, not REMark's! I have not discussed this with Walt Gillespie, REMark's Editor, to see if this fits REMark's schedule. I would suggest that you contact Walt and tell him that you are interested.

Closing

Before next month's "SPREADSHEET Corner", please obtain the spreadsheet software that you have chosen for your computer. When you have obtained this software or if you already have one, review your manual that came with the software IN DETAIL!!! Learn how to "setup" and "configure" the software to your computer, operating system, and printer (if you are going to use one). REMEMBER! If you purchased a new product or you had an unused spreadsheet product, that the very FIRST thing that you should do is BACK UP THE ORIGINAL SOFTWARE DISKS!!! Make WORKING COPIES! Put the original disks away in a SAFE PLACE, not where the kids can play with them or you can spill coffee, ashes, etc. on them.

When you have the software configured to your computer setup, RUN the tutorial lessons that come with most spreadsheet products. The first time through, follow the manual as you go step-by-step as you proceed. Then, the next time you go through the lessons, on another day, see how much you can do without referring to the manual. If your software has "HELP" screens, use these as you need them. Make as many RUNS as you need to "really" know your software and computer setup. Now, make a list of the "functions" and "commands" that you do not COMPLETELY understand. Look each of these up in your manual and then RUN through the lessons again. This time you will find that you will know better what you are actually doing. This preliminary work will help you next month!

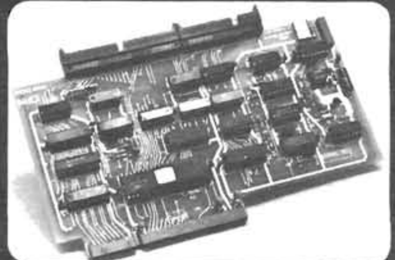
In Part 2, we will review the most basic (beginning?) functions and commands. We will compare them between MultiPlan, SuperCalc/PeachCalc, and LOTUS 1-2-3. Then, we will take a look at how you start a worksheet on each of these software products and we will create a few small models that we will build on in future articles. When you complete Part 2, you will know how to use the worksheet and how to save it. We are going to move along at a pretty fast pace at the beginning. I want you to be able to build a Federal Income Tax Return Computation model (Form 1040) by January, 1985. Thus, we will all be able to do our Income Tax by computer for the 1984 Return. Maybe we can even try a "WHAT IF" on our returns. I must believe every reader can use such a model.

Once again, do not forget! "SPREADSHEET Corner" is for all REMark reader's participation. If you have any kind of questions, comments, suggestions, criticisms, "models", etc.; please let us hear from you. This will make the series more useful to everyone. We want to help the novice but we also would like to think that we can stimulate the experts also. PLEASE! No phone calls! If you expect an answer, furnish a business-size SASE. See you next month and be ready to go "SPREADSHEETING".

International Heath/Zenith Users' Group Conference
Heath Zenith Users' Group
Saint Charles, Illinois • July 27, 28 & 29, 1984

Be There!

1 CONTROLLER



FOR 8" & 5.25" DRIVES

Now be able to run standard 8" Shugart compatible drives and 5.25" drives (including the H37 type) in double and single density, automatically with one controller.

Your hard sectored 5.25" disks can be reformatted and used as soft sectored double density disks. The FDC-880H operates with or without the Heath hard sectored controller.

PRICED AT $395

Includes controller board CP/M boot prom, I/O decoder prom, hardware/software manuals BIOS source listing.

HDOS driver now available for $50.00.

5-20 day delivery–pay by check, C.O.D., Visa, or M/C.

Contact:
C.D.R. Systems Inc.
7210 Clairemont Mesa Blvd.
San Diego, CA 92111
Tel. (619) 560-1272

ANNOUNCING THE MODIFIER

A disk utility that modifies the CP/M BIOS to be able to read and write to a number of 5.25" CP/M disk types.

There is a growing need for the everyday user of computer systems to be able to take data files home from the office to continue to work on them. The computers at home and at work may both run a version of CP/M, but the disk structures may be incompatible. This is especially a problem in the 5.25" world. MODIFY 89 was designed to address this problem. MODIFY 89 makes the CP/M operating system access a specified 5.25" drive as one of the below disk types.
Disks placed in that drive that are of the specified type can be used as if they were one of the standard disk types accepted by the H8, H/Z89 or H/Z90 computers. Thus PIP, STAT, DIR and others will work for that disk also. The price for MODIFY 89 is $49.95.

MODIFY 89 is set for the following disk types:

- Access S.S. D.D.
- Cromemco S.S. D.D.
- DEC VT180 S.S. D.D.
- IBM PC/Zenith 100 (CP/M) S.S. D.D.
- IBM PC/Zenith 100 (CP/M) D.S. D.D.*
- Kaypro II S.S. D.D.
- Morrow Micro Decisions S.S. D.D.
- NEC PC-8001A S.S. D.D.
- Osborne S.S. S.D.
- Osborne S.S. D.D.
- Otrona D.S. D.D.*
- Superbrain Jr. S.S. D.D.
- TI Professional S.S. D.D.
- TRS-80 Model I (Omnikron CP/M)
- TRS-80 Model III (MM CP/M)
- Xerox 820 S.S. S.D.
- Xerox 820-II S.S. D.D.
- Standard (Tests for H/Z37 and C.D.R. Disk types)

* = Double sided 5.25" drive required

S.S. = single sided, D.S. = double sided, S.D. = single density, D.D. = double density

Limitations: MODIFY 89 is not a disk duplicate program. It is currently available for use with an H/Z89 or H/Z90 computer that has an FDC-880H double density 8" and 5.25" controller, using C.D.R.'s BIOS V.2.9 or with an H8 computer using the FDC-H8 by C.D.R. Systems, Inc.

MODIFY 100 will soon be released for the Z100 line of computers at a price of $75.00.

Contact:
C.D.R. Systems, Inc.
7210 Clairemont Mesa Blvd., San Diego CA 92111
Telephone: (619) 560-1272
Or a C.D.R. Systems, Inc. dealer near you.

Practical File Management

David E. Warnick
RD #2 Box 2484
Spring Grove, PA 17362

Part 2 - We Begin Adding Modules

With the completion of last month's menu and control program, we're ready to do the actual file-handling steps. There are seven possible functions to be performed. They are:

1) Add information to the file.
2) Change an existing record.
3) Delete data from the file.
4) Look up information from the file.
5) Name a file to be worked on.
6) Print a chart of the files contents.
7) Sort the file.

Before we can do anything with our file, we've got to open it. Before we open it, we've got to know its file name. Thus, the first module we must develop is the new file name subroutine. From last months flow chart we see that we begin this module on program line 14000.

Before we go any farther, though, we should discuss key files as we'll open them with this portion of the program. In past articles, when we sorted a file and two items had to be swapped, we had to swap the strings being compared and all the associated strings for the two record numbers being exchanged by the swap command. This could amount to quite a few bytes of information if our records were long. The ability to read the entire file into arrays in RAM for sorting also limited us in regard to the amount of data or the size of the file we could sort.

Key files won't remove all these limits, but they will allow us to handle larger files and perform our sorting faster. Suppose we were to set up two arrays for our sorting. The first would contain the field of data we were sorting on (names, dates, etc.) to reorder our file. The second array would contain the number of the record within the random file. We'll soon see that any record number (up to 32767) can be stored in just two bytes. Thus all the computer has to swap is our key field (we had to do that before) and a two-byte record number, rather than all the bytes of the original record. We can also put many more records into our arrays as each takes up less space. This permits us to sort larger files.

After we've sorted our data, the second array contains the record numbers in the order they should be. If we want the 50th record of the sorted file, we'd look at array item 50 and get the actual record number shown there. This array is the KEY to the correct order of the data file. When we save this information on a disk for later use, we call it a KEY FILE. Think of the possibilities. Suppose we have a name and address file of 100 records, each 80 bytes long. We'd like a copy of it in alphabetical order. That's 80 * 100 or 8000 bytes. We'd also like a copy in order by zip code. That's another 8000 bytes. On and on it goes. The bigger the file, the worse it is. With a key file, the original data file would still need 8000 bytes. Each key file would need 100 * 2 or 200 bytes. This is the essence of a data base. A large file contains all our data, and smaller key files hold the record numbers for only those records we want in the order we want them for each application we have. You'll see more clearly how it works as we progress through our program.

I keep making claims that only two bytes of memory are required to store a record number. How can this be? The number may have up to five digits. Let's look at the way our computer handles numeric information when running MBASIC.

The numbers we use every day can appear in three distinctly different ways to a computer within MBASIC. In order of complexity, these are called:

1) Integers
2) Single Precision Numbers
3) Double Precision Numbers

The simplest of these, the integer, may be any whole number with a value between -32768 and +32767. Stored in the computer it takes only 2 bytes (16 bits) of RAM. Fifteen bits show the value (2↑15=32768) and the sixteenth bit is available for the sign (plus or minus).

Single precision numbers are stored as 7 digits, but are printed with only 6 digits. They may be positive or negative numbers, or be represented in exponential form. The price paid for this is four bytes of space. This is the way your H8 or H/Z-89 handles numeric data if it's not told to do otherwise.

The third way of handling numbers, double precision, will really be appreciated by accountants, scientists, engineers, and the like. It provides sixteen digits of accuracy in handling, storage, and printing. When mathematical precision is more important than available RAM (those of us working in technical fields think it always is, even for adding 2+2), this is the way to go. The price paid for this accuracy is eight bytes of RAM.

Not only do the more precise methods of handling numbers require more space in RAM and on disks, they also require that more bytes be handled for each operation we ask the microprocessor to perform. The more bytes it must handle, the longer the operation takes, and the slower the program will execute. The lesson to be learned here is that **"For the fastest possible execution of an MBASIC program, always specify numeric variables in the lowest usable (or acceptable for program needs) precision".**

How do we do this? The instructions are in the first section of our MBASIC manuals. This is the part we skip over first on the way to skipping over most of the manual. If we do nothing, all numeric variables (A, B, C, etc.) are handled as single precision numbers. We see six digits and can have decimal values. We're only using four bytes per number, so it could be worse. But, it could be better. Much better!!! Every time we execute a FOR- NEXT loop, the counter could be working with two bytes rather than four. When handling record numbers for random files the same is true. If we were to write the program line

```
FOR X=1 TO 100
```

we would be in single precision using four bytes 100 times. To force the computer to use integers we use a percent sign (%). Write the above line as

```
FOR X%=1 TO 100
```

and we've accomplished our task. Variables can be forced to take any of the three forms above by using the correct signs as follows:

Type	**Sign**	**Example**
Integer	%	A%
Single Precision	!	A!
Double Precision	#	A#

We'll do this throughout our programming. We could also define all variables beginning with certain letters as one of the above types or as a string variable by using the commands DEFINT, DEFSGL, DEFDBL, or DEFSTR. However, this eliminates easily using A as single precision and AL$ as a string. Therefore, for our program, we'll use the appropriate signs with each variable.

You've been using the above type of notation every time you place a dollar sign after a variable name to declare it as a string, so the concept should be an easy one to master.

Now that you understand the principals of key files and numeric variable types, we can go on with our programming. Figure 1 shows the first part of our New File Name routine. This will permit us to open a file and work on it, then close that file and open another. Of course we'll also open a key file with each data file we use. The operation of this part of the routine is simple. As with each module we write, we first erase the display.

Because we're going to open new files and we may already have had other files open, we close the old files just to be sure. If none were open, no harm is done. Next we put instructions on the screen and ask for a file name. We follow that with requests for drive names for the data file and the key file. They need not be on the same disk. (HDOS users, change lines 14100 - 14200 and 15000 - 15090 to allow SY0 - SY2 as inputs with appropriate error messages.) Error messages are provided in case an illegal input (operator error) is encountered.

Figure 1

```
14000 PRINT ED$               'ERASE SCREEN
14010 CLOSE #1                'CLOSE DATA FILE
14020 CLOSE #2                'CLOSE KEY FILE
14030 PRINT FNCA$(2,16);
      "SELECT THE NAME OF THE FILE TO BE WORKED ON"
14040 PRINT FNCA$(7,16);"FILE NAMES MUST CONSIST OF ONE
                        TO EIGHT CHARACTERS"
14050 PRINT FNCA$(8,16);
     "BEGINNING WITH A LETTER. MEANINGFUL NAMES SUCH AS"
14060 PRINT FNCA$(9,16);
      "THE STOCK MARKET ABBREVIATION FOR THE COMPANY IN"
14070 PRINT FNCA$(10,16);"THIS FILE ARE SUGGESTED."
14080 PRINT FNCA$(4,30);
14090 INPUT"",F$
14100 PRINT FNCA$(22,16);
      "DRIVES ARE SELECTED BY PRESSING A"
14110 PRINT FNCA$(23,16);"SINGLE LETTER, A THRU F"
14120 PRINT FNCA$(13,16);"SELECT A DRIVE FOR THE FILE"
14130 PRINT FNCA$(15,30);
14140 D1$=INPUT$(1)
14150 IF D1$<"A" OR D1$>"F" GOTO 15000
14160 PRINT D1$
14170 PRINT FNCA$(17,16);
      "SELECT A DRIVE FOR THE KEY FILE"
14180 PRINT FNCA$(19,30);
14190 D2$=INPUT$(1)
14200 IF D2$<"A" OR D2$>"F" GOTO 15050
14210 PRINT D2$
15000 PRINT FNCA$(15,16);
      "THE DRIVE YOU SELECTED IS NOT IN THE RANGE ";
15010 PRINT "A - F --RE-ENTER"
15020 FOR X=1 TO 600:NEXT X
15030 PRINT FNCA$(15,1);EL$
15040 GOTO 14130
15050 PRINT FNCA$(19,16);
      "THE DRIVE YOU SELECTED IS NOT IN THE RANGE ";
15060 PRINT "A TO F --RE-ENTER"
15070 FOR X=1 TO 600:NEXT X
15080 PRINT FNCA$(19,1);EL$
15090 GOTO 14180
```

Having taken the necessary inputs, we move to Figure 2 to continue our program. We use lines 14220 and 14230 to build file names for our data and key files. For this program we will always use the extensions .DAT and .KEY for our data and key files. These could just as easily be taken as inputs from the console. If we wished to use multiple key files, this may have been necessary. The other choice, and the one I prefer, would be to ask for an additional name for the second key file. As our program is written, both files share the same name with different extensions to distinguish them apart.

When assigning extensions to file names you should always check your Operating System (HDOS or CP/M) Manual for the conventions used by that system. You wouldn't want to assign .SYS to an HDOS data file or .COM in CP/M. Either system would recognize them as something they're not, an executable file.

With the file names made up, they are opened and fielded. Then we use the binary search routine to determine how many records exist in our data file. If none exists (as in creating a new file) line 14340 sends us to the end of this module which will return us to the menu. Why not go to the menu directly? This method uses an extra step. I did it this way so that our module would have only one entry and one exit point. You'll appreciate this as you modify the program for your own application later. In any case, the number of the last record in the data file is assigned to the variable NR%. That stands for Number of Records. Note that we've used single-precision values here to reduce storage space and speed up execution.

Figure 2

```
14220 FL$=D1$+":"+F$+".DAT"        'BUILD DATA FILE NAME
14230 FK$=D2$+":"+F$+".KEY"        'BUILD KEY FILE NAME
14240 OPEN "R",#1,FL$,10           'OPEN DATA FILE
14250 OPEN "R",#2,FK$,2            'OPEN KEY FILE
14260 FIELD #1,6 AS A$,4 AS B$     'FIELD DATA FILE
14270 FIELD #2,2 AS C$             'FIELD KEY FILE
14280 A%=1                         'BINARY SEARCH ROUTINE
14290 C%=32767
14300 B%=INT((A%+C%)/2)
14310 GET #1,B%
14320 IF EOF(1) THEN C%=B%-1 ELSE A%=B%+1
14330 IF A%<C% GOTO 14300
14340 IF C%=0 THEN NR%=0:GOTO 14440
14350 GET #1,C%
14360 IF ASC(A$)=0 THEN C%=C%-1:GOTO 14340
14370 NR%=C%
```

We round out the new file module with the steps shown in Figure 3. We'll put all the info from the key file into an array. To use an array it must first be dimensioned. If we do that at the beginning of the program, we're locked into a maximum number of records. I find it more desirable to dimension the array to the exact size the file needs.

But, if this is the second file we're working on during a session with this program, we'll get an "ARRAY ALREADY DIMENSIONED" error. To eliminate this problem, line 14380 first erases the existing array B%(). Then it can be redimensioned with no problem ..... unless this is the first file you're working on during this session. You just can't ERASE an array that's not there. The fix for this is easy. When we begin our program run we'll dimension B% just so it's there. The place to do this is on line 400. The completion of the module looks like this:

Figure 3

```
400 DIM B%(10)
14380 ERASE B%                'ERASE ARRAY TO DIMENSION
14390 DIM B%(NR%)             'DIM ARRAY TO SIZE OF FILE
14400 FOR X%=1 TO NR%         'READ KEY FILE INTO ARRAY
14410 GET #2,X%
14420 B%(X%)=CVI(C$)
14430 NEXT X%
14440 GOTO 1000               'BACK TO MENU
```

Add all the program lines presented in this article to the control and menu routines from last month. Save the program. It's a good idea to save a backup copy on a separate disk as there's quite a bit of effort invested up to this point and there's more to go. After saving the program, run it. Select the "N" option and enter a file name and drive designators. Reselect the option and do it again as often as you like. If you have to make corrections to the program, be sure to save and back-up the corrected version. A back-up copy with typing errors is no good once you've debugged the program. Now exit MBASIC to the Operating System and look at the DIRectory or the CATalog for each disk you specified while running the test. The files you named are there. There's nothing in them yet, but you've created data and key files for each file name you specified.

There's one more thing to do when developing a modular program, especially one this large, or one we'll be looking at over a period of several months. Keep track of the variables used and their meaning. They may contain a value, as NR% does, which will be used in other modules. By making a chart of those we use and what they represent, we won't run into problems of duplication and we won't have to search through the program to find the one we need. This kind of information should be stored permanently in a .DOC (document) file for future reference. Our chart looks like this.

Variable	Used For
FL$	Data file name with drive and extension
FK$	Key file name with drive and extension
A%	Low end of binary search range
B%	High end of binary search range
C%	Middle of binary search range
NR%	Highest numbered record in data file
B%()	Key array

There's one last line to add to our program. It seems foolish when first running this program to get the menu and select the "N" option. It's always required at this point so why not let the computer select it for us? We do that by directing program execution to line 14000 just before we get to the menu. Add line 600 to your program and save it. Don't forget to save a backup copy on a second disk.

```
600 GOTO 14000
```

That's it for this month. The introduction of variable types and key files should be enough to digest at one time. Next month we'll begin putting data into our files and getting it back out. Then the whole thing will work and become a useful tool.

See you next month.

Heath/Zenith Users' Group

OPEN TO ALL HEATH/ZENITH USERS

Western Regional HUG Conference

November 10th & 11th, 1984

Disneyland Hotel
Anaheim, California

• SPEAKERS • DISPLAYS
• PANELS • PRIZES

Win a computer:
H/Z-100 with Winchester

Write for Registration Info to:
Conference Coordinator
Western Regional HUG Conference
1555 N. Orange Grove Ave.
Pomona, Calif. 91767

WordStar and the Prowriter Printer

Gary Deley
564 Calle Anzuelo
Santa Barbara, CA 93111

I bought the popular C. Itoh Prowriter 8510A dot matrix printer without being sure that it would interface properly with my Heath H-120 computer or with WordStar. It's not listed as one of the printers in the Z-DOS CONFIGUR program or in the WordStar INSTALL program.

Still, I had heard and read many good things about the quality and reliability of the printer, and I liked its many features. These include 120 characters per second, both friction feed for single sheets and pin feed for continuous paper, several different types of faces and pitches, bit-image graphics, and more. So I plunged ahead and made the purchase of a Prowriter with a parallel interface.

The first step after unpacking the printer is to run the Z-DOS CONFIGUR program. Since the Prowriter is not on the list, I selected a Centronics or MX-80 Parallel. A picture came on the screen, showing me where to plug the cable into the parallel port. So far, so good.

The printer worked immediately in ZBASIC. Encouraging, I thought. But what about WordStar? First, I had to tell WordStar the name of the printer, using the INSTALL program. The INSTALL program has never heard of the Prowriter. Taking hope from the earlier success with the CONFIGUR program, I told INSTALL that it was an Epson MX-80 (with parallel interface). WordStar, not noting my subterfuge, went along with the ruse.

Normal text could now be printed without difficulty, and underlining, boldface (double and triple strike), strikeout type, and strikeover all worked immediately as advertised. But there are many more features on the Prowriter, and I wanted to be able to use them all. Also, i wanted to be able to do super and subscripts, a feature this particular printer lacks.

Using the WordStar 3.21 User 4 patch area, I am now able to perform all of the functions available on the Prowriter and super and subscripts. In this article I'll tell you how.

Modifying WordStar

WordStar is modified by patching in characters to the "User 4" patch area. This is not as difficult as it sounds. Following a few simple instructions, you can do this with the DEBUG program that came with your Z-DOS System Disk. The procedure (described later), briefly, is to load DEBUG into memory along with a copy of WordStar, modify certain memory locations in WordStar, and write the result back on the disk, replacing the original copy of WordStar.

Before describing how to do this, it is helpful to understand the User 4 patch area in WordStar. This is an area (beginning at address 746 hex and ending at 7C8 hex) that contains data used by certain printer functions. For example, the character of data at address 7C2 hex is the character to be used for underscore. Of course this character is preset to the underscore character (5F hex). If you had a non-standard printer that required some other code, you could change this character to that code.

We are interested in several of the memory locations in the User 4 patch area that are not set by WordStar when it is delivered. These include the areas for alternate and standard character widths, carriage roll up and down for superscripts and subscripts, ribbon color change, and the four User Printer Functions described on page 2.170 of the Heath/Zenith WordStar Reference Guide.

Each of these memory areas contains five bytes. The first byte is a number from zero to four telling how many of the following four bytes are to be sent to the printer. When you first receive your WordStar disk, all of these bytes are set to zero, which is why when you try to access them through WordStar commands, nothing happens. All of the features of the Prowriter (and other printers) can be accessed by appropriately setting these bytes to something useful.

There are not enough patch areas to be able to obtain all printer functions with single WordStar control characters. Since nearly all functions begin with the escape character (a non printable character in a WordStar file), clearly this character must be provided for. Patching the third User Printer Function (↑E) to the escape character makes this easy to remember.

Also, not all of the patch areas produce the same result. For example, the ribbon color change is a toggle (obtained by ↑Y ↑Y), so whatever function is selected for this area (assuming ribbon color change is not of interest) must be something that can be turned on and off by the same sequence.

Carriage roll up and roll down are also toggles. The superscript toggle (↑T ↑T) causes first roll down to be printed, then roll up. The subscript toggle (↑V ↑V) causes just the opposite. The alternate and standard pitch controls (↑A and ↑N) produce strange results, presumably a requirement of some printers. Namely, setting alternate pitch causes **each** print line to be preceded by the characters ↑N↑A. The alternate pitch must be turned off by using a ↑N before other pitches can be selected. If you try to change to another pitch directly from the alternate pitch, the first line you change will work, but the

About the Author:

***Gary Deley** is presently Director of the Radar Systems Operations at General Research Corporation in Santa Barbara. He received his Ph.D. in electrical engineering at Stanford University, and has been active in computing for over 20 years. Beginning on an IBM 1620 decimal-memory machine, Gary has programmed Control Data 6000-series machines and is presently using DEC VAX machines. The H-120 is his first home computer, but by no means his first Heathkit. Gary has built countless Heathkits of all varieties over the years, beginning with the Model AR-3 shortwave receiver in 1956. His present passion is assembly programming the H-120.*

following lines will be back to the alternate pitch.

The four User Printer Functions (↑Q, ↑W, ↑E, ↑R) operate in a straightforward manner: When, for example, ↑Q is encountered in the text, the zero-to-four characters in the ↑Q user patch area are sent to the printer.

How To Get Super and Subscripts On the Prowriter

Before describing how to set up the User 4 area of WordStar using DEBUG, I'll tell you how to get supers and subs on the Prowriter. It's not easy, but it's possible. (This technique is from Warren White and Frank Gladu of Leading Edge Products, Inc., the distributor of the Prowriter.)

To obtain superscripts and subscripts requires sending a series of codes to the printer before and after the supers and subs. The codes that need to be sent are listed in Table 1, along with a description of what the codes do. For example, to send one or more subscripts to the printer, the codes ESC [ESC T 09 LF must first be sent to the printer. These codes position the printer properly before printing the subscripts. After the subscripts are printed, the codes ESC r LF ESC T xx ESC f must be sent to the printer. (The symbols ESC, LF, and xx are described in the table footnote.) This seems like a lot of work, but it can be automated in WordStar to greatly simplify the process.

The table also shows the characters to print (using LPRINT) in BASIC. This could be simplified by putting these into subroutines.

How To Patch Using DEBUG

DEBUG is a program that, among other uses, lets you change individual bytes in an executable program (with a .COM extension) and rewrite the results back to the disk. This is exactly what we need to do.

Here are the steps to using DEBUG to change WordStar. Place your Z-DOS disk with DEBUG.COM in drive A: and a disk with a copy of WS.COM in drive B:. The copy of WS.COM must not be write protected. Enter the statement

```
A:DEBUG B:WS.COM <return>
```

where <return> means press the RETURN key. After reading both drives, you will see on the screen

```
DEBUG   version 1.08
>
```

The symbol > is a prompt from DEBUG that it is expecting a command from you. The first addresses to change begin at 76B hex. After the > prompt, on the same line type

```
E76B  <return>
```

DEBUG will return with the address you typed plus its present contents, which should be 00, as follows (Note: the 0805 segment address could be different in your case):

```
0805:076B  00:
```

After the 00: type the following three (two-digit) bytes:

```
02 1B 51   <return>
```

Make sure to type spaces between the three bytes. At this point you should have succeeded in changing the addresses 76B through 76D to the above bytes. These bytes mean that the 02 following bytes, which are ESC A, are to be sent to the printer.

Continue the above process until all of the patches are made. At the end your screen will look like this, with you typing in the bytes after 00:

```
>E76B  00:02 1B 51
>E770  00:02 1B 4E
>E775  00:03 1B 72 0A
>E77A  00:03 1B 66 0A
>E77F  00:08 1B 54 32 34 1B 66 1B 5D
>E789  00:01 1B
>E78E  00:04 1B 54 30 39
>E793  00:01 0E
>E798  00:01 0F
```

Table 2 contains the complete list of patches to make along with their description. The column labeled CTRL gives the WordStar commands to print each patch area. (The LABEL column describes the name given by MicroPro to the individual areas within the User 4 patch area.)

The final step is to save the changes made to WS.COM. To do this simply type W after the > prompt and return:

```
>W      <return>
```

The DEBUG program will respond with the message

```
Writing 5180 bytes
```

To exit DEBUG type Q after the > prompt:

```
>Q      <return>
```

Use this version of WordStar in all of your subsequent printing with WordStar.

Using the New Commands in WordStar

Table 3 describes how to obtain the new printing features on your Prowriter. Note that to enter the control symbol ↑N in a WordStar document, you hold down the CTRL key while pressing the "P" key followed by the "N" key. Also note that printer is very particular; the "N" must be upper case.

As can be seen, superscripts and subscripts require quite a few extra characters to be typed, but at least they are possible. A drawback to supers and subs, as well as some of the other printer control sequences is that they contain non-control characters, which WordStar counts as a part of the line length. If you are using justified lines, this can sometimes cause a line to come out a few characters short.

Now that you understand how the user areas are changed, you can make different changes that suit your purpose better. For example, if superscripts and subscripts are something you do frequently to the exclusion of almost everything else, you can combine User Printer Function patch areas ↑E and ↑R to obtain a combined patch area that is 10 bytes long. (This is done with contiguous patch areas ↑Q and ↑W in Table 2.) The printer sequence ↑E[↑R that begins each superscript or subscript can then be placed into this larger patch area and entered into a document by the single control character ↑E. The drawback is that the ability to send a single escape character to the printer has been lost.

Another possibility is to eliminate superscripts and subscripts altogether. This will free up User Printer Functions ↑Q, ↑W, and ↑R for other uses. For example, ↑Q could be used for changing to elite type (ESC E), and so forth.

This article was written using my patched version of WordStar on my H-120 computer. I've been very pleased with both WordStar and the Prowriter. WordStar is an amazingly competent word processing program with very fast keyboard and screen response. The Prowriter is also fast and versatile, and has proven reliable.

Table 1
Superscripts amd Subscripts On the Prowriter Printer

Description	Code	BASIC LPRINT
SUBSCRIPT(S)		
Enter incremental print mode	ESC [	CHR$(27);"[";
Set line feed to 09/144 inch	ESC T 09	CHR$(27);"T09";
Send line feed	LF	CHR$(10);
Print subscript(s)	subscripts	"subscripts";
Set reverse line feed	ESC r	CHR$(27);"r";
Send line feed	LF	CHR$(10);
Reset line feed to xx/144	ESC T xx	CHR$(27);"Txx";
Set forward line feed	ESC f	CHR$(27);"f"
Leave incremental print mode	ESC]	CHR$(27);"]"
SUPERSCRIPT(S)		
Enter incremental print mode	ESC [	CHR$(27);"[";
Set line feed to 09/144 inch	ESC T 09	CHR$(27);"T09";
Set reverse line feed	ESC r	CHR$(27);"r";
Send line feed	LF	CHR$(10);
Print superscript(s)	superscripts	"superscripts";
Set forward line feed	ESC f	CHR$(27);"f"
Send line feed	LF	CHR$(10);
Reset line feed to xx/144	ESC T xx	CHR$(27);"Txx";
Leave incremental print mode	ESC]	CHR$(27);"]"

```
Notes: ESC = escape character (1B hex, 27 decimal)
       LF  = line-feed character (0A hex, 10 decimal)
       xx  = xx/144 inch line spacing.  Set for normal line spacing:
                xx = 24      (single space)
                xx = 36      (space and a half)
                xx = 48      (double space)
```

Table 2
WordStar Patches

LABEL	CTRL	ADDRESS	PATCH	DESCRIPTION
PALT	↑A	76B	02 1B 51	ESC Q
PSTD	↑N	770	02 1B 4E	ESC N
ROLUP	↑T↑V	775	03 1B 72 0A	ESC r LF
ROLDOW	↑T↑V	77A	03 1B 66 0A	ESC f LF
USER1,2	↑Q	77F	08 1B 54 32 34 1B 66 1B 5D	ESC T 24 ESC f ESC]
USER3	↑E	789	01 1B	ESC
USER4	↑R	78E	04 1B 54 30 39	ESC T 0 9
RIBBON	↑Y	793	01 0E	SO
RIBOFF	↑Y	798	01 0F	SI

Table 3
WordStar/Prowriter Printer Commands

ITEM	CONTROL*	DESCRIPTION
Change font		
Pica	↑N	Not normally needed except to return to pica (printer comes on in pica mode)
Compressed	↑A	Use ↑N to return to pica before using any other font except elongated
Elite	↑EE	
Proportional	↑EP	
Elongated	↑Y ↑Y	Put elongated text between two ↑Y's (after second ↑Y it will return to previous font)
Change line spacing		
Single space	↑ET24	Not normally needed except to return to single space (printer comes on in single space mode)
1-1/2 space	↑ET36	
Double space	↑ET48	
Type faces		
Greek/math	↑E&	Put ↑E& just before Greek/math characters and ↑E$ after to return to normal alphanumeric type
Graphics	↑E#	Same as Greek/math
Alphanumeric	↑E$	Normal type mode (use to return from Greek/math or graphics modes)
Multiple strikes		
Underscore	↑S ↑S	Put underlined text between two ↑S's (note that spaces will not be underlined)
Boldface	↑B ↑B	Same as underscore (this will cause triple strike)
Double-strike	↑D ↑D	Same as underscore (this will cause lighter boldface by only printing twice)
Strike-out	↑X ↑X	Same as underscore (will print hyphen over characters)
Strike-over	↑H	Put ↑H before each character that is to overprint previous character
Super and subscripts		
Superscript	↑E[↑R↑T ↑T↑Q	Put ↑E[↑R↑T before superscript(s) and ↑T↑Q after (note that spacing after super or subscript will be single space; if a different spacing is being used, it must be reselected right after the ↑T↑Q)
Subscript	↑E[↑R↑V ↑V↑Q	See superscript

***Note:** To get ↑X on screen, hold down CTRL key while typing P, then type X. (You can hold down CTRL key while typing X if you want.)

HUG NEW PRODUCTS

NOTE: 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.

885-1235-37 CP/M RDZDOS $20.00

Introduction: RDZDOS is a CP/M program that can copy files from Z-DOS (or any 8-sector 5.25-inch MS-DOS) disks to CP/M disks. It will run on any Heath/Zenith CP/M compatible computer under CP/M-85 version 2.2.101 or higher, or CP/M 2.2.03 or 2.2.04. RDZDOS can also examine files in ASCII (text) or in hexadecimal.

Requirements: RDZDOS requires an H8, H/Z-89,90, or H/Z-100 type computer and Heath/Zenith CP/M. H8 and H/Z-89,90 computers must have a soft sector controller. A 5.25-inch double sided 40 track disk that has been formatted under CP/M-85 or CP/M-86 is also required for the program to "log in" on to regardless of what kind of computer it is run on. Z-DOS/MS-DOS disks must be 5.25 inch 40 track with one or two sides formatted. The drive the disk is read in can be a 40 or 80 track drive, but must be double sided. The destination drive can be any type.

The RDZDOS disk contains the following files:

README	.DOC
RDZDOS	.COM
RDZDOS	.DOC
CONTROL	.CHR
RDZDOS	.ASM

"Soft Sector Only"

Author: Glenn F. Roberts

RDZDOS.COM -- This is the assembled RDZDOS program. It is menu driven and operated with function keys, which are labeled on the bottom screen line. Once the program is started, both the destination and source disks can be changed any number of times so that many files can be copied without restarting the program. **Note:** Z-DOS/MS-DOS programs cannot be run under CP/M, but RDZDOS is useful for transferring text files that have been edited under Z-DOS/MS-DOS, or BASIC programs that do not use graphics, or data files created by business programs, etc.

RDZDOS.DOC -- Detailed instructions for RDZDOS. The program is very easy to use, and the instructions explain every aspect of it.

CONTROL.CHR -- This is a replacement for ALTCHAR.SYS for CP/M-85 users that is useful when using the hex dump mode of RDZDOS. An ASCII interpretation of the hex codes is printed, and if CONTROL.CHR is loaded, non-printable characters show up as abbreviations, such as LF for Line Feed, etc.

RDZDOS.ASM -- The assembly source code for RDZDOS.

TABLE C Rating: (0),(1),(7),(10)

885-3014-37 Z-DOS/MS-DOS UTILITIES II $20.00

Introduction: This disk contains several useful Z-DOS utilities. Most of them will also run under MS-DOS on a Z-150.

Requirements: An H/Z-100 type computer with Z-DOS, or a Z-150 or Z- 160 or IBM PC with MS-DOS (PC-DOS) (except as noted).

This disk contains the following files:

README	.DOC
FONTED	.EXE
FONTED	.BAS
IBM	.CHR
H19	.CHR
DIR150	.COM
DIR150	.ASM
PDIR	.COM
PDIR25	.COM
PDIR	.ASM
DISKID	.COM
DISKID	.ASM
SCRNCLK	.WHT
SCRNCLK	.RED
SCRNCLK	.ASM
CLOCK	.COM
CLOCK	.ASM
DATETIME	.COM
DATETIME	.ASM
SEE	.COM
SEE	.ASM
ONECOPY	.COM
ONECOPY	.ASM
SWAP	.EXE
SWAP	.ASM
ERA	.COM
ERA	.ASM
ZBREF	.COM
ZBREF	.ASM
BRNCHREF	.COM
BRNCHREF	.ASM
ASM	.BAT

Authors:
FONTED, SWAP -- Mark Aagenas, Heath Co.
DATETIME -- Frank Clark, ZDS
DIR150, PDIR, DISKID, SCRNCLK, CLOCK, SEE, ONECOPY, ERA -- Patrick Swayne, HUG Software Engineer
ZBREF, BRNCHREF -- Original program by Rudi Daniel Z-DOS versions by P. Swayne

FONTED -- This is a compiled version of the character font editor presented in REMark magazine. It allows you to easily create new character fonts or modify the keyboard response on your H/Z-100 type computer. This program will not run on Z-150's.

IBM.CHR -- This is a replacement for ALTCHAR.SYS, created with FONTED, that provides IBM-like characters. These characters are bolder than normal and some may find them easier to read, especially on color monitors. H/Z-19 graphic characters are supported. For H/Z-100 only.

H19.CHR -- This character set provides true H/Z-19 style characters, with full descenders on p, y, etc. for your H/Z-100. Since the H/Z-100 has only 9 scan lines per character instead of 10 like the H/Z-19, the catch is that the tops of the characters are on the top line, with the result that reverse video does not look as good as normal. For H/Z-100 only.

DIR150 -- This program is like the DIR100 program released on the first HUG Z-DOS Utilities disk except that this version will run on H/Z-100's or Z-150's (and IBM PC'S). It provides an easy-to-read alphabetized directory with graphic lines separating columns of files. If a label was created on the disk by DISKED (below), it is displayed at the top of the screen. Disk space usage information is presented at the bottom of the screen. Up to 80 files can be displayed on the screen, and if there are more, the program will prompt you to hit RETURN before displaying a new screen of files. DIR150 can handle up to 1024 file names.

PDIR -- This program is similar to DIR150 except that its output goes to a printer. PDIR25 is for H/Z-25 and H/Z-125 printers, and uses graphic lines to separate file columns. PDIR is for other printers.

DISKID -- This program lets you write a volume number and label to a disk, which are displayed at the top of the screen or page by DIR150 and PDIR.

SCRNCLK -- This is the screen clock program that was presented in the May 1984 issue of REMARK. It provides a digital clock display at the upper right corner of your screen that runs while you are running other programs. This version is improved over the published version, and does not "mess up" the top line of text on the screen like the other version occasionally did. SCRNCLK.WHT produces a white clock display, and SCRNCLK.RED produces a red display. For H/Z-100 only.

CLOCK -- This program can turn the screen clock off and on after SCRNCLK is loaded. For H/Z-100 only.

DATETIME -- This is the program presented in REMark that maintains the latest time and date in a directory entry called DATETIME.TMP. If you create an AUTOEXEC file that runs DATETIME at cold boot, your date and time will always be set to the latest date and time on your disk when you boot up.

SEE -- This is a replacement for the TYPE command, for examining text files. It allows you to scroll forward and backward through a file, to search for text strings in the file, and to print screens of the file on a printer. It can scroll sideways to show lines up to 132 characters long.

ONECOPY -- This is a single drive file copy utility. It is useful in situations where one drive of a two drive system is bad, and the "system" still thinks there are two drives. Files up to the maximum allowable size in Z-DOS may be copied.

SWAP -- This is the console swapper program that was presented in REMark magazine. It allows you to use an external terminal on your computer, and to switch back and forth between the main keyboard/screen and the other terminal.

ERA -- For CP/M users who have switched to Z-DOS/MS-DOS, this program allows you to erase files by typing ERA instead of DEL or ERASE.

ZBREF -- This is a variable cross reference utility for ZBASIC or GWBASIC. It lists all of the variables in a program in alphabetical order followed by the line numbers on which the variable can be found. Output can be sent to a printer or the screen. It can quickly process large BASIC programs.

BRHCNREF -- This utility lists all of the branch statements (GOTO, GOSUB, etc.) in a program along with the line numbers they reference.

ASM.BAT -- This is a batch file that lets you assemble programs from start to .COM file in one command line entry. Not for use with .EXE programs.

TABLE C Rating: (0-4),(10)

885-5003-37 CP/M-86 UTILITIES BY PS: $20.00

Introduction: This disk is a useful collection of utilities for the H/Z-100 CP/M-86 user.

Requirements: An H/Z-100 type computer (including the expanded ET-100) and CP/M-86.

This disk contains the following files:

README	.DOC
HXSUB	.CMD
HXSUB	.A86
SCRNCLK	.WHT
SCRNCLK	.RED
SCRNCLK	.A86
CLOCK	.CMD
CLOCK	.A86
TIME	.CMD
TIME	.A86

Ordering Information

For Visa and MasterCard phone orders; telephone Heath Company Parts Department at (616) 982-3571. Have the part number(s), description, and quantity ready for quick processing. By mail; send order, plus 10% postage and handling, up to a maximum of $3.50 to Heath Company Parts Department, Hilltop Road, St. Joseph, MI 49085. Visa and MasterCard require minimum $10.00 order.

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

DATETIME	.CMD
DATETIME	.A86
DIR86	.CMD
DIR86	.A86
PDIR	.CMD
PDIR25	.CMD
PDIR	.A86
DISKID	.CMD
DISKID	.A86
SEE	.CMD
SEE	.A86
SCR	.CMD
SCR	.A86
UNSCR	.CMD
UNSCR	.A86
ONECOPY	.CMD
ONECOPY	.A86
COLOR	.CMD
COLOR	.A86
PSETMX	.CMD
PSETMX	.A86
PSET25	.CMD
PSET25	.A86
SWAP	.CMD
SWAP	.A86
ASM	.SUB

Author: Patrick Swayne, HUG Software Engineer

HXSUB -- This program is a CP/M-86 version of the XSUB program that is supplied with 8-bit CP/M 2.2. It allows you to pass SUBMIT batch command lines to running programs. Operation is identical to the 8-bit XSUB program except that it properly processes a Control-C (entered as "↑C") in a submit file so that you can use Control-C to exit from a batch process involving DDT86 or a similar program.

SCRNCLK -- This is a CP/M-86 version of the screen clock program presented in the May 1984 REMark. It maintains a clock in the upper right hand corner of your screen all the time regardless of what programs you are running. This program uses the clock interrupt vector as set up by the Heath/Zenith version of CP/M-86, and may not work with other versions. SCRNCLK.RED produces a red clock, SCRNCLK.WHT produces a white clock (rename one of them to SCRNCLK.CMD). The source file, SCRNCLK.A86, may be edited and re-assembled to produce other colors.

CLOCK -- This is a program for turning off and on the screen clock after SCRNCLK is loaded.

TIME -- This program works like the TOD program supplied with CP/M-86 except that it sets only the time, not the date. It is easier to use than TOD when you are not concerned with the date.

DATETIME -- This program maintains the latest time and date in a file called DATETIME.TMP. If you CONFIGURe your CP/M to run DATETIME on cold boot, the date and time will automatically be set to the values that were in effect the last time you ran DATETIME, even if the computer was turned off.

DIR86 -- This is a CP/M-86 version of the directory program previously released as DIR19 for 8-bit CP/M and DIR100 for Z-DOS. It lists the files on your disk in an easy-to-read format using graphic lines to separate columns of files. Files can be alphabetized or listed in their "natural" order. DIR86 can list up to 80 files on the screen at a time, and if there are more than 80 files, it will pause for you to hit RETURN to see more files. DIR86 can handle up to 1024 files.

PDIR -- This program is like DIR86 except that its output goes to a printer. PDIR25 is for H/Z-25 and H/Z-125 printers, and uses graphic lines to separate file columns. PDIR is for other printers.

DISKID -- This program allows you to assign a volume number and label to a CP/M disk that can be read by DIR86 or PDIR. If the volume number and label are present, they are displayed at the top of the screen or page by DIR86 or PDIR.

SEE -- This program is a fancy replacement for the CP/M TYPE command. It allows you to scroll forward or backward through a file, to search for strings within the file, and to print screens of the file to a printer. It can also scroll sideways to allow you to view files up to 132 characters wide.

SCR -- This is a textual (not graphic) screen dump program. While it is loaded, anything on the screen is dumped to your printer when you type Control-D. SCR becomes part of the system when it is loaded so that any program can be run, and screens of it "dumped". SCR requires the Heath/Zenith version of CP/M-86, and is not compatible with KEYMAP (885-5001-37), which must be unloaded before SCR can be loaded.

UNSCR -- This program unloads SCR from your system.

ONECOPY -- This is a single drive file copy utility that is useful in situations where you have one bad drive in a two drive system, and the operating system "thinks" that you still have two drives.

COLOR -- This program lets you set the foreground and background color on your screen. It is a translation of the Z-DOS COLOR program by Robert Metz.

PSETMX -- This program lets you set up the print format on your Epson MX-80 or compatible printer prior to use. You can set Normal, Double strike, or Emphasized printing, Normal or Compressed width, and the lines per inch setting.

PSET25 -- This program lets you set up the print format on your H/Z-25 or H/Z-125 printer prior to use. You can set the character spacing and line spacing.

SWAP -- This is a CP/M-86 version of the SWAP program published in REMark magazine, which allows you to use an external terminal on your computer, and switch back and forth between the normal keyboard/screen and the other terminal. This program may run on a Z-150, but has not been tested on one.

IBM.CHR -- This is a replacement for your ALTCHAR.SYS file that provides an IBM-like character font. The characters are bolder, and some may find them easier to read, especially on color monitors. The normal H/Z-19 graphic characters are supported.

ASM.SUB -- This is a SUBMIT file that lets you assemble a program from start to .CMD file with one command line. Parameters for GENCMD are entered after the file name, as in

```
A>SUBMIT ASM FILE DATA[X400]
```

TABLE C Rating: (0-4),(10)

885-3013-37 Z-DOS Checkbook Manager $20.00

Introduction: Checkbook Manager is a ZBASIC program that has the capability of displaying your checkbook register with as many as nine entries at a time. Items can be individually entered and/or edited, and after entering transactions, checks are printed in a format suitable for window mailing envelopes. There is a reconciliation routine which is used at the end of the month which marks cleared transactions, inputs bank statement data (ending balance, bank charges, etc.), then prints a reconciliation statement.

Checkbook Manager is easier to use than other accountant programs because of the on-screen display of multiple transactions, the ability to search and edit individual entries, and the use of H/Z-100 graphics to display the register.

Requirements: This program requires an H/Z-100, color monochrome, 128K of RAM, ZBASIC, and a printer if check printing is desired.

The following files are included on the HUG P/N 885-3013-37 Checkbook Manager Disk:

```
README    .DOC
AUTOEXEC .DOC
CM        .BAS
CPRINTER  .BAS
```

Author: Stephen Ruks

Program Content: With Checkbook Manager, you can essentially discard your checkbook ledger. All normal checkbook functions are performed by this program. The following is the main menu extracted directly from Checkbook Manager:

1. Enter Transactions In Checkbook
2. Search For Transactions
3. Edit Transactions
4. Print Checks
5. Reconcile Bank Statement
6. Start New Disk File
7. Change Account Disk
8. Reconfigure (Color/Mono)
9. End Program

The SEARCH and EDIT features are particularly useful. Transactions can be searched for by check number, date, payee, or amount. Once the proper transaction is found, you are then given the option of editing it. The balance is calculated after each entry is made thus maintaining the correct figures on screen at all times.

Comment: The program is self-prompting and no problems encountered the first time through.

TABLE C Rating: (0),(9)

CORRECTION

Last month, in the June Issue of REMark, the part number for ZDOS FAST EDDY Text Editor and BIG EDDY was incorrect. The correct part number is 885-8029-37. The part number that was printed was for FAST EDDY CP/M-85/86. If you have ordered and received the wrong part, return it to Heath Users' Group, Hilltop Road, St. Joseph, Mi 49085, Attn: Nancy Strunk, and the correct version (ZDOS) will be sent to you. We are sorry for any inconvenience this may have caused.

HUG Price List

The following HUG Price List contains a list of all products not included in the HUG Software Catalog. For a detailed abstract of these products, refer to the issue of REMark specified.

Part Number	Decription of Product	Selling Price	Volume - Issue
HDOS			
885-1030[-37]	Disk III, Games II	$ 18.00	5-2
885-1096[-37]	MBASIC Action Games	$ 20.00	5-2
885-8026	Space Drop	$ 16.00	5-2
885-8027	HDOS SCICALC	$ 20.00	5-3
CP/M			
885-1234[-37]	CP/M Ham Help	$ 16.00	5-2
885-5001-37	CP/M-86 KEYMAP	$ 20.00	5-4
885-5002-37	CP/M-86 HUG Editor	$ 20.00	5-5
885-8025-37	CP/M 85/86 FAST EDDY	$ 20.00	5-2
ZDOS			
885-3009-37	ZBASIC Dungeons & Dragons	$ 20.00	5-3
885-3010-37	ZDOS KEYMAP	$ 20.00	5-4
885-3011-37	ZDOS ZBASIC Games Disk	$ 20.00	5-5
885-3012-37	ZDOS HUG Editor	$ 20.00	5-5
885-8028-37	ZDOS SCICALC	$ 20.00	5-2
885-8029-37	ZDOS FAST EDDY	$ 20.00	5-6
MISCELLANEOUS			
885-0004	HUG 3-Ring Binder	$ 5.75	
885-4001	REMark Volume 1, Issues 1-13	$ 20.00	
885-4002	REMark Volume 2, Issues 14-23	$ 20.00	
885-4003	REMark Volume 3, Issues 24-35	$ 20.00	
885-4004	REMark Volume 4, Issues 36-47	$ 20.00	
885-4700	HUG Bulletin Board Handbook	$ 5.00	5-2

NOTE: 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.

IT'S IMPORTANT!!

International Heath/Zenith Users' Group Conference
Heath/Zenith Users' Group
Saint Charles, Illinois • July 27, 28 & 29, 1984

Setting Time On The Watzman/ HUG ROM

John F. Smith
Ford Aerospace
300 Welsh Road
Horsham, PA 19044

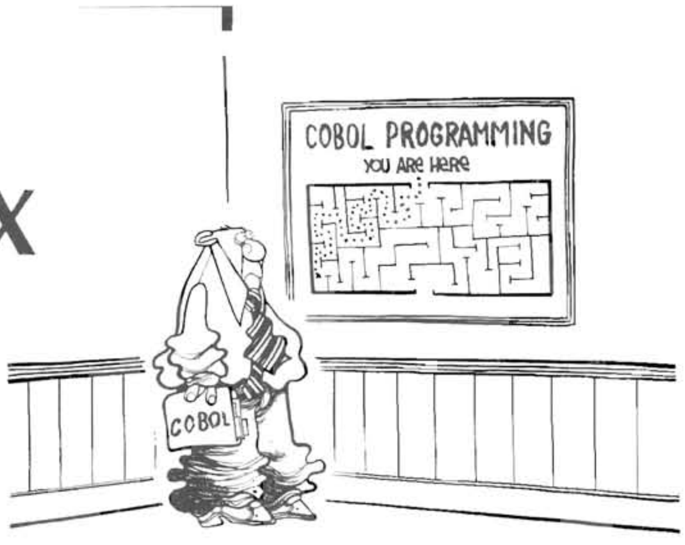
The Watzman/HUG ROM set (HUG P/N 885-4600) adds some really great features to the H/Z-89,90 or the H/Z-19 terminal. One of the most useful is the real time clock displayed on the 25th line. The clock can be enabled, disabled, set or read (into a program by sending the appropriate software terminal escape sequences.

In the H/Z-89 or -90, the clock may be set to the current time by going "off line" and sending the sequence "<ESCAPE> X hh:mm:ss<RETURN>", where "hh:mm:ss" are digits representing the time in hours, minutes and seconds. The colons between the digits are optional and may be omitted. The clock may be set from within an MBASIC program with the line: PRINT CHR$(27) "Xhhmmss".

The following assembly language listing is another alternative -- a simple way to set the screen clock from a directly executable CP/M .COM file. It may be used in a SUBMIT batch command or, as I use it, called from the Automatic Command Line on a CP/M cold boot. The program returns to CP/M without a warm boot, so it may be called from within another program, e.g. by using the "R" command from WordStar.

```
;       TIME - A Program to Set the H-89 Screen Clock
;              With the Watzman/HUG ROM
;
;       **********************************
;       *                                *
;       *    By J.F.Smith, Cairo, Egypt  *
;       *          04-Mar-84             *
;       *                                *
;       **********************************
;
;
;
;               *** CP/M Addresses ***
;
BOOT    EQU     0000H           ;Warm-start exit point
BDOS    EQU     0005H           ;Service request entry
TPA     EQU     0100H           ;Standard program origin
;
;               *** Miscellaneous Equates ***
;
LF      EQU     0AH             ;Linefeed
CR      EQU     0DH             ;Carriage return
ESC     EQU     1BH             ;Escape
;
;               *** CP/M Function Calls ***
;
PSTRING EQU     9               ;Print string at console
CONBUFF EQU     10              ;Read console buffer
;
        ORG     TPA             ;CP/M programs start here
;
;               *** Main Program ***
;
MAIN    LXI     H,0             ;Zero HL Register
        DAD     SP              ;Add stack pointer value
        LXI     SP,STACK        ;Set our stack pointer
        PUSH    H               ;Save CP/M's stack on ours
        LXI     D,PROMPT        ;Point to prompt message
        MVI     C,PSTRING       ;Get console print function
        CALL    BDOS            ;Print prompt at console
        LXI     D,TIME          ;Point to where time input goes
        MVI     C,CONBUFF       ;Get console line input function
        CALL    BDOS            ;Input the time
        LXI     H,TIME+1        ;Point to the number of characters typed
        MOV     E,M             ;Get the count
        MVI     D,0             ;DE = count
        INX     H               ;Move to the start of the string
        DAD     D               ;Add a count to get to the end
        MVI     M,CR            ;Insert a CR
        INX     H               ;Move to the next location
        MVI     M,LF            ;Insert a line feed
        INX     H               ;Move to the next location
        MVI     M,'$'           ;Insert a string terminator
        LXI     D,TIMESET       ;Point to time setting string
        MVI     C,PSTRING       ;Get console print function
        CALL    BDOS            ;Send time code to console
        LXI     D,TIME+2        ;Point to time
        MVI     C,PSTRING       ;Get console print function
        CALL    BDOS            ;Send time
;
;               *** Close and Return to CP/M ***
;
        POP     H               ;Restore CP/M'S stack
        SPHL                    ;Set it
        RET                     ;Return to CP/M
;
;               *** String Storage ***
;
PROMPT  DB      CR,LF,'Please enter the time (HHMMSS):$'
TIMESET DB      ESC,'X','$'
;
TIME    DB      6,0             ;Maximum allowed characters, count
        DS      9               ;Allow 6 characters + LF + CR + '$'
;
;               *** Our Stack ***
;
        DS      32
STACK   EQU     $
        END     START
```

COBOL Corner IX



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

Introduction

How did you do with Program #3? I hope you have the program documented, designed, CODED and KEYED-IN! If you did the "homework", you will find it easy to follow along as I explain the "NEW" programming that was introduced.

Data Division -- File Section

Our data-description PICTURE clause for the FILEL3.DAT Purchase Amount shows a "V" to represent the assumed decimal point. If you print out FILEL3.DAT, you will not find a decimal point in any of the fields. The decimal point is assumed and not actually there. Data, such as a decimal point and other characters are not put in data files as they take extra typing, file space, and would serve no useful purpose. In fact, they would cause us trouble when we get to advanced programs where we will edit the data file before we process it! More on that subject in future articles.

```
05  OF-PURCHASE-AMT                        PIC 9(07)V99.
```

The "V" does not count towards the size of the PICTURE clause!

Next, the Output Data-Description of SALES-REPORT includes three (3) 01 Level Entries:

```
01  SR-SALES-REPORT.
01  TL-TOTAL-LINE-1.
01  AT-TOTAL-LINE-2.
```

Do you like these Names? I do not! I put them in to get your reaction and to get you thinking about what is a good, self-documenting Name! With COBOL you can use any Name you like (if it meets the few rules specified in an earlier article). I would suggest the following for starters:

```
01  SR-SALES-RPT.
01  TL-TOT-LINE-1.
01  TL-TOT-LINE-2.
```

Do you like these? You may think of better ones! We will be using abbreviations more in future programs to save CODE line length. Remember abbreviations are good, provided that anyone that may use the program now, or in the future, would understand what they stand for! I do not usually like to abbreviate the KEY word in a Name--SALES. I used AMT for AMOUNT. How about TRANS for TRANSACTIONS? You do not need to use the same name that I use. Just be sure that the Name is self-documenting!

Now look at the NUMERIC EDITED PICTURE clauses. Remember that Input fields and WORKING-STORAGE fields must be NUMERIC, not NUMERIC EDITED, because we usually use the numbers in calculations! The Output fields (receiving fields) can and usually are NUMERIC EDITED because we do not use them in calculations! Let's look at a NUMERIC EDITED field:

```
PIC Z,ZZZ,ZZ9.99.
```

What can we tell from this NUMERIC EDITED field? The "Z" will cause zero suppression with BLANK replacement and it is used to suppress non-signficant zeros. The comma, ",", is used to aid in the readability of the number; but, the comma cannot be used as a "rightmost" character. The decimal point, ".", prints the decimal point unconditionally when it appears in a PICTURE clause. Each "9" represents one decimal digit. The "9" prints either the Number or a Zero unconditionally. Every "Z", ",", "." and "9" count towards the size of the PICTURE clause. A PICTURE clause may have UP TO 30 characters and the symbols "P", "V" and "S" do not count. A maximum of 18 "9" and "Z" characters may be used in each clause. Here are a few examples of results using the NUMERIC EDITED PICTURE clause shown above:

```
1. Input = 123456789
   Output = 1,234,567.89
2. Input = 987321
   Output =       9,873.21
```

Let's look at a couple of other NUMERIC EDITED PICTURE clause examples:

```
1. PIC 9,999.99.            Input = 654321
                            Output = 6,543.21
                       OR
                            Input = 4321
                            Output = 0,043.21

2. PIC ZZZ9.                Input = 4321
                            Output = 4321
                       OR
                            Input = 321
                            Output =  321
```

A few PICTURE clause RULES follow:

1. A PICTURE clause is required for elementary items.

2. A PICTURE clause cannot be used for a group item.

3. The PICTURE clause tells how many and the type of characters that can be stored in that field.

4. The character "S", "V" and "." can appear only once in a PICTURE clause.

Now, a few remarks about the three (3) 01 Level Entries:

1. There are better and more efficient ways to handle total lines as we will see in future programs. However, I felt you should know and use the several ways.

2. More on Name abbreviations:

a. How about NBR in place of NO for NUMBER? I like NBR!

b. Wouldn't AVG be as good as AVERAGE?

c. Why use TL-ASTERISK-1, -2 AND -3? They could all be Named TL-ASTERISK because they will all receive the same Literal, "*"! So, TL-ASTERISK could be used for the Name for all three.

d. What do you think of the use of the prefix AT? It leaves me cold! We sure could do better! What if we decided on "TL1" and "TL2" for the prefixes and LINE-1 and LINE-2? Which do you like? I want you to decide on the Name.

Working-Storage Section

Remember that a few COBOL Corner articles ago we said that this section could have "100 Lines of Code"? Program #3 does not have that many but it does show additional use of this section. The WORKING-STORAGE SECTION sets up memory storage space for fields, that are not part of the Input or Output fields, but none-the-less required for program processing. These include Program Constants, Work Areas, Hold Areas, etc.

Program #3 needs a temporary Work Area for storing the calculated Sales Tax:

```
05  WS-WORK-SALES-TAX                   PIC S9(06)V99.
```

This PICTURE clause has the symbol "S". This represents the Operational Sign. It tells the computer whether the number is positive or negative. There can only be one "S" per clause and it does not count towards the size of the clause. Signed numbers are used in calculations. Unsigned numbers are used for numbers describing street numbers, telephone numbers, etc.

Program #3 sets up a Hold Area for storing Accumulator values:

```
01  WS-TOTAL-ACCUMS.
    05  WS-TOT-NBR-TRANS                    PIC S9(04).
    05  WS-TOT-PURCHASE-AMT                 PIC S9(08)V99.
    05  WS-TOT-SALES-TAX-AMT                PIC S9(07)V99.
    05  WS-TOT-TRANS-AMT                    PIC S9(08)V99.
```

Again, we need a memory storage place for holding the calculated results that we are putting into the accumulators. Notice that I have added some ideas for abbreviations in the Names. Do you like them? Do you have better ones? Also, note that the PICTURE clauses are NUMERIC (not NUMERIC EDITED) because the numbers in WORKING-STORAGE are usually used in calculations!

Procedure Division

Note: We have added a new MODULE in this program! I called it "INITIALIZE-VARIABLE-FIELDS". It is "good" programming to use a new MODULE if you are going to do several related processes one or more times in a program. In this case it is a borderline decision, but I wanted to get you started with the use of MODULES.

First, we put our "switch" as we have used in previous programs in this MODULE, it needs to be initialized to "NO " at the start of processing:

```
MOVE "NO "                          TO WS-END-OF-FILE-SW.
```

I have abbreviated "switch" to "SW". Would that Name be self-documenting to you?

Second, we must set our four (4) accumulators to zero before we start using them. Notice that I have used a GROUP MOVE, which allows us to initialize all of the accumulators to zero with one MOVE statement. This is a handy thing to remember! I have used a "figurative constant" called ZERO or ZEROS as described in a previous article:

```
MOVE ZEROS                          TO WS-TOTAL-ACCUMS.
```

I wonder if you are curious why we have used the read FILEL3 twice in the program design? If you made the "flowchart" as part of your program documentation, you will see the reason! We have a "conditional loop logic" in our program design. We READ FILEL3 and check for END-OF-FILE. If not EOF, we go to the "PROCESS-SALES-REPORT" Module, where we continue processing until we find EOF. If we did not READ FILEL3 once before processing, we would have to have a READ FILEL3 statement as the first statement in the "PROCESS-SALES-REPORT" MODULE. If we did that we would have a conflict of conditions testing for EOF, and a program ERROR! If you did not do your "homework" and do not have your "flowchart", make one now! It will clarify this point that I could not make clear with 10,000 words.

Process-Sales-Report Module

Our program design has set up a memory storage area labeled, "SR-SALES-REPORT", per our data-description specified size of 132 characters. However, we do not know what kind of data (garbage) the computer might have in this memory area. Thus, we want to "clear" this area. We do this by putting SPACES (figurative constant) into the area:

```
MOVE SPACES                         TO SR-SALES-RPT.
```

Remember, always "clear" a memory area of "garbage" before using it! Always do it at the beginning! If you do it later, you may have already put data into the area and would then lose the data when you "clear" the area.

Next, we start moving our data into the report as we have done in previous programs. However, we do not read all the data from FILEL3 that we need! We must calculate some of the data. The order that we do this in is not important, except that for "good" programming we would like to keep the sequence in a logical and readable order.

The COBOL verb MULTIPLY will be needed first. Its format is:

```
         {data-name-1       }
MULTIPLY
         {numeric-literal-1}

      {data-name-2       }
   BY
      {numeric-literal-2}

        [GIVING data-name-3] [ROUNDED] [SIZE-ERROR-clause].
```

The MULTIPLY statement multiplies two (2) numeric data items and stores the product. It has three (3) options:

1. GIVING option. If used, puts the calculated result of the arithmetic operation into the data-name specified after the word GIVING. This data-name is not used for further calculations and therefore may be a NUMERIC EDITED item.

2. ROUNDED option. When specified, COBOL makes the calculation and carries it out one more decimal position than actually specified in the PICTURE clause of the result field. Then, if the extra decimal position is 5 or greater, the answer is rounded up; if it is 4 or less, the answer is not changed. The extra decimal position is then no longer used and it is truncated. Unless there is a specific reason for not rounding, the ROUNDED option should generally be used whenever the result of the calculation may contain more decimal positions than have been specified in the PICTURE clause.

3. SIZE-ERROR option. This will not be dealt with at this time.

When the GIVING option is omitted, the second operand must be a data-name and the product replaces the value of data-name-2. I would recommend using the GIVING option for better program readability.

Now looking at Program #3 statement:

```
        MULTIPLY OF-PURCHASE-AMT
            BY .065
                GIVING WS-TAX-AMT-WRK.
```

Notice that each verb is on a separate CODING line and that there is no period until the end of the statement! The tax rate of 6.5% is used in decimal form -- .065, not 0.065! The result is put into the WS-TAX-AMT-WRk Work Area so we may use it in other calculations. Remember it must be NUMERIC, if we will use it for further calculations.

Next, we have a MOVE statement:

```
        MOVE WS-TAX-AMT-WRK                 TO SR-SALES-TAX-AMT.
```

Here we are moving the NUMERIC sending field , WS-TAX-AMT-WRK, to the NUMERIC EDITED receiving field, SR-SALES-TAX-AMT. The editing is always performed according to the receiving PICTURE clause. Great Caution should be taken when moving data from a sending field to a receiving field. The receiving field must be the same size or larger than the sending field or you will be greeted with unpleasant results!

The COBOL verb ADD is the next one we will need. The ADD statement adds two (2) or more numeric values and stores the resulting SUM. Its general Format is:

```
     {numeric-literal-1} [numeric-literal-2] ...
ADD
     {data-name-1       } [data-name-2      ] ...

   {TO     }
             data-name-n [ROUNDED] [SIZE-ERROR-clause].
   {GIVING}
```

The three (3) options Giving, Rounded, and Size-Error described above, also apply to the ADD statement. When the TO option is used, the value of all the data-names (including data-name-n) and literals are ADDED, and the resulting SUM replaces the value of data-name-n. This is the usual Format used for accumulators. When the Giving option is specified, at least two (2) data-names and/or literals must be coded between ADD and GIVING. The SUM of the values of these data-names and literals (not including data- name-n) repiaces the value of data-name-n.

We will now look at a Program #3 ADD statement:

```
        ADD OF-PURCHASE-AMT WS-TAX-AMT-WRK
            GIVING SR-TRANS-AMT.
```

This statement will add OF-PURCHASE-AMT and WS-TAX-AMT-WRK (both are NUMERIC only) and place the SUM into SR-TRANS-AMT (NUMERIC EDITED). Any previous value in SR-TRANS-AMT would be "cleared" first. The values stored in OF-PURCHASE-AMT and WS-TAX-AMT-WRK would be unchanged.

We now have all the data for the SR-SALES-REPORT line, so we will DISPLAY the line on the CRT and WRITE the line to the printer with double-spacing. This processing could be made into another MODULE if we desired. We will do this in a future program.

Next, we must calculate the new SUM to put into each of the accumulators. These calculations could also be put in a separate MODULE. More on this in later programs. The four calculations in Program #3 are:

```
1. ADD 1                             TO WS-TOT-NBR-TRANS.
```

Where "1" is a numeric literal being added to the value in WS-TOT-NBR-TRANS and the new SUM replacing the value in WS-TOT-NBR- TRANS.

```
2. ADD OF-PURCHASE-AMT               TO WS-TOT-PURCHASE-AMT.
```

Here the NUMERIC field, OF-PURCHASE-AMT, is added to the NUMERIC field, WS-TOT-PURCHASE-AMT, and the SUM replaces the value in WS- TOT-PURCHASE-AMT. This is the accumulator "form" of the ADD statement that is usually used.

```
3. ADD WS-TAX-AMT-WRK                TO WS-TOT-SALES-TAX-AMT.
```

Note that we need the Work Area field, WS-TAX-AMT-WRK, in NUMERIC to add to the accumulator NUMERIC Hold Area field, WS-TOT-SALES- TAX-AMT. Again, the SUM replaces the value in WS-TOT-SALES-TAX- AMT.

```
4. ADD OF-PURCHASE-AMT WS-TAX-AMT-WRK
                                     TO WS-TOT-TRANS-AMT.
```

This is an example of adding three (3) numeric values to each other and the resulting SUM replaces the value in WS-TOT-TRANS- AMT.

We have now completed processing a SR-SALES-REPORT line. The program will READ another Record from FILEL3 if it has not found EOF and repeat the process. If EOF is found, the program will do the next MOVE:

```
MOVE "YES"                           TO WS-END-OF-FILE-SW.
```

when the program finds that the condition specified in the PROCESS-SALES-REPORT UNTIL WS-END-OF-FILE-SW is EQUAL to "YES". So the program drops to the next CODING line.

PRINT-TOTAL-LINE-1 Module

We must "clear" the memory storage area, TL-TOT-LINE-1, of garbage, so:

```
MOVE SPACES                          TO TL-TOT-LINE-1.
```

Next, we MOVE an ALPHANUMERIC LITERAL (thus, it must be in quotes) to an ALPHANUMERIC receiving field. Remember that the receiving field must be the same size as the sending field. If the receiving field is shorter, the "rightmost" characters of the sending field will be truncated. If the receiving field is longer, the sending field will be left justified in the receiving field and the "rightmost" extra positions will be padded with SPACES.

```
MOVE "TOTAL TRANSACTIONS"            TO TL-TRANS-WORDS.
```

The next four (4) MOVE statements move the NUMERIC accumulator fields into their receiving fields. Remember that the decimal points in the sending fields and receiving fields will first be aligned. If the two (2) fields on each side of the decimal point are the same size, we will get the expected correct results. If the receiving field is shorter on either side of the decimal point, the excess characters will be truncated resulting in possible program ERRORS. If the receiving field is longer on either side of the decimal point, the excess positions will be padded with zeros or spaces depending on the receiving field PICTURE clause:

```
MOVE WS-TOT-NBR-TRANS              TO TL-TOT-NBR-TRANS.
MOVE WS-TOT-PURCHASE-AMT           TO TL-TOT-PURCHASE-AMT.
MOVE WS-TOT-SALES-TAX-AMT          TO TL-TOT-SALES-TAX-AMT.
MOVE WS-TOT-TRANS-AMT              TO TL-TOT-TRANS-AMT.
```

Next, the program will MOVE the ALPHANUMERIC LITERAL, "*", (thus it will be in quotes) to the three (3) ALPHANUMERIC fields named either of two ways described earlier in this article:

```
MOVE "*"                           TO TL-ASTERISK-1
                                      TL-ASTERISK-2
                                      TL-ASTERISK-3. (PERIOD HERE)

OR

MOVE "*"                           TO TL-ASTERISK.
```

Now, we have completed the first total line and will display the line on the CRT and write the line to the printer triple-spaced down from the Body Area of the SALES-REPORT. This completes this MODULE, so the program drops to the next CODING line.

PRINT-TOTAL-LINE-2 Module

Again, we will "clear" the memory storage area for TL-TOT-LINE-2:

```
MOVE SPACES                        TO TL-TOT-LINE-2.
```

Next, MOVE the ALPHANUMERIC LITERAL field:

```
MOVE "AVERAGE PURCHASE AMOUNT"  TO TL-PURCHASE-WORDS.
```

Now, we need another COBOL verb called DIVIDE. The DIVIDE statement divides two (2) numeric values and stores the quotient. The general Format is:

```
       {data-name-1       } {BY  } {data-name-2       }
DIVIDE
       {numeric-literal-1} {INTO} {numeric-literal-2}

   [GIVING data-name-3] [ROUNDED] [SIZE-ERROR-clause].
```

The three (3) options, Giving, Rounded, and Size-Error, are described above under MULTIPLY. They apply the same to DIVIDE. The BY option signifies that the first operand (data-name-1 or numeric-literal-1) is the dividend (numerator) and the second operand (data-name-2 or numeric-literal-2) is the divisor (denominator). If Giving option is not used, then the first operand must be a data-name, in which the quotient is stored. If GIVING option is used, the quotient is stored in data-name-3. The INTO option signifies that the first operand is the divisor and the second operand is the dividend. If Giving option is not used, then the second operand must be a data-name, in which the quotient is stored. When using the Giving option, the quotient is stored in data-name-3.

Program #3 uses the DIVIDE statement to calculate an average using the BY option and the Giving option along with the Rounded option as follows:

```
       DIVIDE WS-TOT-PURCHASE-AMT
        BY WS-TOT-NBR-TRANS
              GIVING TL-AVG-PURCHASE-AMT ROUNDED.
```

Again, we DISPLAY the total line and WRITE the total line to the printer double-spaced down from the first total line. This completes this MODULE and the program drops down to the next CODING line.

Notes

1. Division by zero always causes a Run Time ERROR.

2. COBOL is an English language so we use the English words in place of arithmetic operators. We do have a COMPUTE arithmetic verb which usually uses the Operators. We will find a use for this verb in a later program.

COMPILE and LINK/EXECUTE Program #3

With the above explanations and your "homework" you should complete your program. Transfer (using PIP) FILEL3.DAT to your disk A from HUG COBOL Corner Disk-I. COMPILE and LINK/EXECUTE your Program #3. Hopefully, you have done your work carefully and obtained an ERROR-FREE COMPILE and the specified SALES REPORT print-out. If you can't (please try!), debug your program as we have in previous articles.

If you still cannot find your problems, transfer PRGM03.COB and FILEL3.DAT from your HUG Disk-I to a NEW disk as we did in previous articles. COMPILE and LINK/EXECUTE this program. Now you will have a correct RUN! Compare a LISTING of this disk with the LISTING from your Program #3 line-by-line and find your ERRORS. Use your EDITOR to correct your Program #3 PRGM03.COB source code. Now, COMPILE and LINK/EXECUTE again and you should now have a "good" RUN! Do you? Do you like the Report better with Total Lines?

Closing

The next COBOL Corner article will provide you with the information for Program #4 -- DISCOUNT-REPORT. I will expect you to do this program completely by yourself. I will only explain the COBOL verb called SUBTRACT. Are you ready?

Getting Started With Assembly Language

The CP/M Contest Winner

Pat Swayne
HUG Software Engineer

For those of you new to REMark, this article is actually the tenth in a series that started last year. The purpose of the series was to teach practical 8-bit assembly language, with emphasis on using the operating system. In the last article of the series (in the February issue), I announced a programming contest for those who had been following the series. Now that I have selected the winners of the contest, it is time to start the series again, and examine the winning programs.

There were actually two contests, one for HDOS (Heath Disk Operating System) users, and one for CP/M users. The response to the contests itself was interesting in that there were many more CP/M entries than HDOS entries. I know that there are still a lot of die-hard HDOS users out there in HUG land, but if the contest entries were any indication, CP/M is taking over the Heath 8-bit world, folks.

This is one of the reasons that I am presenting the winning CP/M program first. The other is that since CP/M is not as protective as HDOS, it is harder to write a goof-proof CP/M program than it is to write an HDOS one. In fact, the winning program was the only one to pass all of the tests I applied.

For those of you who missed the contest, it was to write a program RENM to rename files that used the following syntax

```
RENM d:oldname TO newname
```

where d: is a drive designation (optional), and oldname and newname are file names. I allowed CP/M entrants to leave out the TO, but most chose to put it in, and none of the programs that left it out could pass all of the tests, anyhow. There were four basic tests that I applied to the CP/M programs. The first was to just run the program without any arguments.

```
A>RENM
```

In this case, the program should have detected that there were no arguments, and should have indicated proper usage. Some programs went into an interactive mode at this point, which is OK as long as you tell the user what is going on. To prepare for the next test, I created a null directory by entering

```
A>SAVE 0 X.XXX
```

Then I entered

```
A>RENM X.XXX TO X.X
```

A large percentage of the programs failed here, because they could not properly handle extensions of less than 3 characters. The new directory entry created should have had two spaces after the last X, but many had zeros, carriage returns, and other illegal stuff. I examined the RENM-created directory entry after each test with a disk dump program to make sure it was correct. If the program got through this test, I made a new X.XXX entry with SAVE, and then entered

```
A>RENM X.X TO X.XXX
```

Here, the program should have detected that there already was an entry called X.XXX on the disk and alerted me some how. Some of the programs went ahead and performed the rename operation with the result that there were now two files called X.XXX on the disk, which is a real mess. If the program passed this test, I entered

```
A>RENM X.X TO X.*
```

The program should have detected the wild card symbol and not allowed the rename operation. If it did not detect the problem, an entry called X.* was created on the disk. That wouldn't be so bad except that when you try to erase such an entry (ERA X.*), you wind up erasing all files that start with X. So a rename utility should detect and not allow wild card symbols.

The listing following this article is the winning CP/M entry, submitted by John F. Smith whose mailing address is Horsham, PA, but who now lives in Cairo, Egypt. It allows the file names to be entered with or without the TO between them, and it passes all of the tests described above.

The program starts with a section defining the constants that will be used in it. At the opening, it saves the CP/M stack and sets its own so that it can later return to CP/M without warm booting. Then it tests for at least two argument words, and prints an error message if none are present. It tests for arguments by looking at the default FCB areas, which is one of the two methods discussed in this series.

If the arguments are found, the program tests to see if the second one is "TO ", and if not, the program assumes two file names without a TO between. The only drawback to this is that you could not rename

☞

```
;       *** RENM.ASM - Program to RENaMe CP/M Files ***
;                     using the syntax:
;
;                     RENM d:OLDFILE.NAM NEWFILE.NAM
;                                  or
;                     RENM d:OLDFILE.NAM TO NEWFILE.NAM
;
;
;               **********************************
;               *                                *
;               *  By John F. Smith, Cairo, Egypt *
;               *         08-March-1984          *
;               *                                *
;               **********************************
;
;
;               *** Version 1.1, 11-March-84 ***
;
;
;               *** Important addresses ***
;
BDOS    EQU     0005H           ;Service request entry
FCB1    EQU     005CH           ;Default file control buffer
FCB2    EQU     006CH           ;Second file control buffer
TPA     EQU     0100H           ;Standard program origin
DMA     EQU     0080H           ;Default I/O buffer
;
;               ***  ASCII character set names ***
;
LF      EQU     0AH             ;Linefeed
CR      EQU     0DH             ;Carriage return
BLANK   EQU     20H             ;Blank
;
;               *** CP/M function calls ***
;
PSTRING EQU     9               ;Print string at console
FIND    EQU     17              ;Find directory entry
RENAME  EQU     23              ;Rename file
;
;
        ORG     TPA
;
;               *** Save CP/M stack and set ours ***
;
START   LXI     H,0
        DAD     SP              ;Locate stack
        LXI     SP,STACK        ;Set new one
        PUSH    H               ;Save old one
;
;               *** Check syntax ***
;
        LDA     FCB1+1          ;Look at first char in oldfile name
        CPI     ' '             ;Is there an entry?
        JZ      SYNERR          ;If not, tell the operator
        LDA     FCB2+1          ;Now look at first char in new name
        CPI     ' '
        JZ      SYNERR          ;No entry, print message
;
```

```
        JNZ     MOVEXT1         ;until counter = 0
FILLEXT MVI     A,BLANK
        STAX    D               ;Pad with blanks
        INX     D
        DCR     B
        JNZ     FILLEXT         ;Until end of FCB2
;
;               *** Make new file name driveref = old driveref ***

DRIVE   LXI     H,FCB1          ;Point to driveref in old file
        LXI     D,FCB2          ;Point to driveref in new file
        MOV     A,M
        STAX    D               ;Move it
;
;               *** Check for existing file = NEWFILE.NAM ***
;
        LXI     D,FCB2          ;Point to new file name in FCB
        MVI     C,FIND          ;Call CP/M directory search function
        CALL    BDOS            ;Search
        CPI     0FFH            ;Does file exist?
        JNZ     FILXSTS         ;If so, tell the operator
;
;               *** Check for ambiguous filerefs ***
;
        LXI     H,FCB1          ;Point to start of FCB
        MVI     B,32            ;Set up counter = FCB1+2
AGAIN   MOV     A,M             ;Get a char
        CPI     '*'             ;Asterisk?
        JZ      AMBERR          ;Ambiguous, not permitted
        CPI     '?'             ;?
        JZ      AMBERR          ;Ambiguous, not permitted
        INX     H               ;Move pointer
        DCR     B               ;Decrement counter
        JNZ     AGAIN           ;until = 0
;
;               *** Rename the file ***
;
        LXI     D,FCB1          ;Point to the old file name
        MVI     C,RENAME        ;Call CP/M rename function
        CALL    BDOS            ;Try to rename it
        CPI     0FFH            ;Does OLDFILE.NAM exist?
        JZ      FNFND           ;If not, tell the operator, else
        LXI     D,SUCCESS       ;Point to "Successful Rename" string
        MVI     C,PSTRING       ;Call CP/M console string function
        CALL    BDOS            ;Print it at the console
;
;               *** Restore stack and return to CP/M ***
;
EXIT    POP     H               ;Get old stack
        SPHL                    ;Set it
        RET                     ;Return to CP/M
;
;               *** Error routines ***
;
SYNERR  LXI     D,ERMSG1        ;Point to syntax error string
        MVI     C,PSTRING       ;Call CP/M console string function
        CALL    BDOS            ;Print it at the console
        JMP     EXIT            ;and return to CP/M
```

```
;
;               *** Find if second command form used ***
;
        CPI     'T'             ;See if "TO " command syntax used
        JNZ     DRIVE           ;No, proceed
        LDA     FCB2+2          ;Maybe, look at next char
        CPI     'O'             ;Is it "O"?
        JNZ     DRIVE           ;No, proceed
        LDA     FCB2+3          ;Maybe, look at next char
        CPI     ' '             ;Space?
        JNZ     DRIVE           ;No, it's the first form
;
;               *** Find NEWFILE name in command buffer ***
;
        LXI     H,DMA+1         ;Point to command line in buffer
        CALL    SOS             ;Call skip on space routine
FNS     MOV     A,M             ;Get next char
        CPI     ' '             ;Space?
        JZ      FNDTO           ;Go on
        INX     H               ;Else, increment pointer
        JMP     FNS             ;and look for next
FNDTO   CALL    SOS             ;Look for "TO "
        INX     H
        INX     H               ;Skip past it
        CALL    SOS             ;to start of NEWFILE name
;
;               *** Move new file name to FCB2 ***
;
        LXI     D,FCB2+1        ;Point to where file name goes
        MVI     B,11            ;Set counter for length of FCB2
MOVNAM  MOV     A,M             ;Get next char
        CPI     '.'             ;Look for start of .EXT if present
        JZ      MOVEXT          ;If so, go to last 3 char in FCB2
        CPI     ':'             ;Look for drive designator
        JZ      SYNERR          ;Not permitted in "TO" file
        CPI     00H             ;Look for end of command line
        JZ      FILL            ;If encountered before ".", no .EXT
        STAX    D               ;If not, move character to FCB
        INX     H               ;Move pointer in command line
        INX     D               ;Move pointer in FCB2
        DCR     B               ;Count down
        JNZ     MOVNAM          ;Move another character
FILL    MVI     A,BLANK         ;Put a blank in A
        STAX    D               ;Move it to FCB2
        INX     D
        DCR     B               ;Count down
        JNZ     FILL            ;Do it until 11 char moved
MOVEXT  MVI     A,BLANK         ;Replace last char with a blank
        STAX    D               ;and put it in the FCB
        INX     H
        LXI     D,FCB2+9        ;Point to location of .EXT in FCB
        MVI     B,3             ;Set up counter = 3 char
MOVEXT1 MOV     A,M
        CPI     00H             ;If .EXT truncated to < 3 char
        JZ      FILLEXT         ;Pad it with blank(s)
        STAX    D               ;Move .EXT
        INX     H
        INX     D               ;Advance both pointers
        DCR     B
```

```
FILXSTS LXI     D,ERMSG2        ;Point to "file exists" message
        MVI     C,PSTRING
        CALL    BDOS            ;and print it
        JMP     EXIT            ;Return to CP/M
FNFND   LXI     D,ERMSG3        ;Point to "file not found" string
        MVI     C,PSTRING
        CALL    BDOS            ;and print it
        JMP     EXIT            ;and return to CP/M
AMBERR  LXI     D,ERMSG4        ;Point to ambiguous fileref string
        MVI     C,PSTRING
        CALL    BDOS            ;Print it
        JMP     EXIT            ;and return to CP/M
;
;               *** Skip over spaces ***
;
SOS     MOV     A,M             ;Get a character
        CPI     ' '             ;Is it a space?
        RNZ                     ;If not, return
        INX     H               ;else, move to next character
        JMP     SOS             ;Do it again
;
;               *** Message strings ***
;
ERMSG1  DB      CR,LF,LF,'The correct syntax for this command is:',CR
        DB      LF,LF,'    RENM d:OLDFILE.NAM NEWFILE.NAM',CR,LF,LF
        DB      '              or',CR,LF,LF
        DB      '    RENM d:OLDFILE.NAM TO NEWFILE.NAM',CR,LF,LF
        DB      '    where d: is an optional drive reference.',CR,LF,'$'
ERMSG2  DB      CR,LF,'New file name already exists.',CR,LF
        DB      'You must ERAse it before renaming.',CR,LF,'$'
ERMSG3  DB      CR,LF,'File to be renamed not found.',CR,LF,'$'
ERMSG4  DB      CR,LF,'Ambiguous file references (* or ?)',CR,LF
        DB      'are not permitted.',CR,LF,'$'
SUCCESS DB      CR,LF,'File renamed.',CR,LF,'$'
;
;               *** Data area ***
;
        DS      32
STACK   EQU     $
        END     START
```

Vectored to 49 ☞

Controlled Data Recording Systems Inc.

ANNOUNCING THE FDC-H8

DOUBLE DENSITY 8" AND 5.25" CONTROLLER FOR THE H8 COMPUTER

Has all of the capabilities of our popular FDC-880H controller, with the added features of;

- Direct memory access (DMA) data transfer.
- Hard sectored controller (H17) incorporated on the board.
- Runs with the standard 8080 CPU card and with Z80 CPU upgrades.
- Accesses both hard sectored disk formats and soft sectored disk formats through the same drives attatched to the FDC-H8 without hardware additions. **Price $495.00**

NEW PRODUCTS FOR THE FDC-880H

DM-1 DUAL BOARD MODIFICATION KIT **$29.95**
Allows for both the FDC-880H and the H88-4 controller cards to interface with the same 5.25" drives. Drives will run as both hard sectored format and soft sectored format depending upon the logical drive letter.

CDR BIOS by Livingston Logic Labs **$60.00**
Enhanced version of Heath/Zenith CP/M 2.203 BIOS with ZCPR. Supports all Heath/Zenith disk formats through the FDC-880H and the H17 controllers.

CDR DVD by Livingston Logic Labs. **$40.00**
HDOS driver for running double density HDOS through the FDC-880H

Shugart Slimline 5.25" 40 track double sided drives **$275.00**

Shugart Slimline 8" double sided drives **$525.00**

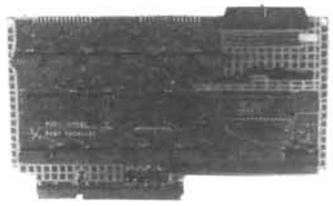
Contact: **C. D. R. Systems Inc.**
7210 Clairemont Mesa Blvd., San Diego CA 92111
Telephone: (619) 560-1272

5-20 day delivery–pay by check, C.O.D., Visa, or M/C

H/Z89 PERIPHERALS from SECURED COMPUTER SYSTEMS

2/3 PORT SERIAL CARD I/O PORT PARALLEL

" . . . not your typical vanilla-flavored serial and parallel interface . . . "



Features:

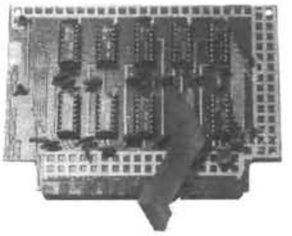
Chip independent design • Reduces computer data buss loading from 3 to 1 • Choice of Centronics or Epson parallel drivers for HDOS or CP/M • Complete documentation and installation instruction.

- 2 Serial Ports
- Supports: Ring Input, External Clock, Auto Dialer
- 3 Port Parallel with 2 Level Interrupt Control
- Fully compatible with all models of H/Z 88, 89, 90 using CP/M or HDOS.
- Fully tested, 90 day warranty, two serial cables and a parallel cable (internal to computer) and software driver.

PRICE $199.00
Shipping & Handling $10.00

16K RAM EXPANSION CARD

Expands your H/Z89 RAM Memory capacity to a FULL 64K!



Fully compatible with:
H/Z 89 • H/Z 88 • Magnolia Microsystems CP/M and disk drive I/O interface cards

NOW INCLUDING SUPPORT MOUNTING BRACKET

Featuring:
Complete installation instruction • 90 day Warranty
Field reliability record now entering its 21st month

Now Only $65.00

HDOS is a registered trademark of Heath Company
CP/M is a registered trademark of Digital Research

Shipping & Handling $5.00

PRICES ARE LESS SHIPPING AND TAX IF RESIDENT OF CALIFORNIA
MAIL ORDER: 12011 ACLARE, CERRITOS, CA 90701 (213) 924-6741
TECHNICAL INFO/HELP:
8575 KNOTT AVE., SUITE D, BUENA PARK, CA 90620 (714) 952-3930
Terms and specifications subject to change without notice.

ZENITH data systems
SERVICE CENTER

IBM PC Compatibility ... Real or Imagined?

William M. Adney
P. O. Box 13186
Arlington, TX 76094-0186

IBM PC compatibility ... what is it and why has it become such a hot issue? As far as I'm concerned, computer compatibility is one of those things that you don't talk about in mixed company. It's like sex, politics, and religion ... the subjects tend to be very controversial. The H/Z-100 computer compatibility with the IBM Personal Computer (IBM PC) has become a real issue. It's obvious that Heath and Zenith have taken this compatibility issue seriously with the announcement of their new H/Z-150 and H/Z-160 series computers.

Some History

In late 1981, IBM released the Personal Computer (PC) which, unlike a lot of others, included features that encouraged the development of third party software and hardware. The PC hit the market like an atomic bomb and was nearly an overnight success which surprised even the developers. The implied promise of a large application software library and the open architecture of the system combined to make the PC an industry standard. Everyone was discussing PC compatibility, and most microcomputer manufacturers were trying to develop micros to conform to the de facto standard.

There is another significant factor which I believe has been overlooked. Soaring software development and hardware costs were being critically reviewed by most companies. And all were looking at ways to cuts those costs. Enter the microcomputer into the corporate environment as a way for the user to develop his own applications ... spreadsheets, data bases, and so on.

I know of one company that purchased several thousand PC's for use by their management. Why has the PC enjoyed such a success in the corporate marketplace? Large IBM mainframes and other equipment have been around in a lot of companies for years. IBM is a known entity in the corporate environment. Any manager who purchases an IBM widget cannot be faulted(or fired) if it doesn't work out the way it was expected. The whole problem was obviously the user's fault since the equipment was IBM. In short, any IBM equipment is a "safe" investment for any manager regardless if it was appropriate for the application or not. And the level of support for the IBM cannot be overlooked either. An IBM representative is usually readily available to get support and answer questions.

But what are the real compatibility issues? Why doesn't a lot of the IBM PC software run on the H/Z-100? What causes the "Wild Interrupt" message? And what's different about the H/Z-150 and the H/Z-160 that make it "more" compatible? In order to find the answers to those questions, we have to look at two things, hardware and software. But before we look at the actual compatibility problems, why is the compatibility issue so important to us?

It's Called Software Availability!

Within a few months of the release of the IBM PC, the market was flooded with new software designed for the PC. Sales of the PC skyrocketed to the point that even IBM was surprised. What caused it? IBM has an absolutely superb marketing group. And EVERYONE has heard of IBM. Name recognition and product identification are the name of the game in marketing. Have you ever known anyone who wasn't aware that IBM made computers? I haven't, but then I've been in data processing for something approaching 20 years.

As a direct result of that marketing expertise, existing and potential software vendors decided to get a piece of the action. The size of the market was so vast that even a small percentage represented a rather significant income. Whether you like it or not, the bottom line is money, and that is strictly a business decision.

The remarkable availability of software has caused the rather untoward interest in compatibility. In some cases, vendors have not taken the time to adapt their programs for "similar" computers (like the H/Z-100) because they apparently don't believe that it would be profitable to do so. While this is obviously a short sighted philosophy, it remains a fact of life. That particular point, by the way, really burns me up.

Hardware Compatibility

Although there are a number of issues that relate to PC compatibility, I believe that there are three main ones that must be considered: the IBM PC architecture, the copyrighted PC ROM (read only memory), and the operating system (PC-DOS).

It seems to me that the biggest difference is in the hardware bus structure. The PC uses a bus specifically designed for it, and that bus was not designed to any particular standard. On the other hand, the H/Z-100 uses the popular (and standard!) S-100 bus which has a number of advantages, not the least of which is that you can use almost any standard S-100 board in your computer. That bus difference accounts for part of the compatibility problems, but it's not the major one. That particular problem can usually be taken care of by judicious coding of the BIOS, but it's not responsible for the Wild Interrupt message.

Three little letters are the basic problem - ROM (Read Only Memory). The PC ROM is the culprit in virtually all cases where the Wild Interrupt message appears. Many of the hardware features available on the PC are available ONLY through a call to the ROM. Since I don't have a PC technical manual (yet!), I'll use an example which may or may not be a fact. The H/Z-100 (and the H/Z- 89) use an Escape E sequence to clear the screen. However, you may have to code a ROM call to clear the screen for the PC since that function is not even available in the BIOS. The only real advantage of the ROM implementation is speed since the BIOS need not be accessed (disk drive access is slow compared to a ROM call). However, the PC ROM is copyrighted so that it cannot be simply duplicated by other hardware manufacturers.

Software Compatibility

PC-DOS was developed based on the MS-DOS operating system from Microsoft. It was customized for the hardware features and system architecture available in the PC as are most operating systems these days. The generic version of MS-DOS, when customized by a manufacturer, will generally provide access to terminal control features through the use of a standard software interrupt (INT 21H). Cursor movement, delete functions, and other terminal features are used by all word processors. If you make those functions difficult (or impossible) to access through the BIOS, and then place those features in a copyrighted ROM, what do you have? An IBM PC! And then ANY application software, like WordStar, MUST use the ROM calls in order for the application to perform its function. The application program expects to find certain information in a specific location in memory, and if it's not there, Wild Interrupt time begins. What can you do to solve this rather puzzling array of problems?

Help is Available

With all of these compatibility problems, how can you get around it? Probably the best known, or perhaps the most infamous, is Graham Wideman's IBEm package. It's a PC-DOS emulator that runs on the H/Z-100, and it will allow you to run some of the PC software. I've heard that he has received a lot of, what I feel is, unjust criticism for IBEm. Among other things, it doesn't run ALL of the PC software, but I don't think that's a fair criticism. Of course it doesn't! We've looked at some of the technical reasons that some of the PC software can't possibly run on the H/Z-100, but it appears that these problems have been solved with the introduction of the H/Z-150 and H/Z-160. IBEm has implemented some software to cope with the hardware differences in the keyboard, printer interface, equipment status, memory availability, and standard disk access routines. Other disk routines, which can be used to copy protect software, are not available in IBEm but i don't think this is a big disadvantage since I don't care much for copy protected software in the first place.

Using IBEm

As you might expect, I don't have a lot of PC software to really check out IBEm, but I have run the PC version of AutoDex (see the May 1984 REMark for the software review) under IBEm with no problems. The only comment that I have is that program execution seems to be slower, but I expected that. IBEm version 1.3 is installed with a batch file, and the manual is reasonably clear on the installation process. This version of IBEm seems to work fine with both versions of Z-DOS, 1.1 and 1.25.

The manual provides some basic cautions on the uses and limitations of IBEm which are worth repeating. Programs which directly access the PC's video memory for pixel graphics or text will generally not work under IBEm. Terminal communications packages will usually access the PC's communication port which also has not been implemented. As previously mentioned, copy protected programs will probably not run either. And finally, the IBM BASIC is not supported.

The bad news is that it's almost impossible to find out if an application program has any of these limitations. According to the manual, PC versions of VisiCalc, MultiPlan, and Lotus 1-2-3 will definitely not work under IBEm. WordStar version 3.24 apparently works as expected, but version 3.3 will not because of direct memory access for the video. As a side note, I would not expect to find that any of the games available for the PC would run under IBEm since many games use graphics. The good news is that many of the popular programs either provide an install program that allows customization for MS-DOS compatible computers (e.g. dBase II) or Zenith has customized the programs for us (e.g. WordStar and Lotus 1-2-3).

IBEm is clearly the result of a lot of expert programming, testing, and hard work. If you find an application that is only available for the PC, and you just can't live without it, talk to the manufacturer about how it was programmed with the above limitations in mind. Otherwise, I would suggest that you write to Graham Wideman and ask him about it. I think you'll find him very cooperative, and he may be able to give you some helpful advice.

Changes Still!

Moving to a new job and state are not real helpful to a writer! The new Z-DOS FlipFast book is done and includes the version 2.0 commands. For those of you who have asked about this book, we decided to delay it so that the 2.0 commands could be included. The June release date of version 2.0 provided us with very little time to add the changes and new commands in order to get this book to you as soon as possible. My personal view is that I would rather delay a book (or column) for a short time to make sure that it is accurate and complete. Experience shows that, no matter how careful you are in editing and proofreading, small errors inevitably creep into the final result. Back to 2.0. In addition to the changes required for the new piping and pathing commands, there are over 20 new commands which had to be tested and written. Printing also takes some time. But this whole process should be nearly complete by the time you read this.

July HUG Convention

From all indications, the HUG Convention should really be an event to remember this year. I've reviewed the agenda, and I'm looking forward to it. But this year I have a personal interest since I'll be one of the speakers. My topic will be a discussion on the differences between CP/M-80, CP/M-85, and CP/M-86. The presentation will primarily focus on the command differences between these operating systems since I'll use a "how-to" approach instead of a lot of theory. There are some interesting differences in the 8-bit and 16-bit versions, and I'll also be looking at the CP/M-86 feature which allows you to run 8-bit software. If you're using any version of CP/M, you might find this presentation interesting. See you there!

Time Goes By...

Due to some extensive traveling related to my new job, I haven't really had the chance to really give some recent software all of the attention that it deserves. I will spend some time on that for the next column.

In The Mail

For those of you who have taken the time to write, I appreciate your comments and have not forgotten about them. Please note the change in address above which will help me give you a more timely response to your letters. If you expect a personal reply, don't forget to enclose a stamped, self addressed envelope.

Next Month

If you haven't heard, version 2 of Z-DOS/MS-DOS has been released. Since I thought that there might be some interest in the differences between version 1 and version 2, it seems appropriate that we take a look at the two versions next month. What's new in version 2? A lot!

Products Reviewed

AutoDex Automatic Software 1035 Santa Barbara St. Santa Barbara, CA 93101 (805) 963-5861	$150.00
IBEm Wideman Computer Consulting 1320 Pepper Villa Drive El Cajon, CA 92021	$79.95

Vectored from 45

a file to "TO" with this program unless you included a TO between the filenames. Small price to pay for allowing both entry forms.

If the the program detects a "TO ", it locates the second name in the command line buffer (at the default DMA address) and decodes it to the form required by CP/M. The decoding code separates the name from the extension, places each in their proper place in the File Control Block (FCB), and fills any left over characters spaces in the name and extension areas with spaces.
After the program decodes the second file name (if that was necessary), it puts the drive code from the first FCB into the second FCB. This step is unnecessary for the rename operation, but is done so that the next step can be done, which is to check to see if the second file name already exists. After that, it checks both FCB's for wild card characters, and if it gets through these tests, it finally performs the rename operation, and exits.

Following the main program are the error exits and a subroutine to skip spaces that is used in the program. Next is the data area that contains the text strings used by the program, and some stack space.

Congratulations to John Smith for producing a program that is well commented, logically laid out, and most importantly, that works! Next month, I will present the winning HDOS program.

See page 5 for the
International HUG Conference Registration Form

Welcome to.. It's Great!

DOODLER

Graphics Package

. . . for the Zenith Z-100

- **Full Featured Graphics Design Package.**
- **Pallette of 36 different color choices.**
- **Store Designs on Disc for later Reference.**
- **Playback Mode for Error Correction.**
- **Extended Text Capabilities including. . .**
  User Designed Character Fonts.
  Italic or Backslant Style and scaling.

See DOODLER at your local HEATHKIT Store.
. . .or Send $ 79.95 directly to. . .

PAUL F. HERMAN
DATA SYSTEMS CONSULTANT
P. O. Box 535
St. James City, FL 33956
813-283-2227
Specification Sheet available on Request.

EPSON
GRAPHICS
for
H89 and H100

Our *CALLIGRAPHY* program package for the *EPSON* printer can combine dot graphics images from disk files and standard text to create custom graphics for advertisements (such as this ad), letter heads title pages and much more!

The set of programs includes a screen character editor, a program to print text and graphics and a starter set of alphabets, pictures and borders. The H89 version requires CP/M and the H100 version operates under CP/M-85. When ordering, please specify computer, disk format and memory size. Price of package, including 60 page manual with tutorial, is 49.95 (Calif. residents add tax).

★ Expand the disk library by adding *FONT SET 2.* 14.95 when ordered with the *CALLIGRAPHY* package. Additional font sets are 19.95 ordered separately.

Write for more information or call our evening customer service number (415) 254-3931.

SKYCASTLE
Computer Products
P.O. BOX 1412 SAN CARLOS, CA 94070

Advanced Assembly Language Programming

Using Z-DOS Resident Programs

Frank T. Clark, M.S.C.S.
402 West Ferry Street
Berrien Springs, MI 49103

Introduction

The Z-DOS operating system provides some new features for the assembly language programmer that did not exist in CP/M. One of these features is the ability to write resident programs. Resident programs are run once and then become a part of the operating system and remain resident in memory even after you return to the operating system prompt and run other programs. This type of program is a little bit similar to the device drivers found in the HDOS operating system.

Appendix I, page 5 of the Z-DOS manual states "This interrupt is used by programs which are to remain resident when COMMAND regains control. Such a program is loaded as an executing COM file by COMMAND. After it has initialized itself, it must set DX to its last address plus one in the segment it is executing in, then execute an interrupt 027H. COMMAND will then treat the program as an extension of Z-DOS, and the program will not be overlaid when other programs are executed."

This is of course very interesting but the problem is that it is not obvious how one makes use of such a feature. The first question one would ask is 'How is a resident program run after it is already in memory?' This question indicates a slight misunderstanding of the purpose of resident programs. Resident programs are not usually run using console prompts and input like regular programs but rather they have a predefined function and run in the background. The question then remains 'How does a resident program execute?' The answer is simple. A resident program usually executes from an interrupt. Resident programs are primarily useful for interrupt handlers.

There are two different kinds of interrupts. There are hardware interrupts and there are software interrupts. Hardware interrupts are generated by the hardware design due to the servicing needs of ports, peripherals, or devices. In the standard hardware/software design, the Z-DOS operating system handles all the hardware interrupts. Software interrupts are generated by the operating system itself to allow the user to interface to some of the operating system functions. The user is also allowed to generate some software interrupts mostly to communicate with the operating system.

It is important to note that an interrupt handling routine does not have the same freedom of a normal user program to use just any system function it might like. In particular most operating system functions and some BIOS functions cannot be used by an interrupt handling routine. It takes a lot of experience and sometimes trial and error to know just what can and cannot be done by an interrupt handling routine.

At this point it is important to note a few points about resident programs. These programs become permanently attached to the operating system, therefore any memory they use is no longer available for other programs to use. The only way to get rid of a resident program is to reboot the computer. If the resident program is run in an AUTOEXEC.BAT file, then you must abort the file before it is run or it will be back in the memory again. Resident programs should not use too much memory or they could cause other programs to run out of memory.

Did you know that there is an example of a resident program on your Z-DOS distribution disk? The PSC.ASM program is the source for a resident program to handle the print screen function. The SHIFT of the F12 function key generates a unique interrupt (number 5) which is usually ignored. This is the only key that generates a unique interrupt by the way. When the PSC program is assembled and run, it attaches itself to the operating system and the interrupt number 5. The program does not do anything (sometimes it is said to be asleep) until it receives control from the interrupt and then it wakes up, performs the print screen function, and goes back to sleep.

Most people are disappointed when they find out that the PSC program only sends the ASCII characters on the screen to the printer. It would be equally possible for the routine to send actual graphics to a suitable printer but the program would have to be much more complex. Some other time I may get into a discussion of how to improve that program.

The PSC program is actually rather simple as far as resident programs go because it handles only one interrupt and that interrupt is generated from the keyboard on a very infrequent basis. In this article I will describe a far more complex resident program that will demonstrate the full power of interrupt handling using resident programs. This example program will handle three interrupts which are occurring constantly and perform a useful video control function.

Example Program

This program will perform a function found built-in to the Z-29 terminal called the CRT saver. What this function does is to turn the CRT screen off when there has been no activity for a certain time period. This function saves the CRT from having a pattern burnt into it when it is left unattended for long periods of time. It also has a secondary advantage in that if the information displayed on the screen is of a confidential nature, it is not left displayed on the screen for just anyone to read.

The way the program works is to watch the keyboard input, the CRT video output, and the clock. If no character has been typed on the keyboard or output to the screen for a certain time period, then it will turn the video screen off. Once it has turned the screen off, it must watch and turn it back on when a character is typed on the keyboard or a character is output to the video screen.

A careful reading of the Z-DOS BIOS source shows that there are software interrupts that are generated for each of the three conditions we are interested in. There is an interrupt generated for every key that is typed on the keyboard and every character that is output to the CRT video screen. There is also an interrupt generated for the timer that counts in hundredths of a second. It is important to note that when we write routines that are going to intercept these interrupts, the routines will be called very often. We must avoid slowing the operation of the system down so the routines must be very efficient and use a minimum of instructions to do what is needed.

A careful reading of the Z-100 Technical Manual's description of the video logic board indicates that there is a video control register which will allow us to have direct control of the video display. With careful modification of this register we can turn on or off the display of the video memory on the screen without changing the video memory itself. There are other functions handled by this control register so we must be careful of just how we modify it.

In our assembly language program we will make frequent use of the standard definitions for the system values. These are found in the files DEFMS.ASM, DEFCONFG.ASM, and DEFIPAGE.ASM on the Z-DOS distribution disk. Whenever we use one of these values it will appear in uppercase and all the rest of the program is entered in lowercase letters. These files must be on the default disk when you attempt to assemble the program.

Program Design

The program will consist of the following parts: the keyboard input interrupt handler, the video output interrupt handler, the timer interrupt handler, and the interrupt installation routine. The interrupt installation routine is not needed after the program is run. When the interrupt 027H is executed we will put a value in DX that includes only the code that is needed after the program is installed. This keeps the amount of memory used by the program to an absolute minimum.

The first thing the program does when it is executed is to jump over all the code for the interrupt handlers to the interrupt installation routines. This code must be at the end of the program so that it can be discarded when it has been used. The interrupt installation routine is identical for each of the three interrupts. First the data segment is modified to point to the interrupts. Second the old value of the interrupt is retrieved, and finally the new value is installed. The old value of the interrupt is used as the termination address for the interrupt handler so that the routine that would have handled the interrupt still gets the chance. When the program is ready to quit, it prints a message to indicate that it has done its job.

The keyboard input interrupt handler and the CRT output interrupt handler are identical except for the termination address. First, the routine tests to see if the video output has been turned off. If the video has been turned off, it is turned back on. Finally, the interrupt handler resets the counter for the timer.

The timer interrupt handler counts the hundredths of a second. The counter goes from -32766 to 32767 which at one hundred counts a second takes about ten minutes. When the counter overflows, the video display output is turned off.

Program Operation

After the program has been created using a suitable editor such as EDLIN it must be assembled. I will give an example of the commands necessary to create the program assuming that we shall call it CRTSAVER.

```
remark This is an example of a BAT file to create
remark the program CRTSAVER
edlin crtsaver.asm
masm crtsaver;
link crtsaver;
erase crtsaver.obj
exe2bin crtsaver.exe crtsaver.com
erase crtsaver.exe
pause till ready for test run
crtsaver
```

After the program is run, if there is nothing typed on the keyboard or output to the screen for ten minutes, the screen will go instantly blank. The screen will return instantly if any character generating key is typed on the keyboard or any character is output by a program. Regrettably the CRTSAVER program cannot tell if the SHIFT keys, the CAPS LOCK or the FAST REPEAT have been typed without interfering with the normal operation of computer programs. This limitation is because these keys do not normally generate characters. The BREAK key is particularly useful for turning the screen on since it generates actual characters but these characters are intercepted by the BIOS and do not interfere with the normal operation of computer programs. This is not very well documented but the BREAK key tells the BIOS to discard type ahead characters. Type ahead characters are any typed characters which have not yet been input by a computer program.

Conclusion

The ability to have resident programs gives a lot of flexibility to assembly language programming under MS-DOS. Particularly when using the BIOS software interrupts a little imagination can see a lot of interesting programs that can be written. As an example, a program could be written which would display a clock on the screen based on the timer interrupt. A program could be written which would echo output to the printer or the AUX device without depending on the CTRL P command. A program could intercept the keyboard input and convert special codes to multiple character strings. The possibilities are endless.

Just remember to be careful in whatever programming you do. With assembly language programming in particular it is very easy to crash the system and even to clobber disks. The writing of resident programs, interrupt handlers or any program that makes direct access to hardware ports is especially tricky. Be sure to always have backups of any data in the computer that might become damaged. If you follow a few simple precautions you should have no problems. You can always be glad for the fact that it is almost impossible to physically damage the hardware from programming mistakes. If worse comes to worse, you can always reboot the computer, reformat your disks, and start over. Good luck!

Program Source

```
        title   crtsaver - turn off crt when not used
        include defms.asm
        include defconfg.asm
        include defipage.asm
```

```
;This resident program is designed for turning off the crt when
;no activity has occurred for a specific time and turn it back on
;if any character is input or output.  The BREAK key is particularly
;effective for this purpose.
;
code    segment
        assume  cs:code
        org     PHD_CODESTART
;
;       Entry point
;
begin:
        jmp     near ptr start
        subttl  interrupt handlers
;
;       interrupt handler for keyboard input
;
int_kb:
        call    testit
        jmp     dword ptr cs:nxt_kb
;
;       interrupt handler for crt output
;
int_crt:
        call    testit
        jmp     dword ptr cs:nxt_crt
;
;test for activity
;
testit  proc    near
        test    cs:crt_off,-1
        je      on
        push    ax
        in      al,ZVIDEO
        and     al,0f8h
        or      al,08h          ;data from the vram displayed
        out     ZVIDEO,al
        mov     cs:crt_off,0
        pop     ax
on:
        mov     cs:timer,-1
        ret
testit  endp
;
;       interrupt handler for timer
;
int_tm:
        sub     cs:timer,ax     ;countdown
        jc      off_crt
        jmp     dword ptr cs:nxt_tm
off_crt:
        test    cs:crt_off,-1
        jne     off
        push    ax
        in      al,ZVIDEO
        mov     cs:crt_off,-1
        and     al,0f7h         ;disable vram display
        or      al,07h          ;disable all planes display
        out     ZVIDEO,al
        pop     ax
off:
        jmp     dword ptr cs:nxt_tm
        subttl  data area
crt_off db      0       ;crt off flag 0=on
timer   dw      ?       ;countdown timer
nxt_crt dd      ?       ;address of next crt handler
nxt_kb  dd      ?       ;address of next keyboard in handler
nxt_tm  dd      ?       ;address of next timer handler
;
;       Mark end of resident portion
;
int_end label   near
        subttl  resident installation routine
;
;       Main entry point
;
mesg    db      13,10,'CRTSAVER routines installed.$'
        assume  cs:code,ds:code,ss:code
start:
;
;install int_crt handler
;
        mov     si,INT_UCRTA*4          ; si points to interrupt
        mov     dx,offset int_crt       ;my interrupt handler
        call    interrupt
        mov     nxt_crt,di
        mov     nxt_crt+2,es
;
;install int_kb handler
;
        mov     si,INT_UKBA*4           ; si points to interrupt
        mov     dx,offset int_kb
        call    interrupt
        mov     nxt_kb,di
        mov     nxt_kb+2,es
;
;install int_tm handler
;
        mov     si,INT_UTMA*4           ; si points to interrupt
        mov     dx,offset int_tm
        call    interrupt
        mov     nxt_tm,di
        mov     nxt_tm+2,es
;
;tell them we did it
;
        mov     dx,offset mesg
        mov     ah,DOSF_OUTSTR
        int     DOSI_FUNC
        mov     dx,offset int_end               ; LWA of resident portion
        int     DOSI_TERMR
;
;set interrupt addresses
;
;input si=interrupt, dx=offset new handler
;return es:di=old handler
 interrupt proc  near
        push    ds
```

☞

```
        xor     ax,ax
        mov     ds,ax                   ; Clear ds
        les     di,dword ptr [si]               ;fetch the next handler
        cli
        mov     word ptr [si],dx
        mov     word ptr [si+2],cs              ; Set my interrupt vector
        sti
        pop     ds
        ret
interrupt endp
code    ends
        end     begin
```

*

About the Author:

Frank is a software consultant for Zenith Data Systems in St. Joseph, Michigan. He is also vice-president of the Blossomland Heath Users' Group. His background includes Xerox Sigma systems APL, COBOL, FORTRAN, Meta-Assembler, BASIC, word processing, IBM VS/APL, Ohio Scientific OS65D, 6502 assembler, BASIC, Commodore DOS, BASIC, word processing, Heath-/Zenith word processing, CP/M, MBASIC, COBOL-80, CP/M MAC 8080 assembler, MicroSoft C, MS-DOS MASM 8088 assembler. He graduated from Andrews University, Michigan in August 1979 with a B.S.C.S degree and in August 1982 with an M.S.C.S. degree. His interests center on anything even remotely related to computer software including music, graphics, and education as well as reading, mostly computer books and science fiction.

If you wish to contact the author, write to the above address, do not telephone, and include a self addressed stamped envelope. An advanced version of this program including source code is available on disk for a small fee. Write for details.

NOTE:

This program is designed to work using Z-DOS version 1.25 for the Z-100 with BIOS version 1.10, release date of 4-15-84 or later. The program should work under MS-DOS version 2 for the Z-100 when it is released, but this can't be guaranteed at this time.

H/Z-100 AND 150 PC GRAPHICS SOFTWARE

MICROSERVICES continues to expand its support of Heath/Zenith computers.

For H/Z-100s:

ZANIMATE. Make color animation sequences using ZBASIC **$64.95**

ZPALETTE. Draw images in 92 colors using ZBASIC. **Improved** with three (3) painting modes and graphics generator **$59.75**

ZPATTERN. Test your color monitor **$24.95**

Z3D from **COLORWORKS.** Make three-dimensional objects in color **$75.00**

For H/Z-100s and 150 PCs:

SOFTKITS from **KERN INTERNATIONAL.** We are now carrying a full line of this excellent series of books and disks for graphics and business/scientific applications.

We are continuing to introduce new products. Write for our catalog.

VISA

MICROSERVICES
P.O. Box 7093
Menlo Park, CA 94026
Phone: (415) 851-3414

master charge

Include 3% p/h ($3.00 min.). California add 6½% tax.

Z FORCE

ARCADE ACTION

7 LEVELS OF PLAY
1 or 2 PLAYERS

ARCADE SPEED

WARNING!

Even if you destroy the enemy ships, you still have the mother ship to contend with!

REQUIRES: H/Z 100 system with color (64K chips), Z DOS 192K. Mem.

$39.95

CALIFORNIA RESIDENTS ADD APPROPRIATE SALES TAX

GET IT AT YOUR LOCAL HEATH/ZENITH SUPPLIER OR FROM:

WESTCOMP

517 N. Mountain
Ste. 229 • Upland

(714) 982-1738

NOTICE: Westcomp is not responsible for any increase in blood pressure or sleepless nights from the result of excessive use of this product!

The Software Toolworks presents: MYCALC™

CAR PAYMENTS. BANK STATEMENTS! TAXES... ARRGH!
YO, FREDDIE. STILL GLAD YOU FIRED THE ACCOUNTANT?
BUDGETING. ACCOUNTING. TAXES... ARRGH!
THIS IS MYCALC™, FREDDIE. IT'S A SPREADSHEET. AND FOR SIXTY BUCKS IT'LL SOLVE YOUR PROBLEMS.
COMES WITH TEMPLATES FOR IRS FORMS 1040 & SCH. A
TWENTY THREE HELP SCREENS! THIS IS EASY, I MEAN MY BUDGET'S ALMOST DONE.
NICE WORK, FREDDIE. NOW PLOT MY CATNIP FUTURES.
MYCALC IS $59.95. IT'S AVAILABLE FOR CP/M (R) AND HDOS (64K) AT MOST HEATH/ZENITH DEALERS. WRITE OR CALL FOR A FREE 41 PRODUCT CATALOG AND NEAREST DEALER.
A SOFT BUY IN A HARD WORLD!

Full featured MyCalc™ with sort, bar graphs, multiple files and more is ideal for financial planning and budgets. It sells for $59.95 from The Software Toolworks, 15233 Ventura Blvd., Suite 1118, Sherman Oaks, CA 91403 (818) 986-4885.

LEARN A NEW WORD FOR YOUR VOCABULARY

ZPAY\zē pā\ n. 1: A computer payroll system for the Zenith and Heath computer systems 2: The act or fact of paying or being paid 3: The status of being paid by an employer

ZPAY PAYROLL SYSTEMS Presents

ZPAY available for ZDOS, IBM-PC Compatable or CPM

Some of the features available in the **ZPAY** Payroll System

- User Friendly — Menu Driven
- Pays by Hourly, Salaried & Commissioned
- No interpreters necessary
- Super FAST operation
- Attractive 3 ring binder manual
- User changeable State & Federal tax tables
- User selected pay periods
- Maintains payroll records required by law
- Prints out easy to use and read reports
- Choice of two different check formats
- *Plus* many features of systems costing much more

May be seen at many Heathkit Electronic Centers or order yours today from **ZPAY** Payroll Systems — **Only $100.00.**

Please send me **ZPAY** for ☐ ZDOS ☐ IBM-PC ☐ CPM

Name ______________________________

Street ______________________________

City ________________ State ________ Zip ________

Phone ____________ Visa/MC Card # ____________ expires ________

Orders shipped by UPS. Enclose $100.00 plus $4.00 per order for shipping and handling. Payment by check, money order, or Visa/Master Card. *Allow 2—4 weeks for delivery.*

ZPAY PAYROLL SYSTEMS

3516 Ruby Street • Franklin Park, IL 60131 • (312) 671-3364

HOW MUCH FREE SOFTWARE COULD YOU USE?

FIND OUT WITH OUR GIANT PUBLIC DOMAIN DIRECTORY

- SUPPLIED ON DISK FOR EASY COMPUTER ACCESS
- MORE THAN 4,500 ENTRIES
- SUBJECT AREAS INCLUDE:

ASTRONOMY, AVIATION, BUSINESS, EDUCATION, ENGINEERING, GAMES, GRAPHICS, HAM RADIO, MUSIC, PROGRAMMING, TEXT EDITING, VOICE SYNTHESIS, UTILITIES AND MUCH MORE.

Yes! I need to know what free software is available. Send me the public domain directory on the Heath CP/M* 5¼ format checked ☐ ON 3 HARD SECTORED SD $17.00 ☐ ON 2 SOFT SECTORED DD $12.00 ☐ ON 1 DOUBLE SIDED DD $7.00

HEADWARE
2865 AKRON STREET
EAST POINT, GA. 30344

TERMS: NO RISK, MONEY back guarantee. Add $2 domestic $4 foreign per order for S&H. Enclose your check or M.O. with your order. Sorry, no charge or phone orders.

Name ______________________________

Address ______________________________

City, State, Zip ______________________________

*CP/M Reg. TM Digital Research Corp.

IBM-PC/ZENITH Z-100 users.

Expand your computer universe with—

micro/VERSAL ™

A utility program to READ/WRITE over 20 different 5¼ (CPM, CPM/86, MS-DOS & USER DEFINABLE) disk formats. Now you can easily transfer text, data, or programs between many different micro computers by simply loading **micro/VERSAL**™ and the disk you want to READ/WRITE from or to. **micro/VERSAL**™ also includes comprehensive utilities to DUMP any 5¼ disk by track, RDSECT to read disk sectors and FAPP, a program to append files together to produce a large file. For disk formats not directly supported, **micro/VERSAL**™ provides customization routines that allow users to write their own directory routines.

Now With Fomatting

micro/VERSAL™ **$79.99**

plus $4.00 shipping & handling

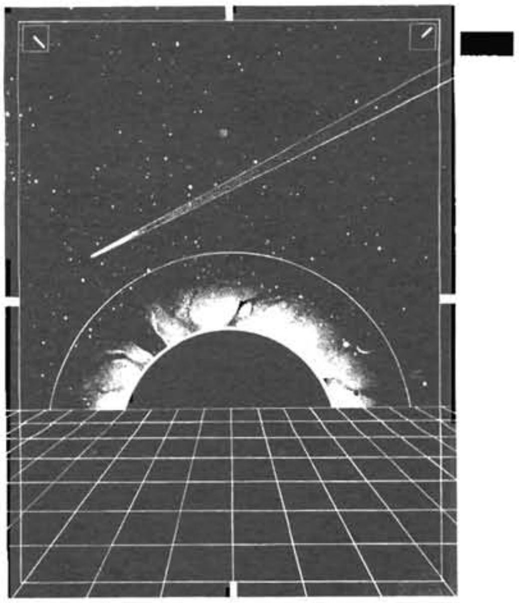
Also **COED**™ full screen editor **$34.97**
Includes: Stack arithmatic, MACRO commands, multiple files, definable function keys and much more.

CREDIT CARD ORDERS:	Master Charge / VISA
MAIL ORDERS:	Checks or money orders. N.J. resident add 6% sales tax.

ADVANCED SOFTWARE TECHNOLOGIES
417 Broad Street
Bloomfield, N.J. 07003
(201) 783-7298

A ZBASIC Color Game For The H/Z-100

Shooting Stars



Craig A. Pearce
2529 S. Home Avenue
Berwyn, IL 60402

The game "Shooting Stars" has been published in machine code for the Intel 8080 chip in an early issue of Byte Magazine, and was originally included in the Hewlett-Packard Program Library as "Teaser". It has now been rewritten to take advantage of the superior color and graphics abilities of the Z-100 series.

When a Star (yellow) is shot it begins "heating", turning white, then light blue, then deep blue, finally becoming a "neutron star" (the deep blue sphere), then lastly collapsing into a black hole (same graphic in black). As a black hole is transformed back into a star, it goes first through the "neutron stage" again, then a small, deep blue "burst" effect, then a larger, white burst which turns yellow and then into a yellow star.

Full instructions for play are included in the game itself. Turns are counted by the game, a chance to quit at any time is allowed. The game will announce when the player loses (with a special display), will produce a special win display, informs the user of having shot a black hole (not allowed), will transform all inputs to positive integers and error check against illegal inputs (0 or greater than 9), allows UPPER or lower case input when asking questions, and will heckle the player with one line quips when the minimum number of moves to complete the game is passed.

Program Lines

The program is broken down into groups with some parameter changes allowed in various areas. Lines 2 through 8 list author, date, title, and program revision. Lines 100 to 490 create the graphic displays for the game. Lines 500 to 530 inquire if instructions are desired. **Note:** Lines 30 through 60 are used as a subroutine later on in the program. This is accomplished by setting a variable called "FLAG" equal to -1 (or "NOT 0"). If this FLAG shows as a non-zero value in line 60, a RETURN is encountered. Otherwise, the RETURN is ignored. This allows a piece of code to be used both in the main body of the program and as a subroutine as well.

Line 1000 starts the program by drawing the initial display and resetting some of the program variables. At line 1190, a random number is used to select from one of the 14 different "heckle" lines used in the program. However, if the number of turns is still less than 11, the line is not printed. In addition, if it is the first turn, a "We're Ready to Go" message is printed instead. These "heckle lines" are in data statements beginning at line 10010. If the user desires to change any, he or she must keep the number of lines equal to 14 or change the value in the random number generator in line 1190.

Note: The variable "FLAG" is again used here to tell the program whether to erase the "Shot a Black Hole" warning line or not.

Lines 2000 up to 3000 actually play the game, incrementing the turn counter, checking for a win or a loss, and selecting what characters to change.

Lines 3000 up to 4000 set up the order and define exactly which graphic images must be altered for a shot entered. These lines are called as a subroutine from the 2000 series and, once the graphics are selected, will go to the next set at 4000.

Lines 4000 to 5000 do the actual work. Here each star or black hole is changed when a shot is made. In line 4020 a delay factor is set to slow down the graphics on the screen. Note that the first item to be changed (this will always be the star just shot) is slower than the other changes. These values can be modified by the user to suit his or her needs.

Lines 5000 to 6000 handle a loss in the game (all images have been changed to Black Holes). It gives the user the option of starting over. If the user decides NOT to play again, the screen is cleared and the text is returned to standard white on black.

Lines 6000 to 7000 are used when the user inputs a "Q" or "q" to quit the game. A review of the number of turns used is given. Again the text is reset to white on black.

Lines 7000 to 8000 are used to display the instructions while lines 8000 up to 10000 wait between instruction pages until the user presses a key. In the event the user wants to reread the instructions, he or she may press the "HELP" key at the end and the instructions will be rerun as many times as necessary.

Skipping to lines 10000 on are the data statements. Line 10000 contains the upper left coordinates (x-y order) for all of the locations on the screen for each graphic. They are used through out the program for "PAINTING" and image location.

Variables

The variables for the graphics are based on a box from 0,0 to 60,30. This results in the use of 724 bytes as per the formula provided. Using double precision (8-bytes per variable) requires 90.5 as the DIMentioned array. As a safety factor, the value of 95 was used for all graphics.

STARLOC$(x) is the string that contains the nine numbers (in string form) from the first of the data lines. These variables can be easily broken down into the first and last three digits and the values taken to provide the upper left corner coordinates for each graphic image. As an example, STARLOC$(1)="120020". Breaking this down into two equal strings results in "120" and "020", which are the x and y coordinates for the first graphic image on the screen. In order to "PAINT" these images during the game, the center point of any image (at least as far as PAINTing goes) is found by adding 30 and 15, respectively, to the x and y values. Thus the center of the graphic at location 1 would be 120+30 or 150 for the x-coordinate and 020+15 or 35 for the y-coordinate.

STARVAL(x) is the actual value of any particular location. If STARVAL(x) equals '0', then that location contains a black hole. If STARVAL(x) equals '-1' (or a "NOT 0"), the position contains a star. By counting the STARVALues, the program can tell of a win or a loss. If all nine STARVAL add up to '0', then there are no stars left and the game has been lost. If the total equals '-8', then it MAY be a winner (since there are 8 stars in a winning game). The program would then check the value of STARVAL(5), the center position. If it is a zero, then it is a black hole and since the count of stars is eight, a winning display MUST now exist.

MODIFY$ is set in lines 3000 up to 4000 as the graphics that must be changed. The order of the numbers in MODIFY$ is the order in which the graphics will be changed. Thus the star shot is always first since the routine will always slow down the first graphic.

HOLD# is used to save a modified graphic and then replace it on itself so that it may be erased (via the Exclusive Or feature).

```
2130    BEEP:PUT(280,76),HOLE1#
2140    FOR J=1 TO 150:NEXT J
2150 NEXT I
2160 SPIN$="12369874"
2170 FOR X=3 TO 15 STEP 6
2180    FOR Y=11 TO 51 STEP 20
2190            LOCATE X,Y
2200            PRINT "  "
2210    NEXT Y
2220 NEXT X
2230 FOR I=1 TO 20
2240    FOR J=1 TO LEN(SPIN$)
2250            STAR=VAL(MID$(SPIN$,J,1))
2260            X=30+VAL(LEFT$(STARLOC$(STAR),3))
2270            Y=15+VAL(RIGHT$(STARLOC$(STAR),3))
2280            COLOUR=I MOD 8:IF COLOUR=0 THEN COLOUR=6
2290            PAINT(X,Y),COLOUR,0
2300    NEXT J:BEEP
2310 NEXT I
2320 LOCATE 20,1:COLOR 6,6:PRINT CHR$(26):COLOR 7,0:PAINT(0,0),6,4
2330 LOCATE 20,5:COLOR 0,6
2340 PRINT"HURRAY!  You've Solved the Puzzle!  You Have Given Light to the"
2350 IF TURNS=12 THEN LOCATE 21,12 ELSE LOCATE 21,29
2360 PRINT"Galaxy in";TURNS-1;"Turns";
2370 IF TURNS<>12 THEN PRINT".":COLOR 7,0:END
2380 PRINT"...And With A Perfect Score As Well!"
2390 COLOR 7,0:END
3000 'define the characters to change:
```

```
2 ' *** SHOOTING STARS ***
4 ' By Craig A. Pearce
6 ' Version 2.16.C1
8 ' Date: 07/27/83;  Updated: 07/28/83; 07/29/83; 08/01/83
10 CLS:COLOR 1,0:PRINT"":COLOR 7,0:LOCATE 1,1:PRINT"":LOCATE 1,1
20 DIM STAR#(95),HOLE1#(95),HOLE2#(95),BURST1#(95),BURST2#(95),HOLD#(95),
   STARLOC$(9),STARVAL(9)
30 FOR I=1 TO 9
40      IF FLAG THEN 50 ELSE READ STARLOC$(I)
50      STARVAL(I)=0
60 NEXT I
70 STARVAL(5)=-1:IF FLAG THEN FLAG=0:RETURN
80 'create the star
90 LINE(30,2)-(14,27)
100 LINE(30,2)-(46,27)
110 LINE(46,27)-(6,12)
120 LINE(6,12)-(54,12)
130 LINE(54,12)-(14,27)
140 PAINT(30,7),7,7
150 PAINT(10,13),7,7
160 PAINT(30,17),7,7
170 PAINT(47,13),7,7
180 PAINT(18,24),7,7
190 PAINT(42,23),7,7
200 PAINT(30,15),6,0
210 'save star in array
220 GET(0,0)-(60,30),STAR#
230 PUT(0,0),STAR#              'clear area for next pattern
240 'create black hole #1
250 CIRCLE(30,15),27,1
260 CIRCLE(30,8),6,1
270 GET(0,0)-(60,30),HOLE1#     'save hole #1 in array
280 'create black hole #2
290 PAINT(30,15),1,1
300 GET(0,0)-(60,30),HOLE2#     'save hole #2 in array
310 PUT(0,0),HOLE2#             'clear area for next pattern
320 'create burst #1
330 CIRCLE(30,15),5,1
340 PAINT(30,15),1,1
350 LINE(30,2)-(30,28),1
360 LINE(2,15)-(58,15),1
370 GET(0,0)-(60,30),BURST1#    'save burst #1 in array
380 PUT(0,0),BURST1#            'clear area for next pattern
390 'create burst #2
400 CIRCLE(30,15),7,7
410 PAINT(30,15),7,7
420 LINE(30,2)-(30,28),7
430 LINE(2,15)-(58,15),7
440 LINE(5,5)-(55,25),7
450 LINE(6,6)-(56,26),7
460 LINE(55,5)-(5,25),7
470 LINE(55,6)-(6,25),7
480 GET(0,0)-(60,30),BURST2#    'save burst #2 in array
490 PUT(0,0),BURST2#            'clear screen of graphics
500 'ask for instructions...
510 LOCATE 5,10
520 LINE INPUT"Would You Like Instructions? <NO> ";X$:X$=LEFT$(X$,1)
530 IF X$="Y" OR X$="y" THEN GOSUB 7010 ELSE CLS
1000 'start program
```

```
1010 TURNS=1:RANDOMIZE TIME/DATE
1020 'print out current field:
1030 FOR Y=20 TO 132 STEP 56
1040   FOR X=120 TO 440 STEP 160
1050           IF X=280 AND Y=76 THEN PUT(X,Y),STAR#:GOTO 1070
1060           PUT(X,Y),HOLE1#
1070   NEXT X
1080 NEXT Y
1090 NUMBER=1
1100 FOR X=3 TO 15 STEP 6
1110   FOR Y=11 TO 51 STEP 20
1120           LOCATE X,Y
1130           PRINT NUMBER
1140           NUMBER=NUMBER+1
1150   NEXT Y
1160 NEXT X
1170 LOCATE 20,1:PRINT"This is Turn No. =";TURNS;SPACE$(52):
     IF FLAG THEN FLAG=0:LOCATE 21,1:PRINT STRING$(79," "):RETURN
1180 LOCATE 20,30:RESTORE 10010
1190 R=INT(RND*14)+1
1200 IF TURNS=1 THEN PRINT"We're Ready To Go!":GOTO 1240
1210 RESTORE 10010:FOR I=1 TO R
1220   READ COMMENT$
1230 NEXT I:RESTORE 10010:IF TURNS>11 THEN PRINT COMMENT$;
     STRING$(50-LEN(COMMENT$)," ") ELSE PRINT STRING$(50," ")
1240 LOCATE 22,1
1250 LINE INPUT"Shoot What Star (or 'Q' to Quit) => ";X$
1260 IF X$="Q" OR X$="q" THEN 6000
1270 S=INT(ABS(VAL(X$)))
1280 IF S<1 OR S>9 THEN 1300
1290 IF NOT STARVAL(S) THEN 1340 ELSE FLAG=-1:GOSUB 1170:GOTO 2000
1300 'illegal input
1310 BEEP
1320 LOCATE 22,36:PRINT STRING$(30," ")
1330 GOTO 1170
1340 'shot a black hole
1350 X=VAL(LEFT$(STARLOC$(S),3))
1360 Y=VAL(RIGHT$(STARLOC$(S),3))
1370 FOR I=1 TO 6
1380   PUT(X,Y),HOLE2#:BEEP
1390   PUT(X,Y),HOLE1#:BEEP
1400 NEXT I:TURNS=TURNS+1
1410 LOCATE 21,2:COLOR 6,4:
     PRINT"Your Shot Was Just Devoured By The Black Hole!
     YOU CAN'T SHOOT BLACK HOLES!":COLOR 7,0
1420 GOTO 1320
2000 'now play the game
2010 TURNS=TURNS+1
2020 ON S GOSUB 3010,3040,3070,3100,3130,3160,3190,3220,3250
2030 'check for a win or loss
2040 H=0
2050 FOR I=1 TO 9
2060   H=H+STARVAL(I)
2070 NEXT I
2080 IF H<>0 AND H<>-8 THEN 1320
2090 IF H=0 THEN 5000           'a loss - all are now black holes
2100 IF H=-8 AND STARVAL(5)<>-0 THEN 1320
2110 '*** A WIN! ***
2120 FOR I=1 TO 10
```

```
3010 'star #1 shot...
3020 MODIFY$="1254"
3030 GOTO 4000
3040 'star #2 shot...
3050 MODIFY$="213"
3060 GOTO 4000
3070 'star #3 shot...
3080 MODIFY$="3256"
3090 GOTO 4000
3100 'star #4 shot...
3110 MODIFY$="417"
3120 GOTO 4000
3130 'star #5 shot...
3140 MODIFY$="52684"
3150 GOTO 4000
3160 'star #6 shot...
3170 MODIFY$="639"
3180 GOTO 4000
3190 'star #7 shot...
3200 MODIFY$="7458"
3210 GOTO 4000
3220 'star #8 shot...
3230 MODIFY$="879"
3240 GOTO 4000
3250 'star #9 shot...
3260 MODIFY$="9856"
3270 GOTO 4000
4000 'do the actual changes necessary for the shot:
4010 FOR I=1 TO LEN(MODIFY$)
4020   IF I=1 THEN DELAY=125 ELSE DELAY=65
4030   MODIFY=VAL(MID$(MODIFY$,I,1))           'find the next postion to change
4040   X=VAL(LEFT$(STARLOC$(MODIFY),3))        'x-coordinate on screen
4050   Y=VAL(RIGHT$(STARLOC$(MODIFY),3))       'y-coordinate on screen
4060   IF STARVAL(MODIFY)=0 THEN 4150          'go change a black hole to star
4070   'change a star:
4080   PAINT(X+30,Y+15),7,0:GOSUB 4280
4090   PAINT(X+30,Y+15),3,0:GOSUB 4280
4100   PAINT(X+30,Y+15),1,0:GOSUB 4280
4110   GET(X,Y)-(X+60,Y+30),HOLD#:PUT(X,Y),HOLD#       'clear star
4120   PUT(X,Y),HOLE2#:GOSUB 4280:PUT(X,Y),HOLE2#      'clear no. 2
4130   PUT(X,Y),HOLE1#:GOSUB 4280
4140   GOTO 4250
4150   'change a black hole:
4160   PUT(X,Y),HOLE2#:GOSUB 4280
4170   GET(X,Y)-(X+60,Y+30),HOLD#:PUT(X,Y),HOLD#       'clear hole no. 2
4180   PUT(X,Y),BURST1#:GOSUB 4280
4190   PUT(X,Y),BURST1#                                'clear burst no. 1
4200   PUT(X,Y),BURST2#:GOSUB 4280
4210   PAINT(X+30,Y+15),6,0
4220   GET(X,Y)-(X+60,Y+30),HOLD#:GOSUB 4280
4230   PUT(X,Y),HOLD#                                  'clear burst no. 2
4240   PUT(X,Y),STAR#
4250   'now change the value code:
4260   STARVAL(MODIFY)=-(STARVAL(MODIFY)+1)
4270 NEXT I
4280 'do the delay:
4290 FOR J=1 TO DELAY:NEXT J
4300 RETURN
5000 'a loss!
```

```
5010  FOR I=1 TO 8
5020   BEEP
5030   FOR J=1 TO 200:NEXT J
5040 NEXT I
5050 PAINT(0,0),1,1
5060 LOCATE 20,9
5070 COLOR 6,1
5080 PRINT"WELL, ARE YOU HAPPY NOW?  YOU'VE JUST MANAGED TO DESTROY THE"
5090 LOCATE 21,9
5100 PRINT"ENTIRE GALAXY WITH YOUR POOR SOLAR ENGINEERING!  HAVE A NICE"
5110 LOCATE 22,7
5120 PRINT"*COLD* TIME...IF YOU AREN'T TRAPPED BY AN EVENT HORIZON THAT IS!"
5130 LOCATE 24,10
5140 LINE INPUT"Willing to try again? <YES> ";X$:X$=LEFT$(X$,1)
5150 IF X$<>"N" AND X$<>"n" THEN COLOR 7,0:CLS:FLAG=-1:GOSUB 30:GOTO 1000
5160 COLOR 7,0:CLS:END
6000 'end the game
6010 COLOR 7,0
6020 CLS
6030 LOCATE 8,14:COLOR 6,4
6040 PRINT"Ah HA!  It WAS harder than you thought, wasn't it?":COLOR 7,0
6050 LOCATE 10,5
6060 IF TURNS=1 THEN WORD$="turn" ELSE WORD$="turns"
6070 PRINT"You used a total of";TURNS;WORD$;" in futile attempt at solving"
6080 LOCATE 12,7
6090 PRINT"this puzzle.  Well...just relax and try'again a little later on!"
6100 LOCATE 20,1:END
7000 'the instructions:
7010 CLS
7020 PRINT TAB(5);
     "This game is called 'SHOOTING STARS', because that is exactly what you"
7030 PRINT
     "are to do!  You will begin with a play field that looks like the follow-"
7040 PRINT"ing (only much bigger):":LOCATE 7,35
7050 COLOR 1,0:PRINT"o     o     o";
7060 LOCATE 9,35:PRINT"o     ";:COLOR 6,0:PRINT"*";:COLOR 1,0:PRINT"     o"
7070 LOCATE 11,35:PRINT"o     o     o"
7080 LOCATE 15,5:COLOR 7,0
7090 PRINT"The ";:COLOR 6:PRINT"STARS";:COLOR 7:PRINT" are in ";:COLOR 6:PRINT
     "yellow";:COLOR 7:PRINT" while the ";:COLOR 1:PRINT"blue circles";:
     COLOR 7:PRINT" are ";:COLOR 1:PRINT"BLACK HOLES";:COLOR 7:PRINT"."
7100 PRINT"You can only shoot the STARS (hence the name of the game) and NOT the"
7110 PRINT"BLACK HOLES.  If you DO attempt such a shot, the game will warn you"
7120 PRINT"and you will lose a turn!":GOSUB 8000
7130 PRINT TAB(5);
     "When you shoot a STAR, it becomes a BLACK HOLE and all of it's"
7140 PRINT
     "neighbors change states...that is...BLACK HOLES become STARS and STARS"
7150 PRINT"become BLACK HOLES.  The Stars and Black Holes are numbered as shown"
7160 PRINT"below:"
7170 COLOR 1:LOCATE 9,35:PRINT"1     2     3"
7180 LOCATE 11,35:PRINT"4     ";:COLOR 6:PRINT"5";:COLOR 1:PRINT"     6"
7190 LOCATE 13,35:PRINT"7     8     9"
7200 LOCATE 17,5:COLOR 7
7210 PRINT
     "Only certain neighbors change when a Star is shot.  These neighbors will"
7220 PRINT"be shown on the next page of the instructions, with the star that"
7230 PRINT"was shot AND those effected by the shot, shown in ";:COLOR 2:
     PRINT"green";:COLOR 7:PRINT"."
7240 GOSUB 8000
7250 PRINT"Star #1 Shot:";TAB(33);"Star #2 Shot:";TAB(67);"Star #3 Shot:"
7260 COLOR 2:PRINT"1     2";:COLOR 7:PRINT"     3";TAB(33);:COLOR 2:
     PRINT"1     2     3";:COLOR 7:PRINT TAB(67);"1     ";:COLOR 2:PRINT"2     3"
7270 LOCATE 4,1
7280 COLOR 2:PRINT"4     5";:COLOR 7:PRINT"     6";TAB(33);"4     5     6";TAB(67);
     "4     ";:COLOR 2:PRINT"5     6":COLOR 7
7290 LOCATE 6,1
7300 PRINT"7     8     9";TAB(33);"7     8     9";TAB(67);"7     8     9"
7310 LOCATE 8,1
7320 PRINT"Star #4 Shot:";TAB(33);"Star #5 Shot:";TAB(67);"Star #6 Shot:"
7330 COLOR 2:PRINT"1     ";:COLOR 7:PRINT"2     3";TAB(33);"1     ";:COLOR 2:
     PRINT"2     ";:COLOR 7:PRINT"3";TAB(67);"1     2     ";:COLOR 2:PRINT"3"
7340 LOCATE 11,1:PRINT"4     ";:COLOR 7:PRINT"5     6";TAB(33);:COLOR 2:
     PRINT"4     5     6";TAB(67);:COLOR 7:PRINT"4     5     ";:COLOR 2:PRINT"6"
7350 LOCATE 13,1:PRINT"7     ";:COLOR 7:PRINT"8     9";TAB(33);"7     ";:COLOR 2:
     PRINT"8     ";:COLOR 7:PRINT"9";TAB(67);"7     8     ";:COLOR 2:PRINT"9":
     COLOR 7
7360 LOCATE 15,1
7370 PRINT"Star #7 Shot:";TAB(33);"Star #8 Shot:";TAB(67);"Star #9 Shot:"
7380 PRINT"1     2     3";TAB(33);"1     2     3";TAB(67);"1     2     3"
7390 LOCATE 18,1:COLOR 2:PRINT"4     5     ";:COLOR 7:PRINT"6";TAB(33);
     "4     5     6";TAB(67);"4";:COLOR 2:PRINT"     5     6"
7400 LOCATE 20,1:PRINT"7     8     ";:COLOR 7:PRINT"9";TAB(33);:COLOR 2:PRINT
     "7     8     9";TAB(67);:COLOR 7:PRINT"7     ";:COLOR 2:PRINT"8     9":COLOR 7
7410 GOSUB 8000
7420 PRINT TAB(5);"The Object of the game is to reverse the original field so"
7430 PRINT"that instead of a single Star surrounded by eight Black Holes, you"
7440 PRINT"end up with a single Black Hole in the middle, surrounded by all"
7450 PRINT"Stars as below:":LOCATE 7,35:COLOR 6
7460 PRINT"*     *     *":LOCATE 9,35:PRINT"*     ";:COLOR 1:PRINT"o     ";:
     COLOR 6:PRINT"*":LOCATE 11,35:PRINT"*     *     *":COLOR 7:LOCATE 15,5
7470 PRINT"If you end up with all Black Holes, you lose.  You can give up at"
7480 PRINT"any time by entering 'Q'.  (The minimum amount of moves to solve"
7490 PRINT"this game is eleven).  GOOD LUCK!":GOSUB 8000
7500 LOCATE 5,6
7510 PRINT"If you care to REREAD these instructions, press <HELP> otherwise,"
7520 PRINT TAB(20);"press ANY other key to Start the Game!"
7530 X$=INKEY$:IF X$="" THEN 7530
7540 IF ASC(X$)=1 THEN 7010 ELSE CLS:RETURN
8000 'wait before displaying next text page
8010 COLOR 2,0
8020 LOCATE 23,1
8030 PRINT"Press ANY Key to Continue:";
8040 COLOR 7,0
8050 X$=INKEY$:IF X$="" THEN 8040
8060 X$=""
8070 CLS
8080 RETURN
10000 DATA 120020,280020,440020,120076,280076,440076,120132,280132,440132
10010 DATA Luke Skywalker You're Not!,Glad I'm Out of YOUR Galaxy!
10020 DATA You ARE Trying to Win Aren't You?,Way to Go 'DEADEYES'
10030 DATA Capt. Kirk Know You're Loose?,Good! Now Try for the Right One!
10040 DATA Darth Vader Wants YOU!,What a Shot (?)
10050 DATA It Can Be Done in 11 Turns You Know!,Are You STILL At It?
10060 DATA Wake Me When You're Through,Want the Instructions Again?
10070 DATA It's a Joke Right? You Can't Be This Bad,Nice Shot...No Win Though
```

Direct Cursor Addressing In A Real Time Environment

(or Having Fun With Your Computer)

Rex G. Bennett
1605 S. Riverbend
Green River, WY 82935

Many of you HACKERS out there have been reading the series of tutorials for better screen control by David E. Warnick each month. And you found yourself wanting to explore these methods more thoroughly and adapt the ideas to your every day skills. The use of direct cursor addressing in the computer game environment provides a sense of accomplishment as you and your family enjoy playing the games.

There are three ways to control the cursor on the H/Z-89 that I have used. In each case, the first thing that you must do is enable direct cursor addressing by sending ESC Y to the terminal. The way I like to do this is with the steps listed below as an MBASIC program.

```
10 ESC$=CHR$(27)  : REM escape character
20 DCA$=ESC$+"Y"  : REM enable direct cursor addressing
```

A this time you are now able to move the cursor to any place on the screen that your program requires. The three ways that I mentioned above are:

(1)

```
30 DEF FN PC$=DCA$+CHR$(31+ROW)+CHR$(31+COL) :
   REM define user function
40 ROW=12:COL=40 :REM row 12 column 40
50 PRINT FN PC$(ROW,COL);"HELLO" : REM move cursor and
   print "hello"
```

(2)

```
30 ROW=12:COL=40 : REM row 12 column 40
40 PRINT DCA$+CHR$(ROW+31)+CHR$(COL+31);"HELLO" :
   REM move cursor and print
```

(3)

```
30 PRINT DCA$+"+G";"HELLO" :REM move cursor,
   print "hello at row 12,column 40
```

Method three is the way that is explained in the HEATHKIT manuals. It does require less typing than the previous two examples. But it has even a more serious drawback. The problem lies in the addressing of the row and column coordinates. Once you run the program, the program cannot change the row and column coordinates. This is fine for titles and such but it does not allow us to have a game program with moving graphics awaiting player input. Also I do not like to look up the ASCII character that equals the row and column number wanted added to the number 31 each time I want to move the cursor. Thus I use the first two methods only. Method number one is the best to use for ease of program entry, but it has been my experience that this method is slower then method number two. This is important when you are trying to keep the action fast.

In both methods 1 and 2 we can allow our program to control the variables ROW and COL based on what the player inputs via the INKEY$ function. Please keep in mind that when we move a character on the screen, we must erase the old position by printing a space to the old row and column coordinates after we print the new character. The loop continues until a player presses a key that causes the operating parameters to change or the program branches out of the loop due to values of the variables matching control parameters.

In this game of PONG the program has four sets of row and column variables labeled as:

LR	= left player row position
LC	= left player column position
RR	= ight player row position
RC	= right player column position
BR	= all row position
BC	= ball column position
RO	= saved ball row position
CO	= saved ball column position

There is also control variables that determine paddle and ball direction and ball bounce. Such as the variable BV that controls ball bounce direction, this variable can take on three different values, 1, -1, or 0. It is added to the variable BR at each execution of the program loop. Let's say that BR (ball row) is now at the value of 10 and BV is set to 1. When we print the ball to the screen with the command line of PRINT FN PC$(BR+BV,BC);"↑" the ball is placed at row 11. This set of coordinates is then saved into the variables RO and CO so during the next loop of the program, the old ball position is erased after the new ball position is printed. If the variable BV is -1, then BR+BV=BR-1 and the ball move towards the top of the screen. Of course if BV=0 then the ball bounce is straight. The program is filled with remarks quite heavily so you can see the method to my madness. I recommend that you leave them out to help speed up the action. The only player input required is the four paddle control keys for left and right players (see listing). You may change these keys to any you prefer.

This game is somewhat slow at 2Mhz. But those of you that have converted to 4Mhz will find the action plenty fast and exciting to play. It will also get you well underway in designing and writing your own moving graphics to be used in as many applications as your imagination will allow. Also, any of you guys that are in the southwest Wyoming area, drop me a line, maybe we can get a users' group started.

```
1 REM ****** PONG.BAS ******
2 REM * BY REX G. BENNETT *
3 REM     * 02 Jan. 84 *
10 DEFINT A-Z:
   REM define all varibles as integer,runs faster
20 ESC$=CHR$(27): REM equates ESC$ to escape character
30 CLR$=ESC$+"E": REM equates CLR$ to clear screen
40 EGR$=ESC$+"F":
   REM equates EGR$ to enter graphics mode
50 XGR$=ESC$+"G": REM equates XGR$ to exit graphics mode
```

```
60 COFF$=ESC$+"x5": REM equates COFF$ to turn curser off
70 CON$=ESC$+"y5": REM equates CON$ to turn curser on
80 ECP$=ESC$+"Y":
   REM equates ECP$ to direct curser addressing
90 PRINT COFF$: REM turn curser off
100 DEF FN PC$(R,C)=ESC$+"Y"+CHR$(R+31)+CHR$(C+31):
    REM user defined function
110 PRINT CLR$;EGR$:
    REM clear screen and enter graphics mode
120 GOSUB 510: REM jump to screen draw and score routine
130 LC=3:RC=77:RA=1:LA=1:RO=10:CO=10:BC=40:BR=12:LR=12:
    RR=12: REM varibles set
140 RANDOMIZE PEEK(11): REM seed random generator
150 REM varible BD=ball direction (-2 = right to left),
    (2 = left to right)
160 BD=RND(1):IF BD>.5 THEN BD=2 ELSE BD=-2:
    REM random ball serve direction
170 REM start of program loop
180 PRINT FN PC$(LR,LC);"q":PRINT FN PC$(RR,RC);"q":
    REM left and right paddles
190 PRINT FN PC$(LR+LA,LC);" ":
    PRINT FN PC$(RR+RA,RC);" ": REM erase old paddles
200 RO=BR:CO=BC: REM save old ball position
210 A$=INKEY$:
    REM keyboard interrupt for player paddle control
220 IF A$="" THEN 280:
    REM jump to line 280 if null. saves time
230 IF A$="Z" THEN LR=LR+1:LA=-1:
    REM left player paddle down
240 IF A$="Q" THEN LR=LR-1:LA=1:
    REM left player paddle up
250 IF A$="/" THEN RR=RR+1:RA=-1:
    REM right player paddle down
260 IF A$="\" THEN RR=RR-1:RA=1:
    REM right player paddle up
270 REM move ball across screen bounce ball if it hits
    a border line
280 BC=BC+BD:BR=BR+BV:
    IF BR=<2 OR BR=>22 THEN IF BV=1 THEN BV=-1 ELSE BV=1
290 PRINT ECP$+CHR$(BR+31)+CHR$(BC+31);"↑":
    REM draw in ball at new position
300 PRINT ECP$+CHR$(RO+31)+CHR$(CO+31);" ":
    REM erase ball at old position
310 IF LR=BR AND BC=4 THEN BD=2:GOTO 380:
    REM ball is hit it bounces back
320 IF RR=BR AND BC=76 THEN BD=-2:GOTO 380:
    REM ball is hit it bounces back
330 IF BC=<2 AND BR<>LR THEN RS=RS+1: REM left player
    misses then right scores
340 IF BC=>78 AND BR<>RR THEN LS=LS+1: REM right player
    misses then left scores
350 IF BC<>2 AND BC<>78 THEN 180
360 IF RS>=15 OR LS>=15 THEN GOTO 430: REM a player
    scored 15 points game over
370 PRINT CLR$:GOTO 120: REM clear screen. jump to
    screen draw and score routine
380 RANDOMIZE PEEK(11):BV=RND(1): REM random vector for
    ball bounce when hit
390 IF BV<.45 THEN BV=-1: REM ball bounces upward
400 IF BV>.55 THEN BV=1 ELSE BV=0: REM ball bounces
    downward or straight
410 PRINT CHR$(7): REM ring bell when hit by paddle
420 GOTO 180: REM jump back to start of program loop
430 PRINT CLR$:PRINT FN PC$(13,35)" GAME OVER ":
    REM clear screen . game over
440 IF RS>=15 THEN PRINT FN PC$(14,32);
    " RIGHT PLAYER WINS "
450 IF LS>=15 THEN PRINT FN PC$(14,32);
    " LEFT PLAYER WINS "
460 PRINT FN PC$(15,35);LS;" TO ";RS: REM print the
    scores
470 PRINT XGR$: REM exit graphics mode
480 PRINT CON$: REM turn curser back on
490 STOP
500 REM redraw screen and update the scores
510 FOR I=1 TO 80
520 PRINT FN PC$(1,I);"i"
530 NEXT I
540 FOR I=80 TO 1 STEP -1
550 PRINT FN PC$(23,I);"i"
560 NEXT I
570 FOR I=1 TO 23
580 PRINT FN PC$(I,1);"i"
590 NEXT I
600 FOR I=23 TO 1 STEP -1
610 PRINT FN PC$(I,80);"i"
620 NEXT I
630 PRINT FN PC$(1,8);" SCORE ";LS;FN PC$(1,65);
    " SCORE ";RS
640 RETURN
```

About the Author:

***Rex Bennett** is a Production Automation Forman for a large oil company located in the southern Wyoming area, where they install and maintain automation systems that are used for data aquisition and control of gas producing wells. He has been teaching himself MBASIC, Fortran and Assembly languages on an H-89 with 48K and 1 hard sectored H-17 drive.*

It's your last chance!
Do it now!
Sign up for the International HUG Conference

See registration on page 5

Expand your H/Z89-90

from KRES

Make your computer a more useful friend

- Now you can add all of those extra boards you've always wanted (color graphics, speech, robotic control, parallel—to name a few)
- The Kres Expansion enables you to put seven full sized boards inside your H89 cabinet; no external cables or boxes.
- Plug in any boards without any software modification
- Simple to install and use

EXP-700
$395.00
+ $6.00 shipping and handling

KRES

P.O. Box 17328, Irvine, CA 92713
(818) 957-6322 or
(714) 559-1047

"My Favorite Subroutines"

Dear HUG,

I think your feature "My Favorite Subroutines" is a GREAT idea and i thank fellow HUGgie, Bob Moskus, for getting it started. I offer a humble contribution in support. This subroutine is written for MBASIC v. 5.21 (CP/M) and illustrates a neat use of the logical features of MBASIC for retrieving single-letter answers to program branching questions. The short program below demonstrates the use of MBASIC's relational operators to compare two values in string expressions. The result of the comparison is either "true" (-1) or "false" (0). The arithmetic result can be used in an ON ... GOTO ... statement (as shown below) to make a decision regarding program flow.

```
 10 PRINT CHR$(27);"E RELATIONAL OPERATOR SUBROUTINE";
 20 PRINT " DEMONSTRATION"
 30 A=0:PRINT
 40 PRINT "Please try a YES or NO answer ...";
 50 GOSUB 1000
 60 ON R GOTO 80,90,100
 70 ' *** The subroutine returns R as 1, 2 or 3 only ***
 80 PRINT "You answered YES":IF A=0 THEN 110 ELSE 30
 90 PRINT "You answered NO":IF A=0 THEN 110 ELSE 130
100 PRINT "You answered something other than YES or NO!"
110 A=1:PRINT "Would you like to try it again?  ";
120 GOTO 50
130 END
1000         *** S U B R O U T I N E ***
1010 R$=INPUT$(1):PRINT R$
1020 R=2*((R$="y")+(R$="Y"))+(R$="n")+(R$="N")+3
1030 RETURN
```

Keep up the fine publication. It gets better and better!

David M. Kletter
15 Russel Avenue
Fort Monmouth, NJ 07703

Dear HUG,

Your helpful subroutine articles are indeed very helpful. I have incorporated a routine sent in by Ralph Stiewe into the enclosed short program which transforms even the lowliest computer into a word processor.

It would greatly aid these old eyes if most programs submitted for reprint could be listed in upper case. Very small lower case printing in your magazine and others is extremely difficult to read without misreading a few characters several times, especially in 'C' which uses so many brackets, curly brackets, and the like.

```
10 'Program name JUSTIFY.BAS             programmer F Kent
20 'Purpose: Right justify printer output   19-Aug-83
30 'Method: Program parses entire file on the first pass
40 'in order to determine the length of the longest line
50 'therein.  Following this it recommends that line
60 'length and proceeds to insert extra spaces between
70 'words in order to fill out each line unless the line
80 'is too short and prints out hard copy.  Can be easily
90 'modified to permit user to select his own line length.
100' Great for those who don't want to go broke buying
110 'word processors.  ENJOY!!!
120 '
130 PRINT CHR$(27)+"E":PRINT:'Clear screen for CP/M
140 LINE INPUT "Enter filename to be printed   ";FILE$
150 OPEN "I",1,FILE$
160 IF EOF(1) THEN 190
170 INPUT #1,A$:IF LEN(A$)>W THEN W=LEN(A$)
180 GOTO 160
190 W=W+1:PRINT"Recommended column width=";W:
    REM W=LENGTH OF WIDEST STRING
200 GOSUB 440
210 CLOSE 1:OPEN "I",1,FILE$
220 DIM Q$(80)
230 IF EOF(1) THEN CLOSE:POKE 3,IOBYTE:END
240 FOR R=1 TO W
250 X$=INPUT$(1,1):REM   INPUT LINE AS 1 CHAR STRINGS
260 Q$(R)=X$:REM  SAVE 1 CHARACTER STRINGS IN DIM Q$(80)
270 IF X$=CHR$(10) THEN 290:
    REM LINE TERMINATOR CHARACTER--START NEW LINE
280 NEXT R
290 COUNT=W+2-R:REM   LINE LENGTH
300 IF R<W-10 THEN 400:
    REM   SKIP SPACING IF LINE IS SHORT
310 FOR X=1 TO R
320 PRINT Q$(X);:IF Q$(X)=" " THEN GOSUB 380:
    REM ADD SPACE IF NEEDED
330 NEXT X:REM    & OUTPUT CHARACTERS TO PRINTER
340 GOTO 230
350 '
360 'Subroutine to locate spaces in input line
370 '
380 COUNT=COUNT-1:IF COUNT>0 THEN PRINT" ";:
    REM   DECREMENT COUNT TO ZERO
390 RETURN
400 FOR X=1 TO R:PRINT Q$(X);:NEXT X:GOTO 230:
    REM  SHORT LINE  NO EXTRA SPACES
410 '
420 ' -Printer switch subroutine-
430 '
440 IOBYTE=PEEK(3):TEMP=IOBYTE AND 252 OR 2
450 PRINT"Do you wish output on the console";
    " or on the printer?"
460 LINE INPUT "TYPE 'C' OR 'P'";A$
470 IF A$="C" THEN PRINT"This file will";
    " appear on the console":RETURN
480 IF A$="P" THEN PRINT "This file will appear on";
    " the printer":POKE 3,TEMP:RETURN
```

Fred W. Kent
1057 Lake Rd.
Conneaut, OH 44030

Dear HUG,

Here is a subroutine to print the system date (and time, with appropriate software) on the printer. It is for use with MBASIC and HDOS 2.0. If the time is desired, "CK.DVD" and "TIME.ABS" from HUG disk IX (P/N 885-1064) are also required. The routine assumes the CK: has been set (by TIME.ABS) and both CK: and LP: (the printer device) have been loaded.

```
10000 'TIME AND DATE ON PRINTER
10010 D9$=""
10020 FORI=8383 TO 8391: D9$=D9$+CHR$(PEEK(I)):NEXT
10030 FOR I=1TO2
10040 OPEN "I",2,"CK:": T$(I)=INPUT$(8,2): CLOSE#2: NEXT
10050 IF T$(1) <> T$(2) THEN 10040
10060 OPEN"O",1,"LP:"
10070 PRINT#1,TAB(70);CHR$(15);D9$;CHR$(18):
      PRINT#1,TAB(70);CHR$(15);t$(1);CHR$(18):
      PRINT#1,:PRINT#1,: CLOSE: RETURN
```

Lines 10000 - 10020 access the system date and place it in D9$. Lines 10030 - 10050 access the time from CK: and place it in T$(1). Lines 10060 - 10070 print the date and time on the printer, and return to the main program. If you do not have CK.DVD, delete lines 10030 and 10040. Also delete the portion of line 10070 that prints t$(1). The CHR$() codes in line 10070 cause the date and time to be printed in condensed mode on an Epson printer. If this is undesirable, delete them.

Once the routine is entered, it should be saved as an ASCII file (i.e. SAVE "PRINTDAT.BAS",A). It can then be MERGEd to any existing MBASIC program, and accessed via the statement GOSUB 10000.

Walter M. Scott III
7608 Luscombe
Knoxville, TN 37919

Dear HUG,

Here is an alternative timing method to the one that appeared in the last issue, under My Favorite Subroutines.

W = Wait Interval in seconds. Store in ASCII mode under WAIT.BAS. Requires only a one time merge. GOSUB 64000 from main program as often as you like. Especially good if you intend to compile later as no changes will be required.

```
64000 T=TIME:WHILE TIME<=T+W:WEND:RETURN
```

George Holt
403 2nd St. West
BAFB, LA 71110

Dear HUG,

This subroutine converts the hexadecimal number (HI$) to the decimal number (DEC). This program can be changed to convert any number system to decimal by changing the number 16 in line 260 to the base of the new number system. If the new number system's base is larger than 10, then you have to use characters and numbers so you will need to change the IF THEN statement in line 280 to check for the characters you are using.

```
350 DEC=0:A=0
360 L=LEN(HI$)
370 C=16↑(L-1)
380 FOR I%=1 TO L
390 IF MID$(HI$,I%,1)=>"A" AND MID$(HI$,I%,1)<="F" THEN 420
400 A=VAL(MID$(HI$,I%,1)):DEC=DEC+A*C
410 C=C/16:NEXT I%:RETURN
420 A=ASC(MID$(HI$,I%,1)):A=A-55:DEC=DEC+A*C
430 GOTO 410
```

William Sandlin
8 Georgian Drive
Newnan, GA 30263
(404) 253-4071

Local HUG Club News

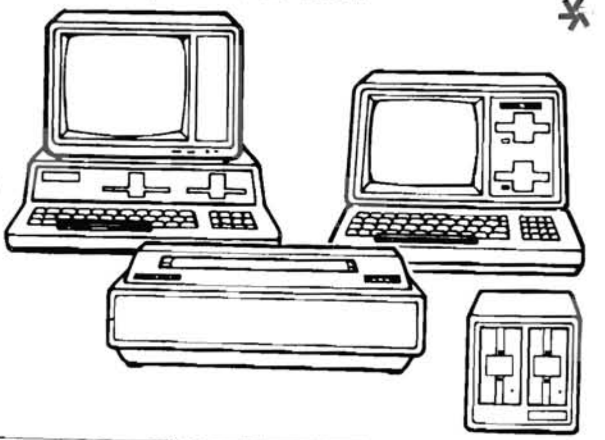
Rex Bennett
1605 S. Riverbend
Green River, WY 82935
(307) 875-3779
Looking to contact others in his area to possibly start a group in the Rock Springs - Green River area.

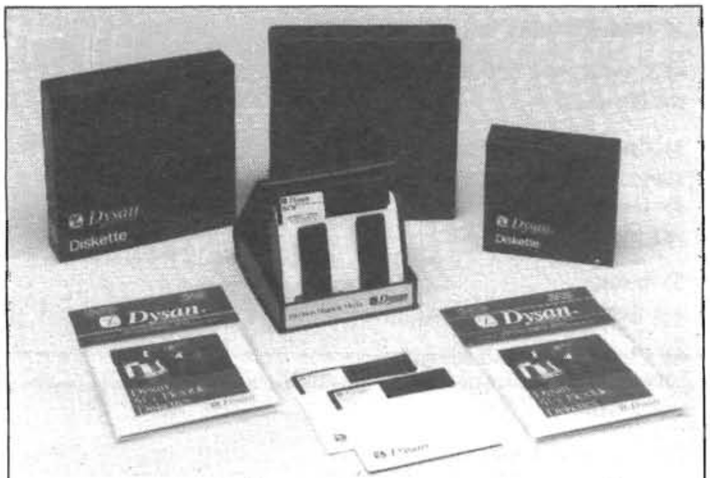
NY, Syracuse
SYRHUG
Box 6
Oran, NY 13125
(315) 682-2358
Contact Person: Garrett Voss
Group Size: 8 (1st meeting)
Meet 1st Wed at 7:00 p.m.

OH, Cleveland
NOHUG (Northeastern Ohio HUG)
7838 Valley Villas Dr.
Parma, OH 44130
Contact Person: Don Danko, Sec.
Phone: (216) 845-6752
Group Size: 54
Meet 2nd & 4th Thurs. 7 p.m. at St. Gregorys Church

FL, Hialeah
MACC (Miami Amateur Computer Club)
Contact: Charlie Robertson, Sec.
1542 Texas Ave.
Homestead AFB, FL 33039
(305) 257-3070
Meet 2nd Thurs. ea. month
7:00 p.m. - 9:30 p.m. at
Heathkit Electronics Center
4705 West 16th Ave.
Hialeah, FL 33012
Group Size: 65
Bulletin Board: (305) 823-2281
Fri. 6:00 p.m. - 9:00 a.m.
Sat. 5:00 p.m. - 9:00 a.m.
Sun. all day

UK Users' Group
Zenith Data System Limited,
Bristol Road,
Gloucester,GL2 6EE,England.
Tel: 0452 29451
Contact Person: Philip Meek
I thought it was time to let you know that we have a user group over here in England. We have been a small community up until now, but with the Z-100 series taking off, membership has increased. At the moment we have 75 members with quite a few more in the pipe line. At present we do not hold meetings as the users are far and wide around the UK. With the increase of U.S. forces personnel over here, we are making contact so I wonder if you could put a little piece in the REMark saying that Zenith is alive and well in the UK, as most are surprised we are here!!!





Dysan. The Finest Quality Diskettes Available.

Dysan's 5¼" or 8" diskettes are available single- or double-sided, and in single- or double-density formats.

Dysan's advanced production techniques assure that each Dysan diskette is certified 100% error free. So you can count on Dysan for your personal computers.

So give us a call. Or stop by. We have a complete inventory of the finest quality diskettes on hand. Dysan diskettes.

Call or write for our complete catalog.

Studio Computers inc. 999 South Adams Birmingham, MI 48011 Phone (313)-645-5365

Dysan is a registered trademark of Dysan Corp.

WatchWord

VERSION 2

The word processor and full screen editor for the Z100 and ZDOS.

NEW FEATURES:

- *Files of arbitrary length*
- *Extensive configuration facility*
- *Multiple fonts loaded from ALTCHR files*
- *Support for ZDOS Version 2 subdirectories*

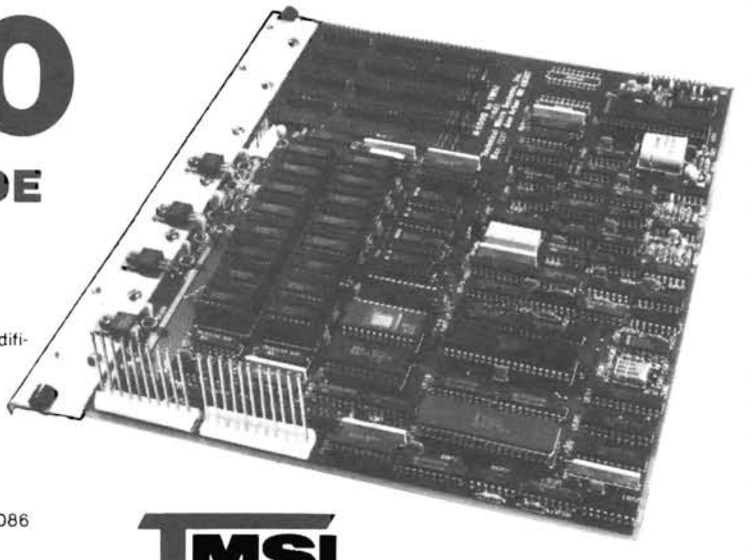
Plus all of the features of Version 1 including: superscripts, subscripts, underlining, boldface, centering, formatting, microjustification, arbitrary line length with horizontal scrolling, word wrap, auto save, split screen, macros, user settable defaults and definable keys, global search and replace with settable zone.

For additional details, send a SASE.
Demo Disk: $3.00
WatchWord with Manual: $100.00

S & K Technology, Inc.
4610 Spotted Oak Woods
San Antonio, TX 78249
512-492-3384
ATTN: Steve Robbins

H-1000

A Z80/8086 UPGRADE FOR THE H89/Z89



HARDWARE

- plug-in replacement for the H89/Z89 CPU board; no modifications required
- dual CPUs: Z80 and 8086
- **256K RAM standard; sockets for up to 1 megabyte RAM**
- 5 I/O slots
- faster program execution: 2/4 MHz for Z80, 8MHz for 8086
- fully compatible with all Heath/Zenith peripherals

SOFTWARE

- runs all Heath/Zenith software without modification
- compatible with Zenith Z100 and IBM Personal Computer
- choice of MSDOS or CP/M-86 for the 8086
- supplied with diagnostic software package
- "soft disk" feature: copies an entire disk in RAM for instant disk access
- supports multi-user and multi-task operating systems

TMSI
Technical Micro Systems Inc.
P.O. Box 7227, Dept. H
366 Cloverdale • Ann Arbor, Michigan 48107
(313) 994-0784

We accept MasterCard and VISA.
Serious Dealer Inquiries Invited

The H-1000 is a quality product of Technical Micro Systems, Inc., manufacturers of innovative microcomputer systems since 1979.

H-1000 and TMSI are trademarks of Technical Micro Systems, Inc. H89, Z89, Z100 and HDOS are trademarks of Heath/Zenith Corp., Benton Harbor, Michigan. MSDOS is a trademark of Microsoft, Bellevue, Washington. CP/M and CP/M-86 are trademarks of Digital Research, Inc., Pacific Grove, California.

Vectored from 6

A Fix For the CP/M DUMP Utility in the Feb. Issue

Dear HUG,

A comment and a fix for the CP/M DUMP utility in the February issue of REMark. I entered the patches immediately after I got that issue, used it, and then went on a trip and did not get back to it until now, so by now you have probably gotten other comments as well.

1) In paragraph 2 on page 28, the last line has JNZ NUNUM, it should be JNZ NONUM, the trivial one.

2) In paragraph 4, which builds the ASCII buffer, the line of code rshould be changed to `"CPI    '~'+1"`. This allows the characters (, l,) and ~ to be displayed.

3) Now for the fix. I noticed that the last line of a file would not display the ASCII characters. Looking at the code, it is apparent that the program exits on the last line of a file before typing the last buffer. One simple fix is:

a. Change the fourth line after GLOOP: from "JC FINIS" to "JC PRTASC".

b. In the print ASCII buffer routine, add the label "PRTASC:" in front of the code containing the PUSHes, and add a line immediately after it with "PUSH PSW   ;Save the program status word as well".

c. Move the label "LINENO:" to a new line in front of the POPs where it now is, to one containing "LINENO: POP PSW   ;balance the pops and pushes". The carry bit now has the correct value from the test done in GNB indicating if the end is at hand.

d. After the POPs line, add a line "JC FINIS".

It works. I tried it on several different files and the program DUMPed the entire file every time. Oh yes, I am using Magnolia Microsystems distribution DUMP.ASM. If there is another version around, my problem may not have shown up.

Keep up the good work. KEYMAP is the best buy around.

Adolph Fejfar
3017 Oriole Drive
Sierra Vista, AZ 85635

Modifications To "FONTED.BAS"

Dear Mr. Gillespie,

I was thumbing through some back issues of REMark (I joined HUG in Jan. 84) and a certain Z-100 article caught my attention. Mark Aagenas' FONTED.BAS in the July 1983 issue is a very good ZBASIC program to create and modify Z-100 ALTCHAR.SYS type files. I typed in the program and ran it. It does indeed allow one to create and/or modify the key map and character fonts in the Z-100. In this article, Mr. Aagenas had encouraged modifications to his program. FONTED.BAS uses 25th line prompts, but the 25th line does not always "behave" (it does not clear completely) or look as aesthetic as it should (typed in entries merge with the existing 25th line characters already on the screen resulting in a confusing presentation). FONTED.BAS, in its present form, will not work properly when compiled. It has been a few months since Mr. Aagenas' program was published in REMark, but I have not found any modifications to FONTED.BAS in any issues since then. For those HUG members who are still interested, I offer the following modifications to FONTED.BAS which will present a clear, much improved 25th line presentation and will allow FONTED.BAS to work properly when compiled (the compiled version runs much faster).

1) Insert:
```
15 BLK$-STRING$(79." ")
```

2) Change line 630 to read:
```
LOCATE 25,41
```

3) Change line 930 to read:
```
INPUT "represented by  the  1st number is struck."
```
4) Insert:
```
931 PRINT
```

5) Insert:
```
935 INPUT "When ready to proceed, hit return";ZZZZ
```

6) Change line 960 to read:
```
LOCATE 25,1:PRINT"Input key number in hex (FF to exit)-->";:
GOSUB 610
```

7) Change line 980 to read:
```
LOCATE 25,44:SCREEN 0,1:PRINT"Present value --> ";
HEX$(KYBD%(A3))
```

8) Change line 1000 to read:
```
LOCATE 25,65:PRINT"New value --> ";:GOSUB 640:SCREEN 0,0
```

9) Change line 1760 to read:
```
LOCATE 25,1:PRINT"Found character, will display"
```

10) Change line 2570 to read:
```
LOCATE 25,1:SCREEN 0,0:PRINT BLK$;:RETURN
```

11) Change line 2620 to read:
```
LOCATE 25,1:PRINT"Save character in font array? (Y/N)";:
GOSUB 2770
```

12) Change line 2750 to read:
```
LOCATE 25,1:PRINT"Exit back to main menu? (Y/N)";:
GOSUB 2770
```

If you are going to compile FONTED.BAS, remember to use SAVE'FONTED',A. By the way, Mr. Aagenas mentioned in his article we need a good chess game for the Z-100 and that this program created the men. For all HUG Z-100 chess enthusiasts, MYCHESS is now available from The Software Toolworks (ZDOS, full graphics, color or monochrome) for $34.95 plus $2.00 shipping.

Richard J. Komar
1408-H Paegelow
Scott AFB, IL 62225

More On the "Dreaded HT Bug"

Dear HUG,

L. T. Scotney, in a letter to Buggin' HUG in the April 1984 issue of REMark, warned readers of the "Dreaded HT Bug". The problem which he spoke of is that ZBASIC and other BASICs "filter" TAB characters which are sent to the printer using the LPRINT statement by turning them into an equivalent number of SPACE characters. Most of the time this is a great convenience since many printers do not automatically expand TABs to every 8th column. This feature becomes a pain, however, when one is sending "bit image" graphics streams to the printer. In this mode the characters sent represent the actual pixels which are to be set. If one of the bit patterns happens to be 00001001 (decimal 09, which is the TAB character), however, we have a problem since this will be translated into one or more SPACE characters, which have the bit pattern 00100000 (decimal 32).

Mr. Scotney suggests one option for handling this problem: compile the program using the ZBASIC compiler which apparently handles printer output in a different fashion. I have several more suggestions.

The simplest way to handle this problem is to tolerate a slightly imperfect picture. If you're not a perfectionist, then simply check each character before it is sent and change all the 9's to, say, 8's. Since a typical picture may contain only one or two bit patterns which result in a 9, this will hardly ever be a noticeable change. (A dump of the Z-100 screen will have 144,000 pixels - what's one or two wrong out of 144,000?)

With regard to the C.Itoh 8510 printer Mr. Scotney is using, there is a second way which should give a perfect picture. This approach is to set the printer to "7 bit" mode (the 8th bit of every byte received will be ignored). He can then set the 8th bit of every character sent to the printer, knowing full well that this bit will be ignored. The TAB characters will then be translated into decimal 137 which will NOT be converted to SPACEs by the LPRINT statement. The disadvantage of this method is that other software (e.g. LOTUS 1-2-3) may expect the printer to be in 8 bit mode.

The third way is to bypass the LPRINT statement altogether using ZBASIC's OUT statement to send the data directly to the printer port. This method is tricky since you'll have to handle all the "handshaking" yourself (using the WAIT statement). This should be possible but frankly I was unable to get it to work when I tried a simple example.

Glenn F. Roberts, Ph.D.
9035 F Countrywood Drive
Knoxville, TN 37923

More On Using 'C'

Dear HUG,

It is certainly commendable that several authors for REMark have written articles about using C. It is a fine language that can suffice for nearly every microcomputer activity. But, many of these writers are, unfortunately, presenting C programs that are written to look like BASIC. An excellent example of this is in Brian Polk's article in REMark 47. The program, MORE.C, is a very nice utility, but the manner in which it is given violates most of the modularity which can be obtained in C. It is extremely difficult to read, and all the gory details are right in the main program itself.

It is terribly important for teachers of C to avoid this style of programming. It is bad practice, and it will encourage this same sort of thing in new C programmers. Enclosed is the same program packaged in a much more readable style. Function calls are made whenever possible, and the main program itself is completely clear. The details of screen formatting are hidden in the various functions. The result is a program easy to write and change. And the modules themselves can be used in other programs as well.

Again, Mr. Polk should be encouraged to continue his writings for REMark. Many can profit from this information. But please, have him do it with style. The reason we program in C around here is so we don't have to use BASIC.

```
#include <printf.h>
#define EOF -1
#define ESC 27
#define stderr 0            /* terminal */

main ()
{
     int c, line_number;

     line_number = 0;       /* initialize the count */
     while ((c = getchar()) != EOF) {
          putchar(c);
          if (c == '\n')
               line_number += 1;
          if (line_number == 23) {
               pause();
               line_number = 0;
          }
     }
}

pause()              /* wait for user to type RETURN */
{
     open25();                 /* open the 25th line */
     savcurs();                /* save the cursor */
     curspos(25,1);            /* go to line 25, column 1 */
     cursoff();                /* turn off the cursor */
     revvid();                 /* enter reverse video */
     printf("—More—");
     normvid();                /* back to normal video */
     while (getc(stderr)  != '\n')        /* wait */
          ;
     rstcurs();                /* restore cursor */
     curson();                 /* turn it back on */
     close25();
     return;
}

open25()                       /* open the 25th line */
{
     printf("%cx1",ESC);
     return;
}

close25()                      /* close the 25th line */
{
     printf("%cy1",ESC);
     return;
}

curspos(x,y)        /* position the cursor. */
                    /* Enter with line and column number */
int x, y;
{
     x = x + 31; /* add the offset */
     y = y + 31;
     printf("%cY%c%c",ESC,x,y);
     return;
}

savcurs()           /* save the cursor position */
{
     printf("%cj", ESC);
     return;
}

rstcurs()           /* restore the cursor position */
{
     printf("%ck", ESC);
     return;
}

cursoff()                 /* turn off the cursor */
{
     printf("%cx5",ESC);
     return;
}

curson()                  /* turn the cursor on */
{
     printf("%cy5",ESC);
     return;
}

revvid()                  /* go to reverse video */
{
     printf("%cp",ESC);
     return;
}
```

```
normvid()               /* normal video */
{
    printf("%cq",ESC);
    return;
}
```

William S. Hall
1811 Willowtree Lane, B6
Ann Arbor, MI 48105

"TERM"

Dear HUG,

The algorithm by William M. Adney (REMark February 1984) to address the screen can be very useful, but it is not necessary to write a separate routine for each of the commands that are wanted. They can all be put into an instructive routine "TERM" and the command "TERM CLS" will clear the screen.

The routines to do this are given in the accompanying listing which illustrates an algorithm using the default file control block to insert a command name while converting it to upper case.

The default file control block is located at 05CH and anything typed after the space following a file name will be found there.

The operating system converts the string to upper case and it can be used by the programmer to locate a command line from those listed in an index of commands.

If the command "TERM" is used alone, it will cause the printing of the whole list of possible commands.

The command "TERM", followed by any other string will result in a search for that string and execution of the code at the address located. If no address is located, an error message is returned.

TERM.COM occupies very little file space and can serve as a simple utility to insert rulers on the screen, turn on graphics, or give you a different kind of cursor.

Once loaded into memory by typing "TERM", the separate commands can be executed by using GO.COM.

GO.COM is created by typing SAVE 0 GO.COM.

GO CLS will then erase the screen.

To place the ruler on the 25th line and then erase the screen, type:

```
TERM RULER<RETURN> GO CLS<RETURN>
```

Assuming that you have previously created the file GO.COM as indicated above.

The index search makes use of the operations of the compiler

When it compiles the program, the DW CLS is matched to the program counter address where it is located in the program. You can check this by looking at the PRN listing that ASM.COM makes. This is located just after the DB 'CLS',EOB which is matched by the command string returned from the file control block. As you can see, we are making the operating system do all of the work in programming for us. A neat way to go.

The listing is generously commented and the details for operation should be apparent by reading your way into it as you type it in with your editor.

Call it TERM.ASM, assemble it with ASM TERM. Create the .COM file with LOAD TERM. It will then be ready to run.

If you use ED.COM with the "v" option turned on, you will have corresponding line numbers on the screen to guide your eyes as you type. It makes it much easier to do an error free job.

Try ED.COM, it is really a very useful utility.

Listing "TERM.ASM"

```
;
;TERM.ASM a program to use the terminal commands in CP/M
; Robert W. Rasch
; 1801 Galen Drive
; Johnson City, TN 37601
; version (2.29.84)
;
BDOS EQU 05H
TPA  EQU 100H
CR   EQU 0DH
LF   EQU 0AH
ESC  EQU 27
CMND EQU 5CH+1
PSTR EQU 09H
PCHR EQU 02H
CHR  EQU 01H
NULL EQU 00H
EOB  EQU 0FFH
        ORG TPA
START:  LXI H,0             ;SET UP THE STACK
        DAD SP
        SHLD OLDSP          ;SAVE THE OLD STACK, AVOID
                            ; WARM BOOT
        LXI SP,STKTOP
        LXI H,CMND          ;POINT TO A POSSIBLE COMMAND
        MOV A,M
        CPI ' '             ;IS THERE ONE?
        JZ PLIST            ;NO GIVE THEM THE LIST OF
                            ; COMMANDS
STAR0:  LXI D,INDEX         ;POINT TO PRIMARY INDEX OF
                            ; COMMANDS
STAR1:  LXI H,CMND          ;RESET OR SET COMMAND LINE TO
                            ; START
STAR2:  LDAX D              ;PUT FIRST CHARACTER IN (A)
        CMP M               ;MATCH?
        JZ CONT             ;MATCHING
        INR A               ;FFH  IN INDEX, FOUND THE EOB
        JZ PMSG             ;(DE) POINTS TO ADDRESS OF
                            ; MESSAGE
STAR3:                      ;LOOP TO EOB OR QUIT AT NULL
        LDAX D              ;END OF INDEX?
        ORA A               ;ZERO?
        JZ ERXIT            ;YES, FOUND END OF INDEX AND
                            ; NO COMMAND
        INR A               ;NO, WAS IT EOB?
        JZ CONT2            ;FOUND AN INDEX END NOW SKIP
                            ; TWO BYTES
        INX D               ;NEXT
        JMP STAR3
CONT:   INX H
CONT0:  INX D               ;BUMP BOTH STILL MATCHING
        JMP STAR2
CONT2:  INX D               ;TWO BYTES
        INX D               ;PAST THE 0FFH
        INX D               ;NEXT TO WORK ON
        JMP STAR1           ;NEW INDEX SET TRY FOR MATCH
;PRINT MESSAGE UNTIL $
PMSG:   XCHG                ;(HL)==> MESSAGE
        INX H               ;PASS EOB
        MOV E,M
        INX H
        MOV D,M
PMSG2:  PUSH H!PUSH B!PUSH D
        MVI C,PSTR
        CALL BDOS
        POP D!POP B!POP H
        JMP EXIT
;THIS IS A LISTING OF THE POSSIBLE COMMANDS
;THE COMPILER INSERTS THE ADDRESS OF THE PROGRAM LOCATION
;WHERE THE COMMAND IS CARRIED OUT
;SEARCHING THIS INDEX RESULTS IN A POINTER TO THE CORRECT
```

```
;EXECUTION LOCATION.  THEY CAN BE IN ANY ORDER.
;
INDEX:
        DB 'QUIET',EOB
        DW QUIET
        DB 'CLICK',EOB
        DW CLICK
        DB 'BLOCK',EOB
        DW BLOCK
        DB 'UNDER',EOB
        DW UNDER
        DB 'CUROFF',EOB
        DW CUROFF
        DB 'CURON',EOB
        DW CURON
        DB 'HSCREEN',EOB
        DW HSCREEN
        DB 'OSCREEN',EOB
        DW OSCREEN
        DB 'RULER',EOB
        DW RULER
        DB 'DISABLE',EOB
        DW DISABLE
        DB 'GRAPHIC',EOB
        DW GRAPHIC
        DB 'NGRAPHIC',EOB
        DW NGRAPHIC
        DB 'CLS',EOB
        DW CLS
        DB 'RVIDEO',EOB
        DW RVIDEO
        DB 'NVIDEO',EOB
        DW NVIDEO
        DB NULL

;COMMANDS EXECUTION IS CARRIED OUT HERE
;THE INDEX LOCATION (ABOVE) POINTS TO HERE
;THE PROGRAM THEN PICKS UP THE PROPER CODE FOR EXECUTION
;OF THE COMMAND
;
QUIET:  DB ESC,'x2$'
CLICK:  DB ESC,'y2$'
BLOCK:  DB ESC,'x4$'
UNDER:  DB ESC,'y4$'
CUROFF: DB ESC,'x5$'
CURON:  DB ESC,'y5$'
HSCREEN:DB ESC,'x3$'
OSCREEN:DB ESC,'y3$'
RULER:  DB ESC,'x1',ESC,'Y8 '
        DB '123456789012345678901234567890'
        DB '12345678901234567890123456789012345'
        DB ESC,'H$'
DISABLE:DB ESC,'y1$'
GRAPHIC:DB ESC,'F$'
NGRAPHIC:DB ESC,'G$'
CLS:    DB ESC,'E$'
RVIDEO: DB ESC,'p$'
NVIDEO: DB ESC,'q$'

PLIST:  PUSH H!PUSH B!PUSH D
        LXI D,MESS1
        MVI C,PSTR
        CALL BDOS
        POP D!POP B!POP H
        JMP EXIT
ERXIT:  LXI D,ERMESS
        JMP PMSG2
EXIT:   LHLD OLDSP
        SPHL
        RET
;MESSAGES AND THE INDEX
MESS1:  DB 'H19 TERMINAL SETTINGS ',CR,LF
        DB 'USAGE: term xxx.'
        DB 'Where xxx is any of the commands'
        DB ' listed below.',CR,LF,LF
        DB 'QUIET: disable the key click'
        DB CR,LF
        DB 'CLICK: enable the key click'
        DB CR,LF
        DB 'BLOCK: display a block cursor'
        DB CR,LF
        DB 'UNDER: display an underline cursor'
        DB CR,LF
        DB 'CUROFF: shutoff the cursor'
        DB CR,LF
        DB 'CURON: display the cursor'
        DB CR,LF
        DB 'HSCREEN: enter the hold screen mode'
        DB CR,LF
        DB 'OSCREEN: exit the hold screen mode'
        DB CR,LF
        DB 'RULER: enable the 25th line and'
        DB ' put a ruler there'
        DB CR,LF
        DB 'DISABLE: disable the 25th line'
        DB CR,LF
        DB 'GRAPHIC: enable Graphic characters'
        DB CR,LF
        DB 'NGRAPHIC: disable Graphic characters'
        DB 'RVIDEO: reverse video'
        DB CR,LF
        DB 'NVIDEO: normal video'
        DB CR,LF
        DB 'CLS: clear the screen and home cursor'
        DB CR,LF,'$'
ERMESS: DB 'SORRY, that command not yet'
        DB ' available',CR,LF,'$'
OLDSP:  DS 2
;STACK AREA
        DS 64
STKTOP:
        END
```

Robert W. Rasch
1801 Galen Drive
Johnson City, TN 37601

Changing your address? Be sure and let us know since the software catalog and REMark are mailed bulk rate and it is not forwarded or returned.

CUT ALONG THIS LINE

HUG MEMBERSHIP RENEWAL FORM

HUG ID Number: ______________________

Check your ID card for your expiration date.

IS THE INFORMATION ON THE REVERSE SIDE CORRECT? IF NOT, FILL IN BELOW.

Name ______________________

Address ______________________

City-State ______________________

Zip ______________________

REMEMBER - ENCLOSE CHECK OR MONEY ORDER

CHECK THE APPROPRIATE BOX AND RETURN TO HUG

	NEW MEMBERSHIP RATES	RENEWAL RATES	
US DOMESTIC	$20 ☐	$17 ☐	
CANADA	$22 ☐	$19 ☐	US FUNDS
INTERNAT'L*	$30 ☐	$24 ☐	US FUNDS

* Membership in France and Belgium is acquired through the local distributor at the prevailing rate.

1984 SPRING/SUMMER CATALOG

If you'd like to receive one of the most comprehensive catalogs related specifically to Heath-Zenith products, call us today! We'll send you our fully illustrated 24 page catalog free for the asking.

Some of the items you'll find include the Zenith Data Systems line of Z100 and Z150/160 computers as well as systems hardware and software accessories. Plus our own line of add-on peripherals such as thinline eight-inch drive subsystems at really great prices. Featured are some of the best prices around on MPI printers, Z100's, Tandon and Qume disk drives and premium quality Dysan diskettes. New products include the Prism line of high speed color printers, U.S. Robotics' S100 modems, Z100 color graphic game software, Transtar letter quality printers and more.

We've been serving the Heath-Zenith community since 1979. We're an authorized Zenith Data Systems dealer and MPI distributor and service center. Our reputation for fast service and excellent support is our most important asset!

We have thousands of happy customers, including some of the largest corporations in the world, governmental agencies and some very distinguished universities. Just call or write, and we'll be happy to send our new catalog out the same day.

Local Customers — Visit Our Retail Store!

999 South Adams
Birmingham, MI 48011
Phone (313)-645-5365

ZENITH | data systems

Studio Computers inc.

Index of Advertisers

Heath/ZENITH Users' Group
Hilltop Road
Saint Joseph, Michigan 49085

BULK RATE
U.S. Postage
PAID
Heath Users' Group

Volume 5, Issue 7

POSTMASTER: If undeliverable, please do not return.

885-2054