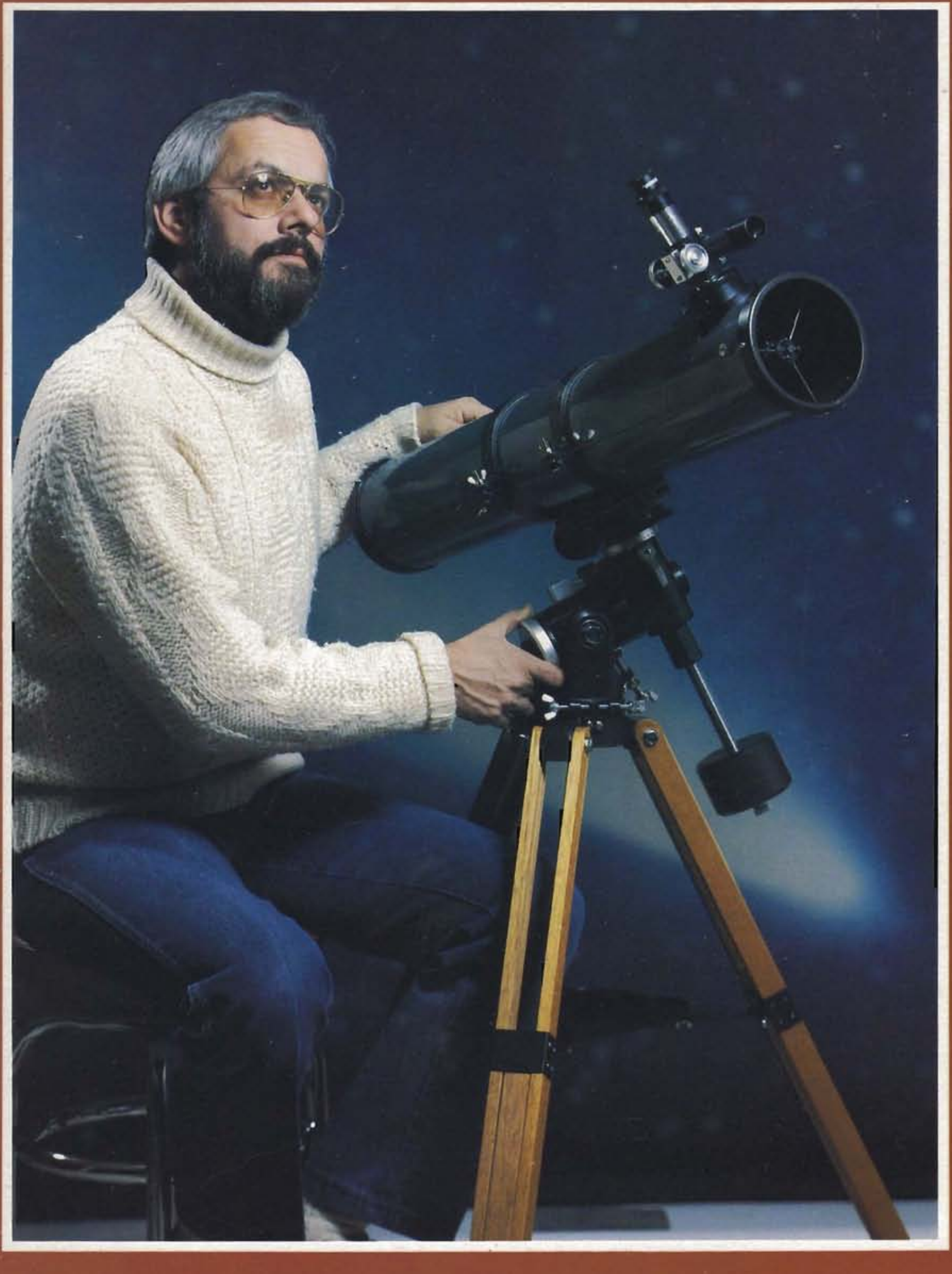
REMark®

$2.50

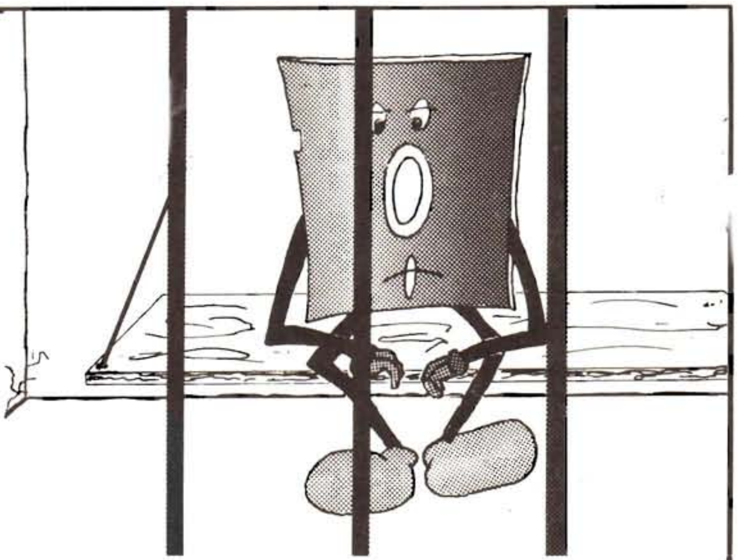
Volume 6, Issue 9 • September 1985
P/N 885-2068 Issue 68

Official magazine for users of HEATH ZENITH computer equipment.



micro/VERSAL™

the complete file transfer package For IBM-PCs and Zenith Z100s



Free your data with micro/VERSAL™

With micro/VERSAL™ you can READ, WRITE and FORMAT any of the following formats:

- 8 inch standard (SSSD)*
- Avatar TC 1 (SSDD)
- Avatar TC 1 (DSDD)
- Cromemco CDOS - I (SSSD)*
- Cromemco CDOS - II (SSDD)
- Cromemco CDOS - II (DSDD)
- Datavue (SSDD)
- Datavue (DSDD)
- Epson QX-10 (SSDD)
- Epson QX-10 (DSDD)
- HDOS (SSSD)*†
- HDOS (SSDD)
- HDOS (DS00)†
- Heath With Magnolia (SSDD)
- Heath With Magnolia (DSDD)
- Heath With Magnolia (SSSD)*
- Heath With Magnolia CP/M 2.24 (SSDD)
- Heath With Magnolia CP/M 2.24 (DSDD)
- Heath With Magnolia CP/M 2.24 (SSSD)*
- IBM-PC/CPM-86 (SSDD)
- IBM-PC/CPM-86 (DSDD)
- Kaypro II (SSDD)
- Osborne (SSSD)*
- Osborne (SSDD)
- Osborne (DSDD)
- Osborne (Ozmosis SSDD)
- Osborne (Ozmosis DSDD)
- Radio Shack TRS-80 CP/M (SSDD)
- Radio Shack TRS-80 CP/M (DSDD)
- Radio Shack TRS-80 CP/M (SSSD)*
- Radio Shack TRS-80 I DOS (SSDD)*†
- Radio Shack TRS-80 DOS 1.3 (SSDD)*†
- Radio Shack TRS-80 DOS 2 (SSDD)*†
- Radio Shack TRS-80 I DOS (SSSD)*†
- Radio Shack TRS-80 NEWDOS (SSSD)*†
- Lobo MAX-80 (SSDD)
- Lobo MAX-80 (DSDD)
- Morrow MD2 (SSDD)
- Morrow MD3 (DSDD)
- Morrow MD11 (DSDD)
- NEC PC (8001) (SSDD)
- NEC PC (8031) (SSDD)
- NEC PC (8031) (DSDD)
- Sanyo (SSDD)
- Sanyo (DSDD)
- Superbrain 2*
- Televideo (SSDD)
- TI Professional (SSDD)
- Xerox 820-I (SSSD)*
- Xerox 820 (SSDD)
- Xerox 820 (DSDD)
- Zenith Z-100/CPM-85 (SSDD)
- Zenith Z-100/CPM-85 (DSDD)
- Zenith Z-90 (SSDD)
- Zenith Z-90 (DSDD)
- Zenith Z-90 CP/M 2.24 (SSDD)
- Zenith Z-90 CP/M 2.24 (DSDD)
- Zenith Z-89 (SSSD)*

*Not available on PC version
† Formatting option not supported

(48 PTI SOFT-SECTORED OR 8 INCH*)

micro/VERSAL™ is easy to use – menu driven, and requires only 64K and 2 floppy drives or 1 floppy and a hard disk; runs under both DOS and DOS2.

You also get utilities to help you transfer your files:

- **ANYCPM™** – define your own CP/M formats for machines not listed on our formats menu.
- **SED** – a disk sector editor lets you examine and modify any bite on any disk, even CP/M disks.
- **COPY/FORMAT** disks to your own specifications.
- **RECOVER** accidentally deleted files.
- Utilities to modify files to **MS-DOS** standards from other operating systems.

only micro/VERSAL™ gives you EVERYTHING YOU NEED FOR ONE LOW PRICE:

$49.95

plus $4.00 shipping & handling

Send check, money order, M/C or VISA numbers to:

Also from AST is our COED full screen editor, with macro commands, user Definable function keys, UNDELETE, Multiple files, built in calculator, on disk tutorial, and much more. COED is FAST, works with BASIC programs and has NO LIMIT on file size . . . Lose those EDLINE blues **$34.95**

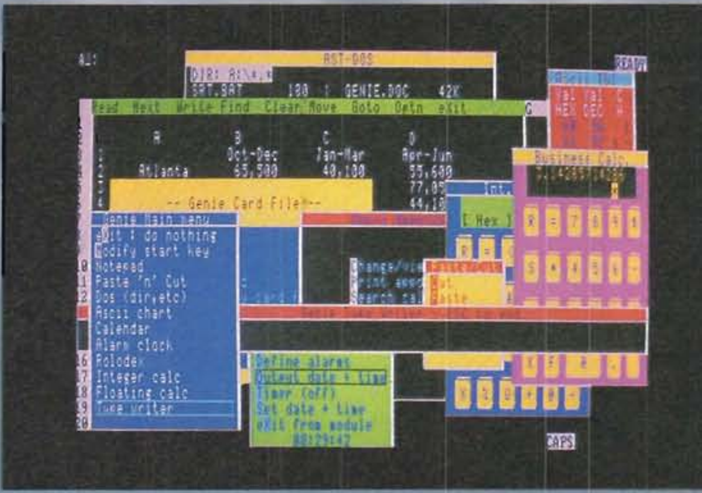
ADVANCED SOFTWARE TECHNOLOGIES

**452 West 47th Street
New York, NY 10036
(212) 247-0150**

NOT PROTECTED

Genie™

Software Magic from Advanced Software Technologies



Shown here is Genie "popped up" on a Z-110 running Lotus 123. From the left are: The Genie main menu, the Genie rolodex style card file, the Genie notepad containing data cut from Lotus, the Genie DOS performing a directory command, the Genie alarm clock (at the bottom,) the Genie typewriter, Genie calendar, Genie Cut and paste, Genie Calculators, and the Genie Ascii table.

	Sidekick	Perks	Genie
Key Macros	+69.95	No	**YES**
Dos Services	No	Yes	**YES**
Notepad	50K	4K	**No Limit**
Ascii Table	Yes	Yes	**YES**
Calculators	One	One	**TWO**
Typewriter	No	No	**YES**
Calendar	Yes	Yes	**YES**
Rolodex	Yes	No	**YES**
Cut and Paste	No	No	**YES**
Alarm	No	Yes	**YES**
Expandable	No	No	**YES**
	$120	$100	**$49.95**

Special Introductory Offer:

$49.95

plus $4.00 shipping and handling
New York residents add 8.25% sales tax

To order send MC/VISA check or M.O. to:

ADVANCED SOFTWARE TECHNOLOGIES

**452 West 47th Street
New York, NY 10036
(212) 247-0150**

Genie is completely resident so that once you load it, it is always there. A single key stroke lets you command the powers of Genie. Have Genie take notes, schedule appointments, move data between your applications, perform calculations, and much more. When you finish, Genie goes away and waits for your next command.

Genie is available for the Zenith Z-100 and IBM-PC compatibles. Genie requires MS-DOS version 2 or above.

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

Printers .... Imperial Printing St. Joseph, MI

	U.S. Domestic	APO/FPO & All Others
Initial	$20.00	$35.00*
Renewal	$17.00	$30.00*

* U.S. Funds

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

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

REMark®

Official magazine for users of HEATH ZENITH computer equipment.

Volume 6, Issue 9 • September 1985

Index of Advertisers

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

21st Century Data, Inc.	66
Advanced Software Technologies	2,3
Analytical Products	43
Barry A. Watzman	68
Bersearch Information Services	16
CDR Systems, Inc.	37
Consumer Product Development	62
DelSoft	54
FBE Research, Inc.	47
FINA Software	63
Floppy Disk Services	6,7
Future Communications	47
International Computer Srvc, Inc.	63
Intersecting Concepts	36
KC Computers	64
M.E. Pittman	49
McGaffey Engineering	66
Paul F. Herman	43
Redwood Development	54
Ross Custom Electronics	26
S&K Technology	64
Secured Computer Systems, Inc.	48
Sextant Publishing Company	27
Software Toolworks	65
Software Wizardry	19
Systems Innovations Inc.	50
Veritechnology Electronics Corp.	17

On The Cover: *Soon to appear in our solar system will be Halley's Comet, which is the subject of this month's feature article on page 10. To view this phenomenon may we suggest the Model TL-400 Telescope available in the current Heathkit catalog.*

GREAT DRIVES!

The H89 TWOET systems Just Got Hard to beat!

We've got a great idea for your H-88, 89 or 90. It's a dual internal half height drive system. Two of our half height 5¼" drives can replace your built-in disk drive, doubling your information storage capacity.

Floppy Disk Services provides you with everything you need. That's two double-sided, double density, half height drives in either 48 or 96 tpi format, all hardware, cables and power connector adaptors. And most important, you get easy, step-by-step instructions, in the Heath/Zenith tradition of good, clear documentation.

We've thoroughly tested the TWOET/Heath set-up. And now by popular demand, the new 10 megabyte hard disk system! We are now able to offer you the storage you want and need for a fair price! The hard disk system comes with all software needed to run with the 17 or 37 controller and/or the Magnolia Double Density controller.

NEW!

Model TWOET 455

2 Shugart SA-455 half height
48 tpi double sided
All hardware
Metal, shielded mounting plates
Data cable with chassis connector
Power "Y" connector
Complete documentation
Price....... $325.00

Model TWOET 55F

Two Teac 55F half height
96 tpi double sided
All hardware
Metal, shielded mounting plates
Data cable with chassis connector
Power"Y" connector
Complete documentation
Price....... $353.00

HARD DISK for H-89 $995.00 Internal

SA-455 DS 48tpi ½ hgt 5.25"	125.00
FDD-100-8 SS/DD 8"	125.0
Teac 55F 96tpi DS/DD half height	139.0
FDD-200-8 DS/DD 8"	***Sale 185.00***
Taxan 415 RGB Monitor	299.00
Data connectors of all types	***CALL***
Power connectors for all drives	***CALL***
Controllers	***CALL***
Hard disk Z150/160	Internal 750.00

Wondering what to do with your internal drive if you buy this system?

Here's the solution. If you purchase a dual half height system for your Heath computer from Floppy Disk Services, just include an extra $65.00 plus shipping and receive a single 5¼ case with power supply and data cable ready to receive your SIEMENS internal drive! The case with data cable is normally an $85.00 item. And the cable that comes with your TWOET system includes the external chassis disk I/O connector.

Due to production deadlines, this ad is 2 months old at time of printing. We encourage you to call for our latest pricing and catalog requests.

Dealer inquiries invited.

We accept cash, MasterCharge, Visa, personal checks (allow 10 days to clear the bank), wire transfers, money orders, purchase orders from government agencies, and approved businesses. Call for info.

Floppy Disk Services inc. has been supplying custom enclosures and disk drives to the hobby and professional market for 6 years now! We have specifically supported the Heath/Zenith community with a high degree of success. We care about you and your system integrity. That's why we have been a success in a time when many others are going out of business. We are proud to be a part of the Heath/Zenith community and will continue our support with new products and ideas.

H-88/89/90 and Z-150/160 are registered trademarks of Heath Corp.

GREAT PRICES!

SA-455

Shugart SA-455 half height 5¼ is the standard for excellence in the disk drive community. Backed by a one year warranty and manufactured by the leader of drive technology, Shugart Corp. Ready to ship at $139.00 ea. Compatible with ALL of the Heath/Zenith computers.

Taxan RGB Monitor

The Taxan (vision III) model 415 monitor is a high resolution RGB monitor designed to work with an IBM, Apple or compatible computer with an RGB output. 640 X 262 line resolution. $299.00 each.
IBM or Apple interface cable $15.00 each.

Maynard and DTC

We can supply PC compatible upgrade boards for your Z-150, Z160 system. Hard disk controllers, memory upgrade, and space saving Maynard modular boards. As an example, you can install a Maynard Modular 5 & 8 controller that will run 5¼, 8 inch and optional boards such as a clock/calendar module, game port, serial port or parallel port. Call for details on the slick way to upgrade and save a slot at the same time...

SA-712 & MicroScience

Half height 10 megabyte (20mb available soon) hard disk drives. The MicroScience has a plated media and is available with a full size bezel option. The Shugart has an on board low current option and is backed by a one year warranty. It also has a full size bezel option. both drives are available for the H/Z-89, H/Z-150, H/Z-160. $525.00 each.

Irwin-110

Data loss! It's a terrible thought. Minimize your chance of lost programs and data files with the new 110 tape drive from Floppy Disk Services inc. It works off the standard 5¼ controller which means you save money over standard tape backups that you must buy the controller for. The data cartridge holds 10mb of data and can retrieve in streaming or individual file form. Software included. Available in many configurations. For the IBM-PC, Z-150 or Z-160 only. (CP/M software being worked on). Complete system only $795.00

Connectors

Floppy Disk Services inc. can supply custom data and power cables for your needs. We use only the best! UL rated or listed components, and all power connectors are crimped by our AMP electric certified crimper. No hand crimping to vary the degree of accuracy. Only the best! Floppy Disk Services inc. can make 1 or 1000 connectors to your specs, fast! Call and ask for details...

Special Sale!

Because of our enormous buying power in the industry we have made an excellent purchase of disk drives for your Heath/Zenith computer. Two Teac 55F 80 track double sided drives, in case with power supply and sheilded data cable, your price $363.00. Single drive system, $229.00 each.

Enclosures!

We manufacture and stock enclosures for every disk drive on this page and for almost any other drive you are likely to find. Available assembled and tested with state-of-the-art power supplies, these enclosures house a wide combination of half- and full-height drives to complement most available computers.

Toll Free Order Line: (800) 223-0306

Tech Help or Info: (609) 799-4440

39 Everett Drive Bldg. D, Lawrenceville, New Jersey 08648

FLOPPY DISK SERVICES™ INC.

BUGGIN' HUG

Clearing WordStar Screens

Dear HUG:

I installed the patches for WordStar (REMark, Vol. 6, No. 5, page 80) and liked all of the improvements, but disliked the fact that the screen did not clear when you quit. Since most of my work is word processing, I use an AUTOEXEC.BAT file when booting up, which loads the keymapper and WordStar among other things. To get the screen to clear, I created a file with EDLIN called CLS. It is one line — ESC E (this is simply F8 followed by E and then ^Z out). I then added two lines to the end of the BAT file:

```
A:
TYPE CLS
```

The reason for the A: is that I keep all my files on B: and log it as the default drive. Adding the A: line prevents the need of having CLS on every disk. I hope this is helpful to someone else.

Sincerely,

James Spinti
121 C Asbury Drive
Wilmore, KY 40390

Looking To Exchange Information

Dear HUG:

I would like to exchange information with Zenith ZW-111-32, MS-DOS 2.18 folk who are trying to do any of the following:

1. Upgrade to 1024 by 1024 color graphics.
2. Interface to a CIPHER model 525 "Floppy Tape" cartridge tape drive.
3. Save normal resolution Zenith color graphics to video tape.
4. Locate Winchester diagnostic software.
5. Interface to an Optimem 1000 optical disk drive via SCSI.
6. Locate word processing software that will handle both English and Russian.

Bob Broedel
Meteorology Department, 404 Love
Florida State University
Tallahassee, FL 32306

Problems With Buffer

Dear HUG:

I recently purchased a Quadram Microfazer to work between my Z-89 and C. Itoh (TEC) model 1500 printer. I have the Z-89 and the buffer input set at 19,200 baud, and the buffer output and printer set at 1200 baud, using no protocol. A 10K file dumps into the buffer in about 20 seconds, and then feeds my printer quite normally.

But I have a problem. The same 10K file fed through WordStar (version 3.0) takes over 3 minutes to load, although it will print out normally. Additionally, if I then try to duplicate the printout using the Microfazer "Copy" control, apparently all the WordStar commands have been stripped from the file!

MicroPro, in answer to my query (and many follow-ups!), says they "don't support buffer operation." This seems odd, to say the least. Most printers have a buffer of some size, and I have not read in the literature of WordStar or any word processing program having such a problem. To date, Quadram has not answered any of my requests for assistance. My dealer has been unable to help me.

Has anyone out there run into this problem? Better yet, do they have a solution?

Winston Bugg
P.O. Box 96
Greensburo, GA 30642

Hints For Operating PIE Under CP/M

Dear HUG:

In reference to Mr. Jack Miner's letter on page 65 of Volume 6, Issue 6, June 1985 REMark, "BIOS Error on B: R/O", I have experienced the same problem, the random failure of the CP/M 2.2.02 or 2.2.03 operating system to save a file while operating under PIE or PIE8. The computer loses track of its disk assignments or tries to write the file to a previously assigned R/O drive or to a non-existent device with the resulting error message "BIOS error on B: R/O" or "BIOS error on (X):R/O" or "BDOS ERROR ON (X) SELECT", thereby hanging the system even if the drive is present and active.

While I am using an H-8, the type of failure that he experienced on his H-89 happens on my machine almost regularly. Although it has been upgraded several times since I built it, the mode of failure is always the same as his, but with the additional problem that the keyboard on the H-19 will die for no apparent reason. This happens only when using PIE/PIE8 and not any other machine language or interpreted program. The only solution is to REBOOT.

I installed a power line monitor and found the machine very stable. I tried to isolate a "goosey" memory chip by re-addressing the memory boards (both dynamic and static) to no avail. I have even stopped my wife from doing the laundry while I am writing files under PIE/PIE8, but the failures still occur.

Since these failures often occur when the dryer is being unloaded or when a universal type appliance is being used in close proximity (50') to the computer, e.g. hair dryer, lawn mower, electric drill, blender, mixer, etc. I suspect that the primary cause of failure is a high frequency or static discharge radio frequency spike that is detected in the computer, scrambling the operating system.

In the operating instructions for PIE there is a section "IN CASE OF EMERGENCY" referring to this problem (page 13), but the solution seems to create an even larger mess to fix. It appears that "Software Toolworks" is aware of the keyboard problem, and that there may be a glitch in the way that PIE communicates with the CP/M BIOS. The author recommends a file save under Control-V at least every fifteen minutes to avoid disaster.

In order to minimize this problem I created a file "SKELTON. FOR" with my "TEXT" instructions in it on my "A" (R/O) disk which I transfer to the object drive (B-G) that I am going to write my new file to under a new name (DEV:Nnnnnn.EXT) and assures (at least in the beginning) that I can access the object drive (B–G), and that the file has been created. I now seldom lose a whole file when these random failures occur.

Good practice dictates that the operator never install a new disk or open the drive door after PIE/PIE8 is loaded since the drive reset procedure is not reliable. Should it ever be necessary to exchange a disk, go to the system level and access it at least once under STAT so that the operating system doesn't lose it after a warm boot or assume that it is R/O.

Respectfully yours,

Paul Woolgar
15272 Oak Drive
Renton, WA 98055

Problems With Printers And Flight Simulator

Dear HUG:

I wrote some time ago about the color problems with Flight Simulator and with the Mannsman–Tally printer. It is now time for a follow–up note, with some questions.

First, I solved the problem with the M–T printer by selling it and buying an Epson FX–80. It is nice to have a printer that responds properly. Maybe much of my problem was inexperience, but there were programs which would never drive it, and it became too much of a source of frustration (especially with a non–computer husband). The manual was written for the very experienced, which didn't help me since I am in such a learning mode. The Epson manual is a joy to use.

I still have some problems which you can possibly help me with. Several programs still do not dump to the printer properly unless I am using PC–DOS 2.11. This is, of course, another source of frustration. I can create the file, but then I have to reboot with PC–DOS to print it. Three programs to note are PCPG from the IBM blue, DRHALO and Mind Prober. In the case of PCPG and Mind Prober, I get a "timeout warning message in line xyz" and the program reverts to DOS. DRHALO does strange line spacing.

It appears the problem must lie with the operating system which Zenith uses, since the same programs work fine on Sperry PC using Sperry MS–DOS 2.11 (with turbo (Snicker, Snicker)). I would like to know if Zenith or anyone else has come up with some kind of solution to this problem. I am sure that I am not the first person to encounter printer problems of this nature.

The second problem is the color in the Flight Simulator. I was in a shop here in Albuquerque (Hi Tech), and they had Flight Simulator running in proper colors! I grabbed the salesman and asked him what had happened to correct the colors. He had bent back a leg on one of the chips which seemed to correct the colors. However, he didn't have any documentation and couldn't remember which chip was altered. He said the new CPU's were coming out with the fix already installed. I would certainly like to have the fix for my machine. I have noticed that on other programs, the Zenith will put a different color border on the screen than the Sperry PC or the IBM will. Maybe the fix will correct the border problem also.

I also wish the HUG magazine was more oriented to the MS–DOS user. It is obviously a good magazine for the CPU side of Zenith, but not that much for the MS–DOS side. I suppose this is as it should be since most magazines are geared for the IBM compatibles.

I will continue my subscription of HUG for what does come my way because (arguments do occur over this) it is specific to the Zenith. Other magazines fill the rest of the bill, such as PC WORLD and PC WEEK.

As another note, I added a 10 MG external hard disk to my system, and I love it. At work I have a Sperry Model 50 (Hi res. color, 1 disk drive, 1 internal 10 MG drive) and I will take my Zenith over the Sperry any day.

Next comes the modem, as soon as I can talk my husband into it.

Thanks for any help you can give me.

Jo Ann Mantych
916 Tramway Lane NE
Albuquerque, NM 87122

Set Up To Talk To Tektronics Terminals

Dear HUG:

I am presently using my H89 to communicate with a mainframe at work. The graphics package for the mainframe is set up to talk to a number of tektronics terminals, namely the 4010, 4013, 4014, and 4062.

I would appreciate any information concerning hardware and/or software, that would allow the H89 to emulate any of these terminals. Thank you.

P. Marshall
716-1375
Prince of Wales Drive
Ottawa Ontario
Canada K2C 3J8

"Interfacing The Prometheus PRO1200 Clock To The H/Z-100"

Dear HUG:

I recently submitted an assembly language program "Interfacing the PROMETHEUS PRO1200 Clock to the H/Z–100," which was rejected as not meeting the criteria for publication in REMark. I find the program to be most useful, and perhaps some of the H/Z–100 users of the Prometheus Modem would like to use the clock to set the 'time/date' at boot–up of their systems. I will be happy to send the source code, as well as the object code to anyone sending an initialized disk and an SASE to me at the address given below.

T.L. Vinson
RR1, Box 175
Pine Island, MN 55963

Simple Memory Expansion

Dear HUG:

I have owned and enjoyed an H89A for some time now, and with the recent price cuts I decided to acquire another one. After I got my new unit up and running, it occurred to me that I was 16k of RAM short. (My original H89A has the 64k upgrade) I wasn't too thrilled with the thought of buying another upgrade, so I de-

Continued on Page 63 ☛

It's A Bird . . . It's A Plane . . . It's HALLEY'S COMET!

Jim Tursa
3489-C Lake Austin Boulevard
Austin, TX 78703

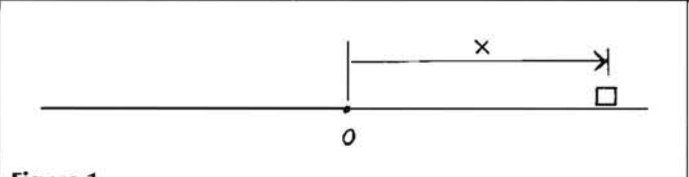


Figure 1

Introduction

For those of you who have access to binoculars or a telescope of any size, this winter will provide an opportunity to view the return of Halley's Comet to the inner solar system. The show will not be as spectacular as past shows mainly for two reasons. First, air and light pollution are much greater now than they have been in the past. Second, the geometry of the Earth–Halley encounter will not be as favorable this winter as it has been in the past (more on this later). In spite of those minor drawbacks, I would urge everyone to make an effort to see this once in a lifetime event. This article is a short tutorial on how to numerically simulate the motion of Halley's Comet around the Sun on a computer. A program is then listed which incorporates the material to be developed and uses the outstanding color graphics capabilities of the Z-100 computer to produce visual displays of the results. This program can be very educational and enjoyable to use even without a thorough understanding of the simulation concepts presented. A general knowledge of algebra and vectors is assumed.

Numerical Simulation

Let the variable x represent the position of some object relative to some reference point O (see Figure 1). The value of x may be positive or negative, indicating that it is either to the right or to the left of O. The velocity of the object will be represented by $\dot{x}$ (pronounced "x dot"). This value may also be positive or negative indicating the direction of motion. The dot above the x is the common notation for "time rate of change" or "derivative with respect to time". For example, the time rate of change of velocity, known as acceleration, is represented by $\ddot{x}$ (pronounced "x double dot"). The value of x may depend on the time t, and will be denoted as x(t). The velocity, however, may depend on both time and position, denoted by $\dot{x}$(t,x(t)).

Suppose we are given the values of the position and velocity at some specific time T. That is, we know the values of x(T) and $\dot{x}$(T,x(T)). (I am using capital T to indicate a specific point in time, as distinguished from lower case t which indicates an arbitrary time variable). We wish to predict the position at some future time T+dT, denoted as x(T+dT), where dT is some small time change. We could use the formula:

```
distance travelled = rate * time
```

which applies only for constant rates, as an approximation to our situation. Algebraically, this is:

```
x(T+dT) - x(T) ~= ẋ(T,x(T)) * dT
```

Solving for the desired quantity x(T+dT) yields:

```
x(T+dT) ~= x(T) + ẋ(T,x(T))*dT
```

The above formula will be converted into an equality as follows:

```
xe(T+dT) = x(T) + ẋ(T,x(T))*dT
```

The xe notation is used to indicate that the quantity xe(T+dT) is only an estimate of the true value x(T+dT), and is not exact. Pictorially we have the situation represented in Figure 2. $\dot{x}$(T,x(T)) is the slope of the tangent line to the x(t) curve at the point (T,x(T)). This method of predicting x(T+dT) is known as Euler's Method, named after the German mathematician Euler (pronounced "Oiler"). Generically we could represent this method by the following equation:

```
x(T+dT) ~= x(T) + slope*dT
```

However, we can do better than this. Consider Figure 3. Euler's Method uses just the slope of line1 to obtain an estimate of x(T+dT). It makes sense that an average of the slopes of line1 and line2 would work better. The slope of line1 is $\dot{x}$(T,x(T)). The slope of line2 is $\dot{x}$(T+dT,x(T+dT)). Plugging the average of these slopes into our generic representation of Euler's Method yields the following:

```
x(T+dT) ~= x(T) + (1/2)*( ẋ(T,x(T)) + ẋ(T+dT,x(T+dT)) )*dT
```

You may have noticed something interesting about this equation. The quantity being estimated, x(T+dT), appears on the right hand side! That is, we cannot plug values into the right hand side to obtain our estimate, since one of those values is unknown. To rectify the situation, we will use our first method to help us out. In place of x(T+dT) on the right hand side we will use xe(T+dT). The resulting formula, where xme(T+dT) is used on the left side for the same reason that the xe notation was used above, is Modified Euler's Method:

```
xme(T+dT) = x(T) + (1/2)*( ẋ(T,dT) + ẋ(T+dT,xe(T+dT)) )*dT
```

The improved accuracy obtained is usually well worth the extra computation involved. This is the method used in the HALLEY program. There are, as you might have already guessed, even better methods available. A particular class of methods known as Runge-Kutta integrators includes methods of varying accuracy and complexity, but they all boil down to the general form:

```
x(T+dT) ~= x(T) + (weighted average of slopes)*dT
```

where the weights add up to one. (Note that the weights of the Modified Euler's Method add up to one, i.e. 1/2 + 1/2 = 1). For example, the 4th Order Runge-Kutta method is essentially:

```
x(T+dT) ~= (1/6)*(slope1 + 2*slope2 + 2*slope3 + slope4)*dT
```

Note that 1/6 + 2/6 + 2/6 + 1/6 = 1. A discussion of how the weights and slopes are obtained is beyond the scope of this article; however, it should be noted here that Euler's Method is in fact a 1st Order Runge-Kutta Method, and the Modified Euler's Method is a 2nd Order Runge-Kutta Method. (Generally, the higher "Order" indicates increased accuracy). Finally, there are even methods that use past information (ie x(T-dT) and ẋ(T-dT,x(T-dT)), as well as present information to predict the future.

Relative Two Body Motion

So how does all of this apply to the motion of Halley's Comet you ask? To answer that we must turn our attention to Newton's Laws of Motion and Gravitation. Together, they combine to form what are known as the relative equations of motion for the Two Body Problem:

$$\ddot{\vec{r}} = - \frac{G*(m+ms)}{r\text{^}3} * \vec{r}$$

where

$\vec{r}$ = the position vector of the comet relative to the Sun
r = the magnitude (positive) of the vector $\vec{r}$
m = the mass of the comet
ms = the mass of the sun .
G = the Universal Constant of Gravitation

That is, given the position of the comet relative to the Sun, $\vec{r}$, we can plug it into the right hand side of the above equation and obtain its acceleration relative to the Sun, $\ddot{\vec{r}}$. Since ms is very much larger than m, the approximation G*(m+ms) ~= G*(ms) is often used, with the notation μ = G*(ms) (μ is pronounced "mu", as in "cute"). Thus we have:

$$\ddot{\vec{r}} = -(\mu/r\text{^}3)*\vec{r}$$

If we use x, y, and z coordinates for r, then we get the three equations:

$\ddot{x} = -(\mu/r\text{^}3)*x$, $\ddot{y} = -(\mu/r\text{^}3)*y$, $\ddot{z} = -(\mu/r\text{^}3)*z$

where r= sqr(x^2 + y^2 + z^2).

That's all fine and well you say, but the numerical simulation method we developed was for equations with single dots (ie ẋ), not double dots. That is true, but suppose we let ẋ = vx, ẏ = vy, ż = vz. Then the three previous double dot equations are equivalent to the following six single dot equations:

$\dot{x} = vx$, $\dot{y} = vy$, $\dot{z} = vz$
$\dot{vx} = -(\mu/r\text{^}3)*x$, $\dot{vy} = -(\mu/r\text{^}3)*y$, $\dot{vz} = -(\mu/r\text{^}3)*z$

To see this, dot both sides of the ẋ = vx equation (this is ok to do)

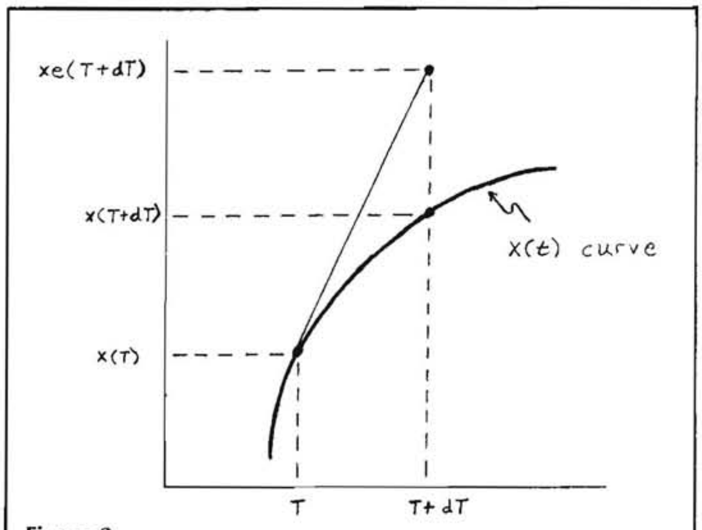


Figure 2

to obtain ẍ = v̇x. This combines with the v̇x = -(μ/r^3)*x equation to give ẍ = -(μ/r^3)*x , the original double dot equation. A similar situation applies to the y and z equations as well. The six single dot equations are the ones used in program HALLEY. In fact, these same equations can be used for any object orbiting the sun. We simply need to use different initial positions and velocities for each object. Thus, with a little extra effort we can include planets in our simulation as well as Comet Halley. All nine planets are supported in the program: Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune, and Pluto. (In addition, we could substitute μ of the Earth, G*(mass of the Earth), for μ of the Sun and use these equations to simulate the motion of Earth orbiting satellites such as the moon or the shuttle orbiter. This last feature is not supported in program HALLEY, however). In reality, of course, there are other forces acting on Comet Halley besides the gravitational attraction of the Sun. These include the gravitational attraction of the planets, solar radiation pressure from the Sun, and out-gassing effects due to the heating of the comet among others. In fact, all of the planets are affected by forces other than just the gravitational attraction of the Sun. These forces are not included in the program because they are not necessary to obtain the needed accuracy. The program is only intended for relatively short term simulations on the order of 2 years, and is accurate to within about 2% of the true positions. This is adequate for the graphics displays used. Of course, you may run the program for many more years than just 2, but I have not checked the accuracy for these cases.

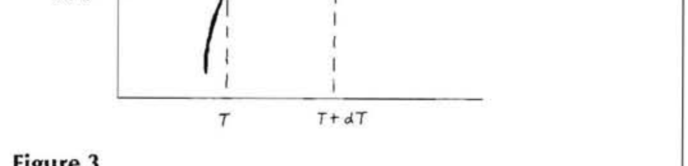


Figure 3

Graphics Displays

There are two different display methods used, which will be called SPACE and RA–DEC. With the SPACE method, the positions of the planets (and Comet Halley) are first rotated to match the user's desired viewing position, and then projected onto the screen using a geometric perspective that gives a sense of depth perception. The orbital plane of the Earth, known as the ecliptic plane, is also drawn on the screen in perspective to facilitate viewing. The viewing angles that the user inputs are ecliptic longitude and latitude. A latitude of 0 degrees means that the user will be viewing the ecliptic plane edge on. A latitude of 90 degrees means that the user will be looking at the ecliptic plane along a perpendicular axis. The longitude input simply rotates the ecliptic plane about this axis. The DFROMSUN input tells the program a hypothetical distance from the Sun to view from. The default value of 3 is adequate for inner planet viewing, but you will need a value of 25 or so to get Pluto into the picture. Some suggested viewing inputs to get you started are: (press RETURN to all questions except)

(R or S)	LONGITUDE	LATITUDE
S	Ø	9Ø
S	12Ø	6Ø
S	22Ø	4Ø

For a discussion of the DT input, see the FINAL COMMENTS section. The remaining inputs for the SPACE viewing option are self explanatory. When a planet is south of the ecliptic plane (below the plane when viewed from a positive latitude), it will alternate color. Also, the planets may appear to be different sizes on the screen. This is an indication of how close they are to the user's viewing position, and is not an indication of the actual relative sizes of the planets. A word of caution here. A planet's motion is not simulated unless it is plotted, thus for example the program will take longer to run when 8 planets are plotted than when 2 are plotted. It takes about 10 minutes to run a 2 planet plot for 1 year under the ZBASIC interpreter. If you try to plot all 9 planets and Comet Halley for the next 50 years . . . well . . . maybe you should start it on a Friday and go camping over the weekend. In any event, the display seems to become too cluttered if more than 4 objects are plotted at once.

An unfortunate situation should become apparent when you use the SPACE viewing option. When Comet Halley is at its perihelion (its closest approach to the Sun on February 9, 1986), it is very nearly on the other side of the Sun from the Earth. This will make it very difficult, if not impossible, for most of us to view the comet when it is perhaps the most interesting. However, we may be consoled by the fact that this will result in two close approaches of the Comet to the Earth on November 27, 1985 and April 11, 1986.

The RA–DEC (Right Ascension – DEClination) graphing method, on the other hand, displays the positions of the planets against background constellations as viewed from the earth. This is the default viewing method. Because we are viewing from the Earth, the Sun will appear to move with respect to the background stars, resulting from the Earth's orbit (i.e. revolution) around the Sun. This movement is different from the apparent movement caused by the rotation (i.e. spinning) of the Earth, which makes the Sun appear to move across the sky once per day. The RA–DEC graph contains a horizontal line in the middle of the screen at 0 degrees Declination. This is a projection of the Earth's equator extended into space. As you run the program, the Sun will trace a "sine-wave" type of curve on the screen. This is the ecliptic plane extended into space. You should notice that all of the planets stay

```
1000 '*************************************************************
1010 '*
1020 '*  PROGRAM   HALLEY              version: 1.Ø
1030 '*
1040 '*  PLANET MOTION WITH HALLEY'S COMET
1050 '*
1060 '*  PROGRAMMER:  JIM TURSA
1070 '*               3489-C LAKE AUSTIN BLVD
1080 '*               AUSTIN, TX  787Ø3
1090 '*  DATE: 4/85
1100 '*
1110 '*************************************************************
1120 '
1130 OPTION BASE 1 : CLEAR
1140 NP= 1Ø   'NUMBER OF PLANETS (INCLUDING HALLEY)
1150 DIM X(1Ø),Y(1Ø),Z(1Ø),VX(1Ø),VY(1Ø),VZ(1Ø),PCLR(1Ø),LN(1Ø)
1160 DIM LASTR(1Ø),LASTC(1Ø),CLR(7),A(1Ø),B(1Ø),PLOT(1Ø),DINM(12),MONTH$(12)
1170 '
1180 CLS : LOCATE 12,28 : PRINT "Halley's Comet Simulation"
1190       LOCATE 14,28 : PRINT "     By James Tursa"
1200 DELAY = 5 : GOSUB 499Ø
1210 DATA 31,JAN,28,FEB,31,MAR,3Ø,APR,31,MAY,3Ø,JUN,31,JUL,31,AUG,3Ø,SEP
1220 DATA 31,OCT,3Ø,NOV,31,DEC
1230 FOR I=1 TO 12 : READ DINM(I),MONTH$(I) : NEXT I
1240 '
1250 ' DISTANCE UNIT IS AU (ASTRONOMICAL UNIT), TIME UNIT IS DAY
1260 ' POS & VEL OF EARTH,HALLEY,MERCURY,VENUS
1270 DATA Ø.6632717,-.7679Ø9,Ø.Ø,.01273999,.Ø11182Ø9,Ø.Ø
1280 DATA .75616Ø1,2.969393,-.2962655,.ØØ27Ø64Ø4,-.Ø12655Ø7,.ØØ28757Ø1
1290 DATA .1693419,-.4Ø92495,-.Ø4889529,.Ø2Ø36314,.Ø1217821,-.ØØØ8774651
1300 DATA  6457331,.3269825,-.Ø3288496,-.ØØ92Ø284,.Ø1795634,.ØØØ7726429
1310 ' POS & VEL OF MARS,JUPITER,SATURN,URANUS,NEPTUNE,PLUTO
1320 DATA -.8888462,1.375545,.Ø5Ø68ØØ7,-.Ø1122166,-.ØØ64Ø6691,.ØØØ1419813
1330 DATA 3.4Ø6998,-3.769919,-.Ø6Ø7Ø352,.ØØ55Ø4353,.ØØ5415Ø48,-.ØØØ1456762
1340 DATA -5.344798,-8.382771,.3586171,.ØØ44Ø98Ø1,-.ØØ3Ø12811,-.ØØØ1226156
1350 DATA -4.33594,-18.592Ø1,-.Ø1268ØØ5,.ØØ38Ø7513,-.ØØ1Ø73142,-.ØØØØ5348167
1360 DATA 1.41455,-3Ø.21365,.5894718,.ØØ31224Ø4,.ØØØ1682596,-.ØØØØ7533318
1370 DATA -23.55598,-16.Ø5764,8.531517,.ØØ2Ø257Ø8,-.ØØ2869793,-.ØØØ2765691
1380  FOR I=1 TO NP
1390  READ X(I),Y(I),Z(I),VX(I),VY(I),VZ(I)
1400  NEXT I
1410 '
1420 ' DETERMINE COLOR STATUS OF COMPUTER
1430  FOR I=1 TO 7 : CLR(I)=I : NEXT I
1440 PSET(Ø,Ø),7 : IF POINT(Ø,Ø)=7 THEN GOTO 148Ø
1450  FOR I=1 TO 7 : CLR(I)=7 : NEXT I
1460 '
1470 ' GENERAL INSTRUCTIONS
1480 CLS : COLOR 7,Ø : PRINT "** PLEASE USE UPPERCASE LETTERS FOR RESPONSES"
1490 PRINT "** BE SURE YOUR RESPONSE IS ACCEPTABLE (BAD RESPONSES NOT CHECKED)"
1500 PRINT "** UNLESS OTHERWISE STATED, THE FIRST ACCEPTABLE RESPONSE LISTED"
1510 PRINT "   WILL BE THE DEFAULT RESPONSE IF A RETURN IS PUSHED"
1520 PRINT " "
1530 '
1540 PI= 4*ATN(1) : TWOPI=PI+PI : RTOD=18Ø/PI : DTOR=1/RTOD
1550 COS23= COS(23.439291#*DTOR) : SIN23= SIN(23.439291#*DTOR)
1560 DFROMSUN=3 : DFROMSCREEN=2
1570 START= -1 : T= Ø : DAY= 3 : MON= 8 : YR= 1985
1580 L1Ø= 1/LOG(1Ø)
1590 MLINES= Ø : PRADEC= Ø
1600 CSCALE= 2ØØ : RSCALE= 96
```

fairly close to this curve. This is because all of the planets orbit in planes that are nearly parallel to the ecliptic plane. Also, note that the Sun passes from the South (negative Declination) to the North (positive Declination) at 0h Right Ascension, 0 degrees Declination. This is how the (0,0) point on the graph, known as the Vernal Equinox, is defined. The Summer Solstice is at 6h RA, when the Sun is highest in the sky, while the Winter Solstice is at 18h RA. These seasons are reversed for viewers in the Southern Hemisphere. A minor point needs to be mentioned here. Right Ascension is measured in hours, where 24h = 360 degrees. Note that this is an angular measure, not a time measure. In fact, the Earth will turn through 24h of Right Ascension in about 23 hours and 56 minutes of clock time. This turns out to be a convenient measuring system. For example, a planet located at 7h RA will rise above the horizon about 1 hour of clock time earlier than a planet located at 8h RA.

Try plotting Mercury and Venus along with the Sun. You will notice that they stay fairly close to the Sun. When they are on the right side of the Sun, they will appear as "Morning Stars" on the horizon just before sunrise. When they are on the left side of the Sun they will appear as "Evening Stars".

Want to know what constellations will be up in the sky at a certain date? Just look at the RA of the Sun on that date. Constellations that are 12h of Right Ascension away from the Sun are on the opposite side of the Earth from the Sun and will be in the sky that night.

Final Comments

When Comet Halley is plotted, the information "MAG = 13.2" (or some other number) will appear on line 2 of the screen. This is the magnitude of the comet as viewed from the Earth. The lower this value is, the brighter the comet will appear from Earth. Each unit of magnitude decrease represents an increase in brightness of 2.5 times. For example, an object of magnitude 4 is about 2.5 times brighter than an object of magnitude 5. Under the best seeing conditions (clear skies, no city lights, etc), the naked eye can see objects as dim as magnitude 6. The brightest Comet Halley will get is about magnitude 3.9. For comparison, the magnitudes of some objects are listed below:

```
                          Sun   Venus   Mars   Uranus   Pluto
                          ---   -----   ----   ------   -----
These vary somewhat -> -27    -4.5     1.4     5.5     13.7
```

One input that needs a short discussion is DT, the integration (simulation) stepsize. Generally, the default value of DT = 1 should be used when inner planets are being plotted, although values up to about 4 may be adequate for your purposes if you will accept a slightly degraded graph in favor of faster execution time. On the other hand, if it is more accuracy you desire, then you should use a smaller stepsize, such as .5 or .25 (this will increase execution time of course). If only outer planets are being plotted, then DT's in the range of 10 to 20 days will be sufficient. The farther from the Sun a planet is, the lower its speed is, which allows for larger DT's without much accuracy loss. Ideally, a different DT should be used for each planet (and perhaps should vary at different points of the orbit), but this is not necessary for our purposes. It should also be pointed out that the RA–DEC graph is essentially a view from the Earth. Thus, regardless of which planets you wish to plot, the motion of the Earth must be simulated to obtain proper plotting coordinates. So DT must remain fairly small for the RA–DEC graph, even if you only plot Pluto.

For those of you who do not relish the thought of typing in all of the constellation data, you may eliminate lines 2470–3120 and the

```
1610 MU= 2.958994E-04      '  IN AU^3/DAY^2
1620 '
1630 ' DETERMINE TYPE OF GRAPH TO USE
1640 INPUT "RA-DEC OR SPACE VIEWING (R OR S)";D$
1650 IF D$<>"S" THEN SRA=-1 ELSE SRA=Ø
1660 PRINT "DATE TO SIMULATE TO (DEFAULT IS 1986/8/3)"
1670 INPUT "YEAR (EXAMPLE, 86 OR 2004)";YEND
1680 YEND=FIX(YEND) : IF YEND<=Ø THEN YEND=1986 : PRINT "(DEFAULT=1986 USED)"
1690 IF YEND<1ØØ THEN YEND=YEND+19ØØ
1700 INPUT "MONTH (NUMBER FROM 1 TO 12)";M
1710 MEND=FIX(M) : IF MEND<1 OR MEND>12 THEN MEND=8 : PRINT "(DEFAULT=8 USED)"
1720 D=DINM(MEND) : IF MEND=2 AND (YEND MOD 4)=Ø AND (YEND MOD 1ØØ)<>Ø THEN D=29
1730 PRINT "DAY (NUMBER FROM 1 TO";D;")", : INPUT DEND
1740 IF DEND<1 OR DEND>D THEN DEND=3 : PRINT "(DEFAULT=3 USED)"
1750 INPUT "STEPSIZE DT (A SMALL POSITIVE NUMBER, DEFAULT=1)";DT
1760 IF DT=Ø THEN DT=1
1770 IF YEND<1985 THEN DT= -DT
1780 IF YEND=1985 AND MEND<8 THEN DT= -DT
1790 IF YEND=1985 AND MEND=8 AND DEND<3 THEN DT= -DT
1800 MUDT= MU*DT
1810 '
1820 'COLORS OF THE PLANETS
1830 DATA 3,EARTH,2,HALLEY,4,MERCURY,5,VENUS,4,MARS,2,JUPITER,5,SATURN
1840 DATA 3,URANUS,2,NEPTUNE,5,PLUTO
1850 ' OBTAIN DESIRED PLANETS TO PLOT
1860 TOP=Ø
1870  FOR I=1 TO NP
1880  READ J,P$ : L=LEN(P$) : PCLR(I)=CLR(J) : PRINT "PLOT "; : COLOR CLR(J)
1890  IF I=1 AND SRA THEN P$="SUN" : PCLR(1)= CLR(6) : L=3 : COLOR CLR(6)
1900  IF I<3 THEN YN$="(Y/N)" ELSE YN$="(N/Y)"
1910  PRINT P$;SPACE$(8-L); : COLOR CLR(7) : PRINT YN$; : INPUT D$
1920  A(I)= PCLR(I) : B(I)= CLR(1)
1930  IF I<3 THEN IF D$="N" THEN PLOT(I)=Ø ELSE PLOT(I)=-1 : TOP=I
1940  IF I>2 THEN IF D$="Y" THEN PLOT(I)=-1 : TOP=I ELSE PLOT(I)=Ø
1950  NEXT I
1960 '
1970 ' OBTAIN DESIRED VIEWING OPTIONS
1980 INPUT "PLOT RANDOM STAR BACKGROUND (Y/N)";D$
1990 IF D$="N" THEN STARS=Ø ELSE STARS=-1
2000 IF NOT PLOT(2) THEN MLINES=Ø : GOTO 2Ø3Ø
2010 INPUT "MARK CLOSE APPROACH POSITIONS (Y/N)";D$
2020 IF D$="N" THEN MLINES=Ø ELSE MLINES=-1
2030 IF SRA THEN GOTO 235Ø
2040 INPUT "DISTANCE FROM SUN TO VIEW FROM (POSITIVE#>3, DEFAULT=3)";D
2050 IF D<3 THEN DFROMSUN=3 ELSE DFROMSUN=D
2060 '
2070 ' CALCULATE VIEWING ANGLE ROTATIONS
2080 INPUT "ECLIPTIC LONGITUDE TO VIEW FROM (Ø TO 36Ø, DEFAULT=Ø)";LONG
2090 LONG= LONG*DTOR : CN= COS(LONG) : SN= SIN(LONG)
2100 INPUT "ECLIPTIC LATITUDE TO VIEW FROM (-9Ø TO 9Ø, DEFAULT=Ø)";LAT
2110 LAT= LAT*DTOR : CT= COS(LAT) : ST= SIN(LAT)
2120 IX=  CT*CN : IY=  CT*SN : IZ=  ST
2130 JX=     -SN : JY=     CN : JZ=   Ø
2140 KX= -ST*CN : KY= -ST*SN : KZ=  CT
2150 '
2160 ' PUT RANDOM STARS AND THE SUN ON THE SCREEN
2170 CLS
2180 IF NOT STARS THEN GOTO 222Ø
2190  FOR I=1 TO 5ØØ
2200  C= CINT(RND*639) : R= 9 + CINT(RND*215) : PSET(C,R),CLR(7)
2210  NEXT I
```

program will still run properly. Of course, you can always type in these lines at a later date. The program will compile without modification.

As an educational exercise, you may wish to see what happens to the accuracy when large DT's are used for inner planet motion. Try the following: (press return to all questions except)

(R or S)	STEPSIZE	YEAR	PLOT HALLEY	LATITUDE
S	2Ø	9Ø	N	9Ø

Where Do I Look?

So you like the program, but now you want to go outside and actually look at the real thing. The RA-DEC graph will tell you the general area of the sky to search by giving you the position of the comet relative to some of the constellations. However, not all of the constellations appear on the graph, and the stars shown are randomly placed for visual effect and are not accurate (I thought typing in the constellation data was enough!). The actual position of the comet in the sky will depend on your position on the Earth, your local time, and the day of the year. If you can locate constellations in the sky at night, then you will be able to locate the general position of the comet easily. If not, then you might want to use a program such as the HUG product SKYVIEWS, which will help locate constellations from your particular place on the Earth at any given time and date. Another option would be to look in periodicals such as ASTRONOMY or SKY AND TELESCOPE magazines, and to watch the local newspapers and TV news shows for information. Finally, you may want to contact a local astronomy club; they will usually be more than willing to help you. But above all, try to get away from the city lights at least once . . . and HAVE FUN!

```
2220 CIRCLE (319,112),42/DFROMSUN,CLR(6) : PAINT (319,112),CLR(6)
2230 '
2240 ' DRAW ECLIPTIC PLANE(S) ON THE SCREEN
2250 SIZE= 1.1 : FLAG= -1
2260  FOR I=-1 TO 1 STEP 2
2270  XX=I*SIZE : YY=I*SIZE : ZZ=Ø : GOSUB 457Ø : C2=C1 : R2=R1
2280  XX=-XX :             GOSUB 457Ø : LINE (C2,R2)-(C1,R1),PCLR(1)
2290  XX=-XX : YY=-YY : GOSUB 457Ø : LINE (C2,R2)-(C1,R1),PCLR(1)
2300  NEXT I
2310 IF FLAG AND DFROMSUN>17 THEN FLAG=Ø : SIZE=7 : GOTO 226Ø
2320 '
2330 GOSUB 4Ø5Ø : GOTO 319Ø    'INITIAL PLANET PROJECTIONS
2340 '
2350 INPUT "PRINT RA AND DEC OF WHAT OBJECT (DEFAULT=HALLEY)";D$
2360 IF D$="SUN" THEN PRADEC=1 : PLANET$=D$ : GOTO 246Ø
2370 RESTORE 183Ø
2380  FOR I=1 TO 1Ø
2390  READ C,A$ : IF A$=D$ THEN PRADEC=I : PLANET$=A$
2400  NEXT I
2410 IF PRADEC>1 THEN GOTO 246Ø
2420 IF PRADEC=Ø THEN PRADEC=2 : PLANET$="HALLEY" : GOTO 246Ø
2430 PRINT "EARTH IS NOT IN THE RA-DEC PLOT . .   SUN WILL BE USED"
2440 PLANET$="SUN"
2450 '
2460 GOSUB 465Ø    'DRAW RA-DEC BOX
2470 LOCATE 1,1 : INPUT "DRAW CONSTELLATIONS ON THE SCREEN (Y/N)";D$
2480 LOCATE 1,1 : PRINT SPACE$(8Ø)
2490 IF D$="N" THEN GOTO 313Ø
2500 '
2510 ' PUT CONSTELLATIONS ON THE SCREEN
2520 DATA  1.6,49 ,  Ø.9,38 ,  Ø.2,29 ,  Ø.3,15 , 23.1,15 , 23.1,28 ,  Ø.2,29
2530 DATA  Ø.6,31 ,  1.2,36 ,  2.1,42 ,   -1,1  , 23.1,28 , 22.7,3Ø , 22.2,33
2540 DATA   -1,1  , 23.1,28 , 22.8,24 ,   -1,1  , 23.1,15 , 22.6,11 ,   -1,1
2550 DATA  1.2,-1Ø,  Ø.7,-18,  Ø.3,-9 ,  1.2,-1Ø,  1.4,-8 ,  1.8,-11,  1.7,-16
2560 DATA  1.2,-1Ø,   -1,1  ,  1.8,-11,  2.3,-3 ,  2.6,Ø  ,  2 7,3  ,  2.4,8
2570 DATA  2.7,1Ø ,  2.9,9  ,  3.1,3  ,  2.7,3  ,   -1,1  ,  1.9,21 ,  2.1,23
2580 DATA  2.7,28 ,  2.8,27 ,   -1,1  ,  5.8,7  ,  5.4,6  ,  5.6,1Ø ,  5.8,7
2590 DATA  5.7,-2 ,  5.8,-1Ø,  5.2,-8 ,  5.5,Ø  ,  5.4,6  ,   -1,1  ,  5.7,-2
2600 DATA  5.6,-1 ,  5.5,Ø  ,   -1,1  ,  4.5,16 ,  4.4,15 ,  4.3,15 ,  4.4,19
2610 DATA   -1,1  ,  4.3,15 ,  4.Ø,12 ,  3.4,9  ,   -1,1  ,  4.5,16 ,  5.6,21
2620 DATA   -1,1  ,  5.3,46 ,  5.9,45 ,  5.9,38 ,  5.4,28 ,  4.9,33 ,  5.3,46
2630 DATA  5.5,5Ø ,   -1,1  ,  6.3,-18,  6.8,-17,  7.2,-27,  7.4,-29,  6.9,-29
2640 DATA  7.2,-27,   -1,1  ,  7 7,5  ,  7.4,9  ,   -1,1  ,  6.7,12 ,  6.6,16
2650 DATA  7.Ø,21 ,  7 7,24 ,  7.7,28 ,  7.5,32 ,  6.7,25 ,  6.4,22 ,   -1,1
2660 DATA  9.7,24 ,  9.8,27 , 1Ø.3,24 , 1Ø.3,2Ø , 1Ø.1,17 , 1Ø.2,12 , 11.2,15
2670 DATA 11.8,14 , 11.2,21 , 11.2,15 ,   -1,1  , 13.Ø,11 , 12.8,4  , 14.Ø,2
2680 DATA 14.7,2  , 14.6,-5 , 14.2,-5 , 13.4,-11, 12.6,-1 , 11.8,2  , 11.7,6
2690 DATA 12.1,9  , 12.8,4  ,   -1,1  , 15.2,33 , 14.2,19 , 14.5,38 , 15.2,33
2700 DATA 15.4,38 , 15.Ø,41 , 14.5,38 ,   -1,1  , 15.6,-28, 15.Ø,-25, 14.7,-16
2710 DATA 15.3,-9 , 15.5,-16, 15.Ø,-25,   -1,1  , 15.7,15 , 15.9,16 , 15.6,2Ø
2720 DATA 15.7,15 , 15.5,11 , 15.8,5  , 15.8,-4 , 16.2,-4 , 16.6,-1Ø, 17.2,-16
2730 DATA 17.3,-13, 17.6,-16, 17 7,-14, 17.9,-1Ø, 18.3,-3 , 18.9,4  ,   -1,1
2740 DATA 17.3,-13, 17.6,-8 , 17.7,2  , 17.6,5  , 17.5,13 , 16.9,9  , 16.5,2
2750 DATA 16.2,-4 ,   -1,1  , 18.2,-37, 18.3,-35, 19.Ø,-3Ø, 18.7,-27, 18.3,-3Ø
2760 DATA 18.3,-35, 18.1,-31, 18.3,-3Ø, 18.4,-25, 18.7,-27, 18.8,-26, 19.1,-28
2770 DATA 19.Ø,-3Ø,   -1,1  , 18.4,-25, 18.9,-21, 19.1,-22, 19.2,-21,   -1,1
2780 DATA 17.5,-37, 17.8,-4Ø, 17.6,-42, 17.2,-43, 16.8,-42, 16.8,-34, 16.5,-26
2790 DATA 16.Ø,-22, 15.8,-29,   -1,1  , 16.Ø,-22, 17.1,-2Ø, 17.2,-19,   -1,1
2800 DATA 2Ø.8,-1 , 2Ø.2,-1 , 19.8,7  , 19.7,11 , 19.4,3  , 19.8,7  ,   -1,1
2810 DATA 19.7,11 , 19.1,14 ,   -1,1  , 19.4,3  , 19.1,-5 ,   -1,1  , 18.6,39
2820 DATA 18.7,38 , 18.9,37 , 19.Ø,32 , 18.8,33 , 18.7,38 ,   -1,1  , 19.5,28
```

```
3420  R2= X(I)*X(I) + Y(I)*Y(I) + Z(I)*Z(I) : R= SQR(R2) : R3= R*R2
3430  ACC= -MUDT/R3
3440  DX= VX(I)*DT  : DY= VY(I)*DT  : DZ= VZ(I)*DT
3450  DVX= ACC*X(I) : DVY= ACC*Y(I) : DVZ= ACC*Z(I)
3460  XX= X(I) + DX   : YY= Y(I) + DY   : ZZ= Z(I) + DZ
3470  VXX=VX(I) + DVX : VYY=VY(I) + DVY : VZZ=VZ(I) + DVZ
3480 ' MODIFIED EULER ESTIMATE
3490  R2= XX*XX + YY*YY + ZZ*ZZ : R= SQR(R2) : R3= R*R2
3500  ACC= -MUDT/R3
3510  X(I)= X(I) + (.5)*(DX + VXX*DT)
3520  Y(I)= Y(I) + (.5)*(DY + VYY*DT)
3530  Z(I)= Z(I) + (.5)*(DZ + VZZ*DT)
3540  VX(I)= VX(I) + (.5)*(DVX + ACC*XX)
3550  VY(I)= VY(I) + (.5)*(DVY + ACC*YY)
3560  VZ(I)= VZ(I) + (.5)*(DVZ + ACC*ZZ)
3570  NEXT I
3580 ' END OF PLANET PROPOGATION *****
3590 '
3600 ' OBTAIN PLANET PROJECTIONS AND PUT THEM ON THE SCREEN
3610 IF SRA THEN GOSUB 424Ø ELSE GOSUB 4Ø5Ø
3620 '
3630 ' DATE COUNTER
3640 DAY= DAY + DT
3650 IF DAY<DINM(MON)+1 THEN GOTO 371Ø
3660 DAY=DAY-DINM(MON) : MON= MON + 1
3670 IF MON<13 THEN GOTO 376Ø
3680 MON=1 : YR= YR + 1
3690 IF (YR MOD 4)=Ø AND (YR MOD 1ØØ)<>Ø THEN DINM(2)=29 ELSE DINM(2)=28
3700 GOTO 376Ø
3710 IF DAY>Ø THEN GOTO 376Ø
```

```
2830 DATA 20.3,40 , 20.6,45 ,   -1,1  , 20.7,34 , 20.3,40 , 19.7,45 ,   -1,1
2840 DATA 20.3,-12, 21.1,-18, 21.8,-16, 21.7,-19, 20.8,-28, 20.3,-12,   -1,1
2850 DATA  5.2,-9 ,  5.2,-5 ,  4.8,-6 ,  4.7,-3 ,  4.5,-3 ,  4.2,-7 ,  3.9,-14
2860 DATA  3.8,-13,  3.7,-10,  2.9,-9 ,  2.7,-14,  2.7,-19,  3.0,-24,  3.3,-22
2870 DATA  3.5,-22,  3.8,-25,  4.0,-24,  4.5,-30,  4.3,-34,  3.9,-35,  3.8,-38
2880 DATA  3.6,-37,  3.5,-40,  3.3,-44,  2.9,-40,  2.6,-40,  2.6,-43,  2.3,-50
2890 DATA   -1,1  , 14.8,-28, 14.3,-28, 13.3,-23, 11.8,-34, 11.1,-28, 10.8,-16
2900 DATA 10.4,-17, 10.2,-12,  9.8,-15,  9.4,-9 ,  9.6,-1 ,  9.1,5  ,   -1,1
2910 DATA  8.9,6  ,  8.7,3  ,  8.6,3  ,  8.6,6  ,  8.7,7  ,  8.9,6  ,   -1,0
2920 '
2930 READ RA,DEC : RA=RA*PI/12 : DEC=DEC*DTOR
2940 LR=112-DEC*103.1324 : LC=259-RA*89.12676 : IF LC<49 THEN LC=LC+560
2950 READ RA,DEC
2960 IF RA=-1 AND DEC=0 THEN GOTO 3020
2970 IF RA=-1 AND DEC=1 THEN GOTO 2930
2980 RA= RA*PI/12 : DEC= DEC*DTOR
2990 R= 112 - DEC*103.1324 : C= 259 - RA*89.12676 : IF C<49 THEN C=C+560
3000 LINE (LC,LR)-(C,R),CLR(3) : LC=C : LR=R : GOTO 2950
3010 '
3020 LOCATE 1,1 : INPUT "PRINT CONSTELLATION NAMES (Y/N)";D$
3030 LOCATE 1,1 : PRINT SPACE$(80)
3040 IF D$="N" THEN GOTO 3130
3050 COLOR CLR(3),0
3060 DATA 14,10,ORION,9,37,PEGASUS,20,37,SAGITTARIUS,21,55,SCORPIUS
3070 DATA 8,70,LEO,6,50,LYRA,6,62,BOOTES,14,67,VIRGO,14,28,CETUS
3080 DATA 21,13,ERIDANUS,6,8,GEMINI,20,70,HYDRA,5,37,CYGNUS,6,20,AURIGA
3090 DATA 9,50,SERPENS,12,38,AQUILA,18,29,CAPRICORNUS,17,61,LIBRA,0
3100 READ R : IF R=0 THEN COLOR CLR(7),0 : GOTO 3130
3110 READ C,LABEL$ : LOCATE R,C : PRINT LABEL$ : GOTO 3100
3120 '
3130 IF NOT STARS THEN GOTO 3190
3140 ' PUT RANDOM STARS ON THE SCREEN
3150  FOR I=1 TO 350
3160  R= 22+RND*180 : C= 49+RND*560 : PSET(C,R),CLR(7)
3170  NEXT I
3180 '
3190 ' PUT LEGEND ON THE SCREEN
3200 RESTORE 1830 : LOCATE 1,1
3210 IF SRA AND NOT PLOT(1) THEN P$="     " ELSE P$="SUN "
3220 COLOR CLR(6) : PRINT P$;
3230  FOR I=1 TO TOP
3240  READ J,P$ : L=LEN(P$) : IF NOT PLOT(I) THEN P$=SPACE$(L)
3250  IF I=1 AND SRA THEN P$=SPACE$(L)
3260  COLOR PCLR(I) : PRINT P$;" ";
3270  NEXT I
3280 IF SRA THEN GOTO 3330 ELSE COLOR CLR(7)
3290 LOCATE 2,67 : PRINT "ECLIPTIC    " : LINE(604,9)-(618,17),PCLR(1),B
3300 '
3310 ' ------------- MAIN SIMULATION LOOP --------------
3320 '
3330 IF PLOT(PRADEC) AND SRA
       THEN LOCATE 2,68 : COLOR PCLR(PRADEC) : PRINT "<--";PLANET$
3340 T= T + DT : SIDE= -1
3350 IF (YR=YEND AND MON=MEND AND DAY=DEND) THEN COLOR CLR(7) : END
3360 IF (YR=YEND AND MON=MEND AND DAY>DEND) OR (YR=YEND AND MON>MEND) OR
       (YR>YEND) THEN SIDE= +1
3370 IF SGN(DT)=SGN(SIDE) THEN COLOR CLR(7) : END
3380 ' PROPOGATE PLANET MOTION *****
3390  FOR I=1 TO TOP
3400  IF (NOT PLOT(I)) AND ((NOT SRA) OR I<>1) THEN GOTO 3570
3410 ' EULER ESTIMATE
```

```
4320  ZEQ=  YDIF*SIN23 + ZDIF*COS23
4330  X2Y2= XEQ*XEQ + YEQ*YEQ : RXY= SQR(X2Y2) : RXYZ= SQR(X2Y2 + ZEQ*ZEQ)
4340  IF ABS(RXY)<.001 THEN DEC=0 ELSE DEC=ATN(ZEQ/RXY)
4350  IF ABS(XEQ)<.001 THEN RA=SGN(YEQ)*PI/2 : GOTO 4380
4360  RA= ATN(YEQ/XEQ)
4370  IF XEQ<0 THEN RA=PI+RA
4380  IF RA<0 THEN RA=TWOPI+RA
4390  COLOR PCLR(I)
4400  IF I<>PRADEC THEN GOTO 4440
4410  RAH=RA*12/PI : HR=FIX(RAH) : RMIN=60*(RAH-HR)
4420  DECD=DEC*RTOD : DEG=FIX(DECD) : DMIN=60*ABS(DECD-DEG)
4430  LOCATE 2,37 : PRINT USING"RA = ##h ##m , DEC = ###  ##'";
                  HR;RMIN;DEG;DMIN : CIRCLE(483,10),2,PCLR(I)
4440  R= 112 - DEC*103.1324 : C= 259 - RA*89.12676 : IF C<49 THEN C=C+560
4450  IF ABS(C-LASTC(I))>100 THEN GOTO 4510
4460  IF START THEN GOTO 4510
4470  LINE (LASTC(I),LASTR(I))-(C,R)
4480  IF T<>116 AND T<>190 AND T<>251 THEN GOTO 4510
4490  IF MLINES AND I=1 AND PLOT(1) AND PLOT(2) THEN CIRCLE(C,R),5
4500  IF MLINES AND I=2 AND              PLOT(2) THEN CIRCLE(C,R),5
4510  LASTC(I)= C : LASTR(I)= R
4520  NEXT I
4530 START=0 : RETURN
4540 '
4550 ' ROW & COLUMN CALCULATIONS    ***********************************
4560 '
4570  RX= XX*IX + YY*IY + ZZ*IZ
4580  RY= XX*JX + YY*JY + ZZ*JZ
4590  RZ= XX*KX + YY*KY + ZZ*KZ
4600  SR= RSCALE*DFROMSCREEN*RZ/(DFROMSUN - RX)
4610  SC= CSCALE*DFROMSCREEN*RY/(DFROMSUN - RX)
4620  C1= CINT(319.5 + SC) : R1= CINT(112 - SR)
4630 RETURN
4640 '
4650 ' DRAW RA-DEC BOX    ****************************************
4660 '
4670 DIM LETT%(29)
4680 CLS : LINE (49,112)-(609,112),7
4690 LINE (49,22)-(609,202),7,B : PAINT(0,0),1,7 : COLOR 7,1
4700 R= 202 : LABEL$= "DECLINATION"
4710  FOR I= -5 TO 5
4720  LOCATE 13-2*I,3 : PRINT USING "###";10*I
4730  LOCATE 13+I,1 : K= 6+I : PRINT MID$(LABEL$,K,1)
4740  LINE (49,R)-(56,R),7 : LINE (602,R)-(609,R),7
4750  CIRCLE (44,R-3),2,7
4760  R= R - 18
4770  NEXT I
4780 '
4790  FOR I=0 TO 8
4800  LINE(49+I*70, 23)-(49+I*70, 27),7
4810  LINE(49+I*70,110)-(49+I*70,114),7
4820  LINE(49+I*70,198)-(49+I*70,202),7
4830  HOUR= I*3 : IF I>3 THEN HOUR= HOUR - 3
4840  LOCATE 1,1 : PRINT USING "##";HOUR
4850  GET (0,0)-(15,8),LETT%
4860  C= 247 - I*70
4870  IF C<0 THEN C= C + 629
4880  PUT (C,206),LETT%,OR
4890  LOCATE 1,1 : PRINT "h"
4900  GET (0,0)-(7,8),LETT%
4910  PUT (C+16,203),LETT%,OR
```

```
3720 MON= MON-1 : IF MON>0 THEN DAY=DINM(MON)+DAY : GOTO 3760
3730 MON=12 : YR= YR-1
3740 IF (YR MOD 4)=0 AND (YR MOD 100)<>0 THEN DINM(2)=29 ELSE DINM(2)=28
3750 DAY=31+DAY
3760 LOCATE 2,1 : COLOR CLR(2) : PRINT YR;MONTH$(MON);DAY;"    ",
3770 '
3780 IF NOT PLOT(2) THEN GOTO 3340
3790 ' MAGNITUDE
3800 RSH2= X(2)*X(2) + Y(2)*Y(2) + Z(2)*Z(2)
3810 XX= X(2)-X(1) : YY= Y(2)-Y(1) : ZZ= Z(2)-Z(1)
3820 REH2= XX*XX + YY*YY + ZZ*ZZ
3830 IF T<190 THEN M0=5.47 : N25=5.55 ELSE M0=4.94 : N25=3.84
3840 MAG= M0 + 2.5*L10*LOG(REH2) + N25*L10*LOG(RSH2)
3850 LOCATE 2,17 : PRINT USING "MAG = ##.#";MAG;
3860 ' SPECIAL DATE SECTION
3870 IF T<>116 OR NOT(SRA OR PLOT(1)) THEN GOTO 3920
3880 LOCATE 2,29 : PRINT "PRE-PERIHELION CLOSE APPROACH TO EARTH";SPACE$(12)
3890 PRINT CHR$(7);:DELAY = 5 : GOSUB 4990
3900 LOCATE 2,29 : PRINT SPACE$(51)
3910 IF SRA THEN GOTO 3330 ELSE COLOR CLR(7) : GOTO 3290
3920 IF T<>190 THEN GOTO 3970
3930 LOCATE 2,29 : PRINT "PERIHELION OF COMET HALLEY";SPACE$(25)
3940 PRINT CHR$(7);:DELAY = 5 : GOSUB 4990
3950 LOCATE 2,29 : PRINT SPACE$(51)
3960 IF SRA THEN GOTO 3330 ELSE COLOR CLR(7) : GOTO 3290
3970 IF T<>251 OR NOT(SRA OR PLOT(1)) THEN GOTO 3340
3980 LOCATE 2,29 : PRINT "POST-PERIHELION CLOSE APPROACH TO EARTH";SPACE$(11)
3990 PRINT CHR$(7);:DELAY = 5 : GOSUB 4990
4000 LOCATE 2,29 : PRINT SPACE$(51)
4010 IF SRA THEN GOTO 3330 ELSE COLOR CLR(7) : GOTO 3290
4020 '
4030 ' --------------- END OF SIMULATION LOOP ---------------
4040 '
4050 ' PLANET PLOTS    ****************************************************
4060 '
4070  FOR I=1 TO TOP
4080  IF NOT PLOT(I) THEN GOTO 4210
4090  XX=X(I) : YY=Y(I) : ZZ=Z(I)
4100  GOSUB 4570
4110  IF RX>DFROMSUN THEN GOTO 4210
4120  RXX= DFROMSUN-RX
4130  R= 12 - 3*SQR(RZ*RZ + RY*RY + RXX*RXX)
4140  IF R<0 THEN R=0
4150  IF RX>0 THEN GOTO 4170
4160  IF SC*SC +4.6*SR*SR < 855/DFROMSUN THEN GOTO 4210
4170  IF Z(I)>=0 THEN CR=PCLR(I) ELSE CR= A(I) : SWAP A(I),B(I)
4180  CIRCLE (C1,R1),R,CR
4190  IF I=1 THEN CE=C1 : RE=R1
4200  IF MLINES AND I=2 AND PLOT(1) AND (T=116 OR T=190 OR T=251)
        THEN LINE (CE,RE)-(C1,R1),CLR(7)
4210  NEXT I
4220 RETURN
4230 '
4240 ' RA-DEC PLOTS    ************************************************
4250 '
4260  FOR I=1 TO TOP
4270  IF NOT PLOT(I) THEN GOTO 4520
4280  IF I=1 THEN XDIF= -X(1) : YDIF= -Y(1) : ZDIF= -Z(1) : GOTO 4300
4290  XDIF= X(I) - X(1) : YDIF= Y(I) - Y(1)   ZDIF= Z(I) - Z(1)
4300  XEQ=  XDIF
4310  YEQ=  YDIF*COS23 - ZDIF*SIN23
```

```
4920  NEXT I
4930 LOCATE 25,35 : PRINT "RIGHT ASCENTION"
4940 LINE (0,0)-(639,17),0,BF : COLOR ,0
4950 RETURN
4960 '
4970 ' DELAY EXECUTION    ****************************************
4980 '
4990 ATT= VAL(RIGHT$(TIME$,2)):ATT=ATT+DELAY:IF ATT>59 THEN ATT=ATT-60
5000 IF VAL(RIGHT$(TIME$,2))=ATT THEN RETURN ELSE GOTO 5000
```

Acknowledgements

The initial position and velocity components of the planets were obtained from the 1985 ASTRONOMICAL ALMANAC. The orbital elements of Comet Halley were obtained from the JET PROPULSION LABORATORY. In particular, I wish to thank Donald K. Yeomans for sending me THE COMET HALLEY HANDBOOK along with the latest orbital element estimates. A BASIC program, written by the author of this article, was used to convert these into position and velocity components. The constellation coordinates were obtained from the EDMUND MAG 5 STAR ATLAS.

*

EVERYTHING YOU NEED . . . $279.00

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

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

The HERO® Macros come with 30 pages of documentation.

Transfer to HERO® with ROBI . . . an affordable interface for the robotics experimenter . . . is simple.

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

VISA and MasterCard accepted

The Cross Assembler with HERO® Macros sells for $100.00; the ROBI Interface sells for $199.00. Both as a package — $279.00.

To order, or for more information, call (303) 670-6137.

BERSEARCH Information Services
26160 Edelweiss Circle
Evergreen, Colorado 80439

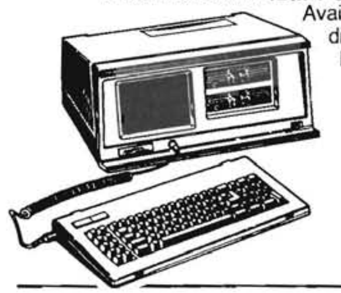
APPLE® is a trademark of Apple Computer. HERO® is a trademark of Heath Electronics.

5 NEW IBM-PC and AT COMPATIBLE COMPUTERS FROM HEATH/ZENITH

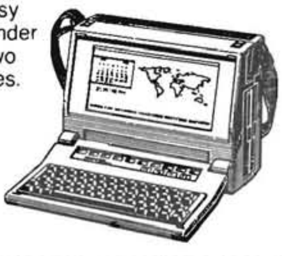
NOW IN STOCK

At Heath/Zenith Computers & Electronics Centers

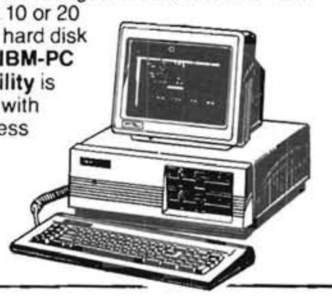
ZA-138 Lightweight, Compact Transportable PC that gives you 60% faster performance in an 8 MHz **IBM-PC compatible**. Available with two disk drives and built-in amber monitor. A lot of business computer for a little money.



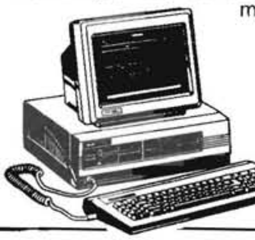
Z-171 Newest extension of the famous Z-150 family, with all-in-one compact design, and full **IBM-PC compatibility.** Electroluminescent backlit LCD display for easy readability. Under 15 lbs. With two 5¼" disk drives.



Z-158 More power comes to you with this new computer in the tradition of the Z-150 series. Choose single or double disk drives, or select a 10 or 20 megabyte hard disk drive. Full **IBM-PC compatibility** is combined with compactness and increased 8 MHz speed.



Z-148 Fast 8 MHz, high-performance IBM-PC compatible offers the features and the memory you need at a price you'll like. 256K RAM standard, also memory expansion up to 640K.

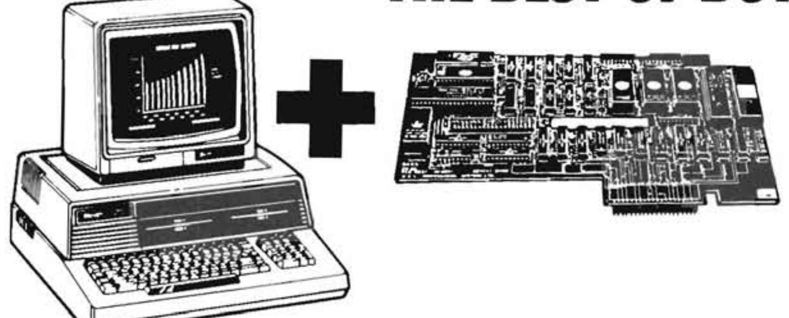


NATIONAL HUG MEMBERS

Get 20% discount on all assembled Zenith products

Z-200 Advanced PC Total performance and advanced technology make this the hottest PC on the market! **IBM-PC and AT compatible.** Can be used as Central Processing Unit for networking/multiuser. Handles both MS-DOS and XENIX operating systems.

Attention H/Z-100 owners: Now you can enjoy THE BEST OF BOTH WORLDS



= IBM-PC Compatibility

You already know what a super computer the H/Z-100 is on its own. Now it can be even more versatile! You can run most IBM-PC compatible software as well as all existing H/Z-100 software just by adding the new Gemini Emulator Board. Easy to install, easy to use, runs most of the 5,000 programs in the IBM-PC library. Supports high resolution and medium resolution graphics. Prices start at $599. HUG 10% discount applies.

Inquire about special corporate and government sales and service programs.

Prices good through September 30, 1985
Phone orders accepted.

VISA MasterCard

ZENITH data systems Authorized Dealer

Your Total Service computer center

Service • Support • Software • Accessories • User Training • Competitive Prices

HZC-146

Heath ZENITH®
Computers & Electronics

Units of Veritechnology Electronics Corporation

FASTIO Patch

Pat Swayne
HUG Software Engineer

The original release of FASTIO program on HUG disk 885-3025 will not work with the Final Word word processing program. It also will not accept input if you run the version of MASM provided with the Programmer's Utility Pack, and try to supply file names at the MASM prompts (instead of on the command line). These problems are caused by the way FASTIO handles DOS functions 6 and 63. To correct the problem with function 6, (for Final Word) load FASTIO.ASM with an editor and make these changes. First, add this macro definition at the beginning of the file, with the other macro definitions.

```
RETFD   MACRO
        DB      ØCAH                ;FAR RETURN WITH DISPLACEMENT
        ENDM
```

Now, locate the label DCINN:, and change the five lines there to this:

```
DCINN.  CALL    BCONST              ;GET STATUS
        JNZ     GOTCHR              ;WE HAVE A CHARACTER
        XOR     AL.AL               ;SET Z FLAG
        JMP     XRET                ;RETURN
GOTCHR: CALL    BCONIN              ;GET CHARACTER
        OR      AL.AL               ;CLEAR Z FLAG
        JMP     XRET                ;RETURN
```

Locate the label XRET:, and change all lines from there to the first blank line to this:

```
        CLC                         ;CLEAR CARRY
XRET:   CLI
        MOV     SS,CS:SYSSTKS       ;RESTORE SYSTEM STACK
        MOV     SP,CS:SYSSTK
        POP     BX
        STI
        RETFD
        DW      2                   ;RETURN, FLAGS SKIPPED
```

Find the label XWRIT2:. A few lines below it, add a CLC instruction before the jump to XRET, as follows:

```
        CLC
        JMP     XRET
```

To correct the problem with function 63, locate the label READCON:, and add this code between the label and the PUSH CX instruction:

```
READCON:CALL    READLN              ;READ CONSOLE
        JMP     MYRET
READLN: PUSH    CX
```

Locate the label READX:, and change the instruction JMP MYRET (two lines down) to RET.

Locate the label XREAD:, and change the lines from there to the comment "ASSUME AUX INPUT" so that they look like this:

```
XREAD:  CMP     BX,4                ;ILLEGAL CALL?
        JNC     JMPSYS              ;IF SO, LET SYSTEM DO IT
        PUSH    DI
        OR      BX,BX
        JNZ     XREADØ              ;NOT STD INPUT
        JMP     XREADSI             ;ELSE, SPECIAL CASE
XREADØ: MOV     DI,OFFSET MYAUXIN   ;ASSUME AUX INPUT
```

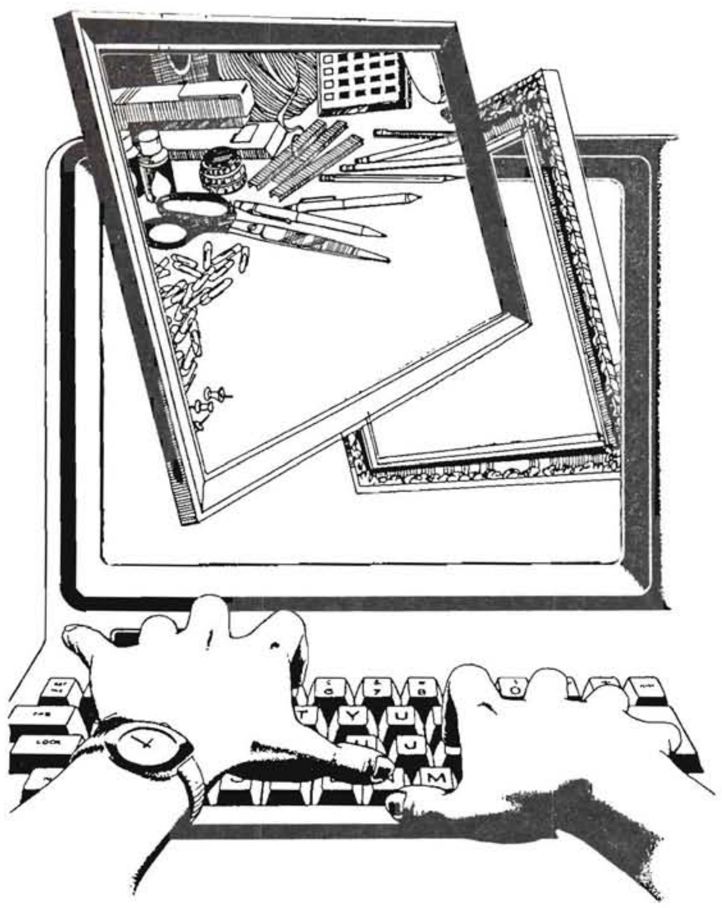
At the end of the XREAD routine, just below the DW 2 statement added earlier, add these lines:

```
XREADSI:PUSH    CX
        PUSH    SI                  ;SAVE REGISTERS
        MOV     SI,DX               ;GET POINTER TO BUFFER
        MOV     BYTE PTR [SI],8Ø    ,INSERT MAX COUNT
        CALL    READLN              ;READ LINE FROM CONSOLE
        MOV     AL,1[SI]            ;GET COUNT OF CHARACTERS
        INC     AL                  ;INCLUDE CR
        XOR     AH,AH               ;AX = COUNT
        PUSH    ES                  ;SAVE ES
        PUSH    DS
        POP     ES                  ;ES = DS
        PUSH    DI
        MOV     DI,SI               ;POINT DI TO BUFFER
        INC     SI
        INC     SI                  ;MOVE TO TEXT
        MOV     CX,AX               ;GET COUNT
        CLD
        REP     MOVSB               ;MOVE TEXT DOWN
        MOV     BYTE PTR [DI],ØAH   ;ADD LF
        INC     AX                  ;COUNT IT
        PUSH    AX
        MOV     CL,ØAH
        CALL    COUT                ;PRINT LF
        POP     AX
        POP     DI                  ;RESTORE REGISTERS
        POP     ES
        POP     SI
        POP     CX
        CLC
        JMP     XRET                ;RETURN TO CALLER
```

After you make these changes, assemble FASTIO.ASM into a .COM file. Assuming that MASM, LINK, and EXE2BIN are on drive A:, and FASTIO.ASM is on drive B:, you can assemble FASTIO with these commands:

```
A>MASM B:FASTIO,B:FASTIO;
A>LINK B:FASTIO,B:FASTIO,
A>DEL B:FASTIO.OBJ
A>EXE2BIN B:FASTIO B:FASTIO.COM
A>DEL B:FASTIO.EXE
```

*



49.95

WHIZ

Memory Resident Utility For The H/Z-100

WHIZ is a multi-purpose utility for the H/Z-100. It remains resident while other programs are loaded and run, and is available at the touch of a key, even while your application programs are running. WHIZ functions are accessed through pop-up windows, which can be moved around on the screen by the user. The initial release of WHIZ incorporates the functions described below. Continuing development will provide future updates which add additional functions to WHIZ.

Context sensitive HELP - provides you with the information you need when you need it.

ALARM CLOCK - beeps once per second for 10 seconds at the desired time of day, no matter what else is going on at the time.

ASCII Table - displays decimal and hex equivalents of ASCII characters, including control characters, and the current graphics set. A boon for programmers.

NOTEPAD - 80 character by 40 line notepad lets you make notes to yourself which are always available. It also serves as a text editor, letting you read, write and edit small text files and view files of any length.

APPOINTMENT CALENDAR - provides a 10 line by 40 character record for each date until well into the 22nd CENTURY! Only active records are stored to minimize disk space usage. You can "step through" the date file by day, week, or month.

SCIENTIFIC CALCULATOR - Built-in, full floating point algebraic expression evaluator. Rather than a cute but inefficient image of a handheld calculator, WHIZ allows direct entry of complex algebraic/trigonometric functions for immediate evaluation. Available under both the Notepad and Appointment Calendar functions, the calculator supports square root, sine, cosine, tangent,arc sine, arc cosine, arc tangent, hyperbolic sine, hyperbolic cosine, hyperbolic tangent, base 10 log,natural log, and natural anti-log.

Requires H/Z-100 at least 192K RAM; MS-DOS 2.13 or higher; does not run under ZDOS 1.xx.

Memory Resident - WHIZ requires 70K to 110K (user specified) of available system memory. All functions work, even on a minimum memory allocation, just not all at the same time.

Software Wizardry
THE MAGIC TOUCH

Software Wizardry, Inc.
1106 First Capitol Drive
St. Charles, MO 63301
(314) 946-1968
TWX 910-380-4822

Operating Systems: Fact, Fiction, And Future

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

As promised last month, this article begins the first of a series of general information topics for all microcomputer users. In most cases, the information presented in this series will apply to all brands of microcomputers, including the IBM PC, which uses the MS-DOS operating system or one of its derivatives, such as PC-DOS. As a result, a significant amount of information discussed in this series will be hardware independent, which means that it applies to the H/Z-100, the H/Z-150/160, IBM PC, and Compaq among others.

Since a number of the terms tend to be quite difficult to use repeatedly, I will use a common set of names as follows:

MS-DOS:	Includes all versions of the disk operating system initially developed by MicroSoft and includes popular names like Z-DOS and PC-DOS.
CP/M:	Includes all versions of the disk operating system developed by Digital Research such as CP/M-80, CP/M-85, CP/M Plus, CP/M-86, and Concurrent CP/M-86.
H/Z-100:	Includes Heath and Zenith versions of the low profile and All-in-One computers such as the Z-110 and Z-120 models, but excludes the IBM compatibles.
H/Z-150:	Includes Heath and Zenith versions of the popular IBM compatible computers including the portable H/Z-160. Sometimes called the Z-100 PC series. Unless otherwise noted, this term also includes the IBM PC and compatibles.

I have a hearty dislike for the currently popular term "computer literacy". It somehow implies that a person is illiterate, which rubs me the wrong way. We will, in this series, talk about general information on microcomputers, and if you choose to call that "computer literacy", that is up to you. I prefer to talk about general information only because this series will provide information for the beginner, as well as the advanced user. Based on some discussions that I have had at local HUG meetings, I have found that even many of the advanced users have not had time to get into some of the details that we will discuss here.

Definitions And Terminology

I will make a concerted attempt to discuss all definitions and terms before I use them in the article. Since I will generally use each article as a building block for future discussions, I suggest that each of these articles be read in the order of their publication.

Many computer definitions and terms tend to be academic and difficult to understand. Therefore, I will use a "working definition" which will be technically accurate, but may not include some of the very subtle intricacies of the term. Two of the very common terms – software and hardware – will be used throughout the series, so we will start with those definitions.

SOFTWARE includes all of the programs and the documentation associated with a computer system. Although many people seem to know that software includes the programs for a computer, most do not seem to recognize that software also includes the documentation for their system. The major portion of this article will focus on operating systems which is a special type of software.

HARDWARE, in contrast to software, includes all of the physical equipment used in the computer system. That includes just about everything else such as the computer itself, the monitor or CRT (cathode ray tube – sometimes called a video display terminal or VDT), disk drives, modem, printer, connecting cables, and so on.

In addition to hardware and software, another term has been used to describe a combination of both – firmware. FIRMWARE is typically a unit of hardware, such as the so-called microcomputer chip or integrated circuit (called an IC), which has been programmed and contains software. Not being content with the introduction of a single term, the industry has seen fit to introduce some additional terms which describe specific types of firmware. A ROM (Read Only Memory) or PROM (Programmable Read Only Memory) are typical examples of firmware. Virtually every computer (mainframes to micros) on the market has firmware in one form or another. For those of you who read my column in July 1984, you may recall that the ROM in the IBM PC is one of the significant problems related to the PC compatibility. It contains copyrighted program code which performs special functions (and parts of IBM BASIC) for the IBM PC. Because the ROM code is copyrighted, other microcomputer manufacturers cannot simply DUPLICATE that code since it would be a copyright infringement. That ROM is one of the key reasons that the H/Z-150 cannot run virtually all of the IBM PC programs. Since the ROM also contains parts of IBM BASIC, that also explains why H/Z-150 systems cannot run IBM BASIC – you must use GW BASIC which does not require that particular ROM code. From all reports, you can run programs developed in IBM BASIC under the GW BASIC interpreter for the H/Z-150. That is one way around the ROM copyright problem.

A Look At Software

Before we go too much farther, there are some other terms that I will use consistently to describe certain kinds of software. Although these terms are also used extensively in the mainframe data processing environment, I find them very useful for discussing all types of software.

SYSTEM software is all programs supplied on the distribution disk for your operating system, whether it's MS-DOS, PC-DOS, or one of the many versions of CP/M. That includes FORMAT, DISKCOPY, and PIP. System software also includes some of the "programs" that you can not see with the DIR command...we will take a look at that later in this article.

APPLICATION software is everything else, whether it's WordStar, WatchWord, Lotus 1-2-3, SuperCalc or dBase III. In that context, the application is word processing, spreadsheets or a data base. Some application software can obviously support the requirements of different business functions (e.g. spreadsheets or data bases). For example, a general purpose data base can be used in such diverse business functions as recording invoices for accounts payable, inventory control, and even payroll processing. Although it's generally better (and cheaper) to buy software for a specific application like accounts payable, some businesses have unique requirements which dictate that they must have custom software.

What Does An Operating System Do?

The primary function of an operating system is to allow users and programmers to conveniently use a computer system's resources. Resources, in this context, simply means the keyboard, memory, disk drives, CRT (cathode ray tube) or monitor, printers, etc. As an example, pressing an "A" key on the keyboard results in the display of an "A" on the CRT. The operating system takes care of interpreting the electrical signals from the keyboard and translating them into a video display form. Most operating systems provide features which also aid the programmer who develops application software. Both CP/M and MS-DOS provide system calls which allow the programmer to read the keyboard through the operating system. In short, operating systems take care of the details of the resource management and use of the microcomputer system.

Some History

Operating systems for microcomputers have been around longer than many people currently believe. Most will be surprised to learn that the popular microcomputer operating system, CP/M, has been around since late 1973. CP/M, which means Control Program/Monitor was invented by Gary Kildall, now president of Digital Research. Since paper tape and cassettes were clumsy and inefficient for microcomputers, the Disk Operating System or DOS was born. At this point, it seems appropriate to note that DOS, in one form or another, has been used on mainframe computer systems for years.

The actual definition of the term "DOS" has become somewhat blurred over the years. Many people like to think of DOS as being located on a disk (i.e. disk based) instead of on paper tape or cassettes. Others maintain that the "Disk" part refers to the primary form of input and output (e.g. storage) as opposed to paper tape and cassettes. For our purposes, both definitions are true. CP/M and MS-DOS are located on disk, and either floppy or hard disks are used as the primary form of input and output.

CP/M became the standard operating system for virtually all of the 8-bit microcomputers. One of the key reasons, I believe, is that it provides a very clever hardware interface (called the BIOS) which allows it to work with virtually any 8-bit system. This particular innovation was unique, and even MS-DOS uses the same idea. I suspect that all future operating systems will use a BIOS in one form or another only because that makes it easy for manufacturers, like Heath and Zenith, to customize the system for their particular hardware.

Until 1980, CP/M had virtually no competition even though some manufacturers, like Heath, wrote operating systems for their hardware. IBM was embarking on a secret venture, later to become known as the IBM PC, and visited MicroSoft in order to find out about programming languages. They asked Bill Gates, President of MicroSoft, about the CP/M operating system for their new secret project. Gates called Gary Kildall at Digital Research to provide an introduction for the IBM representatives and an appointment for the next day. In one of the most famous stories in the industry, reports are that "Gary was out flying" (for fun) while the IBM representatives waited. Although Kildall apparently disagrees with this version, it is one of the biggest blunders perpetrated by any leader in the microcomputer industry. The IBM people returned to MicroSoft to arrange for an operating system. That's how MS-DOS, and its derivative PC-DOS, became the "standard" operating system for 16-bit computers.

The follow-on to the story is also interesting...MicroSoft did not really write the first version of MS-DOS. It was initially written by Seattle Computer Products and purchased by MicroSoft. As part of the work for IBM, MicroSoft had to modify the system to work on the IBM PC. And obviously that work was successful beyond anyone's dreams at MicroSoft.

For those of you interested in a lot of the historical details related to the microcomputer industry, I suggest that you read "Fire in the Valley" by Paul Freiberger and Michael Swaine. It's an extremely interesting book which provides a lot of background on the developments in the Silicon Valley. I happened to find a copy for $9.95, but you're on your own as far as price is concerned. It contains 288 pages with a number of photographs. If you have a microcomputer, this is MUST reading. It is a highly recommended book.

The Structure Of A Microcomputer DOS

Since the basic structure of MS-DOS and its derivatives is very similar to CP/M, we can take a look at both. This similarity is not an accident. It happens that CP/M was used as a model for MS-DOS, and although some of the technical details are different, the user interface is quite similar. For example, the DIR command displays the disk directory for both operating systems. And both also use the FORMAT command.

The Basic Input/Output System (BIOS)

I mentioned earlier in this article that one of the unique features of the microcomputer DOS was that most of the popular ones provide for easy customization by manufacturers for different hardware. That is generically called the Basic Input/Output System (or BIOS for short) or I/O (Input/Output) Manager. The BIOS is the hardware dependent component of the DOS.

In CP/M, the file name is generally something on the order of BIOS.SYS, although the BIOS for the H/Z-100's CP/M-85 has two BIOS files - BIOS85.SYS and BIOS88.SYS. In MS-DOS, the file name is IO.SYS, and PC-DOS uses IBMIO.SYS as a file name. Regardless of the actual file name, the BIOS contains the hardware unique information for a specific computer. With the exception of the PC compatibles, that is one reason that one manufacturer's version of an operating system will not run on another manufacturer's computer...the BIOS is different. It also explains why the H/Z-89's CP/M-80 will not run on the H/Z-100's.

Now that we know that the BIOS is hardware dependent, let's take a look at some of the information that the BIOS actually contains. I like to think of the BIOS as a translator. For example, all of the codes generated by the keyboard are translated into values that the computer can use. The H/Z-100 has one set of keyboard codes and the H/Z-150 (and IBM PC) has a completely different set. Screen (i.e. CRT) control codes are different. The H/Z-100 uses an ESC E sequence to clear the screen, and the H/Z-150 uses ESC J. Regardless of the sequence used, the code to clear the screen is hardware dependent and is contained in the BIOS.

In order to access any kind of floppy or hard disk, a disk controller must be used. The BIOS contains the appropriate code for the specific disk controller(s) available to the system. It also defines the type and characteristics of the disk drives. The current Zenith operating systems generally allow a maximum of two 5-1/4 inch drives, two 8 inch drives, and up to 4 hard disk drives (not partitions). I won't even try to cover all of the exceptions to that statement since it depends on the specific operating system. The point is that you would have to modify the BIOS to add a third 5-1/4 inch drive or 8 inch drive.

I mentioned the ROM in the IBM PC which has been one of the compatibility issues. I purposely omitted its full name - it's really called the ROM BIOS - since I felt that it would be confusing to introduce too many terms at the same time. In the IBM PC (and the H/Z-150), the entire BIOS for the system includes the code in IBMIO.SYS, as well as the ROM BIOS. The justification for including part of the BIOS in the ROM is speed...memory is faster than disk I/O. While that sounds just fine in theory, it doesn't quite work that way as a practical matter. Changes in the ROM BIOS have caused compatibility problems within the IBM PC line. That is, some of the early PC's won't run some software developed on a later ROM version because of changes. In addition, some software developed for the PC and XT will not run on IBM's new AT because of the ROM differences. In my opinion, I would much rather sacrifice a little speed for the "upward compatibility." PC and XT owners will have to be sure to check all of their software before they buy an AT or they may be in for an unpleasant surprise.

Now that we know something about the BIOS, let's take a look at an example. Many of you may have seen either computer hardware reviews of the IBM compatible micros or advertisements which note that the system will run the MicroSoft Flight Simulator program. Why do manufacturers make such a big deal about that? The answer is that the Flight Simulator exercises virtually all of the ROM BIOS "features" and is a very reasonable "test" for compatibility.

Since the purpose of this discussion is to give you a feel for the BIOS, I will not cover all of the contents or any actual technical details. That requires a knowledge of assembly language which is beyond the scope of this article. For those of you who are interested, the BIOS source code for the H/Z-100 and the H/Z-150 is part of the Programmer's Utility Pack (PUP).

The System Kernel

The second major component of the DOS is the system kernel. Typical file names are MSDOS.SYS or IBMDOS.SYS. In CP/M, this component is also known as the Basic Disk Operating System or BDOS and does not have a separate file name. The system kernel is hardware independent. It essentially contains the system interface used by most system and application programs. The system kernel primarily contains system management functions related to file handling and memory, as well as the standard system calls.

The file handling function keeps track of the directory information such as the physical location and the size of the file.

The memory management function controls the Random Access Memory - known as RAM - which contains programs and/or data. The available RAM in CP/M is also known as the Transient Program Area (TPA) since the programs/data are only loaded as needed and may be overwritten when completed. ALL Programs that have a COM, EXE, or CMD (16 bit CP/M only) suffix are loaded into RAM for execution.

Standard system calls (sometimes called a service routine) are generally provided by all operating systems. An example of a standard system call is to "read a character from the keyboard". In assembly language, a certain register is set to the value 1 to indicate this function. Then an instruction is sent to the system to read the character. The system kernel identifies the request and goes through the BIOS (for translation to the hardware) and returns the character "value" to the calling program.

In my opinion, it is good programming practice to use the operating system calls whenever possible. Again, the primary reason for that is to provide for the upward compatibility that I mentioned earlier. In some cases, it may be preferable (or mandatory) to bypass the system calls and use the BIOS directly to perform some functions. You can argue that a direct jump to the BIOS is faster and therefore more efficient, but the disadvantage is that the BIOS addresses usually change in new releases in the operating systems. If you code an exact address in your program and assume that the appropriate BIOS code is ALWAYS there, the program may perform some interesting gyrations in a new release of the DOS if the BIOS was changed at that address.

An example of a program that bypasses at least one of the system features is Lotus 1-2-3. For a reason that I have not been able to determine, perhaps speed, Lotus "takes over" the keyboard directly. That could lead to all kinds of interesting results if the keyboard codes were ever changed.

In summary, the kernel performs the file and memory management functions, as well as providing an interface to the operating system known as system calls or service routines. The system calls usually remain stable so that upward compatibility between releases can be maintained.

The Command Interpreter Or Console Command Processor

The last major component of an operating system is the MS-DOS Command Interpreter (COMMAND.COM) and its CP/M counterpart, the Console Command Processor (CCP). No, I didn't forget the file name for the CCP – it is not a separate file. The CCP is part of the system kernel in CP/M. But before we get too involved in the differences, let's look at the similarities. As a matter of definition, the MS-DOS COMMAND.COM is also called the CCP.

The function of the CCP is to act as the interface between the computer user and the system. It is responsible for interpreting the commands given at the system prompt and passing the appropriate information to system and application programs. The MS-DOS and CP/M CCP's also contain built-in commands which are not located on a disk as a COM, EXE or CMD file.

At this point, I will digress a moment to define built-in and disk resident commands. When I began writing the FlipFast books, there was no standard name for the two types of commands. CP/M uses built-in and transient, MS-DOS uses resident (or system) and transient, and PC-DOS uses internal and external. NUTS! It's no wonder that a lot of people go crazy trying to figure out the terminology in computers. Isn't technology great? Since I was never convinced that any of those combinations intuitively described the command types, my publisher and I spent lots of hours trying to come up with a combination that we thought would be more obvious. We thought the contrast between built-in and disk resident was the best.

BUILT-IN commands are contained in the CCP for both operating systems. These commands are not stored as separate files on a disk like FORMAT and DISKCOPY. Built-in commands may be executed at any command prompt for the system regardless of drive and/or directory (or user number in CP/M). Most versions of CP/M have about half a dozen built-in commands. The latest versions of MS-DOS and PC-DOS have over 30.

DISK RESIDENT commands are stored on disk as separate files (i.e. programs) and must be loaded into memory before they can be executed. These commands have a file type of COM or EXE (CMD in 16 bit CP/M), and may be preceded by a drive designator and/or path name (MS-DOS/PC-DOS only). The MS-DOS CCP, COMMAND.COM, is also a disk resident command and can be executed just like any other command.

One other significant feature of both CCP's is that they are usually reloaded after the execution of disk resident programs just in case part or all of the code was overwritten by a program or data.

Additional Features Of The MS-DOS CCP

Because of the size and complexity of the MS-DOS CCP, it contains three major sections of code – resident code, transient code, and initialization code which is only available during system boot.

The resident portion of the CCP allows the DOS to load programs into memory and handle device errors. It also checks the transient portion of the CCP, and reloads COMMAND.COM if it has been overwritten by a disk resident program.

The transient portion of the CCP contains all of the MS-DOS built-in commands as well as the batch processor for the BAT files. This section of code contains the search order for the MS-DOS commands – built-in, COM, EXE, and BAT files in that order. For those of you who are not familiar with the search order, I suggest that you memorize it.

As an example, let's say that we wanted to develop a BAT file which displays some kind of directory listing. And so we might want to name that file DIR.BAT. MS-DOS will first look for the DIR command in the transient portion of the CCP and will execute the built-in DIR command. The DOS will never "find" our DIR.BAT file. Similarly, the DOS will never find FORMAT.BAT if it is able to locate FORMAT.COM first.

The initialization portion of the CCP is only resident during the initial cold boot of the system. It contains the code necessary for the setup of the AUTOEXEC.BAT file execution and also determines the appropriate memory (for those of you familiar with assembly language, the exact word is segment instead of memory) address for program loading. After performing its function, this part of the CCP is overwritten by the first program that COMMAND.COM loads after startup (i.e. cold boot).

A Summary Of The DOS Components

The DOS contains three major components. The BIOS is the only hardware dependent component and acts as a translator between the software and the hardware.

The system kernel is a hardware independent component of the DOS. It provides file and memory management capabilities as well as the standard operating system calls for use in programming.

The Command Interpreter or Console Command Processor is the primary interface between the user and the DOS. It contains built-in DOS commands and processes all disk resident commands. Most operating systems usually reload the part or all of the CCP after the execution of a disk resident command.

What Is REALLY On A System Disk?

Now that we have a good idea of the DOS components, let's take a look at what you can actually find on a system disk. And then we'll find out about a data disk followed by a discussion of why you need both.

One of the things that I have ignored up to now is a discussion of the Boot Loader. Some manuals consider the boot loader part of the DOS because it is required to load or boot an operating system. Other manuals either ignore it entirely or pass over it quickly because it is only used when the DOS is initially loaded. I will pass over it quickly since it is important that you know its purpose as well as the fact that it is recorded on EVERY disk by the FORMAT program.

The boot loader (technically called the bootstrap loader) is a very short (1,000 bytes or so in MS-DOS) and simple program that provides the initial instructions to load the DOS into memory from a disk. From that description, you may be able to correctly deduce that the boot loader is hardware dependent (like the BIOS) and is generally written by each manufacturer of computer hardware (like Zenith). The boot loader is always located in the first part of a floppy disk or hard disk partition. For those who are so inclined, the technical description of a floppy disk is side 0, track 0, sector 1. Unlike every other file on the disk, the boot loader does not have a file name (or a directory entry), and you won't see it unless you use DEBUG or DDT to look at the first sector of a formatted disk.

Although CP/M was used as a model for MS-DOS, it happens that MS-DOS is very particular about the location of system files on a disk. Therefore the following will only apply to MS-DOS and PC-DOS since CP/M is not quite as fussy.

The BIOS (like IO.SYS) file name for MS-DOS must be the first entry in the disk directory. Similarly, the kernel (e.g. MSDOS.SYS) must be the second entry in the directory. If the BIOS and the kernel are not located EXACTLY where the boot loader expects to find them, you don't have a correctly formatted system disk as defined to the boot loader. The CCP for MS-DOS, COMMAND.COM, is usually the third file in the directory, but that isn't a requirement.

I have mentioned in a previous column that the MS-DOS SYS command gets my vote as the most useless command in any version of MS-DOS. Two special notes are also listed in the Requirements section of the MS-DOS FlipFast book which bear repeating here. First, the SYS command will NOT transfer files to a newly formatted disk. Second, the SYS command will only replace the BIOS and system kernel when the new file size is equal to or smaller than the existing files on the disk. If you didn't format the disk with the /S option, you'll have to format another disk and copy the existing files to that disk.

There are several ways to fix that problem. One way is to make the boot loader more "intelligent" so that it searches the disk directory for the files like CP/M does. Another possibility is to provide a way for the FORMAT program to "reserve" the first two directory entries and required disk space for those files. I suspect that Zenith will find a way to correct that problem in a future release of MS-DOS.

For the sake of completeness, I will simply note in passing that the FORMAT program also creates the disk directory and the File Allocation Table (FAT) regardless of any command options. We will take a look at that in more detail when we take a look at the DOS commands that you must know.

In summary, a MS-DOS system disk is created by the FORMAT command with the /S (System) option. The boot loader is always the first "program" on the disk, and it does not have a file name or directory entry. The BIOS (IO.SYS or equivalent) and the system kernel (MSDOS.SYS or equivalent) must be the first two files in the disk directory. The CCP, COMMAND.COM, must also be on the disk, but may be located anywhere in the disk directory.

The CP/M System Disk

Now that you understand everything about an MS-DOS system disk, you also understand everything about a CP/M system disk except perhaps the commands. Many manufacturers (notably Heath and Zenith) provide a SYSGEN command for their 8 bit versions of CP/M which transferred the BIOS and the system kernel to the destination disk after the disk had been formatted with the FORMAT command. It's interesting to note that a system disk could be created at any time provided that sufficient directory and disk space was available on that disk. The CP/M boot loader was "intelligent" and searched the disk directory for the appropriate files. The 16 bit versions of CP/M use slightly different commands and also require that you copy the CPM.SYS file or equivalent to the destination disk.

The only real difference in the CP/M system disk is that it contains only the boot loader, the BIOS, and the system kernel. The CCP is included in the system kernel and is not a separate file like it is in MS-DOS. Like its MS-DOS counterpart, the CP/M FORMAT program also creates a disk directory (but not a FAT) on every disk regardless of command options.

What Is A Data Disk?

Since you know all of the files that are contained on a system disk, the definition of a data disk is easy. A data disk does not contain the BIOS file, the system kernel, and the CCP. That's all well and good, but why, you might quite reasonably ask, would you want to have two kinds of disks?

In order to make a point, I'd like you to try an experiment with MS-DOS or PC-DOS. Take two blank disks, and label one as a "system" disk and the other as a "data" disk. Format the system disk using the FORMAT/S command. Format the data disk using the FORMAT command with no options. Press RETURN or ENTER when asked for a label. Now run CHKDSK on both disks and note the values shown as "bytes available on disk" is significantly less on the system disk. And that of course is the first point...you can effectively have more disk space for data on a data disk since you don't need the system files on EVERY disk.

Disk space is a key concern in any system, but can become a significant problem in one that does not have a hard disk. For that reason, it is especially important to use the system/data disk concept. Since we know that a system disk contains some key DOS files, we can stretch that definition to say that our system disk may contain other system AND application programs.

A typical system disk of mine might contain an AUTOEXEC.BAT file, FORMAT, CHKDSK, WordStar, and SuperCalc files in addition to the standard DOS system files (e.g. FORMAT, DISKCOPY, etc.) that we have discussed. I then use that disk to boot the system, do word processing (WordStar), and run a spreadsheet (SuperCalc). The data disk is in drive B and the content will depend on the work that I'm doing at the time. One data disk contains my REMark articles and another is for miscellaneous correspondence. The same concept applies to the spreadsheet disks. One disk is for personal use and another disk is for business. The whole point of this discussion is that I don't have COM or EXE files on a data disk. Why copy those program files and use up valuable disk space?

Although there are lots of reasons for using this concept, I think the best is the time that it takes to do things. I have about 50 floppy disks that I use for MS-DOS. About 5 of those disks contain software in the form of system and application programs. When I changed from Z-DOS version 1 to MS-DOS version 2, it was a relatively simple matter to FORMAT 5 new disks (with the /S option) and copy the appropriate system and application software to them. The rest of the disks contained data of one form or another and required no change. The same thing is true when I received the recent update to MS-DOS 2.21. I only had to update 5 disks. Aside from providing for more effective use of disk space, the time savings are also evident when you create backup copies of data disks (you are doing that, aren't you?) since you don't have to copy programs in addition to the data. If you don't have a lot of extra time and floppy disks, I highly recommend using the system/data disk concept.

I have assumed that you have at least two disk drives in your system. If you don't, you have my deepest sympathy, and I strongly recommend that you get a second drive for your system. I've worked with an IBM PC Jr. which has a single disk drive, and it's like shooting yourself in the foot...very painful and it takes a

LONG time to do things. For those of you with a hard disk drive, you can substitute the word "partition" for disk drive, however, if you decide to try the experiment described above, be sure to backup your hard disk BEFORE you try the FORMAT command. In case you didn't know it, FORMAT completely erases a disk, and there is absolutely no hope of recovering any previous files on a formatted disk.

More On Operating Systems

There is another dimension of operating systems that is certainly worth some review. It has to do with how many tasks or users an operating system can handle. Windows, Concurrent CP/M, Concurrent DOS (MS-DOS and PC-DOS), and even IBM's Topview are certainly part of this discussion. But before we can intelligently discuss Windows and other pains, let's take a look at some terminology.

The existing implementations of MS-DOS (2.xx) and PC-DOS (3.0) are capable of handling a single user performing a single task. That is, one person can execute one task (e.g. command), wait for the system to respond, and then execute the next task or command. One example of this is to print a file with a word processor (task 1) followed by some calculations using a spreadsheet program (task 2). You have to wait for the word processor to complete the printing process before you can execute the spreadsheet program. The DOS is a SINGLE USER, SINGLE TASK system, and is sometimes referred to as a single user system since the "single task" is understood. That's obviously not the most efficient way to get your work done. CP/M-86 and the 8 bit CP/M versions are also in the same category.

Since loading a program into memory can be very time consuming, one step forward is to provide multiple views, virtual consoles or Windows to the system and application software. In general, the "Windows" approach by itself simply "freezes" the current program (like Lotus 1-2-3 or a batch file) and displays a window which allows access to another program (e.g. Perks). Since the current application is "frozen", all of the computer resources are still devoted to a single task which is Perks in this case. In this example, we are still talking about a single user, single task system even though we can look through different windows at other programs.

Wouldn't it be nice if we could run a batch file (to make backup copies of course!), print a letter, and work on a data base? From a practical view, we have entered a batch file command on one console or window, executed the word processing program to print the letter in another window, and started our data base program in yet another window. In other words, the computer is executing multiple tasks consisting of the batch file, word processor, and the data base at the SAME time. This process is called multitasking in mainframe computers, which is simply the concurrent execution of two or more tasks by a computer. The microcomputer vendors apparently seem to think that concurrent is a better term since Concurrent CP/M-86 is now available for the H/Z-100, and MicroSoft keeps talking about Concurrent DOS.

It's also important to understand that concurrent processing tasks is NOT the same as simultaneous processing even though it may APPEAR to be simultaneous to the user. The essence of concurrency is to allocate processing "time slices" of the computer's resources to each task. In order to demonstrate the process, let's assume a basic reference time of one second and assume that we have two, 4 second tasks to perform. The "time slice" is the reference time (i.e. 1 second) divided by the number of tasks (2 in this example) or 1/2 second. Of course all computers operate much faster than this, but the principle is the same. The computer then begins execution of task 1 and continues for 1/2 second. Task 1 is stopped, and the execution of task 2 begins and continues for another 1/2 second. At the end of that 1/2 second (1 second elapsed time), the computer resumes processing of task 1 for another 1/2 second. Task 1 is again stopped, and task 2 resumes for another half second. This process, called INTERLEAVING, continues until both tasks are complete in 8 seconds of elapsed time. If you have followed this discussion so far, you probably can see that a predictable consequence of any concurrent processing is that any SINGLE task will take LONGER when multiple tasks are being executed.

What This Means To You

All of this sounds great, but there is at least one major impact to all microcomputer users. Simply stated, it's TANSTAAFL -there ain't no such thing as a free lunch! You may have seen some reports about the IBM PC II and the AT. You probably have seen some of the latest Zenith computers like the Z-200, which is AT compatible or the speed up/memory expansion kit to 8 megahertz/768K for the H/Z-100. I made a statement in a previous column about some of these new features requiring additional memory and a hard disk. It appears that I didn't carry that far enough. The computer must also process faster in order to obtain any kind of reasonable performance from these new features. The bottom line is that, if you want these features, be prepared to pay for them.

Windows Are A Pain

According to a recent issue of InfoWorld (July 1, 1985 to be exact), MicroSoft has begun shipping (as of June 28) the Windows software to manufacturers. They are well over a year late in shipping Windows since it was announced in November 1983. As for Windows, I'll believe it when I see it. In addition, I don't see any real advantage to Windows until concurrent processing is also available even though InfoWorld states that it has multitasking capabilities. I can't help but wonder if the author of that article understands what multitasking really is. InfoWorld further reports that Windows will include some standard "desk accessories" like a card file, notepad, clock, and calculator.

According to the same article, Windows is expected to require about 128K of memory, and you can add about 40K (use CHKDSK to find out the exact amount for your system) for the MS-DOS operating system itself.

Other Operating System Additions

Another product that has received a lot of attention lately is IBM's Topview. Topview also provides a Windows-like view of the system as well as concurrency. I've seen Topview, and I've not been very impressed with the configuration requirements. It looks like it will take 3 Computer Science PhD's and a dog to successfully set up Topview for most applications. One of the particularly frustrating requirements is that you must define the amount of memory required by a program that runs under Topview. That is not the same as the hardware requirement for the software. If a word processor requires 256K of memory according to the manual, subtract 40K for the operating system. Of the remaining 216K, how much is required for the actual program and how much is required for the data? Try to find THAT information in a manual! We ended up using the 216K for the program requirement, but I suspect that was well over the actual requirement. Incidentally, the dog is required to provide moral support for the PhD's while they are trying to set up the Topview configuration.

Another particular disadvantage of Topview is that you can't access a lot of DOS utilities – it's very restrictive. In the version that I saw, you had to exit Topview in order to FORMAT a disk. Topview also requires more memory than Windows...about 150K or so, but does not support graphics.

Digital Research's Graphics Environment Manager (GEM) is another similar package. Reports indicate that it does not provide multitasking (because Concurrent CP/M-86 is available?) although it does provide graphics. Of these three packages, GEM is the biggest memory hog and requires about 169K.

The Future

It's always difficult – not to mention dangerous – to predict the future. I don't like to be wrong. However, I see some definite trends developing that are worth discussing.

I see a clear trend that a lot of the new operating system software and add-on software (including application software) will require hardware changes. Although it may be as simple as adding more memory and a hard disk, I doubt it. Computer clock speeds will also have to be increased in order to provide any kind of reasonable user performance. Manufacturers have apparently recognized that hardware need since we are now looking at 10 megahertz Z-200's and AT's plus upgrades to H/Z-100's and H/Z-150's. The current clock speed of 4.77 megahertz just won't provide reasonable performance with complex software.

I also expect the popularity of the so-called integrated software to be short lived. That includes programs like Symphony, Framework, and WordStar 2000. Although integrated software provides a number of features like word processing, spreadsheet, and data base functions in a "single" package, the disk space requirements alone can be staggering. The complete WordStar 2000 software, for example, comes on six distribution disks and requires a monstrous 2 megabytes to download the entire package.

I am also not convinced that the "common" command structure in these integrated packages is all that great either. Why should I spend the time (and money!) trying to learn new commands when I already have a word processor, spreadsheet, and data base? Perhaps the only rationale for buying integrated software is that it is quite cost-efficient (i.e. cheaper) compared to the sum of the costs of individual programs for new computer users.

Integrated software is related to operating systems only by the fact that add-on system software, like Windows, may provide a similar capability with existing software. Topview is about the most expensive at $149.00 according to the IBM Product Center. Suggested retail prices for Windows and GEM are considerably less, but I won't quote some of the prices I've seen.

Next Month

I hope that you have found the information in this article to be helpful. If you have to read it a couple of times, that isn't very surprising. I've discussed a lot of new terms and concepts which are relatively new in the microcomputer world. I've spent nearly 20 years in data processing learning about these things...maybe I'm just a slow learner.

If you would like me to continue articles on various subjects like this, be sure to let me know. You'll find my preliminary list of topics in last month's column. When you write, be sure to enclose a stamped, self addressed envelope if you expect a reply. You can generally expect to receive a reply within three weeks from the date that you mail the letter since I answer all mail in the same week that I receive it, usually on the weekend.

Since I have a few of the topics in various stages of completion, I won't try to guess which one will strike my fancy for next month.

Products Discussed

"Fire in the Valley"	See text for price
Osborne/McGraw-Hill	
2600 Tenth Street	
Berkeley, CA 94710	
Programmer's Utility Pack (CB-5063-16)	$149.00
MS-DOS Version 2	
Z-100 only (OS-61-8)	$150.00
Z-150 only (OS-63-50)	$150.00
CP/M-86 (OS-63-2)	$ 99.00
Concurrent CP/M-86 (OS-61-12)	
Z-100 only	$299.00
Heath/Zenith Computer Centers	
Heath Company Parts Department	
Hilltop Road	
St. Joseph, MI 49085 (616) 982-3571	
Perks	
MS-DOS (Z-100 only)	$99.97
Heathkit Stores	
Barry Watzman	
560 Sunset Road	
Benton Harbor, MI 49022 (616) 925-3136	

INTELLIBURNER

EPROM-EEPROM-MICROCOMPUTER PROGRAMMER

UNIVERSAL PROGRAMMING CAPABILITIES AT AN AFFORDABLE PRICE

- **Ultra Fast Programming – 2716's in 16 Seconds**
- **Programs & Verifies 8K thru 256K Single Voltage EPROMs**
- **Erases, Programs & Verifies 2815, 2816 & 2817 EEPROMs**
- **Programs & Verifies 8748 and 8751 Series MICROCOMPUTERS***
- **Programming Characteristics Selected by Convenient Personality Jumper Plug (DIP Header)**
- **Program, Verify, Status, & Diagnostic Display with Tricolor LED**
- **Serial Interface — 3, 4, or 5 wire — 1200 to 19200 Baud**
- **Supports ACK/NAK, XON/XOFF and READY/BUSY Protocols**

NO SPECIAL SOFTWARE REQUIRED. Transfer disk files (Intel Hex or Motorola 'S' Format) to EPROM with your system's line printer or modem software. Transfer EPROM contents to disk file with your system's modem software. Or use the supplied software ** to transfer any binary or ASCII file to/from EPROM.

PROGRAMS:

2758	2716	27128	2815	8741*	8748*
2516	2732	27128A	2816	8742*	8749*
2532	2732A	27256	2817	X2212*	8751*
2564	2764	68764	27C Series		8755*

*Requires low cost personality adapter

SOFTWARE AVAILABLE FOR: **

CP/M systems on 8" SSSD – many 5¼ formats
TRS-80 Model I & III TRSDOS – NEW! TRSDOS 6.X
Heath H8/H89 HDOS & CP/M
ZENITH Z90 & Z100 CP/M – Z100 ZDOS – Z150
KAYPRO II/IV/10/16/286/2000
IBM PC, PC/XT, AT – many MS-DOS systems

IntelliBurner Programmer with Software	$ 299.00
Stand-Alone RAM OPTION with 64K Bit RAM (Expandable to 256K Bit)	89.00
RS-232 Interconnect Cable	12.00
IntelliBurner PC Boards, EPROM, Plans & Software	99.00

ROSS CUSTOM ELECTRONICS
INCORPORATED
1551 Sandra Drive
Boulder City, Nevada 89005
PHONE (702) 293-7426

Low Cost "DumBurner" serial programmers harness the power of your personal computer with the supplied software for full programming capabilities:

DumBurner II Programmer for 28 Pin and 24 Pin EPROMs and EEPROMs (with Software)	$ 199.00
DumBurner II Bare PC Board, Plans & Software	59.00
16K/32K DumBurner for 24 Pin EPROMs (with Software)	149.00
16K/32K DumBurner PC Board, Plans & Software	39.00

! ! NOW SHIPPING with ALL 'BURNERS ! !
TOOLKIT Software for Editing EPROM Information, Plus HEX/MOTOROLA 'S'/Binary File Conversion Utilities

Ultra Violet Products EPROM Erasers
Model DE-4 · Holds 8 EPROMS · Special Prices

Add $3 Shipping & handling ($2 Bare Boards), C.O.D.'s accepted. Foreign Orders add required postage. Specify Environment and Media Requirements.
H8, H89, Z100 are ™ Heath/Zenith. CP/M is ™ Digital Research. TRS-80 is ™ Tandy Corp.; IBM-PC, PC/XT, AT are ™ IBM. MS-DOS is ™ Microsoft.

Get *more news* about your Heath or Zenith computer system *more often*—AND get a *FREE* directory of over 400 sources of support for your computer.

Subscribe to **Buss: The Independent Newsletter of Heath Co. Computers** and also receive the directory *at no additional cost*

You can't get more frequent news about your system than with a subscription to *Buss*. Twenty times a year, in each and every eight-page issue, you'll find out what other users and independent suppliers have to say about:

☐ H8	☐ terminals	☐ graphics
☐ H/Z89	☐ disks and drives	☐ utilities
☐ H/Z100	☐ compatible software	☐ Z-DOS
☐ H/Z150	☐ business applications	☐ MS-DOS
☐ compatible hardware	☐ home/hobby applications	☐ CP/M
☐ printers	☐ games	

Twenty times a year, you'll get a complete report on the latest news about your Heath/Zenith system—delivered directly to you by first class mail. *Buss* helps you:

- Learn what new products are being developed, and where you can get them.
- Benefit from the advice of users when you write in to *Buss*'s "Requests for Assistance" column.
- Save money when you add to your system! *Buss* offers a free "For Sale" listing for Heath/Zenith users where you can pick up a bargain.
- Avoid bugs in your system as you read how other users have solved similar problems with their computers.
- Discover news and events in the Heath/Zenith community that affect you.
- Meet other users in your area through the "Local Club Notes" section of *Buss*.
- Get unbiased coverage of your system from the oldest independent newsletter for Heath/Zenith users. Since 1977, *Buss* has been committed to supplying information exclusively for Heath/Zenith users.

The BUSS Directory

Over 400 sources to help you complete your Heath® or Zenith computer system.

FEATURING:
- products for the H8 · H11 · H88 · H/Z89/Z90 · H/Z100 · H/Z150 · HERO
- an improved index to help you find the products you need
- a directory of local Heath Users' Groups
- an index to *Buss* issues in 1983–84

And, as a free bonus with your *Buss* subscription, you'll receive your FREE copy of the 1985 edition of *The Buss Directory*—a 92-page reference tool which provides you with information on over 400 sources of independent support for your Heath/Zenith system.

The Buss Directory is a $12.50 value—and it's yours FREE with your subscription to *Buss!* We'll send it to you by first class mail as soon as we receive your subscription order.

Order your subscription today, and start getting MORE NEWS, MORE OFTEN about your Heath/Zenith system!

Call Toll Free: 800/341-1522

(DATATEL 800™, for orders only, Monday-Friday 8 a.m.–9 p.m., and Saturday 9 a.m.–5 p.m., Eastern Time)

Or use this coupon, and mail your order to:
Buss, Dept. R85
716 E Street, S.E.
Washington, D.C. 20003
202/544-0900

Your satisfaction with *Buss* is completely guaranteed. If at any time you are not satisfied, your money will be refunded—in full.

Yes! Sign me up for *Buss*, and send me my free copy of *The Buss Directory*!

I want:
☐ 30 issues for $38 ($55 overseas)
☐ 20 issues for $28 ($40 overseas)
☐ 10 issues for $18 ($24 overseas)

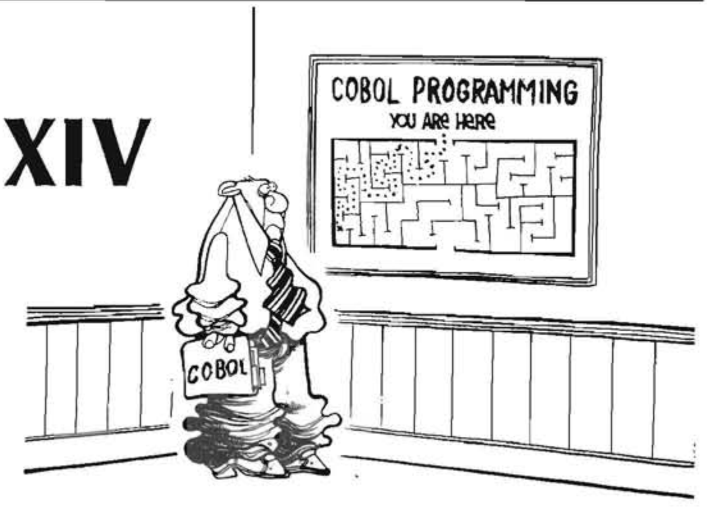
☐ Payment enclosed (checks must be in U.S. dollars payable on a U.S. bank).
☐ Bill me.
☐ Charge my: ☐ Visa
☐ MasterCard

Name____________________
Address____________________
____
____

Card #____________________
Expiration date__________
(MC code #__________)

Buss, Dept.R85, 716 E Street S.E., Washington DC 20003

COBOL Corner XIV



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

Introduction

"COBOL Corner" readers now know how to program a "useful" REPORT with program Headings, Column Headings, Page Totals, and Program Totals. Do you like the format of our last REPORTS we have programmed? I DO! The next step is to see if we can make them look a little more "professional" and easier for the user to use. I am sure that all of you readers have seen and used reports with "Control Breaks". What's that I hear you ask. I will show you soon. Most users just take them for granted. But, how many users have wondered what kind of programming design has gone into their presentation.

Sample Sales Report Without Control Breaks

```
STORE     DEPARTMENT        PRODUCT           SALES
NUMBER      NUMBER        DESCRIPTION        REVENUE

 Ø11        5ØØØ7         PRINTER           1,895.95
 Ø12        6ØØ12         DISK DRIVE          375.5Ø
 Ø12        6ØØ11         DSDD DISKS           42.ØØ
 Ø11        5ØØØ7         PRINTER RIBBON       11.ØØ
 Ø11        5ØØ17         PRINTER PAPER        41.ØØ
 Ø12        6ØØ12         DRIVE CABINET       21Ø.ØØ
 Ø12        6ØØ11         SSDD DISKS           29.ØØ
 Ø11        5ØØØ7         PRINTER CABLE        34.ØØ
 Ø11        5ØØ17         POWER CORD            8.ØØ

                          REPORT TOTAL      2,646.45 *
```

Sample Sales Report With Single-Level Control Breaks

```
STORE     DEPARTMENT        PRODUCT           SALES
NUMBER      NUMBER        DESCRIPTION        REVENUE

 Ø11        5ØØØ7         PRINTER           1,895.95
 Ø11        5ØØØ7         PRINTER RIBBON       1Ø.ØØ
 Ø11        5ØØØ7         PRINTER CABLE        34.ØØ
 Ø11        5ØØ17         PRINTER PAPER        41.ØØ
 Ø11        5ØØ17         POWER CORD            8.ØØ

           TOTAL FOR STORE NUMBER Ø11       1,989.95 *

 Ø12        6ØØ11         DSDD DISKS           42.ØØ
 Ø12        6ØØ11         SSDD DISKS           29.ØØ
 Ø12        6ØØ12         DISK DRIVE          375.5Ø
 Ø12        6ØØ12         DRIVE CABINET       21Ø.ØØ

           TOTAL FOR STORE NUMBER Ø12         656.5Ø *

                          REPORT TOTAL      2,646.45 **
```

Sample Sales Report With Multiple-Level Control Breaks

```
STORE     DEPARTMENT          PRODUCT            SALES
NUMBER      NUMBER          DESCRIPTION         REVENUE

 Ø11        5ØØØ7           PRINTER            1,895.95
 Ø11        5ØØØ7           PRINTER RIBBON        1Ø.ØØ
 Ø11        5ØØØ7           PRINTER CABLE         34.ØØ

        TOTAL FOR STORE Ø11 DEPT. 5ØØØ7        1,94Ø.95 *

 Ø11        5ØØ17           PRINTER PAPER         41.ØØ
 Ø11        5ØØ17           POWER CORD             8.ØØ

        TOTAL FOR STORE Ø11 DEPT. 5ØØ17           49.ØØ *

            TOTAL FOR STORE NUMBER Ø11         1,989.95 **

 Ø12        6ØØ11           DSDD DISKS            42.ØØ
 Ø12        6ØØ11           SSDD DISKS            29.ØØ

        TOTAL FOR STORE Ø12 DEPT  6ØØ11           71.ØØ *

 Ø12        6ØØ12           DISK DRIVE           375.5Ø
 Ø12        6ØØ12           DRIVE CABINET        21Ø.ØØ

        TOTAL FOR STORE Ø12 DEPT. 6ØØ12          585.ØØ *

            TOTAL FOR STORE NUMBER Ø12           656.5Ø **

                            REPORT TOTAL       2,646.45 ***
```

Report Control Breaks Concepts

Did you recognize the above reports with Control Breaks? Did you ever give them any special consideration? I bet that you just accepted them without further consideration. I will guarantee that you will watch reports in the future for Control Breaks after you complete the programming of the next two reports. We will do a "single-level control break report" and a "multiple-level control break report" in the next few "COBOL Corner" articles. We will be using control breaks in our advanced programs down-the-road-aways!

Records are usually sorted prior to preparing reports. Up until now and continuing for a few more programs, I have been furnishing the "transaction record file" to you sorted. We will be discussing SORTING and we will work with SORTING programs in a few more months. One of the main reasons that the files are sorted is so that they are easier for the report user to understand and refer to. However, our more advanced programs will require sorted input records for logical programming designs.

Consider the first example above of a Sales Report without Control Breaks. This Report could be prepared without the data being sorted as I have shown. It is likely though, that the general manager of this store chain would want to know the total sales revenue from each of the two stores. The most efficient way to process such a report would be to sort the sales transaction records by store number and print totals whenever all the records for one store had been printed. This is what I have shown in the second example above, which would be an example of a "single-level control break report" with the store number as a "control field". The input records would require sorting by store number to provide a logical method to program the data.

Perhaps the general manager wants to distribute the report to each store manager and to each department manager. The store manager would probably be interested in his store total and the department totals. However, the department managers may be more interested in what their department totals are. Thus, a report sorted by department number within the store number (the major field is store number and the minor field is the department number in sorting "lingo") is shown in the third example above. This is an example of a "multiple-level control break report". Notice that the store totals are a summation of the store's department totals.

It is easy to see that if this were a large chain store operation, the department manager would probably like to see the report sequenced by salesperson number within the department within the store. This would allow the department managers to determine how well each salesperson is performing. This control break report can be carried even further. The Buyers at the main office might be interested is seeing the report organized by product within the department regardless of salesperson or store, so they could easily spot product sales trends. Also, maybe someone in the advertising department would like to see sales totals sequenced by product within a date period so that they could determine the effectiveness of advertising programs. You can see the importance of knowing how to program control breaks! For practically all data processing applications, a given set of records may require sequencing (sorting) and then reporting in a variety of formats, depending upon the user of the report. This is the reason that many businesses have a Data Base Management System.

By now, I bet a lot of you "COBOL Corner" readers have seen the LOGIC required to program a control break report. Observe that the "control field" is one of the item code numbers. Whenever the item number changes, a total for a group of records is printed.

Control Break Report Programming

Most COBOL programming allows for more than one way to accomplish the desired results. Control Break Report programming requires a rather limited way to accomplish the result. The programmer must learn the procedure's logic. We will approach our design of control break reports in two main steps:

1. Design and coding of a single-level control break report.
2. Design and coding of a multiple-level control break report.

For this article we will concentrate on the single-level control break report. Remember, that the input records for the control break report must be sorted into the correct sequence by a previous sort routine (I have furnished such a file on the HUG COBOL Corner Disk-II — FILEL5.DAT).

Although control break programs are not overly complex, they can be very difficult if the programmer is not trained in their design and coding. I must advise the readers that they MUST adhere to the logic of the design that we will develop. This becomes even more important with the multiple-level report. There are two tricky things about a control break program. I want you to watch for these two "tricks" as we design Program #7, and see if you can find the two tricks and our solution. Here are the two tricky areas:

1. Bypassing the "false" or null control break, which will occur when the first record is processed.
2. Force out the last control break after all records have been processed.

A good STRUCTURE CHART with MODULES is a must to follow the programming!

Program Description

I want to make our "single-level control break report" fairly easy as far as the data file is concerned. I want to have the reader concentrate on the "logic" of the programming rather than on an elaborate formatted report. Also, I have been having you do all financial reports up until now, so this time we will do something different — School Test Results! As your Program Analysis I will supply the following:

```
----------------------------------------------------------
                  PROGRAM SPECIFICATIONS
----------------------------------------------------------
PROGRAM NAME   ROOM TEST RESULTS       PROGRAM ID:  PRGM07
----------------------------------------------------------
```

Program Description:

This program reads the Room Test Result Records and prints a Room Test Results report with a detail-line for each student. When the Room Number changes a Room Total-line is printed which will provide the total students in the room, the number of correct answers, and the room's average score. After all the input test result records have been processed, a Report Total-Line will be printed showing the total number of students in all of the rooms, the number of all correct answers, and the report's average score.

Input File:

Test Results File — FILEL5. (Presorted by room number)

Output File:

ROOM TEST RESULTS.

List Of Program Operations:

A. READ records from the input Room Test Result Records file.

B. For each Test Result Record:

   1. Print a detail-line containing the following fields in accordance with the format shown on the Print Chart:

      a. Room Number.
      b. Student Name.
      c. Correct Answers.

2. Accumulate the following totals:

   a. Total number of students in each room.
   b. Total correct answers in each room.
   c. Report total number of students.
   d. Report total correct answers.

C. Whenever the Room Number changes, the program will print a room number total-line containing the following fields in accordance with the format shown on the Print Chart:

   a. Room number.
   b. The words "TOTAL STUDENTS: ".
   c. Total students in that room.
   d. Total correct answers for that room.
   e. The words "AVERAGE SCORE: ".
   f. Average score for that room (total correct answers for room divided by total students in that room).
   g. One asterisk following average score.

D. After all input Test Result Records have been processed, the program will print the following total fields on the Report Total-Line in accordance with the format shown on the Print Chart:

   a. The words "REPORT TOTAL".
   b. Total number of all students.
   c. Total number of all correct answers.
   d. The words "REPORT AVERAGE SCORE: ".
   e. Report average score (report total number of all correct answers divided by report total number of all students).
   f. Two asterisks following report average score.

E. Headings (one Report Header and two Column Headers) are to be printed on the first page of the report. After a "range" of 51 thru 54 lines have been used on a report page, the program will skip to the next page and print the Headings again.

   1. The RUN-DATE will be obtained from the date data furnished in the WORKING-STORAGE-SECTION. It will be printed on the Report Header in accordance with the format shown on the Print Chart.
   2. The PAGE-NBR will be incremented each time the Headings are printed and will be located on the Report Header in accordance with the format shown on the Print Chart.

F. Line-spacing will be as follows:

   1. The first Report Header will start one inch from the top of the page. The first Column Header will be triple spaced from the Report Header. The second Column Header will be single spaced from the first Column Header.
   2. The first detail-line will be double spaced from the second Column Header.
   3. Second and successive detail-lines for the same Room Number will be single spaced from one another.
   4. Each control break room total-line will be double spaced from the previous detail-line.
   5. The first detail-line for each Room Number will be triple spaced from the previous control break room total-line.

G. COBOL will be the programming language.

Output Report Line Format

```
 PRINT
POSITIONS    FIELD NAMES          COMMENTS
---------    -----------          --------
                   DETAIL LINE
                   ***********
  1-5     FILLER            PROVIDE LEFT MARGIN
  6-9     ROOM NUMBER
 1Ø-18    FILLER            PROVIDE SPACING.
 19-38    STUDENT NAME      LAST NAME FIRST.
 39-48    FILLER            PROVIDE SPACING.
 49-51    CORRECT ANSWERS   ZERO-SUPPRESS NON-
                              SIGNIFICANT ZEROS
 52-132   FILLER            PROVIDE RIGHT MARGIN

                 ROOM TOTAL LINE
                 ***************
  1-5     FILLER            PROVIDE LEFT MARGIN
  6-9     ROOM NUMBER
 1Ø-18    FILLER            PROVIDE SPACING
 19-34    FILLER            PRINT "TOTAL SPACING:  "
 35-37    TOTAL STUDENTS    ZERO-SUPPRESS NON-SIGNIFICANT
                              ZEROS  INSERT COMMA.
 38-41    FILLER            PROVIDE SPACING
 42-47    TOTAL ANSWERS     ZERO-SUPPRESS NON-SIGNIFICANT
                              ZEROS  INSERT COMMA
 48-65    FILLER            PRINT "   AVERAGE SCORE:  "
 66-68    AVERAGE SCORE     ZERO-SUPPRESS NON-SIGNIFICANT
                              ZEROS.
  69      FILLER            PRINT "*"
 7Ø-132   FILLER            PROVIDE RIGHT MARGIN

                REPORT TOTAL LINE
                *****************
  1-19    FILLER            PROVIDE LEFT MARGIN
 2Ø-33    FILLER            PRINT "REPORT TOTAL"
 34-38    TOTAL STUDENTS    ZERO-SUPPRESS NON-SIGNIFICANT
                              ZEROS  INSERT COMMA
  39      FILLER            PROVIDE SPACE.
 4Ø-48    TOTAL CORRECT     ZERO-SUPPRESS NON-SIGNIFICANT
          ANSWERS             ZEROS. INSERT COMMA.
 49-68    FILLER            PRINT "   REPORT AVERAGE SC"
 69-73    FILLER            PRINT "ORE   "
 74-78    AVERAGE SCORE     ZERO-SUPPRESS NON-SIGNIFICANT
                              ZEROS
 79-8Ø    FILLER            PRINT "**"
 81-132   FILLER            PROVIDE RIGHT MARGIN

                REPORT HEADER LINE
                ******************
  1-5     FILLER            PROVIDE LEFT MARGIN
  6-25    FILLER            PRINT "ROOM TEST RESULTS     "
 26-27    H-MONTH           ZERO-SUPPRESS NON-SIGNIFICANT
                              ZERO
  28      FILLER            PRINT "/"
 29-3Ø    H-DAY
  31      FILLER            PRINT "/"
 32-33    H-YEAR
 34-58    FILLER            PROVIDE SPACING
 59-63    FILLER            PRINT "PAGE: ".
 64-66    H-PAGE-NBR        ZERO-SUPPRESS NON-SIGNIFICANT
                              ZEROS.
 67-132   FILLER            PROVIDE RIGHT MARGIN.

               COLUMN HEADER LINE-1
               ********************
  1-5     FILLER            PROVIDE LEFT MARGIN
  6-9     FILLER            PRINT "ROOM"
 1Ø-46    FILLER            PROVIDE SPACING
 47-53    FILLER            PRINT "CORRECT"
 54-132   FILLER            PROVIDE RIGHT MARGIN

               COLUMN HEADER LINE-2
               ********************
  1-4     FILLER            PROVIDE LEFT MARGIN
  5-1Ø    FILLER            PRINT "NUMBER"
 11-22    FILLER            PROVIDE SPACING
```

```
23-34   FILLER                PRINT "STUDENT NAME".
35-46   FILLER                PROVIDE SPACING
47-53   FILLER                PRINT "ANSWERS"
54-132  FILLER                PROVIDE RIGHT MARGIN
```

Input Record Format

```
 FIELD
POSITIONS     FIELD NAME       DATA CLASS     COMMENTS
---------     ----------       ----------     --------

  1-5         FILLER                          SKIP CODE "L5"
  6-25        STUDENT NAME     ALPHANUMERIC
 26-27        FILLER
 28-30        CORRECT ANSWERS  NUMERIC
 31-35        FILLER
 36-39        ROOM NUMBER      NUMERIC
 40-80        FILLER
```

Procedure

With the above information, you are now ready to start your Program #7 documentation. Here is what I would expect that you would want to do:

1. STRUCTURE CHART.
2. FLOWCHART.
3. PRINT CHART.
4. RECORD CHART.
5. SOURCE CODING
   - A. IDENTIFICATION DIVISION.
   - B. ENVIRONMENT DIVISION.
   - C. DATA DIVISION
     - a. FILE SECTION.

If you need any help with these, refer back to your Program #5 and Program #6. They are very similar to what you need to do for Program #7.

Program #7 Working-Storage-Section

Program #7 will be very similar to your coding for Program #6 and Program #7 for this Section. You should know how to code the following:

```
1  WS-SWITCHES   (We need to add one switch!)
   A--05  WS-FIRST-RECORD-SW                PIC X(03).
          88  FIRST-RECORD                    VALUE "YES"
2. WS-REPORT-CONTROLS
3  WS-DATE-AREA
4  WS-INPUT-RECORD.
5  DETAIL-LINE
6  ROOM AND REPORT TOTAL-LINES
B7  REPORT AND COLUMN HEADER LINES.
```

I will leave this CODING for you to do.

The Control Break will require that we add the following to define the "Control Field":

```
01  WS-CONTROL-FIELD
    05  WS-PREVIOUS-ROOM-NBR               PIC X(04)
```

We must also have an ACCUMULATOR area. I am sure you know how to set this up, but I will go over it once more as a review:

```
01  WS-TOTAL-ACCUMULATORS.
    05  WS-RM-STUDENT-COUNT-ACUM           PIC S9(03)
    05  WS-RM-CORRECT-ANWS-ACUM            PIC S9(05)
    05  WS-RPT-STUDENT-COUNT-ACUM          PIC S9(03)
    05  WS-RPT-CORRECT-ANWS-ACUM           PIC S9(05)
```

You should now code the complete WORKING-STORAGE-SECTION.

Program #7 Procedure Division

```
MODULE 000
----------

000-PRINT-TEST-REPORT
    OPEN  INPUT FILEL5
          OUTPUT TEST-RESULTS-REPORT.
    PERFORM 100-INITIALIZE-VARIABLE-FIELDS
    PERFORM 200-PROCESS-TEST-REPORT
          UNTIL END-OF-FILE
    PERFORM 700-PRINT-REPORT-TOTAL-LINE
    CLOSE FILEL5
          TEST-RESULTS-REPORT
    STOP RUN
```

Do you see anything different with this Module from what we have been doing? We have not put the 870-HEADING or the 800-READ Modules here. Thus, our STRUCTURE CHART will be very different from our previous programs. Watch how we work these into our next Modules. You will need to change your STRUCTURE CHART to match.

```
MODULE 100
----------

100-INITIALIZE-VARIABLE-FIELDS
    MOVE "NO "                     TO WS-EOF-SWITCH
    MOVE "YES"                     TO WS-FIRST-RECORD-SW
    MOVE WS-YEAR                   TO H-YEAR
    MOVE WS-MONTH                  TO H-MONTH
    MOVE WS-DAY                    TO H-DAY
    MOVE ZEROS                     TO WS-TOTAL ACCUMULATORS
```

There is one new item to do in this Module. We must initialize the WS-FIRST-RECORD-SW. If you will remember earlier, I said there were two tricky areas. This is where we start the fix for one of them.

```
MODULE 200
----------

200-PROCESS-TEST-RESULTS
    PERFORM 800-READ-TEST-RECORDS
    IF FIRST-RECORD
        MOVE TR-ROOM-NBR           TO WS-PREVIOUS-ROOM-NBR
        MOVE "NO "                 TO WS-FIRST-RECORD-SW
    IF TR-ROOM-NBR
        IS NOT EQUAL               TO WS-PREVIOUS-ROOM-NBR
            PERFORM 220-PRINT-ROOM-TOTAL-LINE
    IF NOT END-OF-FILE
        PERFORM 210-PRINT-DETAIL-LINE.
```

Now we are getting into the logic of the control break programming. Did you see where we added the 800-READ? Do you see the 88-Conditions working? Note the "NOT EQUAL" and the "NOT" expressions. Do you see how they effect the logic? The three IF statements are the "MEAT" of the Module. Did you see the Second IF providing the Control Break? Be sure to study this Module until you can follow what is happening.

```
MODULE 210
----------

210-PRINT-DETAIL-LINE
    IF FIRST-PAGE OR FULL-PAGE
        PERFORM 870-PRINT-REPORT-HEADINGS.
    MOVE TR-ROOM-NBR           TO DL-ROOM-NBR
    MOVE TR-STUDENT-NAME       TO DL-STUDENT-NAME
    MOVE TR-CORRECT-ANSWERS    TO DL-CORRECT-ANSWERS
    MOVE DL-DETAIL-LINE        TO RESULTS-REPORT-LINE
    PERFORM 890-PRINT-REPORT-LINE
    ADD 1                      TO WS-RM-STUDENT-COUNT-ACUM
    ADD 1                      TO WS-RPT-STUDENT-COUNT-ACUM
    ADD TR-CORRECT-ANSWERS     TO WS-RM-CORRECT-ANWS-ACUM
    ADD TR-CORRECT-ANSWERS     TO WS-RPT-CORRECT-ANWS-ACUM
    MOVE 1                     TO WS-LINE-SPACING
```

This Module uses all items that you have handled in other programs. The main difference is that they are rearranged. I will let you analyze the Module. It is not hard!

```
MODULE 220
----------

220-PRINT-ROOM-TOTAL-LINE
    MOVE WS-PREVIOUS-ROOM-NBR       TO ST-ROOM-NBR
    MOVE WS-RM-STUDENT-COUNT-ACUM   TO ST-TOT-STUDENTS
    MOVE WS-RM-CORRECT-ANWS-ACUM    TO ST-TOT-ANSWERS
    DIVIDE WS-RM-CORRECT-ANWS-ACUM
        BY WS-RM-STUDENT-COUNT-ACUM
            GIVING ST-AVG-SCORE ROUNDED.
    MOVE ST-ROOM-TOTAL-LINE         TO RESULTS-REPORT-LINE.
    MOVE 2                          TO WS-LINE-SPACING
    MOVE 3                          TO WS-LINE-SPACING
    MOVE ZEROS                      TO WS-RM-STUDENT-COUNT-ACUM
    MOVE ZEROS                      TO WS-RM-CORRECT-ANWS-ACUM.
    MOVE TR-ROOM-NBR                TO WS-PREVIOUS-ROOM-NBR.
```

Again, this Module has about the same operations that you have done in previous programs, but they are rearranged. Did you see how the line spacing was handled? The most important item to look at is the last MOVE — TR-ROOM-NBR TO WS-PREVIOUS-ROOM-NBR. This prepares the programming logic for the NEXT control break. Do you follow this? Understanding the program logic depends on your study of this MOVE.

```
MODULE 700
----------

700-PRINT-REPORT-TOTAL-LINE
    MOVE WS-RPT-STUDENT-COUNT-ACUM    TO RT-TOT-STUDENTS
    MOVE WS-RPT-CORRECT-ANWS-ACUM     TO RT-TOT-ANSWERS
    DIVIDE WS-RPT-CORRECT-ANWS-ACUM
        BY WS-RPT-STUDENT-COUNT-ACUM
            GIVING RT-AVG-SCORE ROUNDED.
    MORE RT-REPORT-TOTAL-LINE         TO RESULTS-REPORT-LINE.
    PERFORM 890-PRINT-REPORT-LINE

MODULE 800
----------

800-READ-TEST-RECORDS.
    READ FILEL5
        INTO TR-REST-RESULTS-RECORD
            AT END
                MOVE "YES"            TO WS-EOF-SWITCH
                MOVE HIGH-VALUES      TO TR-ROOM-NBR
```

This Module is nearly the same as the one we used in Program #6. The last MOVE — HIGH-VALUES TO TR-ROOM-NBR is the KEY change! If you look up "high-value" in your manual, you will find that it is a "figurative constant". Do you remember these? We discussed them some articles back. This might be a good time for a review. Anyway, "high-value" is a character whose "octal" representation is 177. How many of you readers know what that means? How many are willing to try to find out what it means? High-value is used to set a field to the highest possible value. This set value will be used to force out the Report Total-Line. Remember the tricks we discussed earlier? I am going to let you study this logic.

```
MODULE 870
----------

870-PRINT-REPORT-HEADINGS
    MOVE SPACES                     TO RESULTS-REPORT-LINE
    PERFORM 880-PRINT-REPORT-TOP-LINE.
    MOVE WS-PAGE-COUNT              TO H-PAGE-NBR
    ADD 1                           TO WS-PAGE-COUNT.
    MOVE REPORT-HEADING-LINE        TO RESULTS-REPORT-LINE.
    MOVE 6                          TO WS-LINE-SPACING
    PERFORM 890-PRINT-REPORT-LINE
    MOVE COLUMN-HEADER-LINE-1       TO RESULTS-REPORT-LINE.
    MOVE 3                          TO WS-LINE-SPACING
    PERFORM 890-PRINT-REPORT-LINE
    MOVE COLUMN-HEADER-LINE-2       TO RESULTS-REPORT-LINE
    MOVE 1                          TO WS-LINE-SPACING.
    PERFORM 890-PRINT-REPORT-LINE
    MOVE 2                          TO WS-LINE-SPACING
```

There is nothing really new in this Module except for the line spacing and the way we use the 890-PRINT Module to print each report line. You must remember that you will NOT be able to follow this program logic without preparing a STRUCTURE CHART! Have you done this? If not, do it NOW! I am not going to go into any long explanations. It is time for me to make YOU think!

```
MODULE 880
----------

880-PRINT-REPORT-TOP-LINE.
    WRITE RESULTS-REPORT-LINE
        AFTER ADVANCING PAGE
    MOVE ZEROS                          TO WS-LINES-USED.

MODULE 890
----------

890-PRINT-REPORT-LINE.
    DISPLAY RESULTS-REPORT-LINE
    WRITE RESULTS-REPORT-LINE
        AFTER ADVANCING WS-LINE-SPACING.
    ADD WS-LINE-SPACING                 TO WS-LINES-USED
```

You know this Module from previous programs. Just note how to handle the line spacing!

Closing

I have completely coded the PROCEDURE DIVISION for you. However, I have not completely explained all of the coding! I want you to use all of your documentation along with the coding to study the LOGIC of Program #7. I am going to expect more study from you in this and with future programs! Did you find the way we solved the two "tricks"? Your extra effort will help you with the next, more complicated program.

Now, KEY-IN the code and do your "walk-through". COMPILE and LINK/EXECUTE the program. Do not expect to get it right the first time! It will require some extra thought. You will learn from the experience! Be sure to have it worked out by next month. Remember, if you cannot find all of your problems, you can go to your HUG COBOL Corner Disk II for the PRGM07 file. Compile and Link/Execute this and compare it to your Program #7.

We will start the "multiple-level control break program" in the next "COBOL Corner". It will not be easier. If you have done your work well on Program #7, it will make Program #8 seem easy. It now depends on your effort!

"COBOL Corner" NOTE: We have completed the "HUG COBOL Corner Disk I" programs. It is time to order the "HUG COBOL Corner Disk II" for COBOL-80 or "II(Z)" for COBOL-86. You will find that this disk will be more valuable to you for the advanced programming we are now getting into! Please order your disk from HUG!

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.

HDOS HARDCOPY SOFTWARE

Part Number	Description of Product	Selling Price	Vol. Issue
885-1008	Volume I Documentation	9.00	
885-1013	Volume II Documentation	12.00	
885-1015	Volume III Documentation	9.00	
885-1037	Volume IV Documentation	12.00	8
885-1058	Volume V Documentation	12.00	

MISCELLANEOUS HDOS COLLECTIONS

Part Number	Description of Product	Selling Price	Vol. Issue
885-1032	Disk V H8/H9	18.00	8
885-1044-[37]	Disk VI H8/H89	18.00	
885-1064-[37]	Disk IX H8/H89 Disk	18.00	
885-1066-[37]	Disk X H8/H89	18.00	10
885-1069	Disk XIII Misc H8/H89	18.00	

GAMES

HDOS

Part Number	Description of Product	Selling Price	Vol. Issue
885-1010	Adventure Disk H8/H89	10.00	4
885-1029-[37]	Disk II Games 1 H8/H89	18.00	8
885-1030-[37]	Disk III Games 2 H8/H89	18.00	8
885-1031	Disk IV MUSIC H8 Only	20.00	25
885-1067-[37]	Disk XI H8/H19/H89 Games	18.00	12
885-1068	Disk XII MBASIC Graphic Games	18.00	10
885-1088-[37]	Disk XVII MBASIC Graph. Games	20.00	14
885-1093-[37]	D&D H8/H89 Disk	20.00	16
885-1096-[37]	MBASIC Action Games H8/H89	20.00	18
885-1103	Sea Battle HDOS H19/H8/H89	20.00	20
885-1111-[37]	HDOS MBASIC Games H8/H89	20.00	23
885-1112-[37]	HDOS Graphic Games H8/H89	20.00	23
885-1113-[37]	HDOS Action Games H8/H89	20.00	23
885-1114	H8 Color Raiders & Goop	20.00	23
885-1124	HUGMAN & Movie Animation Pkg	20.00	41
885-1125	MAZEMADNESS	20.00	41
885-1130	Star Battle	20.00	47
885-1133-[37]	HDOS Games Collection I	20.00	59
885-8009-[37]	HDOS & CP/M Galactic Warrior	20.00	32
885-8022	HDOS SHAPES	16.00	45
885-8026	HDOS Space Drop	16.00	49
885-8032-[37]	HDOS Castle	20.00	59

CP/M

Part Number	Description of Product	Selling Price	Vol. Issue
885-1206-[37]	CP/M Games Disk	20.00	11
885-1209-[37]	CP/M MBASIC D&D	20.00	19
885-1211-[37]	CP/M Sea Battle	20.00	20
885-1220-[37]	CP/M Action Games	20.00	32
885-1222-[37]	CP/M Adventure	10.00	35
885-1227-[37]	CP/M Casino Games	20.00	38
885-1228-[37]	CP/M Fast Action Games	20.00	39
885-1236-[37]	CP/M Fun Disk I	20.00	55

ZDOS

Part Number	Description of Product	Selling Price	Vol. Issue
885-3004-37	ZDOS ZBASIC Graphic Games	20.00	37
885-3009-37	ZDOS ZBASIC D&D	20.00	50
885-3011-37	ZDOS ZBASIC Games Disk	20.00	52
885-3017-37	ZDOS Contest Games Disk	25.00	58

UTILITIES

HDOS

Part Number	Description of Product	Selling Price	Vol. Issue
885-1022-[37]	HUG Editor (ED) Disk H8/H89	20.00	20
885-1025	Runoff Disk H8/H89	35.00	
885-1060-[37]	Disk VII H8/H89	18.00	
885-1061	TMI Load H8 ONLY Disk	18.00	
885-1062-[37]	Disk VIII H8/H89 (2 Disks)	25.00	
885-1063	Floating Point Disk H8/H89	18.00	
885-1065	Fix Point Package H8/H89 Disk	18.00	10
885-1075	HDOS Support Package H8/H89	60.00	
885-1077	TXTCON/BASCON H8/H89	18.00	
885-1079-[37]	HDOS Page Editor	25.00	15
885-1080	EDITX H8/H19/H89 Disk	20.00	
885-1082	Programs for Printers H8/H89	20.00	
885-1083-[37]	Disk XVI Misc H8/H89	20.00	11
885-1089-[37]	Disk XVIII Misc H8/H89	20.00	20
885-1090-[37]	Disk XIX Utilities H8/H89	20.00	22
885-1092-[37]	Relocating Debug Tool H8/H89	30.00	14
885-1098	H8 Color Graphics ASM	20.00	19
885-1099	H8 Color Graphics Tiny PASCAL	20.00	19
885-1105	HDOS Device Drivers H8/H89	20.00	24
885-1116	HDOS Z80 Debugging Tool	20.00	27
885-1119-[37]	BHBASIC Support	20.00	29
885-1120-[37]	HDOS 'WHEW' Utilities	20.00	33
885-1121	HDOS Hard Sec Sup Pkg 2 Disks	30.00	37
885-1123	XMET Robot Cross Assembler	20.00	40
885-1126	HDOS Utilities by PS:	20.00	42
885-1127-[37]	HDOS Soft Sector Support Pkg	30.00	45
885-1128-[37]	HDOS DISKVIEW	16.00	46
885-1129-[37]	HDOS CVT Color Video Terminal	20.00	46
885-8001	SE (Screen Editor)	25.00	28
885-8003	BHTOMB	25.00	28
885-8004	UDUMP	35.00	28
885-8006	HDOS SUBMIT	20.00	31
885-8007	EZITRANS.	30.00	30
885-8015	HDOS TEXTSET Formatter	30.00	42
885-8017	HDOS Programmers Helper	16.00	42
885-8024	HDOS BHBASIC Utilities Disk	16.00	46

CP/M

Part Number	Description of Product	Selling Price	Vol. Issue
885-1210-[37]	CP/M ED (same as 885-1022)	20.00	20
885-1212-[37]	CP/M Utilities H8/H89	20.00	21
885-1213-[37]	CP/M Disk Utilities H8/H89	20.00	22
885-1217-[37]	HUG Disk Duplication Utilities	20.00	26
885-1223-[37]	HRUN HDOS Emulator 3 Disks	40.00	37
885-1225-[37]	CP/M Disk Dump & Edit Utility	30.00	40
885-1226-[37]	CP/M Utilities by PS:	20.00	40
885-1229-[37]	XMET Robot Cross Assembler	20.00	40
885-1230-[37]	CP/M Function Key Mapper	20.00	42
885-1231-[37]	Cross Ref Utilities for MBASIC	20.00	43
885-1232-[37]	CP/M Color Video Terminal	20.00	46
885-1235-37	CP/M COPYDOS	20.00	54
885-1237-[37]	CP/M Utilities	20.00	55
885-1245-37	CP/M-85 KEYMAP	20.00	63
885-1246[-37]	CP/M HUG File Manager & Utilities	20.00	64
885-1247-37	CP/M-85 HUG Bkgrd Print Spooler	20.00	67
885-5001-37	CP/M-86 KEYMAP	20.00	51
885-5002-37	CP/M-86 HUG Editor	20.00	52
885-5003-37	CP/M-86 Utilities by PS:	20.00	54
885-5008-37	CP/M 8080 To 8088 Trans. & HFM	20.00	64
885-5009-37	CP/M-86 HUG Bkgrd Print Spool	20.00	66
885-8018-[37]	CP/M Fast Eddy & Big Eddy	20.00	43
885-8019-[37]	DOCUMAT and DOCULIST	20.00	43
885-8025-37	CP/M-85/86 Fast Eddy	20.00	49

ZDOS

Part Number	Description of Product	Selling Price	Vol. Issue
885-3005-37	ZDOS Etchdump	20.00	39
885-3007-37	ZDOS CP/EMulator	20.00	47
885-3008-37	ZDOS Utilities	20.00	47
885-3010-37	ZDOS Keymap	20.00	51
885-3022-37	ZDOS/MSDOS Useful Programs I	30.00	63
885-3023-37	ZDOS/MSDOS EZPLOT	20.00	63
885-3026-37	MSDOS SMALL C Compiler	25.00	65
885-8029-37	ZDOS Fast Eddy	20.00	53

H/Z100 ZDOS/MSDOS - H/Z150 PC MSDOS

Part Number	Description of Product	Selling Price	Vol. Issue
885-3012-37§§	ZDOS HUG Editor	20.00	52
885-3014-37§§	ZDOS/MSDOS Utilities II	20.00	54
885-3016-37§	ZDOS/MSDOS Adventure	10.00	57
885-3020-37§	MSDOS HUG Menu System	20.00	62
885-3021-37§§	ZDOS/MSDOS Cardcat	20.00	63
885-3024-37§	ZDOS/MSDOS 8080 To 8088 Trans	20.00	64
885-3025-37§§	ZDOS/MSDOS Misc. Utilities	20.00	64
885-3029-37§§	ZDOS/MSDOS HUG Bg. Print Spool	20.00	66

§ All program files run on both
§§ Program files run partially on both

PC/IBM COMPATIBLE

Part Number	Description of Product	Selling Price	Vol. Issue
885-6001-37	MSDOS Keymapper	20.00	59
885-6002-37	CP/EMulator II & ZEMulator	20.00	59
885-6003-37	MSDOS EZPLOT	20.00	65
885-6004-37	MSDOS CheapCalc	20.00	67
885-6005-37	MSDOS Skyviews	20.00	67
885-8033-37	MSDOS Fast Edit	20.00	62

PROGRAMMING LANGUAGES

HDOS

Part Number	Description of Product	Selling Price	Vol. Issue
885-1038-[37]	Wise on Disk H8/H89	18.00	
885-1042-[37]	PILOT on Disk H8/H89	19.00	
885-1059	FOCAL-8 H8/H89 Disk	25.00	13
885-1078-[37]	HDOS Z80 Assembler	25.00	21
885-1085	PILOT Documentation	9.00	
885-1086-[37]	Tiny HDOS PASCAL H8/H89	20.00	13
885-1094	HDOS Fig-Forth H8/H89	40.00	18
885-1132-[37]	HDOS Tiny BASIC Compiler	20.00	59
885-1134	HDOS SMALL-C Compiler	30.00	63

CP/M

Part Number	Description of Product	Selling Price	Vol. Issue
885-1208-[37]	CP/M Fig-Forth H8/H89 2 Disks	40.00	18
885-1215-[37]	CP/M BASIC-E	20.00	26

BUSINESS, FINANCE AND EDUCATION

HDOS

Part Number	Description of Product	Selling Price	Vol. Issue
885-1047	Stocks H8/H89 Disk	18.00	
885-1048	Personal Account H8/H89 Disk	18.00	
885-1049	Income Tax Records H8/H89 Disk	18.00	
885-1055-[37]	MBASIC Inventory Disk H8/H89	30.00	
885-1056	MBASIC Mail List	30.00	
885-1070	Disk XIV Home Fin H8/H89	18.00	
885-1071-[37]	MBASIC SmBusPk H8/H19/H89	75.00	17
885-1091-[37]	Grade/Score Keeping H8/H89	30.00	14
885-1097-[37]	MBASIC Quiz Disk H8/H89	20.00	18
885-1118-[37]	MBASIC Payroll	60.00	30
885-1131-[37]	HDOS CheapCalc	20.00	47
885-8010	HDOS Checkoff	25.00	32
885-8021	HDOS Student's Statistics Pkg	20.00	44
885-8027	HDOS SciCalc	20.00	50

CP/M

Part Number	Description of Product	Selling Price	Vol. Issue
885-1218-[37]	CP/M MBASIC Payroll	60.00	31
885-1233-[37]	CP/M CheapCalc	20.00	47
885-1239-[37]	Spread Sht. Contest Disk I	20.00	
885-1240-[37]	Spread Sht. Contest Disk II	20.00	
885-1241-[37]	Spread Sht. Contest Disk III	20.00	
885-1242-[37]	Spread Sht. Contest Disk IV	20.00	
885-1243-[37]	Spread Sht. Contest Disk V	20.00	
885-1244-[37]	Spread Sht. Contest Disk VI	20.00	
885-8011-[37]	CP/M Checkoff	25.00	32

ZDOS

Part Number	Description of Product	Selling Price	Vol. Issue
885-3006-37	ZDOS CheapCalc	20.00	47
885-3013-37	ZDOS Checkbook Manager	20.00	54
885-3018-37	ZDOS Contest Spreadsheet Disk	25.00	58
885-8028-37	ZDOS SciCalc	20.00	50
885-8030-37	ZDOS Mathflash	20.00	55

Continued on Page 67☛

HUG NEW PRODUCTS

885-3030-37 MS-DOS/Z-DOS Z-100 PC Emulator (ZPC) $40.00

Introduction: ZPC is a program that emulates an IBM PC or compatible computer (such as the H/Z-150) on an H/Z-100 series (dual processor) computer. It represents the ultimate software solution to the problem of IBM PC compatibility on the Z-100. ZPC emulates the keyboard, printer driver, and the color/graphics adapter of an IBM PC or similar computer. It supports all video modes of the color/graphics adapter, including 40 or 80 column text modes, and the medium resolution color and high resolution monochrome graphics modes. If your Z-100 has 768k of RAM, it can even run programs that write directly to PC video memory, and is amazingly efficient at running such programs. For example, WordStar version 3.3 for the Z-150 runs faster on a Z-100 under ZPC than the Z-100's own version of WordStar 3.3. The only way you can make your H/Z-100 more IBM compatible than ZPC does is to purchase and install an expensive adapter board. ZPC requires NO hardware modifications.

Requirements: ZPC requires an H/Z-100 series computer with at least 192k of RAM and the Z-DOS or MS-DOS operating system. For maximum IBM compatibility, your system should have 768k of RAM and use MS-DOS version 2 or higher. It is also desirable that your system be color capable, with either 32k or 64k color RAM chips.

This disk contains the following files:

README	.DOC	WSPCH	.BAT	ZPC	.ASM
ZPC1	.COM	WSPCH	.DAT	COND	.ACM
ZPC2	.COM	SC3PCH	.BAT	DOS	.ACM
ZPC3	.COM	SC3PCH	.DAT	KEY	.ACM
PC	.COM	MPPCH	.BAT	PIXEL	.ACM
Z100	.COM	MPPCH	.DAT	PUTCHAR	.ACM
				SCROLL	.ACM

ZPC also includes printed documentation.

Program Author: Patrick Swayne, HUG

ZPC1.COM, ZPC2.COM, ZPC3.COM — These are three versions of ZPC, that support different levels of IBM compatibility, depending on your system. ZPC1 is for systems with less than 768k of RAM and the Z-DOS operating system. This version will run any IBM program that limits itself to performing I/O via the BIOS. ZPC2 is for systems with less than 768k of RAM and the MS-DOS operating system, version 2 or higher. In addition to supporting the BIOS I/O calls, it also supports the data area at the bottom of the BIOS RAM segment. This data area contains such information as the cursor position, the video mode, and a timer counter. Some IBM programs access this area rather than using BIOS calls to obtain the cursor position, etc. ZPC3 is for systems with 768k of RAM and MS-DOS version 2 or higher. It supports all features of ZPC1 and ZPC2, and in addition, it can run programs that write directly to the memory of the color/graphics adapter to produce screen displays.

ZPC is a background program that remains resident in memory after you run it. It can be turned on to run IBM programs, or turned off to run Z-100 programs, once it has been loaded. The PC.COM and Z100.COM programs, explained below, are used to turn it on or off.

Although ZPC is not as IBM compatible as an add-on hardware modification would be, it offers some advantages over such modifications, including:

1. You only need one operating system, and you can run both Z-100 and IBM programs from the same disk.
2. Assembly programmers can access the Z-100 monitor ROM and video RAM while ZPC is in the PC mode. It is therefore possible to have, for example, both 40 column and 80 column text on the screen at the same time.
3. ZPC is much less expensive than a hardware adapter!

Note: ZPC is still under development at the time of this writing. We are not sure how many IBM or Z-150 programs it will be able to eventually run, but at this time, we have been able to run Turbo Pascal (the IBM version, including graphics) under ZPC2 or ZPC3, and ZPC3 can run WordStar 3.3 (Z-150 version), SuperCalc-3 (including graphs), and Multiplan (Z-150 version, 40 or 80 column mode). Each of the last 3 programs write directly to video memory.

PC.COM — This program is used to turn on the IBM emulation mode after ZPC is loaded into memory. With PC.COM on your system disk, you just enter

```
A>PC
```

to turn IBM emulation mode on. The cursor will change to the double line type normally used on an IBM PC to indicate that PC mode is on, and the character font will change to the IBM PC style. You can also use PC to set a specific video mode by entering the number of that mode in the command line. For example, to set the 40 column color text mode (mode no. 1), you would enter

```
A>PC 1
```

The default mode, when you do not enter a mode number, is the

80 column color text mode (mode 3).

Z100.COM — This program is used to turn off the IBM emulation mode after ZPC is loaded. If you have been running a PC program, and you want to run a Z-100 program, you can just enter

```
A>Z100
```

to turn off the IBM emulation mode. The normal Z-100 cursor will be restored, and the text font will revert to what it was before the PC mode was set.

WSPCH.BAT, WSPCH.DAT, SC3PCH.BAT, SC3PCH.DAT, MPPCH.BAT, MPPCH.DAT — Programs that write directly to video RAM on an IBM PC or compatible computer usually write to or read from ports to check the status of video memory, etc., and these ports conflict with the ports on a Z-100. The above files are batch and data files that patch out the ports in WordStar, SuperCalc-3, and Multiplan. ZPC does not require that the ports be accessed in order for the programs to run, so the sections of code that write to or read from ports are simply patched out with NOP (no operation) instructions. The patch files are easy to use. For example, to patch WordStar, you would copy WSPCH.BAT, WSPCH.DAT, and DEBUG.COM (from your system disk) to your WordStar disk, log on to that disk, and enter

```
A>WSPCH
```

The required patches would then be installed automatically. This only needs to be done once, and from then on, the program can be run under ZPC as easily as it is run on a real IBM PC or compatible. **Note:** There will probably be patch files for other programs besides these three on the ZPC disk when it is released (this description was written more than a month before the release date), and as other patches are developed, they will be printed in REMark.

ZPC.ASM — This is the source code for ZPC.

COND.ACM, DOS.ACM, KEY.ACM, PIXEL.ACM, PUTCHR .ACM, SCROLL.ACM — These are INCLUDE files that contain parts of the source code for ZPC.

TABLE C Rating: (2), (7), (10)

HUG P/N 885-3027-37 and 885-3028-37 HUGPBBS Update

A minor bug has surfaced in MSDOS HUGPBBS Version 1.00.M. The problem occurs when editors like BSE and EDLIN are used to create the User Log file. Some names and passwords in that file may not get recognized by the system. If the HUG Editor (supplied with the software) is used, the problem does not happen. A new version (1.01.M) is being made available to original owners at no extra charge. This new version corrects this bug, as well as contains three small enhancements: character echoing is suppressed upon entering the user password, improper log-ons are now flagged at the users' name on the printer, and the SYSOP can now notify a user that he would like to talk to him using the 'T' command. To receive your update, return your original disk to Nancy Strunk here at the Heath Users' Group, Hilltop Road, St. Joseph, MI 49085, and it will be updated and returned to you free of charge. This update affects both, the executable file and source code.

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

TABLE C
Product Rating

10 - Very Good
9 - Good
8 - Average

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

7 - Has hardware limitations (memory, disk storage, etc.)
6 - Requires special programming technique
5 - Requires additional or special hardware
4 - Requires a printer
3 - Uses the Special Function Keys (f1,f2,f3,etc.)
2 - Program runs in *Real Time**
1 - Single-keystroke input
0 - Uses the H19 (H/Z89) escape codes (graphics, reverse video)

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

ORDERING INFORMATION

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

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.

Transfer "Foreign" Files in Seconds

NOW AVAILABLE FOR Z-100 AND Z-150

No modems and RS-232 links required!

ONLY $39.95!

READ, WRITE and FORMAT more than 70 different CP/M and MS-DOS disks on your computer

with MEDIA MASTER™

Tired of hunting for software in your computer's disk format? Do you need to exchange your Lotus 1-2-3, dBase II, Multiplan, and Wordstar data files among different CP/M and MS-DOS computers?

Then MEDIA MASTER is for you!

MEDIA MASTER supports the following disk formats:

Actrix (SSDD)
Actrix (DSDD)
Avatar TC10 (DSDD)
Casio FP 1000 (DSDD)
Chameleon CP/M-80
Columbia MPC CP/M-80
Cromemco CDOS (SSDD)
Cromemco w/Int'l Term (DSDD)
Cromemco w/Int'l Term (SSDD)
DEC VT180 (SSDD)
Davidge (DSDD)
Digilog (DSDD)
Epson Multifont (DSDD)
Epson QX-10 (DSDD)
Fujitsu Micro 16s (DSDD)
Groupil III CP/M (DSDD)
H/Z Z-100 CP/M (DSDD) - later
H/Z Z-100 CP/M (SSDD)
H/Z Z-100 Z-DOS 1.xx (SSDD)
H/Z Z-100 Z-DOS 1.xx (DSDD)
H/Z Z-100 Z-DOS 2.xx (SSDD)
H/Z Z-100 Z-DOS 2.xx (DSDD)
H/Z Z-90 40 trk, 1 k blk (SSDD)
H/Z Z-90 40 trk, 2 k blk (SSDD)
Heath w/Magnolia CP/M (SSDD)
IBM PC CP/M-86 (DSDD)
IBM PC CP/M-86 (SSDD)
IBM PC-DOS 1.xx (DSDD)
IBM PC-DOS 1.xx (SSDD)
IBM PC-DOS 2.xx (DSDD)
IBM PC-DOS 2.xx (SSDD)
IDEA Bitelex (SSDD)
ISM CP/M (DSDD)
Insight Dev. IQ-120 (SSDD)
Kaypro II/2 (SSDD)
*Kaypro 4, 10 (DSDD)
LNW-80 (SSDD)
Lobo Max-80 (SSDD)
Lobo Max-80 512 (SSDD)
Micral 9050 CP/M-80 (DSDD)
Morrow MD 11 (DSDD)
Morrow MD 2 (SSDD)
Morrow MD 3 (DSDD)
NCR Decision Mate 5 (DSDD)
NEC PC-8001A (SSDD)
NEC PC-8001A (DSDD)
Olympia ETX II (SSDD)
Olympia EX100 (DSDD)
Osborne (SSDD)
Osborne 4 (DSDD)
Osborne Osmosis (SSDD)
Otrona (DSDD)
PMC Micromate (DSDD)
Reynolds & Reynolds (SSDD)
Sanyo (DSDD)
Superbrain (DSDD)
Superbrain Jr. (SSDD)
Systel II (SSDD)
Systel III (DSDD)
TI Professional CP/M-86 (DSDD)
TRS-80 III FEC CP/M (SSDD)
TRS-80 III FEC T805 (SSDD)
TRS-80 III Hurr. Labs (SSDD)
TRS-80 III Mem. Merch (SSDD)
TRS-80 IV CP/M+ (SSDD)
TRS-80 IV Mont. Micro (SSDD)
Teletek 40 trk (SSDD)
Toshiba T100 (DSDD)
TurboDos (DSDD)
Wang Maws CP/M (DSDD)
Xerox 820 II (SSDD)
*Zorba 40 trk (DSDD)

**Requires access to "foreign" computer*

***MEDIA MASTER** is also available for DEC Rainbow, Osborne, and Kaypro computers.*

NEW! — FILE MATE™

A Dozen Utilities in One!

ONLY $29.95

Finally — a powerful, easy to use file utility that performs a host of functions. **FILE MATE's** numerous text sifting and paragraph reformatting modes help you exchange "foreign" text files. Other file inspection and repair tools include two binary dumps, an ASCII/HEX file compare, and CRC "fingerprint." FILE MATE is available for popular CP/M and MS-DOS computers.

INTERSECTING CONCEPTS
(formerly MDG & Associates)

4573 Heatherglen Ct., Moorpark, CA 93021

ORDERING INFORMATION:
Include $3 per order for postage/handling. Overseas airmail add $10. To place COD orders:

Call TOLL FREE 24 hours:
800-824-7888 (Ask for Operator 251)
Alaska, Hawaii: 800-824-7919 (Ask for Operator 251)
Technical questions: call 805-529-5073

Controlled Data Recording Systems Inc.

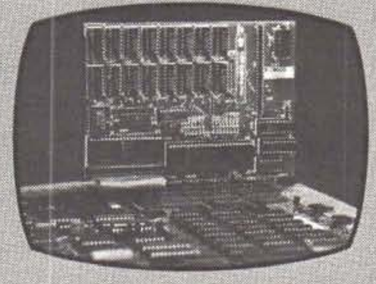
Quality Products and Support for the Heath/Zenith Community

New for the H/Z89-90 Computer Users:
SUPER RAM 89

Heath/Zenith 89 and 90 computer users can now get the speed and power of a high capacity Ram Drive System at a reasonable price.

Using the Super Ram 89 package, standard software shows an immense improvement in speed. Depending on the software being run, programs may execute 10 times faster than when run through standard floppies.

Super Ram 89 comes as a two card set that plugs into the left hand (16k expansion) side of the computer. No computer modifications required.



Board 1 has two banks of 256k chips possible for a total of 512k. Either or both banks are able to use 64k chips instead. Board 1 can be used by itself, with board 2 added at a later date.

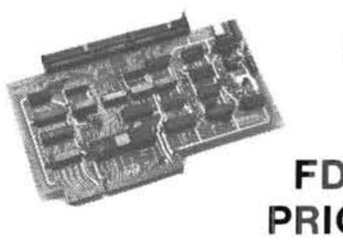
Board 2 has an additional 512k, plus it has a real time clock capability, and a SASI interface hardware capability. Board 2 piggybacks onto board 1.

The Ram Drive Software (SRAM) allows one or two logical ram drives. The ram drive(s) can be located starting anywhere from logical A: to O: (standard drives get relocated). SRAM can be set to start at logical A: and warm boot with ram (no floppy disk accesses needed). Ram drive attatches to any of the versions of CP/M 2.2 bios used in the H/Z89-90.

Super Ram 89 Pricing:
Board 1 includes hardware manual and ram drive software with no ram: $190.00. Each 256k bank, add $90.00. Board 2 (must have board 1). With no ram, no clock, no SASI $90.00. Each 256k bank, add $90.00. Ask about clock, SASI pricing.

FOR THE H/Z89-90 COMPUTERS

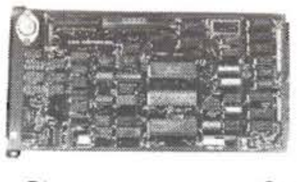
THE ONE CONTROLLER FOR 8" & 5.25" DRIVES THE FDC-880H PRICE: $395



Includes controller board CP/M boot prom, I/O decoder prom, hardware/software manuals BIOS source listing. HDOS driver now available for $50.00.

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. The FDC-880H operates with or without the Heath hard sectored controller.

THE FDC-H8 DOUBLE DENSITY 8" AND 5.25" CONTROLLER FOR THE H8 COMPUTER PRICE $495



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

- 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.

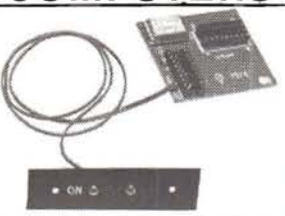
MODIFY89 Allows owners of the FDC-880H or the FDC-H8 to read/write to the following disktypes:

Access S.S.D.D., Cromemco S.S.D.D., DEC VT180 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-8001 A S.S.D.D., Osborne S.S.D.D., Osborne S.S.D.D., Otrona D.S.D.D.*, Zorba D.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.D.D., Zerox 820-II S.S.D.D. Plus more, Standard (Tests for H/Z37 and C.D.R. Disk types)

PRICE $49.95

FOR THE Z100 COMPUTERS

THE ORIGINAL Z100 SPEED MODULE. RUN YOUR Z100 PROGRAMS FASTER



The ZS100 runs the Z100 CPU 50% faster, (7.5 MHZ) in 8088 mode. The ZS100 installs easily with no soldering. The ZS100 is externally switchable between speed mode and normal. The ZS100 improves the time performance of applications packages with no software modifications needed. The ZS100 is for all Heath/Zenith 100, 110 and 120 series computers.

New: a powerful floppy drive package for any computer that can use 8" floppy drives.

Two extended technology 5.25" drives in a package with interface cables case and power supply, with the capability of looking exactly like two standard 8" drives to the computer.

Now have **2.4 megabytes** of **5.25" floppy file storage capacity** in one package. A switch enables the drives to look like 8" drives or 5.25" 96 TPI drives.

ASK FOR PRICE.

For information about these products contact:
Controlled Data Recording Systems Inc.
7210 Clairemont Mesa Blvd.
San Diego, Ca. 92111 Phone (619) 560-1272 or contact a Heath/Zenith dealer that carries the C.D.R. Systems Inc. line of products.

Your Own Personal MS–DOS Amanuensis

Vincent Alfieri, Ph.D.
4118 Los Feliz Boulevard
Los Angeles, CA 90027

About The Author

Vincent Alfieri, Ph.D. *is a software specialist and the very happy owner of a Zenith Z-160. He is the author of Mastering WordStar (Hayden) and a forthcoming book on Display-Write 2.*

Until the time when you can actually speak to your PC to tell it what you want it to do, you are stuck with having to use the keyboard to input commands. (Of course, if you purchase a mouse, this will help you cut down on a lot of keystroking, but it still takes time to move that critter around, and many programs don't work with a mouse yet.) And if you hate to type as much as I do, you undoubtedly hate having to type repetitious commands.

Take heart. MS–DOS, the disk operating system of your Z–150 or Z–160 PC, provides you with built–in features that can make life easier (and less boring) by doing much of the keystroking for you. Using these features is almost like having the services of a personal amanuensis, ready at your beck and call to do most of the work. And your personal amanuensis won't ever ask for a raise, needs no desk space, and doesn't like cheese. This short tutorial illustrates just a few ways to take advantage of DOS's keystroke–saving power.

The Frequently Underused Function Keys

Many application programs take advantage of the PC's function keys as keystroke–savers. But what users often overlook is that the <F1> , <F2>, <F3> and <F4> keys on the function keypad can help you save a lot of keystrokes when typing in DOS commands, too.

The <F3> key is by far the most useful. It allows you to repeat the previous command without having to retype it! Thus, if you are looking at the directories of many diskettes, you merely type dir b: once, and then use <F3> (followed by <RETURN>) for every other diskette. If you make a mistake at the very end of the line, why retype the entire command? Merely hit <F3>, backspace over the mistake, make your correction, and re–enter the command with <RETURN>.

Once you type in another command yourself, however, DOS "remembers" it instead. It always keeps track of the last issued command. It can do this because it retains the command line in a buffer area which it calls the template.

You can also use the <F1> and <F2> keys to edit the previous command template. The former key is helpful when you make a mistake at the beginning of the command line. For example, if you typed format b:/s/v — a lot of keys indeed! — DOS would of course give you an error message. Instead of retyping the entire command, you could hit <F1> once to get to the second letter in the command line. Note that DOS shows you the "f" but not the "u." Type in the "i" and then hit <F3> to show the rest of the command. Press <RETURN> to enter the command.

If you have made a mistake at the other end of the command line, use the <F2> key. For instance, the command copy myfile.txt b;/v (a long one indeed) is also wrong. Instead of hitting <F1> many times to get to the incorrect semicolon, you could hit <F2>, followed by typing the slash. This key instructs DOS to move the cursor in front of that spot on the command line (in other words, to the semicolon). Then type the correction, hit <F3> to complete the command, and then <RETURN> to re–enter the command. These three keys are great for lazy people like me who hate to retype anything if they can help it.

Finally, the <F4> function key will erase part of a command line up to the character that you indicate. For instance, if you typed a word twice by mistake, you can quickly delete it. To correct the command line dir dir b: press <F4> and then "r." DOS will delete the first three characters but leave the rest of the line as is.

Learning how to reuse part of a command, or repeat it in its entirety, is a great keystroke saver and almost a real amanuensis.

An Easy Way To Change The "Default" Drive

Before we get into our discussion of the true DOS amanuensis, here is a quick and easy solution to a common problem: getting around the default drive instruction. Let's take an example or two to explain what I mean.

If you have the marvelous ThinkTank "idea processor" (a must for all writers), you know that the ThinkTank program disk must be in the A drive when you load it, even if you have a hard disk. But you will probably want to store most, if not all, of your ThinkTank "outlines" on another drive (since there is little space on the program disk).

When you load ThinkTank, it thoughtfully opens the last outline file that you were working on during the last session. However, every time you wish to open another outline file, you must tell ThinkTank the drive designator for the file (that is, b: or c: or whatever) as well as the file name. There is no way to change from within the program the default drive where ThinkTank is to look for its work files (as is possible in, say, Lotus 1-2-3). This is a hassle (and a lot of keystroking), since you have to add the drive designator for every file. And if you forget, ThinkTank would not be able to "find" the file.

Similarly, WordStar and DisplayWrite 2 are set up to "assume" that the default drive is A. Of course, you can change the default drive from within the program once you've loaded it (with the L command in WordStar or the <Ctrl><Dir> key in DisplayWrite 2), but you would have to do this each time you start working with the program.

Dear Reader, don't despair! There's a way around this inconvenience: change the default drive before you load the program. This is done, as you know, by typing at the DOS prompt the new drive designator:

```
b: <RETURN>
```

Then, load the program from the A drive by appending the drive designator to the program name, like so:

```
a:tank <RETURN> [for ThinkTank] or
a:ws <RETURN> [for WordStar] or
a:dw2 <RETURN> [for DisplayWrite 2]
```

The program will be loaded from the A drive as normally, but it will use the B drive as the default drive. That's because the DOS instruction governs the default drive. You thus don't have to key in the drive designator for each different file name when you are working in a program. So if you're the lazy type, as I am, you will not have to fumble with the colon key too much.

Using this technique would also be how you could cajole WordStar (or any other program that doesn't "understand" DOS subdirectories) to read files from a different directory on the default drive. As you may know, you cannot change to a different directory, or open a file in another path, from the WordStar Opening Menu. However, you can set up a batch file that first changes to the appropriate subdirectory and then loads WordStar from a different drive.

For example, this batch file changes the "default drive," that is, the subdirectory to be used on the B drive, to an existing subdirectory called work, but loads WordStar from the A drive:

```
cd work <RETURN>
a:ws <RETURN>
```

WordStar will show the current files in the work subdirectory after it is loaded. However, you cannot use the WordStar file utilities (such as O to copy or E to rename a file) between different directories. (They will work in the current directory.) You would have to exit back to DOS for that. The reason for this is that the backslash [\] is a legal filename character in WordStar, even though it is illegal in DOS, so WordStar wouldn't interpret this character as a subdirectory designator.

What happens if you create a new file in WordStar with an illegal character? WordStar will allow you to open the file and work in it, but when you attempt to save the file you will get a "Fatal Error" message and be plopped back to DOS. Don't do it! On a hard disk, you would have enough space to have the program resident in the subdirectory anyway.

Going One Step Further: Batch Files

That's not much of a saving in keystrokes, you may opine, since you still have to type in the drive designator once manually anyway, whether it be from DOS or within the program, and then append a: to load the program. And of course you're right. But be patient, we're not finished yet!

Once you understand the pattern for changing the default drive before you load a program, you can then set up the entire kit and caboodle in a batch file. Batch files are a kind of "shorthand" that you give to your DOS amanuensis, which then does the rest of the typing. They allow you to tell DOS to do many different commands in succession, all without the necessity of your having to key the commands yourself.

Thus, a typical batch file might well be one to change the default drive and then load ThinkTank. It would contain the following two command lines:

```
b <RETURN>
a:tank <RETURN>
```

Here's where the real keystroke saving comes in. You can start a batch file with one keystroke, followed by <RETURN>. DOS does the rest. I have, for example, set up my ThinkTank batch file so that all I type is t <RETURN> to start the ball rolling.

Even though you only type t, the actual file is slightly longer. It must have the file extension .bat. This stands, of course, for "batch." Like regular DOS command files (which have the file extension .com or .exe), batch files need only be invoked by typing the file name without the extension. Hence, t.bat can be typed in as t.

You can create batch files with as many commands as you wish, provided you keep in mind that DOS executes the commands in the order that you have typed them in the batch file. Be very careful about this! An easy way to prevent mistakes is to jot down the commands that you issue during a normal loading and work session so that you have the order correct before you create the batch file.

Let's do one now. Look at the following example:

```
b: <RETURN>
a:print <RETURN>
a:dw2pg a: <RETURN>
a: <RETURN>
```

This batch file first changes the default drive to B, then loads the DOS print command, which is used by DisplayWrite 2. The third line is important: it loads DisplayWrite 2 from the A drive and tells DisplayWrite 2 where its other program files are (note the second a:, which must be included in the command line, since DisplayWrite 2 is programmed to look for its other files on the current drive, which is now B, but the program is still on A).

The rest of the batch file will be run after the operator is finished working with DisplayWrite 2. He or she types the "exit to DOS command" (z). This returns control to the batch file (patiently waiting all the time). The batch file then changes the default drive back to A, and the user is then ready to load another program from the A drive.

This entire batch file (which must be on all three DisplayWrite 2 program diskettes, by the way) can be named d.bat. With one letter (d) and the <RETURN> key, it will do a great deal! Thus, you merely tell your DOS amanuensis the shorthand code letter, and it will take care of the rest.

Setting Up A Batch File

Batch files must be "straight ASCII" text, which means that they can't contain special word processing formatting commands. So if you wish to create a batch file with WordStar, for example, make sure that you use the "non-document" mode (N from the Opening Menu). Other word processing programs may call this "program mode" or "ASCII mode." And always give batch files the .bat extension.

However, for creating short batch files it is often much easier to use DOS itself instead of having to go into your word processing program. In fact, you don't have to use any word processor or text editor (such as the infamous edlin) at all to create a batch file. (Some word processors, like DisplayWrite 2, do not have a "non-document" or "program" mode.) Merely use DOS's copy command instead.

What does copying have to do with typing in batch files? More than you think. In the sometimes bizarre world of computer terminology, a command can have several applications. The DOS copy command can actually do more for you than merely copy a file. In this case, you will be "copying" the screen output to a batch file. (Copy can also "copy" a file to the printer for a quick-and-dirty printout.)

What happens is this: after you tell DOS to start copying, it waits for you to type in the commands (which then appear on the screen after you type them). When you're finished typing in the various commands for the batch file, you tell DOS to copy these commands from the screen into a file.

The trick is to use the word con: in the copy command instead of the first file name. Note the colon: it is very important. What con: does is instruct DOS to copy from the console (that is, the screen), instead of from a normal file. Thus, to create your example DisplayWrite 2 batch file called d.bat on the program diskette (in the A drive), you would type at the DOS prompt:

```
copy con: d.bat
```

The spaces are also important! After you type in the line, hit the <RETURN> key. The cursor will move down a line, and DOS will wait for you to type in the commands that you wish to copy into the batch file.

It is important that you hit <RETURN> after each command, just as if you were entering it manually. No special symbol will appear on the screen, but DOS will "see" the carriage return code in the batch file and enter the command for you when you run the batch. Each separate batch command must therefore be on a separate line. Make sure that the cursor is positioned on the next blank line after the last command line in the batch file when you've typed in all the commands.

DOS doesn't know when to start copying these batch commands from the screen into the batch file until you tell it. With the cursor still on that blank line, hit the <F6> key on the left side of the keyboard. This puts an "end-of-file" marker into the file, which DOS needs so that it stops processing the batch file correctly. (A ^Z will appear on the screen; that means end-of-file). Finally, hit <RETURN> to finish the procedure. DOS responds with the message:

```
1 file(s) copied
```

DOS has thus "copied" the screen output (your batch commands) to a file called d.bat. Always do a trial run of the batch file to see that it really does what you want it to do. If you find that you didn't key in the commands correctly, you can repeat the entire procedure (until you get it right!), or edit the file with your word processor (remember: "non-document" or "program" mode). Don't be put off by the row of .@'s that your word processor will display in the batch file: they represent that CTRL-Z.

The True DOS Amanuensis

There is one special batch file that DOS is instructed to "look for" every time you turn on your computer, or use <Ctrl><Alt><Del> to reboot it. This is the automatic executing batch file, which has its own reserved name: autoexec.bat. DOS looks for this file when you start or reboot your machine. If it doesn't find it on the program disk, it asks you for the date and time as usual and then presents you with the A> or C> prompt.

This file is an amanuensis in the truest sense of the word. You can set it up to load the particular program automatically without having to key in anything at all! You can even include instructions for DOS to ask for the date and time.

For example, if you wanted your sample DisplayWrite 2 batch file to load automatically, you could set up the autoexec.bat file on the DisplayWrite 2 program disks like this:

```
date <RETURN>
time <RETURN>
b: <RETURN>
a:print <RETURN>
a:dw2pg a: <RETURN>
a. <RETURN>
```

Notice the addition of the DOS date and time commands in the batch file. If you don't want them, merely leave them out, and DOS will then not ask for the date and time. You can have one command and not the other: it's entirely up to you. Then, whenever you turn on or re-boot your computer with this diskette in the A drive, autoexec.bat does everything for you.

For programs like DisplayWrite 2 and Framework that use several program disks, the batch file must be copied onto each program disk, because DOS "looks" for the batch file when you exit the program. For instance, Framework uses two program disks. If you have the batch file only on the first program disk, when you exit Framework, DOS wouldn't be able to find the batch file on the second program disk, which is the one currently in the A drive. By copying the batch file to all necessary program disks, you avoid a lot of disk-swapping.

Of course, if you want to make best use of this special batch file, you should have autoexec.bat files on each program diskette, each file customized for the particular program. Even so, these files must all be named autoexec.bat exactly. No other name will work, and there can only be one such file on each diskette. You may have noted that many programs come with an autoexec.bat file, which you can also customize to suit your own needs. (How do you see what's in one of these batch files? With the type command.)

Other Useful DOS Batch Files

Here are a couple batch files that can be very useful at times. The first one, which I have named d.bat, gives me a directory listing of the B drive (where my work files are usually kept):

```
dir b:/p <RETURN>
```

But what is the /p? Many people don't know this built-in feature. It is called page mode and will give you the directory listing by pages, so that you don't have to worry about the entire directory scrolling past you off the screen before you have time to hit the <Ctrl><NumLock> command to stop the scrolling. This is especially useful if you have a hard disk. At the end of each page, DOS will pause and wait for you to strike any key to resume the directory listing. (You could also use the /w — "width" — mode to give you a wide directory listing, but this won't show the size of the file or the date and time stamp.)

The next batch file (c.bat) copies all the work files from the B drive to a backup disk in the A drive, but first erase all files with the .bak extension (such as those created by WordStar):

```
erase b:*.bak <RETURN>
copy b:* * a /v <RETURN>
```

The /v parameter verifies the copying.

A Word About Filters And Pipes

Here's a very useful batch file that need only be set up once. I call it s.bat, for "sort." It also illustrates some other nifty DOS features.

```
dir b: ¦ sort
```

When I type s and hit <RETURN>, the batch file gives me a sorted directory of the B drive. I never have to type in the entire command line again, nor do I have to worry about trying to find and hit the vertical bar key (|).

By the way, if I wanted this sorted directory to be printed, I would have the batch file set up as dir b:| sort > prn. The > is a pipe that redirects the output from the screen to the printer (provided, of course, that I remember to turn the printer on).

You do know about sort, don't you? It is one of three DOS filters that change ("filter") the normal output of a command. In this instance, the sort filter takes the directory listing and sorts it alphabetically.

By the way, you can also sort in reverse order (that is, from Z to A), or by file size (great when you want to review and then move larger files to a new diskette to release diskette space.) These operations are done with special "switches" added to the command line. For example, the command dir b:| sort /r will sort the directory in reverse order. The command dir b:| sort /+14 will sort by file size, smallest to largest. The command dir b:| sort /r/+14 will sort by file size, largest to smallest. If you set up these long and complicated commands in batch files, then you save yourself time and work.

The other two DOS filters are find, which finds a particular string of characters in a file (such as when you want to find which file contains that letter you sent to Geraldine Ferraro) and more, which pauses the screen. The latter saves you fumbling with <Ctrl><NumLock> to stop the scrolling of files when you type it. Get to know filters and pipes! (If you are using MS-DOS instead of PC-DOS, you have the luxury of a fourth filter, cipher, which can encrypt and decrypt your files so that no one, not even a real amanuensis, can read them!)

Batch Files On The Hard Disk

If you have a hard-disk system, you can have similar batch files, including an autoexec.bat file, too. And you can include in the batch files commands to move over to other directories, thus saving even more keystroking. For example, let's assume that you have DOS on the "root" directory, but DisplayWrite 2 and its work files on a subdirectory called dw2. Your d.bat file (called from the root directory) might look like this:

```
cd dw2 <RETURN>
print <RETURN>
c:dw2pg c: <RETURN>
cd  <RETURN>
```

The first line of this batch file changes the directory from the root to the existing subdirectory called dw2. It then loads the print command (which you have previously copied into this subdirectory) and DisplayWrite 2. When you are finished with DisplayWrite 2 and exit back to DOS, the batch file returns you to the root directory. It's entirely possible, of course, to have much more complicated batch files that go through many different directories.

Redoing DOS

Lazy typists, rejoice! Here is a way to get around having to type all those repetitious and boring DOS commands all the time. You can set up a batch file which will type most of the command, but allow you to enter the variable information each time you use the batch file. It's almost as if you were redoing DOS to suit your personality!

For example, in the following command the file name will be different each time you use the command, as you know:

```
erase filename.txt
```

To set up a batch file called e.bat (for "erase"), after using the copy command as outlined above (that is, copy con: e.bat), type in this line:

```
erase %1 <RETURN>
```

Then hit the <F6> and <RETURN> keys to save the batch file.

The %1 is an instruction to DOS to substitute whatever file name you supply when you run the batch file. Thus, if you wish to erase a file called junk.bak, you would type:

```
e junk.bak <RETURN>
```

Note the space between the e and the file name.

Here is an even better example of how to save a lot of boring typing. This is a batch file called r.bat (for "rename"):

```
rename %1 %2 <RETURN>
```

Whenever you wish to rename a file, you would first type the batch file name, a space, then the name of the file to be renamed, another space, and finally the new name. DOS will take care to substitute the first file name for the %1 in the batch file, and the second file name for the %2. Thus, to rename the file junk.bak to good.txt, the command would be:

```
r junk.bak good.txt <RETURN>
```

How would you set up a similar batch file called to copy one file to another? Try it!

This substitution option is just one of the fancier features of batch files. In fact, you can even have batch files with rudimentary programming in them, such as "if/then" constructs and "goto" statements that work only on certain conditions. You can instruct DOS to display messages and prompt lines, to pause for operator input (such as a file name) from the keyboard during a batch file, or to clear the screen, among other things. (The installation program for Lotus 1-2-3, to give one example, is actually one such programmed DOS batch file.) Why, you can even set up a simple menu "shell" to govern other batch files, almost like a real turnkey system. Check the DOS manual under "Batch" or "Automatic Command Entry" for more information.

Saving keystrokes means saving work. With a very little amount of planning and typing on your part, you can instruct your own personal DOS amanuensis to spare you the boredom of repetitious typing so that you can get down to using your computer (and your fingers) more productively.

For Those Times When You Really Need A Typewriter

Fred T. Ormand
140 Galloping Hill Road
Basking Ridge, NJ 07920

Now that you have your computer, printer and word-processing program in good working order, have you ever been faced with a form with spaces to be filled in with typed answers? Has addressing an occasional envelope using a word-processing program proved to be more trouble than it is worth? I found myself in this situation several times after my elderly typewriter reached a stage of unreliable senility. Since it offended my sense of technology and economics to buy another typewriter, I wrote a short BASIC program to enable my system to act as a typewriter. It should be useful for anyone who faces the same problem.

The program, TYPIST.BAS, is listed in Table 1. It is written in Microsoft BASIC-80 for an H-89A with CP/M 2.2.04. It should work with any BASIC and H/Z operating system that supports the INKEY$ function and LPRINT. This 24-line program is liberally supplied with REM (comment) statements; these may be omitted if desired.

When any key is pressed, the corresponding character is sent to the printer and the screen. The Space Bar will send a blank space, the Back Space will cause back-spacing, the Return key will start a new line, etc. Of course the form or envelope must be positioned manually so that printing begins on the desired line or position.

Statements 30 - 70 define strings of ASCII characters to be used by the program. This is more efficient than redefining CHR$(whatever) every time and simplifies programming. Statements 90 -110 clear the screen, remind the user to make sure that the printer is ready, and explain how to exit the program. The major working portion of the program is in statements 130 - 220.

The INKEY$ function is used for direct keyboard input without using the return key (see statement 130). If no key has been pressed, INKEY$ returns a NULL value for CH$. Statement 140 tests CH$ for NULL and goes back to statement 130 to try again if a NULL is found. This loop continues until a key is pressed. When a key is pressed, INKEY$ returns the ASCII value of that key to CH$, CH$ is found to be not equal to NULL, and the program proceeds to statement 150. If the Escape key is pressed, CH$ is found to be ESCAPE by statement 150 and the program ends. If any other key is pressed, the program continues to statement 170 to output the corresponding character to the screen and the printer. The character in CH$ is output to the screen and the printer by statements 170 - 180. The semicolon at the end of the PRINT statement and the LPRINT statement prevents an automatic carriage return after the printing of the character. Of course if the Return key was pressed, a Carriage Return character is sent to the screen and the printer.

The H-89 screen needs both a Carriage Return and a Line Feed to move the cursor to the beginning of a new line, so a Carriage Return must be followed by a Line Feed to the screen. Many printers, including my Smith-Corona TP-1, automatically advance to a new line after receiving a Carriage Return. Statement 190 tests CH$ to see whether a Carriage Return was sent to the screen and printer; if not, the program goes back to statement 130 to get another character from the keyboard. If CH$ was a Carriage Return, statement 200 sends a Line Feed to the screen following the Carriage Return. If your printer needs a Line Feed following a Carriage Return, you should omit the initial REM from statement 210 so that a Line Feed is LPRINTed to the printer after a Carriage Return. If your printer automatically advances to a new line after receiving a Carriage Return, you may either omit statement 210 or include it as a comment with the initial REM. After Line Feeds have been sent, as needed, following a Carriage Return, statement 220 returns the program to statement 130 to get another character from the keyboard.

This "bare bones" program does not automatically send special characters to the printer to set margins, tabs, etc. Such features are not usually needed for simply filling out forms or addressing envelopes. However, any character or control character entered from the keyboard will be transmitted to the printer, so printer setup commands can be sent, if desired. Since these commands vary from printer to printer, you will have to figure them out for yourself and enter them from the keyboard, if needed. If you need to send Escape sequences to set these parameters, you should change statement 150 to use a different special character to end the program (or perhaps omit 150 altogether and rely on Control-C to exit this BASIC program).

Although TYPIST.BAS is in no sense a substitute for a word-processing program, it should prove useful "for those times when you really need a typewriter."

```
10 REM PROGRAM   "TYPIST.BAS"        F.T.ORMAND  5/12/85
20 REM ---- DEFINE PARAMETER STRINGS ------------------
30 NU$ = CHR$(0)             :REM NULL
40 LF$ = CHR$(10)            :REM LINEFEED
50 CR$ = CHR$(13)            :REM CARRIAGE RETURN
60 ES$ = CHR$(27)            :REM ESCAPE
70 CL$ = ES$+"E"             :REM CLEAR SCREEN
80 REM ---- CLEAR SCREEN AND GIVE PROGRAM MESSAGE -----
90 PRINT CL$
100 PRINT "THE PRINTER IS NOW A TYPEWRITER
    MAKE SURE IT IS ON & READY"
110 PRINT "          Press ESCAPE key to exit program.",
    CR$,LF$
120 REM *** BEGIN PROGRAM ***
    GET A CHARACTER FROM THE KEYBOARD ***
130 CH$ = INKEY$             :REM READ A KEY
140 IF CH$ = NU$ GOTO 130    :REM IF NULL, TRY AGAIN
150 IF CH$ = ES$ GOTO 240    :REM IF ESCAPE GOTO END
160 REM --- PRINT THAT CHARACTER ON SCREEN &
    LPRINT IT ON PRINTER ---
170 PRINT CH$;               .REM SEND CHARACTER TO SCREEN
180 LPRINT CH$;              :REM SEND CHARACTER TO PRINTR
190 IF CH$ <> CR$ GOTO 130   :REM IF NOT CR$ GET ANOTHER
200 PRINT LF$;               :REM SCREEN NEEDS LF AFTER CR
210 REM LPRINT LF$;
    :REM SOME PRINTERS NEED LF AFTER CR, MINE DOES NOT
220 GOTO 130                   :REM GET ANOTHER CHARACTER
230 REM **** END PROGRAM ******************************
240 END
```

Table 1

*

Welcome to DOODLER

It's Great!

Graphics Package

...for the Zenith Z-100 and Z-150/160

- Full feature graphics design package
- Save designs on disk for later use
- Playback mode for error correction
- Extended text capability includes...
  User designed character fonts
  Italic or backslant styles
  All text may be scaled

See DOODLER at your local Heathkit Electronic Center

...or Send $ 79.95 directly to...

PAUL F. HERMAN
DATA SYSTEMS CONSULTANT
P.O. Box 206
Safety Harbor, FL 33572

Specification Sheet available on Request

EMULATE

A program which allows the H89 to read/write to the following disk formats.

Actrix	Eagle II	Morrow MD	Superbrain Jr
AMPRO	Epson QX-10	NCR DecMate 5	Televideo
Beehive Tpr	Fujitsu CP/M86	NEC PC-8001A	TRS80-1 CP/M
CDR	IBM CP/M86	Osborne 1	TRS80-3 CP/M
Cromemco	IMS 5000	Otrona	TRS80-4 CP/M
DEC VT180	Kaypro II	PMC MicroMate	Xerox 820
DEC Rainbow	Magnolia	Sanyo 1100	Zorba

Now includes 42 formats! Uses a modified version of CP/M 2.203 or .04 BIOS which is included with the program. Allows the use of virtual drives and reading of 40-track disks in an 80-track drive.

For H37 with Heath CP/M $59

Check for C.D.R. and Magnolia versions.

Automatic Repeat

Simple plug-in installation of the REP3 gives your H89/H19 keyboard the same auto-repeat function you get with a Z100. Provision for a defeat switch.

A Must For Word Processing!

Kit $32

Assembled $40

Real Time Clock

Install the TIM2 in a left expansion slot of your H89 to have date and time keeping with battery backup. Requires soldering 4 wires to the CPU board.

Kit $55

Assembled $65

Software on Disk $10

(Specify Format)

4MHz Mod for H89

Plug-in installation. Software for Heath and CDR CP/M.

Assembled $35

6MHz Mod for H89

Requires soldering on CPU board. Call or write for details.

Assembled $49

CDR disk controllers

FDC 880H for H89 . . . **$345** FDC H8 for H8 . . . **Call**

Include CP/M s/n when ordering.

CDR Super RAM 89 - up to a Megabyte for the H89!

Main board w/o RAM - specify disk format **$175**

Expander board w/o RAM **$69**

Boards with RAM . . . . **Call**

MAGNOLIA MICROSYSTEMS

Disk controller 77316 . . . . **$345**

CP/M+w/128K RAM board 77318 . . . . **$430**

Specify disk format for software

ZCPR3 - we have it. Call for information.

The Software Toolworks® - We Sell It At Discount!

CALL OR WRITE FOR CATALOG   PRICES SUBJECT TO CHANGE

Terms: Check or Money Order - VISA/MC - C.O.D.
Add $3 per order for shipping and handling
CA residents add 6% tax

ANALYTICAL PRODUCTS   **209/564-3687**

20663 Ave. 352   **Woodlake, CA 93286**

H-89 HDOS/CPM Interface To TRS80/ MODEL 100 Portable Computer

V. N. Fritz
70 Costello Drive
Winnipeg, MB
Canada R2Y 1W7

In our area, several of the Huggies are using TRS80 portable computers. Having used the portable for some time, with its BASIC, text editor, telecom and data base programs, I felt that there had to be a way to interface it with my H89. After many dismal attempts, it became clear that it was not as simple as most articles and manuals would lead you to believe. The solution that I came up with, although not perfect, is certainly workable.

The TRS80 BASIC is relatively straight forward (Microsoft). It was obvious that any programs conceived and written while away from home, could be readily transferred to the H89 upon return. A like observation was made concerning the text editor. It operates very similarly to the current Heath/Zenith word processors.

The TRS80 uses a 80C85 CPU. Surprise! The machine language coding and assembler of the 8080(Z80) is compatible with the TRS80. This meant that I could write assembly language programs on the H89 for the TRS80! Armed with the TRS80 Technical Reference Manual (26-3810), I now had the tools and the TRS80 Model-100 BIOS calls.

Being interested in assembly language programming, I immediately wrote a TRS80 BASIC load program that would take a CP/M Hex file and translate the hexadecimal coding to machine language. A block of 500 bytes was reserved for the TRS80 program. This means that the program must be originated at F400H. After this data is assembled, the hex file is transferred to the TRS80 and poked into memory and then saved with a command that works like the CP/M SAVE command. The HexLOAD.BA program listing follows.

```
10 CLS:PRINT@18,"HexLOAD
20 CLEAR500,MAXRAM-500:ONERRORGOTO500
30 INPUT"Enter hex file name ";Z$
35 IFMID$(Z$,LEN(Z$)-2,1)<>" "THENZ$=Z$+".DO
40 OPENZ$FORINPUTAS1
50 LINEINPUT#1,D$:H$=MID$(D$,4,2)
60 GOSUB300:SA=H*256:H$=MID$(D$,6,2)
70 GOSUB300:SA=SA+H:P=SA-1:GOSUB200
90 LINEINPUT#1,D$:IFLEN(D$)=11THEN400
100 GOSUB200:GOTO90
200 D$=MID$(D$,10,(LEN(D$)-11)):PRINT:PRINTD$'line
210 FORL=1TOLEN(D$)STEP2:H$=MID$(D$,L,2)
220 GOSUB300:P=P+1:POKEP,H:NEXT:RETURN
300 PRINTH$;:H=0:CB$="0123456789ABCDEF
310 FORI=1TO2:FORJ=1TO16
320 IFMID$(CB$,J,1)=MID$(H$,I,1)THENH=H*16+J-1:J=16
330 NEXTJ,I:RETURN
400 CLOSE:Z$=LEFT$(Z$,LEN(Z$)-3)
410 PRINT:IFLEFT$(Z$,4)<>"CAS:"THEN440
420 PRINT"Prepare recorder to receive ";Z$
430 INPUT"Press ENTER when ready!";D
440 SAVEM Z$,SA,P+1,SA:END
500 IFERR=7THENPRINT"Out of memory!":PRINTZ$;SA;P+1:GOTO520
510 PRINT"Structural problem with hex file!
520 BEEP:END
```

Notice that there are few spaces included in the listing. This technique although rendering the program difficult to read, saves a considerable amount of space in a portable computer with already limited memory space. On with the task.

Physically connecting the TRS to the H89 was simple. A 25 lead ribbon cable was inserted between the TRS80 RS232 connector and the H89 RS232 connector that is fastened to port 320Q. This would allow the TRS to act as a TTY: in CP/M or an AT: in HDOS.

The problem was mainly with CP/M and its requirement for a carriage return, as well as a line feed at the end of each string. HDOS would accept just a carriage return as a string terminator, as does the TRS portable. Fortunately, the TRS80 disregards line feed characters. It became apparent that there had to be a way to stuff a line feed into the ASCII file being transferred.

Enter stage left, the solution. A program was written in REMark #33, page 27, CP/M Console Swapper. Although not essential, it makes the interface operation easier. One could have used the STAT CON:=TTY: command on the H89 to turn control of the H89 over to the TRS80 and STAT CON:=CRT: command on the TRS80 to return control to the H89 keyboard. If the CONSWAP program is set to run on warm boot, a [CTRL-C] on the current console will leave the control of the H89 up for grabs by either console.

The second part of the problem was solved with a homebrew program called UPLOAD.COM that bypasses the CP/M BDOS glitches. I attempted to use PIP and the command FILENAME.EXT=TTY: but too many characters were lost. The main pitfall was the fact that BDOS spends too much time checking to see if there are any housekeeping chores to do, when its main task, in this particular application, is getting data from a port and storing it in memory.

One of the biggest time consumers is having to route system calls through BDOS, so why not use a bios call to get the character?

This was the final approach taken. UPLOAD.COM will transfer data at 9600 baud or slower if you prefer. But why?

Ok, what about that line feed? Not to worry, that little problem was solved too. When the incoming character is determined to be a carriage return, UPLOAD branches and puts a line feed in memory immediately thereafter. If the incoming character is a 1AH (CTRL-Z), the program stores the received data in a previously selected CP/M file.

Now it's your turn. The UPLOAD.ASM text should be entered and assembled to UPLOAD.COM. The CONSWAP.ASM text should be entered and assembled to CONSWAP.COM.

In HDOS, the solution is very simple. No programs to write or fudge with the operating system. ATH84.DVD copied to AT.DVD. You may already be using AT: for another device, so use another device driver name (ie: XT.DVD).

Next, came the configuration. I used the SET program to set the driver to 2 stop bits (2SB), allow lower case letters (NOMLC) and 300 baud (BAUD 300). A higher baud rate was tried, but unfortunately like CP/M, HDOS has a lot of housekeeping to do. Consequently, characters were lost in the transfer.

In both CP/M and HDOS, some editing may be required in the transferred file. Most files end up with no serious errors in the destination text file. It is usually carriage returns that will be prefixed or appended to the file and are easily edited. So, on with the show.

The operating procedures follow:

CP/M - TRS80

CP/M SETUP:

Copy to a bootable CP/M disk, the following command files.

```
CONSWAP.COM
UPLOAD.COM
```

Configure the CP/M disk to run CONSWAP on warm boot. Set the TTY: to 9600 baud.

TRS80 SETUP:

Go to the TELCOM program, press F3 (STAT) and enter 8712E [CR] and set to FULL duplex in the terminal mode.

To copy data to the TRS80:

1. On the H89 type [CTRL-C]. CONSWAP should run. Go to the terminal mode on the TRS80 and press ENTER. The TRS80 should now have control of the H89.
2. Enter TYPE FILENAME.EXT ***Do not press [CR].***
3. Press F2 (DOWN load) key and enter the destination file name, and press [CR] ***two times (2).***
4. Press the TRS80 F2 (DOWN load) key to close the file, when the transfer is complete.
5. Press [CTRL-C] to relinquish control of the H89.

To copy data to the H89:

1. On the H89 type [CTRL-C]. CONSWAP should run. Go to the terminal mode on the TRS80 and press ENTER. The TRS80 should now have control of the H89.
2. Type UPLOAD[CR]. UPLOAD should run and request a destination file name.
3. Enter FILENAME.EXT[CR]. UPLOAD will then inform you that a file has been opened under the requested file or that a file already exists under that name. If the latter happens, go to step 2 and enter a new destination file name.
4. Press F3 (UP load) key and enter source filename [CR]. TRS80 will respond with Width:. Press [CR].
5. To close the file, press [CTRL-Z] when the Upload sign turns to normal video.
6. Press [CTRL-C] to relinquish control of the H89.

And now the HDOS solution. 300 baud only. Sorry!

HDOS - TRS80

HDOS SETUP:

Copy AT.DVD=ATH84.DVD and do the following SET commands:

```
SET AT  BAUD 300
SET AT· NOMLC
SET AT. 2SB
REBOOT THE DISK!
```

TRS80 SETUP:

Go to the TELCOM program, press F3 (STAT) and enter 3712E [CR] and set to HALF duplex in the terminal mode.

To copy data to the TRS80:

1. On the TRS80, go to terminal mode, press F2 (DOWN load) key and enter the destination file name
2. On the H89 type COPY AT:=FILENAME.EXT [CR]
3. Press the TRS80 F2 (DOWN load) key to close the file, when the transfer is complete.

To copy data to the H89:

1. On the H89 type:
   ```
   PIP [CR]
   FILENAME.EXT=AT: [CR]
   ```
   On the TRS80 press F3 (UP load), enter source filename [CR].
2. TRS80 will respond with Width:. Press [CR].
3. To close the file, press [CTRL-D], when the UP load sign turns to normal video.
4. To exit from PIP, press [CTRL-D] on the H89.

That's it!

Upload Listing

```
;UPLOAD.ASM
;Copyright 1984 V.N.Fritz
;841024
;SYSCALL.LIB
;CP/M SYSTEM CALL EQUATES

CONIN   EQU     1       ;CONSOLE INPUT
CONOUT  EQU     2       ;CONSOLE OUTPUT
PSTRING EQU     9       ;PRINT STRING->$
LINPUT  EQU     10      ;READ CONSOLE BUFFER
OPEN    EQU     15      ;OPEN FILE
CLOSE   EQU     16      ;CLOSE FILE
WRITE   EQU     21      ;WRITE SEQUENTIAL
MAKEFIL EQU     22      ;MAKE FILE
SETDMA  EQU     26      ;SET DMA ADDR

BDOS    EQU     05H
DFCB    EQU     05CH    ;FILE CONTROL BLOCK 1
DMA     EQU     080H    ;DMA BUFFER
TPA     EQU     0100H   ;TRANSIENT PGM AREA
LF      EQU     0AH
CR      EQU     0DH
CTRLZ   EQU     1AH
ESC     EQU     1BH
```

```
        ORG     TPA
START   LXI     H,Ø
        DAD     SP
        LXI     SP,STACK
        SHLD    STACK

        CALL    TYPTX
        DB      7,ESC,'EUPLOAD',CR,LF+8ØH

FNAME   CALL    TYPTX
        DB      CR,LF,ESC,'KEnter destination file',' '+8ØH
        MVI     C,LINPUT ;GET OUTPUT FNAME
        LXI     D,INBUFF
        CALL    BDOS
        LXI     H,INBUFF+1 ;PTR CHAR COUNT
        MOV     A,M
        CPI     Ø       ;ANY INPUT?
        JZ      FNAME   ;TRY AGN
        MOV     E,M
        MVI     D,Ø     ;DE=STRING LEN
        INX     H       ;HL=START OF STRING
        PUSH    H       ;SAVE 4 LATER
        DAD     D       ;HL=END+1
        MVI     M,Ø     ;Ø TERMINATOR
        INX     H
        MVI     M,'$'   ;$ TERMINATOR
        CALL    TYPTX
        DB      CR,LF,ESC,'KData copied to',' '+8ØH
        POP     D       ;DE=START OF STRING
        PUSH    D
        MVI     C,PSTRING
        CALL    BDOS
        POP     H       ;HL=START OF STRING
        LXI     D,OUTFCB
        PUSH    D
        CALL    BFCB    ;BUILD FCB
        POP     D       ;DE=OUTFCB
        PUSH    D
        MVI     C,OPEN
        CALL    BDOS
        RAL             ;OPEN OK?
        JC      MAKEREC
        CALL    TYPTX
        DB      CR,LF,7,'ERROR! - Output file '
        DB      'exists.',CR,LF+8ØH
        JMP     EXIT
MAKEREC POP     D
        MVI     C,MAKEFIL
        CALL    BDOS    ;MAKE FILE
        INR     A
        JNZ     READY   ;RDY FOR UPLOAD
        CALL    TYPTX
        DB      7,CR,LF,'ERROR! - No directory'
        DB      'space.',CR,LF+8ØH
        JMP     EXIT
READY   CALL    TYPTX
        DB      7,CR,LF,'Ready!',CR,LF+8ØH
        LDA     ØØØ2H   ;BIOS JUMP
        MOV     H,A
        MVI     L,Ø9H   ;CONIN OFFSET
        SHLD    BIOSJMP
        LXI     H,BUFFER
COPYLP  PUSH    H
        CALL    RDBYTE
        POP     H
        MOV     M,A
        CPI     CTRLZ   ;EOF MARKER?
        JZ      DONE
        INX     H       ;POINT 2 NEXT
        CPI     CR      ;CR?
        JNZ     COPYLP  ;NO!
DEWLF   MVI     M,LF    ,INSERT LINE FEED
        INX     H
        JMP     COPYLP
RDBYTE  LHLD    BIOSJMP
        PCHL            ;READ THE CHARACTER
BIOSJMP DS      2
```

```
EOF     DS      2
DONE    INX     H
        SHLD    EOF     ;END OF FILE
        LXI     H,BUFFER
        SHLD    DMAPTR  ;SAVE DMA POINTER
        XCHG            ;DE=DMAPTR
SAVELP  MVI     C,SETDMA
        CALL    BDOS
        LXI     D,OUTFCB
        MVI     C,WRITE ;DMA
        CALL    BDOS
        CPI     Ø
        JNZ     BADWRIT
        LHLD    DMAPTR
        LXI     D,128   ;BYTES IN DMA
        DAD     D
        SHLD    DMAPTR
        XCHG            ;DE=DMAPTR
        LHLD    EOF     ;HL=EOF
        CALL    SUBHL   ;PAST EOF?
        JNC     SAVELP  ;NO

DUNSAVE LXI     D,DMA   ;RESET DMA
        MVI     C,SETDMA
        CALL    BDOS
        LXI     D,OUTFCB
        MVI     C,CLOSE
        CALL    BDOS
        CALL    TYPTX
        DB      CR,LF,'Transfer complete',CR,LF+8ØH

EXIT    EQU     $
        LHLD    STACK
        SPHL
        RET             ;TO CPM

BADWRIT CALL    TYPTX
        DB      CR,LF,7,'ERROR! - Disk full '
        DB      CR,LF+8ØH
        JMP     EXIT

;SUBTRACT HL-DE
;       ENTRY   HL=SUBTRAHAND
;               DE=SUBTRACTER
,       EXIT    HL=DIFFERENCE
,               DE=DE
;               C FLAG SET IF HL<DE

SUBHL   MOV     A,L
        SUB     E
        MOV     L,A
        MOV     A,H
        SBB     D
        MOV     H,A
        RET

,       COSUB.LIB - 84Ø91Ø
;       CONSOLE OUTPUT SUBS

;OUTPUT 'A' TO CONSOLE
;       ENTRY   A=CHAR
;       EXIT    A=A AND 7FH
;       USES    A,F

COUT    EQU     $
        PUSH B ! PUSH D ! PUSH H
        ANI     Ø7FH    ;STRIP PARITY
        MOV     E,A     ;CHR TO E
        MVI     C,CONOUT
        CALL    BDOS
        POP H ! POP D ! POP B
        RET

;       SUB TO WRITE BYTES ADDRESSED
;       BY 'HL', UP TO AND INCLUDING ONE
;       WITH BIT 7=1   ALTERS 'A,F', STEPS
;       'HL' TO LAST BYTE OF STRING+1
```

```
TYPTX   XTHL
TYPTX1  MOV     A,M
        CALL    COUT     ;PRINT IT
        MOV     A,M      ;GET CHR AGN
        INX     H        ;POINT TO NEXT
        RLC              ;CHECK BIT 7
        JNC     TYPTX1   ;RETURN
        XTHL
        RET

;BUILD FCB FROM Ø TERMINATED STRING
;       ENTRY   DE=FCB POINTER
;               HL=FILENAME STRING
;       EXIT    OUTFCB=BUILT FCB
;CALLING SEQUENCE
;       LXI     D,OUTFCB
;       LXI     H,FILENAME
;       CALL    BFCB
;
BFCB    EQU     $
        INX     H        ;TEST 4 ':'
        MOV     A,M
        DCX     H
        CPI     ':'
        JNZ     BFND     ;NO DISK
        MOV     A,M      ;A=DISK LETTER
        ANI     ØØØ11111B
        INX     H        ;JMP PAST DISK
        INX     H        ;JMP PAST .
        JMP     BFSD
BFND    XRA     A        ;DEFAULT DISK
BFSD    STAX    D        ;SAVE DISK
        INX     D        ;DE=FIRST CHAR FNAME
        MVI     C,8      ;LEN FNAME
        CALL    BFXT     ;XFER TOKEN
        CPI     '.'      ;TYPE TOO?
        JNZ     BFNT     ;NO TYPE
        INX     H        ;GET PAST ' '
BFNT    MVI     C,3      ;TYPE LEN
        CALL    BFXT
        MVI     B,Ø
        MVI     C,24
        CALL    BFFT
        RET
BFXT    MOV     A,M      ;XFER BYTE
        ORA     A        ;A=Ø ?
        JZ      BFSFT    ;YES SO SPACE FILL
        CPI     '*'
        JZ      BFQFT    ;FILL WITH ???
        CPI     '.'      ;FNAME DONE?
        JZ      BFSFT    ;SPACE FILL
        STAX    D        ;GUD CHAR
        INX     D
        INX     H
        DCR     C
        JNZ     BFXT     ;DO MORE
BFSKIP  MOV     A,M      ;SKIP TILL   OR Ø
        ORA     A        ;A=Ø ?
        RZ
        CPI     '.'
        RZ
        INX     H        ;UPDATE FNAME PTR
        JMP     BFSKIP
BFSFT   MVI     B,' '    ;SPACE FILL TOKEN
        JMP     BFFT
BFQFT   MVI     B,'?'    ;? FILL TOKEN
        CALL    BFFT
        JMP     BFSKIP
BFFT    PUSH    PSW
        MOV     A,B
BFFTL   STAX    D
        INX     D
        DCR     C
        JNZ     BFFTL
        POP     PSW
        RET

BUFLEN  EQU     15       ;MAX STRING LEN
INBUFF  DB      BUFLEN,Ø
        DS      BUFLEN+3
OUTFCB  DB      Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
        DB      Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
        DB      Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
        DB      Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
        DB      Ø,Ø,Ø,Ø

;STORAGE AREA

        DS      64
STACK   DS      2
DMAPTR  DS      2        ;DMA POINTER SET TO END
BUFFER  EQU     $        ;MUST BE LAST LABEL IN LIST

        END
```

*

CHECKZ

- Continuous check printing
- Report sort on user defined categories
- Report sort on deductibles
- Maintains list of accounts, addresses & account numbers
- Password security
- Searching, reconciling, & archiving facilities
- Ideal for small business or individual

All tangled up in red tape?
Let CHECKZ uncomplicate your home or business records & increase productivity.

CP/M-Z 8Ø, PCDOS, MSDOS Formats
$49.95 Disk and Manual
$5.00 Manual Alone

For further information or orders:
FUTURE COMMUNICATIONS
210 West Benson Street, Atlanta, Georgia 30030

768K Z-100™
WITH OUR
ZMF100™
PACKAGE

$65 EACH

Our ZMF100™ modification package lets you plug 256K RAM chips into your "old" motherboard Z-100™ computer. Up to 768K (27 RAM's) on the part number 85-2653-1 motherboard.

Simple, plug-in installation requires no trace cutting or funny assembly tricks. Fully reversible. No permanent changes to the motherboard.

Requires disassembly of the computer and some easy soldering to connect the pin-1's of the memory sockets together. All materials (except memory chips and solder) are supplied along with complete, detailed instructions.

Price: $65 each.

Quantity Discounts. UPS Shipping Free.
Order Direct or See Your Local Heath/Zenith Dealer.
Literature Available. VISA or Mastercard Accepted.

FBE

FBE Research Company, Inc.
P.O. Box 68234
Seattle, Washington 98168
(206) 246-9815

8 MHz 256k Update

Pat Swayne
HUG Software Developer

In the July issue of REMark, we published an article on how to upgrade an older model H/Z-100 main circuit board to run at 8 MHz and use 256k RAM chips. This article is the result of feedback from readers and personal experience since the original article was published.

First of all, I would like to mention that the 256k modification (page 27, column 1 of the July article) was originally designed by Mike Cogswell, and in the rush to get the article finished by the deadline for the July issue, I forgot to mention that. I apologize to Mike for that oversight.

Old Monitor ROMs

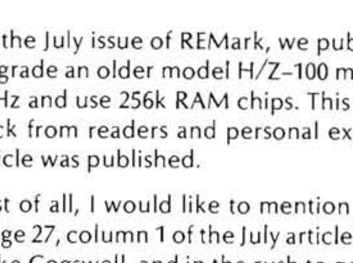
If your H/Z-100 contains a very old version of the monitor ROM (version 1.1 or 1.2), you will need to make some additional modifications besides those in the July article. You can find out which version of the monitor you have by typing V at the "hand prompt" when you first turn on or reset your computer. If you have a version earlier than 2.3, you will need to replace the IC at U161 with a Heath part no. 444-129-1, and you must install the jumper at J102 on position 1.

Wait State Modification

If after performing all of the modifications in the original article you get a "Default controller not ready" message on the screen when you turn your computer on, try removing the modifications made under the heading "Wait State Modification" (page 27, second column), and restoring that section of the board to the original configuration. At this writing, we do not know what the problem is with that modification, but if your computer will work without it, you are probably better off.

Memory Problems

If your modified board seems to work OK, except that you get an occasional system "crash" with the message "ERROR MEMORY PARITY OR BUS", try connecting a 2.2 uF tantalum capacitor (Heath part no. 25-221) to the leads of capacitor C102 (leave C102 installed on the board). The positive lead of the tantalum capacitor should be connected to the C102 lead that is closest to the S-100 connectors.



HEATH/ZENITH 88, 89, 90 PERIPHERALS

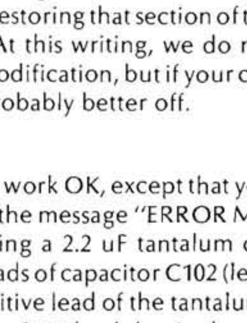
16K RAM EXPANSION CARD
Expand your H/Z 88, 89 RAM Memory to a **FULL 64K** and begin using larger and more powerful programs with our **16K RAM** card.

Fully compatible with: Magnolia Microsystems and CDR CP/M and disk I/O interface cards.

Featuring: Complete installation instructions • Mounting bracket • 90 day warranty

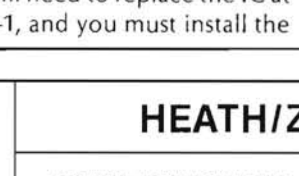
Field reliability record exceeding 3 years

Only $65.00 Shipping & Handling $5.00



PORT SERIAL
2/3 CARD — I/O
PORT PARALLEL

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



Your H/Z 88, 89, 90 can now directly connect and operate EPSON, IDS, ANADEX, GEMINI, SILVER REED, NEC, SUPER 5, PROWRITER, OKIDATA,* and many more line printers using CENTRONICS style parallel interface with our **2/3rds**, 2 port serial, 3 port parallel interface card. Or you may use all 24 digital lines in various configurations of input, output, or bidirectional modes for industrial control or data sampling

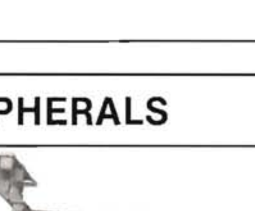
Features:
• 2 Serial Ports Supporting Ring Input, and External Clock • 3 Parallel Ports, 24 Total Digital Input/Output Lines • Fully Compatible with All Models of H/Z 88, 89, 90 using Heath/Zenith CP/M or HDOS • Now Supporting CP/M Version 2.2.04 • Choice of Centronics Line Printer Support Software for the CP/M or HDOS Operating Systems • Reduced Computer Bus Loading and Chip Independent Design

Complete with Installation Instructions, Documentation, 90 Day Warranty, Two Serial Cables and a Parallel Cable Internal to the Computer.

Price $199.00 **Second Operating System Driver $25.00**
Shipping & Handling $10.00

REAL TIME CLOCK

You will be able to perform time and date stamping for point of sales software, and bulletin board software or perform time studies as well as real time data sampling with our **REAL TIME CLOCK**. This peripheral card is a perfect companion to our 2/3rds card for industrial control and data sampling. STOP WATCH time study and alarm demonstration software is included for either the CP/M or HDOS operating systems. You will be able to view the current date and time on screen continuously or simply listen to an audible beep every fifteen minutes and the hour chimed or disable the clock entirely at your option.



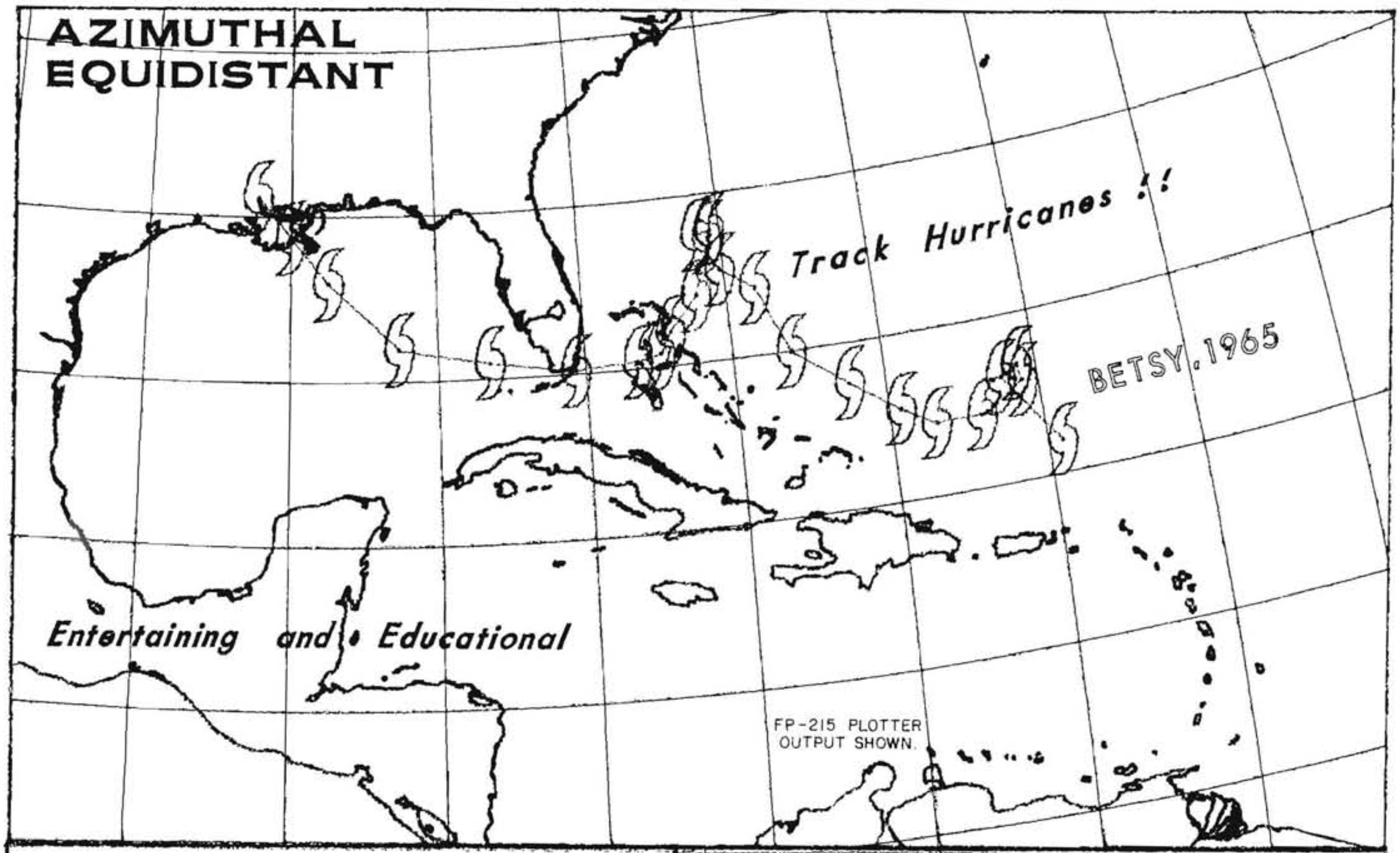
Features:
• True I/O Addressing, Not Memory Mapped • User Selectable Address • Rechargeable Battery Backup Using Commonly Available Batteries • Installable on the Left or Right Side of the Computer (Left side operation requires our I/O expansion module.) • Month, Day, Year, Hours, Minutes, Seconds, 1/10's, 1/100's and 1/1000's of Second Accuracy • Interrupt Capability Based on Tenths of Seconds, Seconds, Minutes, Hour, Day, Week or a Specific Date and Time • Choice of CP/M or HDOS Operating System Software Driver and Demonstration Programs

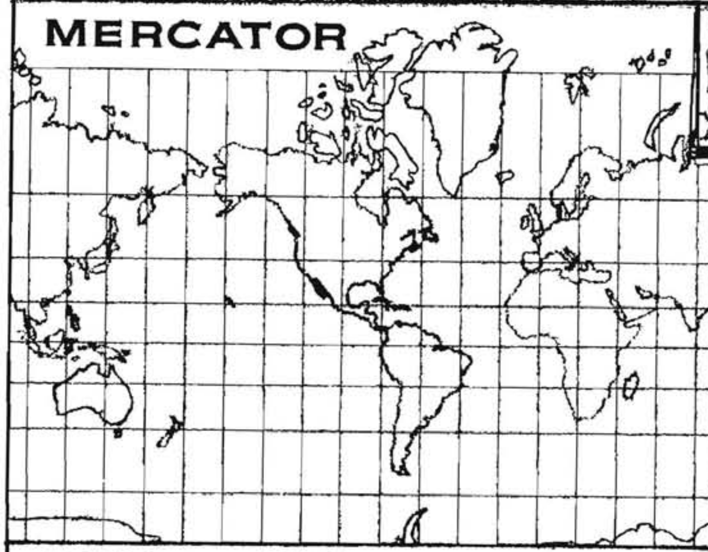
Price $105.00 with Batteries **$89.00 without Batteries**
$ 25.00 Software for Second Operating System
Shipping & Handling $5.00

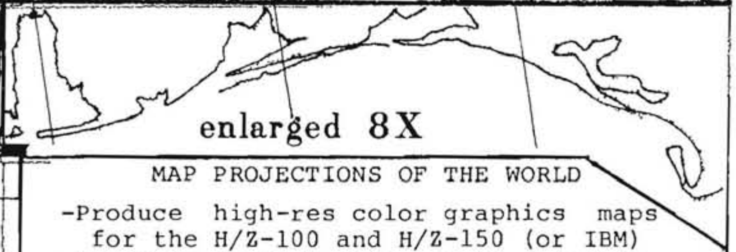
HDOS is a reg. trademark of the Heath Co. CP/M is a reg. trademark of Digital Research
PRICES ARE LESS SHIPPING AND TAX IF RESIDENT OF CALIFORNIA
MAIL ORDER: 12011 ACLARE ST., CERRITOS, CA 90701 (213) 924-6741
TECHNICAL INFO/HELP: 8575 KNOTT AVE., SUITE D, BUENA PARK, CA 90620 (714) 952-3930

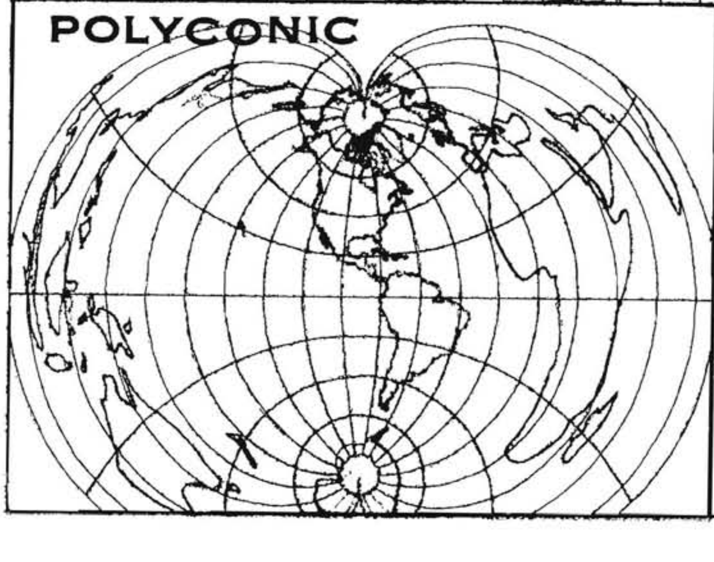
ZENITH data systems
SERVICE CENTER

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









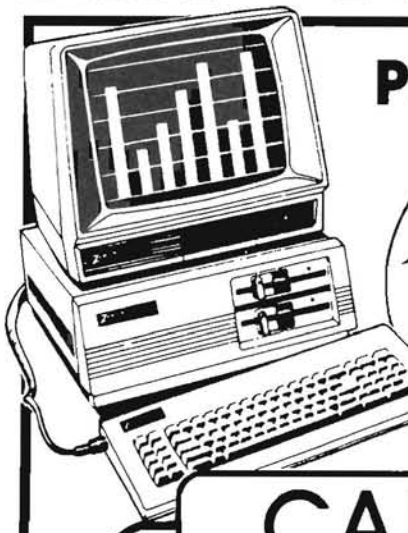
MAP PROJECTIONS OF THE WORLD

- -Produce high-res color graphics maps for the H/Z-100 and H/Z-150 (or IBM)
- -Track up to 10 hurricanes in animation Two historical ones are included
- -Choose any of 7 different projections- 5 Azimuthal & Polyconic & Mercator
- -Pick either of 2 data files: World Coastlines with 5070 digitized points or a very-high-res file of the Caribbean from the Carolinas to Mexico to the Amazon (11940 points)
- -Vary the origin, rotation, window and scale (Global or down to 20 miles/inch on most projections)
- -User's manual includes screen dumps of 16 sample maps to help you learn how to reproduce these and many others
- -Superfast execution- over 1000 pts/min from an optimized compiled basic program that requires only DOS and color graphics. A color monitor does help

"...fascinating look at the world from every conceivable angle... The maps are excellently drawn. It's amazing that Dr Pittman could squeeze that much detail onto a single 5 1/4-inch diskette..."--Dave Felstul--BUSS # 91

Pick up a copy at many Heathkit Centers or order direct by sending $49.95 postpaid (specify Z-100 or Z-150) to: Dr M E Pittman, 57 Emile Av, Kenner, LA 70065

SII VALUE BULLETIN



PRICE PERFORMANCE BREAKTHRUS

ZENITH data systems

Z150 Dual Floppy System	$ CALL
Z150 10 meg Winchester	$ CALL
Z160 Portable Dual Drive	$ CALL
Z160 Portable Winchester	$ CALL

ALL SYSTEMS INCLUDE 320K RAM, COLOR GRAPHICS
Parallel & Serial Ports DOS 2.1 Microsoft Word & Multiplan **WHILE THEY LAST**

CAD-star

INTEGRATED WORKSTATIONS FOR MECHANICAL DESIGN, ARCHITECTURE, PC DESIGN AND MORE.
FEATURING IBM, NEC APC 111, ZENITH DATA SYSTEMS, HITACHI, CALCOMP AND ALPHAMERICS.
FOR APPLICATIONS WITH AUTOCAD, P-CAD, FUTURENET, PERSONAL 870 AND OTHERS.

GRAB BAG SOFTWARE

DBPlus (8" CPM)	$10.00
BOTTOM LINE STRATEGIST (8" CPM)	$10.00
FINANCIAL PLANNER (8" CPM)	$10.00
CROSSTALK (8" CPM)	$10.00
SOFTPLOT BGL (Z100)	$10.00
PALETTE (Z100)	$10.00
CHRONOLOGIC (Z100)	$10.00
ALTCHAR (Z100)	$ 5.00
HDOS TO CPM CONVERSION (Z-89)	$10.00
PRISM MAGIC (APPLE)	$10.00
PEACHTREE HOME LIBRARY (IBM)	$49.00
PEACHTEXT 5000 W.P.	$49.00
4 PT GRAPHICS (IBM)	$49.00
HARD RUNNER (IBM)	$10.00
VEDIT TEXT EDITOR (IBM)	$49.00

ACCOUNTING SOFTWARE

TCS SOFTWARE

PAYROLL (IBM)	$299.00
SALES INVOICING (IBM)	$399.00
FIXED ASSETS (IBM)	$399.00
Q WORD (IBM)	$199.00

PEACHTREE

SALES INVOICING (IBM)	$199.00
INVENTORY (IBM)	$199.00
FIXED ASSETS (IBM)	$199.00
BACK TO BASICS (IBM)	$ 99.00
GENERAL LEDGER (Z100)	$149.00
ACCOUNTS PAYABLE (Z100)	$149.00
ACCOUNTS RECEIVABLE (Z100)	$149.00
INVENTORY (Z100)	$149.00
SALES INVOICING (Z100)	$149.00

DEMOS AND TRADE-INS

EAGLE PC-2 W/256K & MONITOR	$995.00
GEMINI 10X PC PRINTER	$250.00
NEC P3 PINWRITER	$799.00
SIEMENS INK JET PRINTER	$699.00
AMDEK 710 HI-RES COLOR MONITOR	$549.00
HOUSTON INSTRUMENTS DIGITIZER	$549.00
HOUSTON INSTRUMENTS PLOTTER	$649.00
SHARP PERSONAL PLOTTER	$249.00
DUAL 5¼ SINGLE SIDED DRIVES	$249.00
TECMAR JR WAVE 128K	$349.00
Z89 COMPUTERS, 5¼ DRV	$495. - $695.00
IDS 480 DOT MATRIX PRINTER	$249.00

SYSTEMS INNOVATIONS INC.

HOURS: 9AM-6PM, MON.-FRI. • THURS. TIL 8PM • SAT. 10AM-3PM

44 Stedman Street,
Lowell, Massachusetts 01851
(617) 459-4449
"Your Complete Computer Resource Center"

VISA MasterCard AMERICAN EXPRESS

WordStar Filters

FOR THE Z-100 AND Z-100 PC

John F. Smith
239 Colonial Drive
Warminster, PA 18974

If you use WordStar on an H/Z-100 or H/Z-150 running under MS-DOS 2.xx or PC-DOS, this article describes two short, simple filter utilities which will speed up and simplify the task of file conversion between WordStar and other text editors or word processors.

WSCON converts WordStar "document" files to either a standard ASCII text file ("non-document" in WordStar parlance) or to a format which is suitable for input to a different word processor for editing and reformatting.

CONWS performs a complementary filtering function — it converts standard ASCII text files to a format which can be input to WordStar, or another word processor, for editing and formatting.

Don't let the fact that both programs are written in assembly language put you off if you haven't tried it yet! Simply type the listings with a text editor or with WordStar itself (non-document mode, of course) and follow the detailed instructions in the listings to assemble, link and run the programs.

Yes, I know that Heath/Zenith did not furnish an assembler with their release of MS-DOS 2.xx, but the assembler that came with the original ZDOS Version 1.xx — MASM.EXE — will work just fine on these programs. Having said that, I would strongly recommend that any H/Z-100 or H/Z-150 user consider buying the MS-DOS Version 2 Programmer's Utility Pack, Part Number CB-5063-16, whether or not he or she is interested in assembly language programming. Not only does it include a much faster and bug-free version of MASM.EXE, but it is supplied on four (4!) full disks of goodies, including the BIOS source code for both the Z-100 and Z-150 (including the Z-150/Z-100 mode), a RAM disk device driver (much improved over the one furnished with DOS 2.xx), your choice of two excellent full-featured screen editors, a disk full of excellent and useful utility programs and an absolutely outstanding manual, equal in size and quality to the MS-DOS Version 2 manual itself.

WSCON, shown in Listing 1, does the following filtering on a WordStar document file:

1. It strips bit 8 (also known as the parity bit) from all characters. WordStar sets bit 8 true to identify word boundaries and soft non-printing characters. "Soft" characters are those created by WordStar to accomplish word wrapping and justification — carriage returns, line feeds, hyphens and spaces.
2. It removes "dot" commands used by WordStar to control printing format functions, such as page numbering, headings, footings and MailMerge functions.
3. It removes print control functions inserted in the text by WordStar to control printer type style options.
4. Depending on the option you select before assembling the program, WSCON replaces "soft" carriage return/line feed combinations with a space (for word processor output) or a hard CR/LF (for standard text output).

The companion program CONWS, shown in Listing 2, converts files in the other direction — from standard ASCII text to a format that can be handled by WordStar, or most other word processors. CONWS does this:

1. It replaces hard carriage return/line feed combinations within a paragraph by a space, so that the paragraphs can be reformatted by the word processor, while retaining hard carriage return/line feed combinations where they were originally inserted by the typist. The trick here is to determine which is which! The usual approach is to assume that a single CR/LF must be within a paragraph, while a sequence of 2 or more consecutive CR/LF combinations denote a paragraph. Anyone who has converted a long text file with large amounts of columnar or tabular data interspersed with intentional single CR/LF combinations knows what a mess THAT assumption can produce! CONWS is just a bit smarter than that. When it encounters a single CR/LF combination, it looks ahead at the next character. If it is a space or tab, it assumes the preceding carriage return/line feed combination was intentional and restores it in the output file.
2. WordStar, and most other word processors, get indigestion when they see tab characters (09 hex). CONWS expands tabs — replacing them with 5 spaces. If you wish to expand tabs to a different number of spaces, you may do so with the change shown in Listing 2. To expand tabs to 8 spaces, for example, just change the instruction "MOV CX,5" to read "MOV CX,8".

Both of these programs — WSCON and CONWS — are "filters"; they accept input from the standard input (STDIN) and deliver output to the standard output (STDOUT). In MS-DOS 2.xx, STDIN and STDOUT are defaulted to the console. In order to process input from files and send the output to files or another device (e.g. the line printer), you must use I/O redirection. The "<" before the input file name and the ">" before the output file or device name are mandatory. If you inadvertently enter the program name without an argument and find the console staring blankly at you, apparently locked up, it is not necessary to re-boot. Just strike a CTRL-Z, the ASCII end of file marker, and the DOS prompt will reappear.

Both programs are "generic" MS-DOS — that is, they are written using only standard MS-DOS system calls and will run on any MS-DOS machine, including the IBM PC.

Listing 1

```
TITLE - WSCON.ASM - Ver 1.1 - 15-Mar-85
PAGE ,132
;
;       by J.F.Smith
;       239 Colonial Drive
;       Warminster, PA 18974
;
;
;       WSCON is a filter which processes WordStar "document" files to
;       produce an output file in either of two formats, selectable with the
;       assembly option shown below:
;
;               * a standard ASCII text file ("non-document" file in WordStar
;                 parlance)
;
;               * an ASCII text file formatted for input to a different
;                 word processor
;
;       WSCON runs under MS-DOS 2.xx.  It uses only "generic" MS-DOS system
;       calls and will run on the Z-100, Z-150, IBM PC or any standard MS-DOS
;       machine
;
;       It does the following:
;
;               1.  Strips Bit 8 from all characters
;               2.  Removes all non-printing (control) characters except
;                   carriage returns and line feeds
;               3   Removes "soft" spaces inserted by WordStar
;               4.  Deletes "dot" commands
;               5.  Replaces "soft" carriage return/line feed combinations
;                   with hard ones (standard ASCII version, WP=FALSE) or
;                   with spaces (word processing option, WP=TRUE)
;
;       WSCON is a "filter" -- reads STDIN and writes to STDOUT, therefore
;       reguires I/O redirection.
;
;
;       Usage:  WSCON <d:infile >d:outfile
;
;       ASSEMBLY INSTRUCTIONS -- On a disk containing this file, MASM.EXE,
;       LINK.EXE AND EXE2BIN.EXE, use the following.
```

```
        JZ      KILDOT          ;Yes, go get rid of dot command
        CALL    CRLF            ;Else restore CR/LF combination
        MOV     AL,BL           ;Restore the char we saved in BL
        JMP     LOOP2           ;and go process it
;
KILDOT: MOV     AH,7            ;Go back for next character
        INT     21H
        CMP     AL,ØAH          ;Is it a LF?
        JNZ     KILDOT          ;No, drop it and go back for another
        CALL    CRLF            ;Else restore CR/LF combination
        JMP     LOOP1           ;Go back for another character
;
CRLF:   MOV     DL,ØDH          ;Output a CR/LF combination
        MOV     AH,2
        INT     21H
        MOV     DL,ØAH
        MOV     AH,2
        INT     21H
        RET                     ;And return to calling function
;
REMOVE: MOV     AH,7            ;Go back for next character (LF)
        INT     21H
        CMP     AL,ØAH
        JZ      REMOVE          ;
        JMP     LOOP2           ;go back for next character
;
EXIT:   MOV     DL,1AH          ;Make end of file
        MOV     AH,2            ;Call char out function
        INT     21H
        MOV     AH,4CH          ;Recommended Exit procedure
        INT     21H
;
        DW      8ØH DUP (?)     ;Define stack size (required for Int 21H)
TOP_OF_STACK    LABEL WORD      ;Identify top of stack
;
CODE ENDS
        END     START
```

Listing 2

```
TITLE  - CONWS.ASM - Converts ASCII Text File to WordStar Format
;           Version 1.1 - 15 March 1985
PAGE ,132
;
;               By John F. Smith
;               239 Colonial Drive
;               Warminster, PA 18974
;
;
;       This filter utility will convert a standard ASCII text file to a
;       format suitable for input to WordStar or other word processor
;       All system calls are "generic" MS-DOS, so it will run on any
;       computer running under MS-DOS 2.xx, including the Z-1ØØ, Z-15Ø
;       and IBM PC.
;
;       It does the following:
;
```

```
;
;                  MASM WSCON;
;                  LINK WSCON;              (Ignore the "No Stack Segment" warning)
;                  DEL WSCON.OBJ
;                  EXE2BIN WSCON WSCON.COM
;                  DEL WSCON.EXE
;
;
TRUE    EQU     1
FALSE   EQU     Ø
;
WP      EQU     TRUE     ;Output in "word processing" mode, change equate
;                         to 'FALSE' for standard ASCII text output
,
CODE SEGMENT
        ASSUME CS:CODE,DS:CODE,SS:CODE
;
        ORG     1ØØH            ;.COM programs start here
START:  MOV     SP,OFFSET TOP_OF_STACK
        MOV     AH,7            ;Call MS-DOS console in, no echo function
        INT     21H
        CMP     AL,2EH          ;Is it a dot?
        JNZ     LOOPA           ;If not, start processing
START1: MOV     AH,7            ;else, gobble up characters
        INT     21H
        CMP     AL,ØAH          ;until LF appears
        JNZ     START1
LOOP1:  MOV     AH,7            ;Get input function
        INT     21H
LOOPA:  CMP     AL,1AH          ;End of file?
        JZ      EXIT            ;If so, exit
        CMP     AL,2ØH+8ØH      ;WordStar soft space?
        JZ      LOOP1           ;If so, drop it and go back for next

        IF WP                   ;Assemble these 2 lines if WP=TRUE
        CMP     AL,ØDH+8ØH      ;"Soft" carriage return?
        JZ      REMOVE          ;Yes, take it out
        ENDIF

        CMP     AL,ØAH          ;Line feed?
        JZ      CKDOT           ;Yes, see if next character is a dot

        IFE     WP              ;Assemble these 3 lines if WP=FALSE
        CMP     AL,8AH          ;Soft line feed?
        JNZ     LOOP2           ;If not, continue processing
        CALL    CRLF            ;Else, insert a CR/LF combination
        ENDIF

LOOP2.  AND     AL,7FH          ;Remove bit 8
        MOV     DL,AL           ;Move character to DL register
        SUB     AL,2ØH          ,See if control char (<2Øh)
        JL      LOOP1           ;Yes, go back for another
        MOV     AH,2            ;Call output function
        INT     21H             ;Send it
        JMP     LOOP1           ;Go back for next character
;
CKDOT:  MOV     AH,7            ;Go get another one
        INT     21H
        MOV     BL,AL           ;Temporarily save it in BL register
        CMP     AL,2EH          ;Is it a dot?
```

```
;               1.  Removes carriage return/line feed combinations within
;               a paragraph.  A paragraph is assumed if (a) there are 2
;               consecutive CR/LF combinations or (b) a CR/LF combination
;               is followed immediately by a space or tab   The latter
;               is to preserve tabulated or columnar data.
;
,               2   Expands tabs to spaces.  Default expansion is 5 spaces.
;               The default tab can be changed by altering the MOV  CX,n
;               instruction following the EXPAND symbol below and reassembling
;
;       This program reads from STDIN and writes to STDOUT.  Use I/O
;       redirection   Correct syntax is:
;
;               CONWS <d:infile >d:outfile
;
;       ASSEMBLY INSTRUCTIONS -- On a disk containing this file, MASM.EXE,
;       LINK.EXE and EXE2BIN.EXE, use:
.
;               MASM CONWS;
;               LINK CONWS;     (Ignore the "No Stack Segment" warning)
;               DEL CONWS.OBJ
;               EXE2BIN CONWS CONWS.COM
;               DEL CONWS.EXE
;
;
CODE SEGMENT
        ASSUME CS:CODE,DS:CODE,SS:CODE
;
        ORG     1ØØH                    ;.COM programs start here
;
START:  MOV     SP,OFFSET TOP_OF_STACK  ;Point to top of stack
LOOP1:  MOV     AH,7                    ;Call character input function
        INT     21H                     ;Get character
        CMP     AL,1AH                  ;Is it CTRL-Z?
        JZ      EXIT                    ;If so, EOF
        CMP     AL,Ø9H                  ;Is it a tab?
        JZ      TAB                     ;Expand it
        CMP     AL,ØDH                  ;Is it a LF?
        JZ      TEST                    ;Go see if more than one
        MOV     DL,AL                   ;else, let's print it
        MOV     AH,2                    ;Call character output function
        INT     21H                     ;Send it
        JMP     LOOP1                   ;and go back for another character
;
TEST:   MOV     AH,7                    ;Get next LF character
        INT     21H                     ;and send it to the bit bucket
        MOV     AH,7                    ;Go get another one
        INT     21H
        MOV     BL,AL                   ;Temporarily save character in BL
        CMP     AL,ØDH                  ;Is it another LF?
        JZ      PARA                    ;If so, assume it's a para
        CMP     AL,2ØH                  ;Is it a space?
        JZ      COL                     ;Might be columnar data
        CMP     AL,Ø9H                  ;Is it a tab?
        JZ      COL1                    ;Might be columnar data
        CMP     AL,1AH                  ;Is it EOF?
        JZ      EXIT                    ;Then let's get out of here
        MOV     DL,2ØH                  ;Replace CRLF with a space
        MOV     AH,2                    ;Send it
        INT     21H
```

WINCAD

The Easy-to-Learn, Fun-to-Use Graphics Package Designed for Engineers, Architects, Programmers, Illustrators & Artists, and Students

- **Circuit Designs**
- **Floor Plans**
- **Mechanical Designs**
- **Logic Diagrams**
- **Flow Charts**
- **Organization Charts**
- **Preliminary Architectural Designs**

$249.95

Demonstration Disk is Available for
$10.00 + $1.50 postage and handling
Please write for free brochure.

Dealer inquiries are invited.

REDWOOD DEVELOPMENT

7 Redwood Dr.
(406) 494-5610
Butte, MT 59701

WORDKEY™ TAMES THE POWERFUL WORDSTAR

WordKey makes the popular WordStar™ program the easy-to-use professional word processor it should be. More than 90 simple keypad and function-key commands help you use and remember ALL of WordStar's many capabilities, without EVER using WordStar's awkward 'CTRL' key. Your secretary will love it, and so will you and your family.

Full onscreen help is always instantly available. And you'll like its many other features. A letter to H-Scoop from a satisfied user says, "I found [WordKey] made WordStar much easier to use and ...I learned some capabilities that I did not know WS had...I am extremely pleased with it."

Use WordKey with either WordStar version 3.21 or 3.30, Z-DOS 1.25 or MS-DOS 2.13, on your H/Z-100 computer. To order, just mail a check to DelSoft for $49.95. (Calif. residents please add 6% tax.)

PSC-PROWRITER GRAPHICS SCREEN DUMP UTILITY

PSC-PRO prints an exact copy of your H/Z-100 monochrome (B&W or green) screen on the C.Itoh Prowriter printer from within ANY program simply by pressing Shift-F12. $19.95 (Calif. residents please add 6% tax.)

DelSoft
Gary Deley, 564 Calle Anzuelo, Santa Barbara, CA 93111
(805) 967-9566 eves and weekends

```
        MOV     DL,BL           ;Now move in the char we saved
        MOV     AH,2            ;and send it
        INT     21H             ;
        JMP     LOOP1           ;and go back for next character
;
EXPAND: MOV     CX,5            ;Set counter for 5 spaces
AGAIN.  MOV     DL,20H          ;and expand tab
        MOV     AH,2
        INT     21H
        LOOP    AGAIN
        RET                     ;and return to calling function
;
TAB:    CALL    EXPAND          ;expand the tab
        JMP     LOOP1
;
PARA:   CALL    CRLF            ;Send a CRLF combination
        MOV     DL,0DH          ;Insert the LF we captured
        MOV     AH,2
        INT     21H
        JMP     LOOP1           ;and go back for another character
;
CRLF:   MOV     DL,0DH          ;Send a CR/LF combination
        MOV     AH,2
        INT     21H
        MOV     DL,0AH
        MOV     AH,2
        INT     21H
        RET                     ;and return to calling function
;
COL:    CALL    CRLF            ;Send a CR/LF combination
        MOV     DL,20H          ;Restore the space we captured
        MOV     AH,2
        INT     21H
        JMP     LOOP1           ;and go back for another character
;
COL1:   CALL    CRLF            ;Send a CR/LF combination
        CALL    EXPAND          ;and expand the tab we captured
        JMP     LOOP1           ;and go back for another character
;
EXIT:   MOV     DL,1AH          ;Send EOF
        MOV     AH,2
        INT     21H
        MOV     AH,4CH          ;Recommended DOS 2.0 exit procedure
        INT     21H
;
        DW      80 DUP (?)      ;Set up stack space (req'd for INT 21H)
;
TOP_OF_STACK    LABEL WORD
CODE    ENDS
        END     START
```

*

The Use Of Modules In Pascal

Paul W. Simmons
6409 Glenbard Road
Burke, VA 22015

After receiving my Z-100 Computer, I was so pumped up with enthusiasm I began a major software project in a language I've never used, Pascal. "Naive", you say. Unquestionably. However, for years I have heard wonderful things about Pascal and now I have MS-Pascal on my own computer. I just couldn't wait to give it a try. So I'm currently learning all kinds of interesting things about the Z-100, Z-DOS and Pascal. It's challenging, frustrating, time consuming, rewarding, and fun. One useful thing I've learned is how to create independent modules in MS-Pascal. Because modules are such an important tool, I decided to make them the subject of this article.

What precisely is a module? Well, there are three things that can be compiled in MS-Pascal: 1) Programs, 2) Modules, and 3) Implementations of Units. Microsoft calls these compilands, apparently because you can compile them. The first compiland mentioned was the program. We all have a fair idea what a program is. Basically, a program stands alone and may be executed by itself. Modules and implementations of units are only parts of a program. Although they may be compiled, they cannot be executed. Modules are very similar to programs, allowing you to declare variables, constants, data types, procedures and functions. What's missing? Program statements. That's right. You can have procedure statements and/or function statements but no program statements. This is why you cannot execute a module. Generally, modules contain a group of procedures and functions that are related to one another in purpose.

Implementations of units serve a similar purpose as modules, however they are more complicated, flexible, powerful and naturally more confusing. Since what I wish to accomplish with this article can be easily handled with modules, I will not attempt to cover units. Honestly, I don't think I understand units well enough to explain them.

Why do I want to be bothered with modules? After all, I can include all my procedures and functions in the main program, which is a simple, easy, and straightforward process. However, as programs become larger, this becomes exceedingly cumbersome, and the use of modules becomes increasingly advantageous. Below are listed some of the reasons for using modules. There are undoubtedly others you could add. To me, these seem the most significant. This discussion applies equally well to any software language that supports subprograms, such as Fortran, COBOL, PL/1, Algol, assemblers, etc.

Modules make it possible for you to develop a library of procedures and functions. Let's say you have an application that requires use of the matrix processes of add, subtract, multiply, inverse, and transpose. These can be developed as Pascal procedures and compiled into a single module. From then on whenever you need to use matrix algebra routines, all you need to do is call these procedures and link the appropriate module to the executable program. You never need to go through the joy of writing and agony of debugging the matrix algebra routines again. As you can imagine, this is a tremendous time saver. The programmer can develop a library of modules containing frequently used calculations or procedures, such as statistical routines, math routines, graphics routines, cursor control routines, etc. Once a sufficiently complete library is created, the development of new programs approaches the simple matter of calling the appropriate procedures from the library. There are even commercially available libraries, particularly in the area of graphics.

Enough about libraries. Another advantage of modules is the way they make it possible to create large structured programs. This is accomplished by breaking the program into parts. In actuality, large programs are developed in logical parts. This is partly the result of the wide acceptance of structured programming techniques in the last ten years and of course the development and use of structured languages such as Pascal. The power of using structured constructs (do while, case, if-then-else, repeat until, etc.) is even creeping into such languages as BASIC (HP 9000 BASIC) and Fortran 77 (the if-then-else construct).

The use of structured design techniques to create a large program, by its very nature, results in breaking the program into logical parts (at least it does for me). For example, a data base system may be composed of the following parts: data input, data edit, file update, file search, report writer, file backup, data field definitions, query language, recovery mechanism, etc. Each of these tasks could be implemented in a module. Within each module would be the procedures and functions necessary to accomplish that task. If the task is sufficiently complex, several modules might be linked together to accomplish it. These modules would then form a library for the program's code.

Modules, modules, modules! They're wonderful! What else are they good for? Besides what I have already mentioned, they ease the problem of code testing and debugging, whether the testing is top-down (starting with the main program and testing each module from highest level to lowest level) or bottom-up (starting with the lowest level module or procedure and working up to the main program). In either event, modules can be thoroughly tested by passing as many different data configurations to them as possible or practical. Once the module and each procedure and function in it is thoroughly tested it can be compiled and placed in the program library. Theoretically, from this point on, you will not need to be bothered with this module again since it has been thoroughly debugged and tested. This really is a plus, since you will not need to recompile this code during the remainder of the software development. No longer will it be necessary to look through long compilation listings in which 90% of the code has already been debugged. Also, you save a tremendous amount of compilation time since the only portion of code you are compiling is the module you are currently debugging. I once had a friend tell me he thought structured modular code was great because he didn't have the patience to wait for more than 50 to 100 lines of compilation listing print at a time. I realize there are other ways of cutting down on the length of the compilation listing but with modular code they are not needed.

Finally, it is worthwhile establishing a module for the sole purpose of containing hardware or compiler specific code. This eases the burden of moving the code to another computer or using a different compiler, since all the code that needs to be converted is in one place.

In summary, modules are needed because they enable you to build a code library; they support structured code; they ease the burden of code test and debug; they cut down on the compilation and listing time; and they are useful for isolating code.

That's enough about the wonders of modules. Now I'd like to demonstrate their use on the Z-100 Computer using the Z-DOS Operating System and Microsoft's MS-Pascal Compiler. The Calling Card Program was written as an example of using modules. It is a simple program, but at the same time complex enough to show the utility of using modules. The program solicits calling card information from the user via a form displayed on the screen. These data (name, phone number, company, address, etc.) are stored in a file named by the user. Once data entry is complete, all the information input is displayed on the screen, one calling card at a time, for the user's review. As you can see, this program is nothing to write home about.

The Calling Card Program has been written using one main program and six modules. The program and each module are individually compiled using the MS-Pascal Compiler and then linked together to form an executable program using the Z-DOS Link Program. The process of compiling will not be discussed here since it is clearly covered in the MS-Pascal User's Guide. I will touch upon linking later; however, it too is covered in the User's Guide.

The main program and its modules pass data to each other through the use of procedure parameters. This, of course, is nothing new. Procedures normally pass data to one another via parameters and we always have to link our programs after we compile them. What we have here is a slight variation on a familiar theme.

The following table identifies the disk file name, the module name, and the associated procedure names used in this example:

Table 1

Z-DOS File Name	**Module Name**	**Procedure Name**
CALLING.PAS (Listing 1)	(Main Program)	
MODINIT.PAS (Listing 2)	Mod__Initialize	Initialize
MODCDIN.PAS (Lisitng 3)	Mod__Card__Input	Get__Name Card__Input
MODCDOUT.PAS (Listing 4)	Mod__Card__Output	Card__Output
LIBESC.PAS (Listing 5)	Libesc	Clear Locate Color__Monitor Color
MODFORM.PAS (Listing 6)	Mod__Form	Dash Dashes Form
MODFILE.PAS (Listing 7)	Mod__File	Open__Cards Save__Card Get__Card

Listing 1 contains the main program, named Calling. It defines one variable and calls three procedures, one for each of the major functions of the program. The variable 'cmon' is set in the Procedure Initialize and is passed to all the other procedures. Its purpose is to identify if a color monitor is being used. Color adds a little pizazz to any program. The three procedures called are not defined in the main program; only the procedure heading is given followed by the MS-Pascal directive EXTERN. The EXTERN directive is the key to using modules. EXTERN is short for external and it informs the compiler that the actual procedure or function exists in some other module. I find it useful when defining external procedures to include a trailing comment identifying in which module the procedure exists. This is for my own edification and it is not used by the compiler or link programs. As you can see, the Calling program by itself is rather trivial. All the work is done in the three procedures and I chose to put each of these procedures in its own module.

The following is a brief explanation of what each of these procedures does. You probably know more about writing Pascal code than I, so I won't bore you with a line by line explanation of the code. I will, however, point out why I organized the program as I did and the benefits of using modules. Procedure Initialize sets variable 'cmon' to true if a color monitor is being used, prints some general user instructions, and initializes the file named by the user to hold the calling card data. Procedure Card__Input prints the input form on the screen and solicits the user for input. Once the form is filled in with data the information is written to the file. Then, the form is cleared and the process is repeated until all calling cards are entered. Procedure Card__Output then prints the input data on the screen one calling card at a time for the user's review.

Having the Calling Program organized into three procedures, each in its own module, allows me to write the main program first by including all dummy procedure calls. Once the main routine is debugged, I can write the Initialize procedure (in module Mod__Initialize) and debug it independent of the rest of the program, replace the dummy procedure call in the main program, link, and test the system. I then can write the next procedure using the same process and continue in like manner until the program is complete. I find this approach to writing code shortens my debug time.

Before I discuss in greater detail the three procedures called by the main program, I need to explain the remaining three modules. Remember we have six modules total: three are used for the major portions of the main program and the other three support these modules. Module 'Libesc' contains Procedures Clear, Locate, Color_Monitor, and Color (see Listing 5). These procedures form a library of screen manipulation routines that are implemented through the use of escape sequences (see the Z-100 Technical Manual, TM-100 pages 10-38 – 10-51). For brevity's sake, I have included and use only a few of the screen manipulation procedures in this example. I use the 'Libesc' module in almost every program I write. It provides an excellent example of a procedure library. Module 'Mod_Form' contains Procedures Form, Dash, and Dashes (see Listing 6). These are used to place the calling card input form on the screen. Finally, Module 'Mod_File' contains Procedures Open_Cards, Save_Card, and Get_Card (see Listing 7). These procedures do all the disk I/O. For reasons unknown even to me, I prefer to have all the file handling routines in one module. Maybe it makes for cleaner code or it's just my style.

Let's take a closer look at the three procedures that make up the Calling Program.

Procedure Initialize is in Module 'Mod_Initialize' (see Listing 2). As you can see, Procedure Initialize also references procedures in Modules 'Libesc' and 'Mod_File'. Note, those procedures defined in other modules all have the directive EXTERN following their heading. Again, this informs the compiler that the procedure is contained in another module and not to worry about the missing definitions. It is the Link Program's job to tie this all together.

You have probably already noticed that one of the procedures I have included in 'Mod_Initialize' is Mod_File. But Mod_File is a module name, not a procedure name. What gives? As you may recall, I prefer putting all my file I/O routines in one module, in this case Module 'Mod_File', and 'Mod_File' declares the file variable 'cards', which is used to store the calling card data. MS-Pascal has a rule regarding modules and file variables. If a module declares a file variable, it must be initialized prior to its use. You are warned of this when you compile the Mod_File module. The compiler prints the following warning: "Warning 364 — Contains File Initialize Module." This is ok. All you need to do is call the module that contains the file variable (Mod-File) as a parameterless EXTERN procedure before any procedure or function in the module is referenced. This is what is happening in the first procedure statement of the Initialize Procedure. This is the only quirk I have found in using modules in MS-Pascal.

Procedure Card_Input is in Module 'Mod_Card_Input' (see Listing 3). Here again, Procedure Card_Input references procedures in Modules 'Libesc', 'Mod_Form', and 'Mod_File'. These procedure headings are all followed with the directive EXTERN. In Mod_Card_Input the calling_card record is defined with its eight fields, and variable 'card' of type calling_card is defined. Once the calling card data have been entered by the user they are passed to Procedure Save_Card via variable 'card'. Procedure Save_Card stores the record in the cards file. Thus

Listing 1

```
program Calling (input, output),

{This program demonstrates the use of modules  Each of the three
procedures called are in independent modules and each of these
modules calls other modules

The calling program prompts the user for calling card input, stores
this input in a file and then prints all entries on the screen.}

var cmon. boolean; {cmon is true when color memory is availble }

procedure initialize (var cmon: boolean); extern; {mod_initialize}
procedure card_input (cmon: boolean); extern;      {mod_card_input}
procedure card_output(cmon: boolean); extern;      {mod_card_output}

  begin
  initialize (cmon); {Set cmon, print messages and initialize file.}
  card_input (cmon); {Prompt user for calling card input.}
  card_output(cmon)  {Print calling card records.}
  end  {program}
```

Listing 2

```
module Mod_Initialize;

{The initialize module determines if a color monitor is in use,
prints some general messages to the user and opens the calling
card file.}

procedure color_monitor (var cmon: boolean); extern; {libesc}
procedure clear; extern, {libesc}
procedure color (fore,back: char); extern, {libesc}
procedure locate (row, col: integer); extern; {libesc}
procedure open_cards; extern; {mod_file}
procedure mod_file; extern; {mod_file}

procedure initialize (var cmon. boolean);
  begin
  mod_file, {Initialize the mod_file module since it declares a FILE
           variable   See Section 16.2 of Pascal Reference Manual.}

  color_monitor(cmon); {Cmon is set true if color monitor is in use }
  if cmon then color('7','1');
  clear; {Clear the screen.}
  locate(2,4); write('General Comments:'); locate(5,4);
  write('This program demonstrates the use of modules   You will be'),
  locate(6,4);
  write('prompted to input calling card data on a screen form. This');
  locate(7,4);
  write('data is then stored in a disk file   The input cycle is '),
  locate(8,4);
  write('terminated by pressing return in response to the NAME:');
  locate(9,4);
  write('prompt   Then, the data is printed on the screen.');

  if cmon then color('4','1');
  open_cards {Open the cards file.}
  end; {procedure}
end  {module}
```

data are transferred from one module to another using normal procedure parameters. We return to the main program if the user enters only a carriage return when prompted for the next calling card name.

Procedure Card_Output is in Module 'Mod_Card_Output' (see Listing 4). As we have seen before, all the external procedures are declared; then Procedure Card_Output is defined. Record type calling_card is once again defined and variable 'card' of type calling_card is declared. Data are retrieved from the file via Procedure Get_Card and put in variable 'card'. When all the data have been read from the cards file, variable 'done' is set to true and we return to the main program. That's all there is to it.

Once all the procedures and modules are written and compiled they must be linked together into an executable program. This is accomplished by using the Z-DOS Link Program. You basically use the Link Program as you have in the past, with one difference. When prompted with 'Object Modules [.OBJ]' you enter not only the program name but also all the module names separated by plus signs. Assuming you use the same file names I have chosen in Table 1 above, you would enter the following:

```
CALLING+MODINIT+MODCDIN+MODCDOUT+LIBESC+MODFORM+MODFILE
```

The main program must be the first name entered and the remaining modules may be entered in any order. Then you respond to the other Link Program prompts as always.

You have now created an executable program. All you need to do is execute it by typing in its name.

I hope this article has piqued your curiosity about modules and is informative enough to allow you to begin using them in your Pascal programs. Remember, modules allow you to build code libraries; they support structured code and system design; they ease the boredom of testing and debugging; they cut down on compilation and listing time, and they allow you to isolate hardware and compiler dependent code.

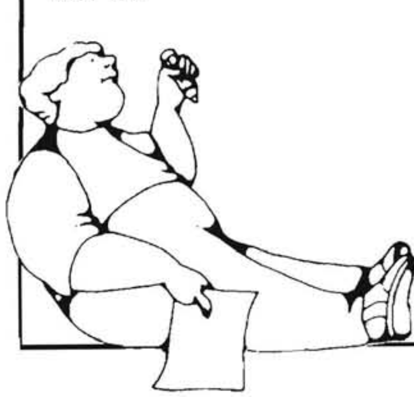
Have You Ever Asked?

1. How can I retrieve programs from the HUG Bulletin Board?

2. How can I tell what programs to choose from the hundreds available?

3. How can I get a quick reference to the commands available on the HUG Bulletin Board?

4. How can I communicate with other HUG members that happen to be on-line on the HUG Bulletin Board with me?



Get The HUG
Bulletin Board
Handbook
(PN 885-4700)

It's $5.00 from the
Heath Parts Dept.

Listing 3

```
module mod_card_input;

{This module prompts the user for calling card input and stores it
in a disk file.  Input is terminated by entering a null name.}

type calling_card = record
                    name: lstring(3Ø);
                    office: lstring(14),
                    home: lstring(14);
                    company. lstring(5Ø),
                    street: lstring(3Ø);
                    city: lstring(2Ø);
                    state: lstring(14);
                    zip: lstring(9);
                    end; {record}

var card  calling_card, {one calling card record.}

procedure form (cmon: boolean), extern; {mod_form}
procedure dashes (cmon: boolean); extern; {mod_form}
procedure locate (row, col: integer), extern, {libesc}
procedure clear; extern; {libesc}
procedure color (fore, back. char), extern, {libesc}
procedure save_card (var card: calling_card); extern; {mod_file}

procedure get_name (cmon: boolean);
  begin
  dashes (cmon); {Clear the form by print dashes.}
  if cmon then color('2','1'),
  locate(6,19); readln(card.name)
  end; {procedure}

procedure card_input (cmon. boolean),
  begin
  clear, {Clear the screen.}
  form(cmon); {Print the input form on the screen.}
  locate(7,14); write('To terminate calling card input, press return ');
  with card do
    begin
      get_name(cmon), {retrieve name}
      while name<>null do {End of input signaled by carrage return}
        begin
        locate(9,29); readln(office);
       locate(9,52), readln(home);
        locate(11,22); readln(company);
        locate(13,21); readln(street);
        locate(15,19); readln(city);
        locate(15,49); readln(state),
        locate(17,18); readln(zip);
        save_card(card); {store record in file}
        get_name(cmon) {get name for next calling card}
        end; {while}
    end; {with}
    if cmon then color('7','1');
    locate(22,1Ø); write('Input complete ')
  end; {procedure}
end. {module}
```

Listing 4

```
module mod_card_output;

{This module prints the calling cards on the screen.}

type calling_card = record
                    name. lstring(30),
                    office: lstring(14);
                    home: lstring(14);
                    company: lstring(50);
                    street: lstring(30);
                    city: lstring(20),
                    state: lstring(14);
                    zip: lstring(9);
                    end;

var card: calling_card, {One calling card record.}
    done. boolean, {set false after last record is read}

procedure form (cmon: boolean); extern; {mod_form}
procedure dashes (cmon: boolean); extern; {mod_form}
procedure locate (row, col: integer); extern; {libesc}
procedure clear; extern; {libesc}
procedure color (fore, back: char), extern; {libesc}
procedure get_card (var card: calling_card; var done: boolean); extern;
          {mod_file}

procedure card_output(cmon: boolean);
  begin
  done := false;
  locate(23,10); write('Press return to display first record.'); readln;
  clear; form(cmon); {clear screen and print form}
  with card do begin
    get_card(card, done); {get first record from the file}
    while not done do
      begin
      dashes(cmon); {print dashes on form}
      if cmon then color('2','1');
      locate(6,19); write(name);
      locate(9,29); write(office);
      locate(9,52); write(home);
      locate(11,22); write(company);
      locate(13,21); write(street);
      locate(15,19); write(city);
      locate(15,49); write(state);
      locate(17,18); write(zip);
      if cmon then color('6','1'),
      locate(23,10); write('Press return for next record '); readln,
      get_card(card, done) {get next record from file}
      end; {while}
    if cmon then color('7','1');
    locate(23,10); write('All Done.                      ')
  end; {with}
end; {procedure}
end. {module}
```

Listing 5

```
module Libesc;

{Libesc module contains procedures to manipulate the screen via escape
sequences  These sequences are issued to ZDOS via DOSXQQ function.}

  var x: byte;

function DOSXQQ(COMMAND,PARAMETER:word):byte;extern;
{Reference: Pascal Compiler User's Guide, page 119}

{Escape Code References:  Technical Manual (TM-100) pages 10-38, 10-51}

procedure CLEAR, {ESC E - clears the screen}

  begin
  x:=DOSXQQ(2,27);            { ESC }
  x:=DOSXQQ(2,wrd('E'))       {  E  }
  end; {procedure}

procedure LOCATE(ROW, COL: integer);
  { Moves the cursor to location ROW and COL on the screen }
  {ESC Y ROW COL - Direct Cursor Addressing}

  begin
  x:=DOSXQQ(2,27);            { ESC }
  x:=DOSXQQ(2,wrd('Y'));      {  Y  }
  x:=DOSXQQ(2,31+wrd(ROW)),   { ROW }
  x:=DOSXQQ(2,31+wrd(COL))    { COL }
  end; {procedure}

procedure COLOR_MONITOR (var CMON: boolean);
  {ESC i 0 - Identify Zenith Terminal Type}

  begin
  x:=DOSXQQ(2,27);          { ESC }
  x:=DOSXQQ(2,wrd('i'));    {  i  }
  x:=DOSXQQ(2,wrd('0'));    {  0  }
  x := DOSXQQ(6,255); x := DOSXQQ(6,255); x := DOSXQQ(6,255);   { ESC i E }
  if chr(ord(DOSXQQ(6,255)))='3'  {Number of banks of color memory }
    then cmon:=true
    else cmon:=false;
  x:=DOSXQQ(6,255); {The amount of video ram.}
  end; {procedure}

procedure COLOR(FORE,BACK: char);
  { Change the foreground and background colors. }
  { ESC m FORE BACK }
  {colors: black='0', blue='1', red='2', magenta='3', green='4',
   cyan='5', yellow='6', and white='7'}

  begin
  x := DOSXQQ(2,27); x := DOSXQQ(2,wrd('m'));           { ESC m     }
  x := DOSXQQ(2,wrd(FORE)); x := DOSXQQ(2,wrd(BACK));   { FORE BACK }
  end; {procedure}
end  {library module}
```

Listing 6

```
module mod_form,

{This module contains procedures to print the screen form and dashes.}

procedure locate (row, col: integer); extern; {libesc}
procedure color (fore, back: char); extern, {libesc}

procedure dash (count: integer);
  var i. integer;
  begin
  for i:=1 to count do write('_')
  end: {procedure}

procedure dashes (cmon· boolean),
  {Prints dashes on the screen form.}
  begin
  if cmon then color('7','1'),
  locate(6,19), dash(3Ø);  locate(9,29), dash(14);
  locate(9,52); dash(14);  locate(11,22), dash(5Ø),
  locate(13,21); dash(3Ø); locate(15,19); dash(2Ø);
  locate(15,49); dash(14), locate(17,18), dash(9)
  end; {procedure}

procedure form (cmon. boolean),
  {Print screen form.}
  begin
  if cmon then color('7','1');
  locate(3,2Ø); write('C A L L I N G     C A R D     D A T A');
  locate(6,14), write('NAME·'),
  locate(9,14); write('PHONE - OFFICE:');
  locate(9,47), write('HOME:');
  locate(11,14); write('COMPANY.'),
  locate(13,14), write('STREET·');
  locate(15,14); write('CITY:');
  locate(15,43), write('STATE·');
  locate(17,14); write('ZIP:')
  end, {procedure}
end  {module}
```

Listing 7

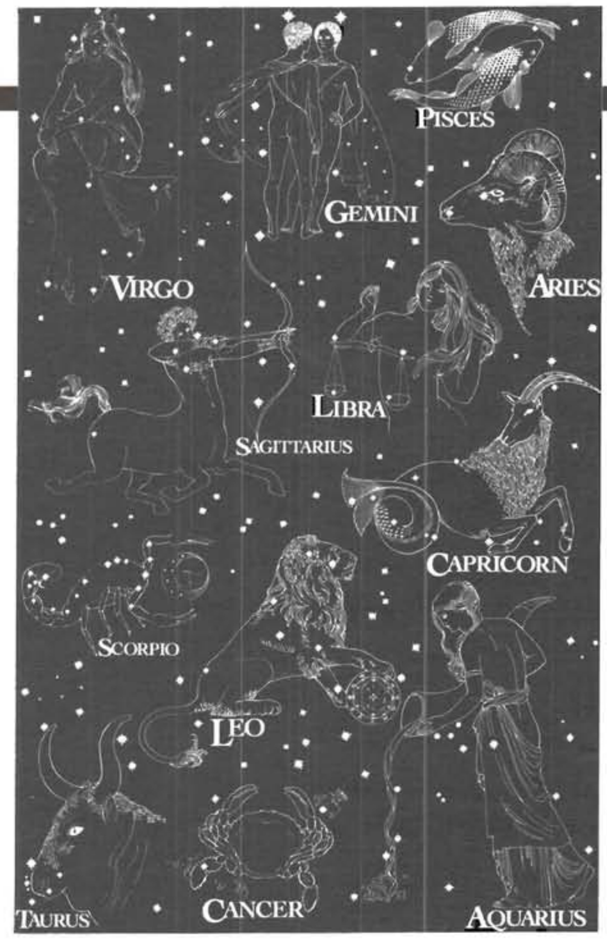
```
module mod_file;

{This module does all the file handling for the calling program.}

type calling_card = record
                    name. lstring(3Ø),
                    office. lstring(14);
                    home· lstring(14);
                    company: lstring(5Ø),
                    street· lstring(3Ø);
                    city. lstring(2Ø),
                    state· lstring(14),
                    zip: lstring(9)
                    end; {record}

var cards· file of calling_card;

procedure open_cards;
  {Queries for the file name, opens the file and rewrites the file.}

  begin
  writeln; writeln, write('      Enter calling card file name.'),
  readfn(input, cards); readln,
  rewrite(cards);
  end; {procedure}

procedure save_card (var card. calling_card),
  {Saves a calling card record in the file.}

  begin
  cards^ := card,
  put(cards);
  end; {procedure}

procedure get_card (var card. calling_card, var done. boolean),
  {Gets records from the file  'Done' is set to true at eof }

  var first[static]· boolean, {Set true for first call and set false
                               on subsequent calls.}
  value first := true;

  begin
  if first
    then begin
      first .= false,
      reset(cards)
      end {then}
    else get(cards);

  if eof(cards)
    then done:=true
    else card := cards^
  end; {procedure}
end {module}
```

*

TELLSTAR

Window To The Universe

Ralph F. Rumpf
6036 Legion Rd.
Stevensville, Michigan 49085



As the time rapidly approaches for another visit from the most notorious rogue in our solar system, heads begin to turn skyward and again ponder the wonders of the universe.

Astronomers make a living at searching the heavens and wondering about what goes on out there and why things tick. Most of the rest of us, with the possible exception of avid amateur astronomers, only gaze up when something catches our eye . . . fireworks, the moon, or perhaps a visiting comet. Astronomy can be a rather engaging hobby that can easily provide a lifetime of enjoyment.

I have had something less than an amateur interest in astronomy since I was quite a bit younger. I've owned several telescopes, nothing fancy, but enough to get the imagination going. Many a night I've spent either swatting bugs or freezing as I gaze at the heavens. My poor wife thinks I'm something of a kook, but what does she know? She's the one who bought my last telescope as a Christmas present!

One of the biggest problems for anyone who tends to stare at the sky more often than once in a blue moon is to determine what is up there and where exactly is it? This becomes even more important when you are using a telescope and can see things that the unaided eye cannot. Well, they say that you can't tell the stars without a program so, maybe I have a program for you.

TELLSTAR is an educationally based astronomy program that runs on a number of computers, from Apple to Zenith, and those blue things in between. The intent of the program is to make the science of Astronomy something that almost anyone can grasp and enjoy. To do this the program provides a number of options and functions that make a natural enjoyment of the heavens even more enjoyable.

TELLSTAR provides both horizontal and overhead views of the sky and can plot the various constellations that are visible. The program also has functions that can be used to identify various plotted objects, set the viewing location and time of day, locate an object in the tables, and print the screen contents to a graphics printer.

The IBM compatible version of this program comes in three flavors. The basic version plots objects in the northern hemisphere only. The level II version plots objects in the southern hemisphere, as well as the northern. The 8087 enhanced version has the same functions as the level II program, but adds 8087 Numeric Co-processor support that greatly speeds up star table and other associated calculations. I don't think that the 8087 version is really that necessary from what I have seen of the program. I could only justify the increased cost if I were doing a great number of star views, since that would entail a lot of re-calculations.

The version of the program that I have is for the northern hemisphere only. There are approximately 240 different objects that are computed and plotted for each view of the heavens, included in this count are the sun, moon and the planets in our solar system. To give you an idea of the way the program works, let's try to imagine a sample session.

It's early fall and an exceptionally clear night is calling you to star gaze. Rather than just stare at the stars, you decide to take a more involved look and see what you can see. You load up TELLSTAR and provide your position and the time that you will be wandering the heavens. TELLSTAR requires your longitude and latitude for calculating your location, but once you have done that you can make it a permanent entry. Once you have provided the necessary information, TELLSTAR performs the required calculations and is ready to display the sky. One distinct view is provided for each compass direction, plus the overhead view. The overhead view is provided because of distortion in the normal views above a 60 degree elevation. Now you can see what there is to see.

You can scan around the compass views to see what will be visible in your location. Once you have found something, you can use the program's Identify function to find out what it is and it's exact location. This feature is very handy if you use a telescope because it provides the exact position to set the telescope, too. Next, to make locating the object easier, you plot the constellations in the area, then, so you have a handy reference, you plot the whole display to your graphics printer for a hardcopy. You can do the same thing for a future or past date and save the view in a file on disk. If you are interested in looking at a specific object, you can request it directly from the tables to speed up your preparation time.

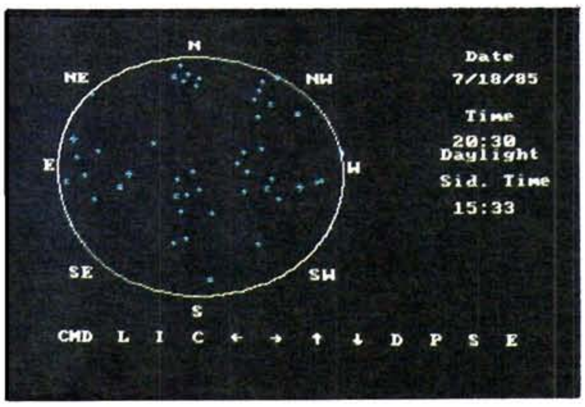
TELLSTAR is not the same as having your own observatory and dedicated computer system, but for what it does do, it does a good job. If you have any interest in the sky or Astronomy, you might give this program a try. Even if you aren't interested, once you start looking you might change your mind. There is more than enough in this package to provide hours of enjoyment for you and your whole family.

TELLSTAR is available in it's various versions from:

SPECTRUM HOLOBYTE
1050 Walnut, Suite 325
Boulder, CO 80302
800-621-8385 Ext. 262

The price for the various versions are:

TELLSTAR I	$ 49.95
TELLSTAR II	79.95
TELLSTAR II/8087	129.95



Overhead View — Date, local time and sidereal time noted.

TIRED OF WAITING, WAITING, WAITING, ON YOUR COMPUTER ? SO, SPEED IT UP

INTRODUCING

THE TIME WORKS'

CPDZ-100 & CPDZ-150

INSTALLS IN JUST MINUTES WITH NO PERMANENT MODIFICATIONS TO YOUR EXPENSIVE MAIN BOARD. A SMALL SWITCH, MOUNTED ON A PANEL FITTED ON THE REAR OF YOUR COMPUTER, ALLOWS THE USER TO CHANGE THE 8088'S SPEED BY 40 % . . FASTER ! YOU'LL SEE THE DIFFERENCE !

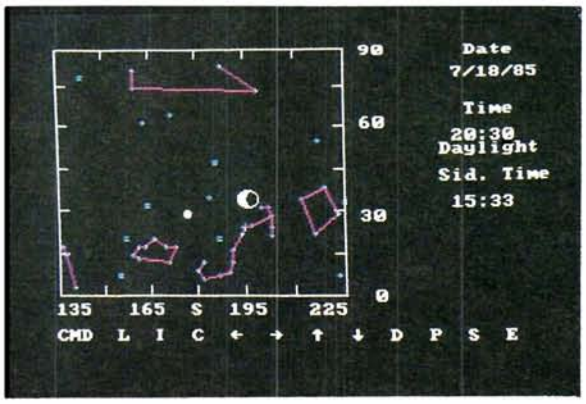
VERSIONS FOR ALL H/Z - 100'S & THE H/Z - 150.

$34 95 PLUS $2 .00 FOR SHIPPING AND HANDLING.

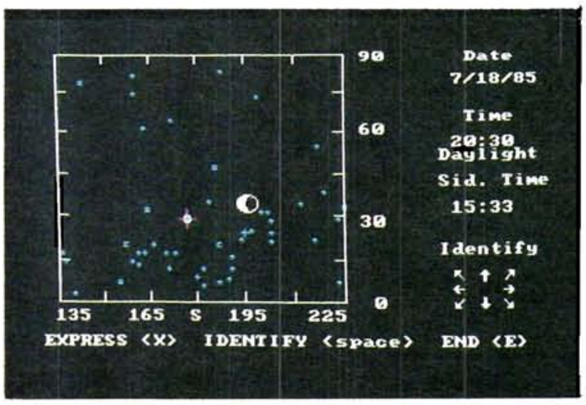
SEND CHECK, MONEY ORDER TO >

CPD
CONSUMER PRODUCT DEVELOPMENT

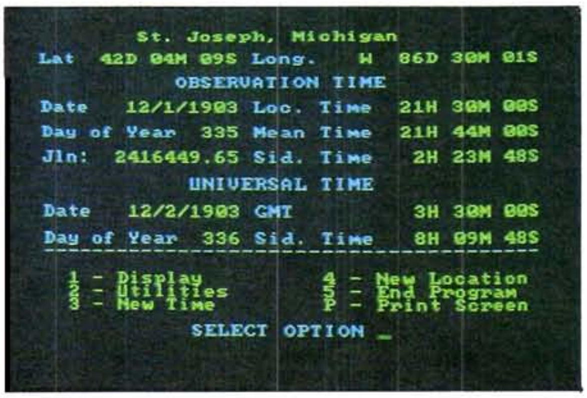
THE
TIME WORKS
6242 B RUMFORD LN
HOUSTON, TX 77084



Horizontal View facing South with constellations highlighted.

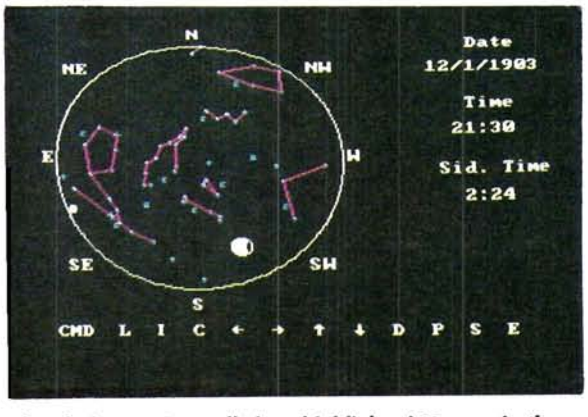


Horizontal View facing South requesting identification for planet in view area.



```
              St. Joseph, Michigan
Lat  42D 04M 09S Long.    W  86D 30M 01S
             OBSERVATION TIME
Date    12/1/1903 Loc. Time  21H 30M 00S
Day of Year  335 Mean Time  21H 44M 00S
Jln:  2416449.65 Sid. Time   2H 23M 48S
              UNIVERSAL TIME
Date    12/2/1903 GMT         3H 30M 00S
Day of Year  336 Sid. Time    8H 09M 48S
------------------------------------------
  1 - Display            4 - New Location
  2 - Utilities          5 - End Program
  3 - New Time           P - Print Screen
             SELECT OPTION _
```

Statistical Display describing current view location, date and time. NOTE: We are looking back 82 years.



Overhead View — Constellations highlighted 82 years in the past.

Continued from Page 9

cided to see just how hard it would be to expand the memory myself. As it turns out, this modification is not only simple, but doesn't entail any irreversible damage to the motherboard.

I started with the schematic for the WH88-16 16k expansion. I expected to find some kind of address decoding circuitry, but such was not the case. It seems the required address line for the extra bank of chips is already available on the motherboard, at pin 17 of the left-side expansion connectors. This being the case, all that was necessary was to obtain 8-4116 RAM chips, bend pin 4 of each chip out to the side, and piggy-back them on 8 existing 4116s. I did this outside the computer, because I didn't want to risk damaging the motherboard. Just solder each pair of chips together (all except pin 4, being careful to orient them properly). Then install the chips on the motherboard and solder a jumper wire connecting the pin 4s together. Finally, connect this jumper to pin 17 of P509. I soldered this connection, but if you don't want to solder a wire to your motherboard, you can fashion some sort of plug-on connector. From here on out it's all in the configuration guide, basically just change one jumper plug (JJ501) from 0 to 1.

The modification has worked great since I first powered it up, and the only risk involved was to the memory chips themselves, which are quite inexpensive now. Right now I'm looking into the possibility of a 128k (or more) modification, but that will undoubtably be more involved. Good luck.

Sincerely,

Chuck Hatcher
Buckley Hill Road
Colchester, CT 06415

Looking To Run Z80 Code On 8-bit Side

Dear HUG:

In response to Barry Watzman's recent advertisement (REMark May 85) for Perks (his Sidekick work-alike for the Z-100). I ordered a copy and am happy to report it is well worth a hundred bucks. No need to review it. His full color ad does a pretty good job of indicating its features. The only thing we don't know is what programs it isn't compatible with (e.g. I've heard that ProKey doesn't work with SideKick. That's supposedly why Borland wrote SuperKey).

That said, I have a couple of problems you or your readers might be able to help me with.

First, ever since my office took delivery of our first Zenith Z-100 about a year ago, I've been looking for a way for it to run Z80 code on the 8-bit side. In one of the last issues of Microsystems, there was a great "how to" on replacing the 8085 on Godbout's dual processor board with a National Semiconductor NSC-800 chip. For those who don't know, the NSC-800 executes Z80 code while preserving the Intel 808x multiplexed address/data bus. I don't think I can use this "how to" on the Z-100 because its 8085 is clocked at 4.9 MHz, but the fastest NSC-800 I know about will only handle 4 MHz. I've written to National Semiconductor about a faster part, but received no reply. Have any readers heard of a 5 MHz NSC-800, and if so, where can I buy one? I've considered slowing the 8-bit system clock with a new crystal, but I think this will lower the 2661B serial port baud rates too — a clearly unacceptable side effect. Can anyone confirm this and/or suggest an alternative? It seems to me that an NSC-800 conversion for the Z-100 (like the 8087 co-processor addition, the RAM-PAL memory upgrade, the 8088 MHz speed-up, etc.) would be a sure-fire success for the garage entrepreneur. I offer the idea to anyone willing to market it, with a request to send me the first one!

Second, I'd like to add my name to the chorus of folks looking for an S-100 expansion interface for the Z-100. The three remaining slots in my hard disk system were filled or reserved even before I took delivery. A 10-12 slot expansion chassis, complete with a power supply and a pair of interface cards (similar to the S-100/IBM-PC bus interface described in the May/June 1985 issue of Micro/Systems Journal) should be marketable in the $500 price range. This would be a great Heathkit!

Sincerely,

Robert G. Savage
General Delivery
Scott AFB, IL 62225

A Better Way To Include Escape Codes In Pascal

Dear HUG:

I am submitting the following only because Paul W. Simmons in the article printed in REMark of May 1985 ask if there is a better way to include Escape Codes in Pascal programs.

I believe that there are at least two ways in Pascal.

My reading of the manuals that were supplied with my H-100 tells me that the Escape Codes are to be sent to the "terminal". That tells me that in BASIC, I should use the PRINT statement, in FOR-

---

ICS ZENITH 8 bit and 16 bit computer components at $ubstantial $avings!

	Price
OEM Replacement Internal 10MB Disk Drive	**CALL**
H89/90 CPU Swap	**$ 231.00**
44MB Internal Disk Drive Kit for your Z-150, includes 640K RAM	**$2995.00**
Take your Z-150 from 320K to 704K for (includes ZPal2 set and (18) 256K chips!)	**Only $ 225.00**
ZPal2 kit only	**$ 42.00**

Call for super prices on service repairs, parts and upgrade kits. Dealer inquiries invited.

CALL TODAY! **312/646-1511**

INTERNATIONAL COMPUTER SERVICE, inc.
13256 SOUTH BRANDON CHICAGO, ILLINOIS 60633

H/Z-25 Super Chip Set

- 4 Chip Set
- 12 New Features
- 256 Characters
- Double Strike
- Underline
- Italics
- more

$84.95

FINA Software
16144 Sunset Bl. #3
Pacific Palisades
California 90272
213/454-6393

TRAN I should use the WRITE statement, and in PASCAL, I should use the WRITE or WRITELN statement. The manual that was supplied with PASCAL provided the answer.

ASCII characters can be represented using the function CHR(?) where ? is the numeric value of the ASCII character.

Above I said that there are at least two ways in Pascal, they are:

I. Declare them as constants:
For example:

```
CONST  CLS=CHR(27)*'E';       {Clear screen}
       ETG=CHR(27)*'F';       {Enter graphics}
       EXG=CHR(27)*'G',       {Exit graphics}
```

In program use:

```
Write(CLS);
Write(ETG),
 etc
```

II. Include them in procedures:
For example:

```
procedure CLS,           {Clear screen}
    begin
    writeln(chr(27),'E');
    end,
procedure LOCATE(L_NO,C_NO:integer), {Locate screen char}
    begin
    write(chr(27),'Y',chr(L_NO+31),chr(C_NO+31);
    end,
```

In the program use:

```
CLS,              {Clear the screen}
LOCATE(20,48);  {put the cursor at line 20
                      and column 48}
```

I trust that this information has been helpful to the readers of REMark.

Frank L. Westerman, Jr.
638 6th Avenue West
East Northport, NY 11731

RESET-Z
ELECTRONIC RESET
FOR ZENITH PCs

- Resets computer without turning power off.
- Prevents possible hard disc damage.
- 5 minute screwdriver installation.
- Installs in rear panel.
- $29.95, includes shipping.

KC COMPUTERS
712 N. Citrus • Hollywood, CA 90038 • (213) 937-8127
For Z/H 150 & 160 computers only.
Personal checks — allow 2 wks. • CA residents — add $1.95 tax.

Number FORTRAN Statements On A Terminal?

Dear Chief HUGger:

After all of the arguments on the numbering of FORTRAN statements, I was sure that someone would come up with the old fashioned method of numbering fixed format FORTRAN statements. It all goes back to the days of steam powered computers when all FORTRAN statements were entered on unit record (IBM) cards. There was nothing more embarrassing than a dropped deck of cards which naturally were picked up in random order. If you'll look in your FORTRAN manual, you will find the following columns reserved:

1 to 5	Label field
6	Continuation field
7 to 72	Statement field
73 to 79	The identification field

In fact, in the old days you could use all of the eight right-hand columns for identification. Then, if you dropped a deck of cards, you could take them over to a device called a Sorter and sort them based on the eight right-hand columns.

So, if you want to number FORTRAN statements on a terminal, place the identification in columns 73 to 79. This ID can be any ASCII character.

Kenneth Mortimer
352 Green Acres Drive
Valparaiso, IN 46383

WatchWord and The Resident Speller

. . . Now Spelling Checking Has Been Integrated With Word Processing . . .

The Resident Speller can be used alone or as background for other programs. In WatchWord, the Resident Speller will check a paragraph or check text as you type. Display similar words or add words to alternate lists at the touch of a key.

For the H/Z100 with ZDOS or MSDOS
The Resident Speller: $100
WatchWord: $100
Demo Disk *(for both):* $3

Memory Requirements:
WatchWord alone: 192K
Resident Speller alone: 192K
WatchWord with Speller: 300K

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

ZCLK(tm) Losing Time Or Date

Dear HUG:

It appears that some purchasers of our ZCLK(tm) Clock/Calendar card for the Z-100 (sold as Heath Model PC-240) have had problems with the unit losing the time or date after power off periods.

All of the ZCLK's returned to us (so far) for repair have been fixed by replacing U1, a 74LS02 manufactured by Signetics, with the same part manufactured by any other company. Motorola and Fairchild 74LS02s seem to be the best replacement.

Replacement 74LS02s should be available through the local Heath stores, but we can provide replacement parts as well.

A copy of our ZCLK(tm) Tech Memo #1 which describes the problem in some detail, is available on request.

Sincerely,

Dave Brockman
FBE Research Company, Inc.
P.O. Box 68234
Seattle, WA 98168

Several Guidelines For Writing "Large" FORTRAN Programs

Dear HUG:

Through working with our Z-GRAPH-100 customers, we have learned about several weak areas in the MS FORTRAN compiler. More importantly, there appears to be several things one can do to avoid or compensate for these areas.

In one non-graphics program, we found that "large" can relate to the size and complexity of individual FORTRAN statements. In this case, we found that the program compiled and linked properly but produced strange and erratic results. The problem was a fairly complex mathematical statement which the compiler could not handle. The FORTRAN reference manual mentions that you can exceed the capabilities of the compiler in this area. However, one would assume that it would complain if its capabilities were exceeded. Clearly, such is not the case. The fix is to simply break the statement up into several more simple expressions.

In another case, a customer was moving a modest size (3500 lines) program from a CYBER to a Z100. The customer found that a number of routines which had been individually tested would not come together during the link phase. His calls to MicroSoft revealed that in "large" programs ALL common blocks must be declared in the main module. Block data statements also cannot be used in subroutines. In fact, he recommends that block data not be used at all. He now initializes all variables with assignment statements. His "large" program is about 168K. When he asked, MicroSoft could not tell him when a program was considered "large".

Finally, "large" FORTRAN programs using Z-GRAPH-100 graphics routines must use the ZGRAPH.OB file as the first module in the link line (as per the example on page 5 of the programmers manual). Since we don't know when a program becomes "large," we recommend that all programs be linked per the referenced example. This applies to all other languages, as well.

Hope these tidbits are of some help. Our best.

Fred Pospeschil
Micro-Doc
3108 Jackson Street
Bellevue, NE 68005

Several Comments Shared About Zenith

Dear HUG:

I very much enjoy REMark and look forward to it each month. Since I am not a Compuserve user — even though their headquarters and computers are about eight blocks from my home — your magazine is my only source of contact with the Heath/Zenith world of computer users.

I want to share several comments with your readers:

1. I agree with Clifford Coughlin of Upper Darby, PA (REMark, June 1985, p. 65) that the computer names are confusing for the new user. The difference between H/Z-100 (MS-DOS, but not IBM compatible) and H/Z-100 PC (MS-DOS and IBM compatible) is pretty subtle! I find that most users seem to be using Z-151 now for the latter computer. Like Mr. Coughlin, I

"C/80 . . . the best software buy in America!"

—MICROSYSTEMS

Now available in MS-DOS

Other technically respected publications like *Byte* and *Dr. Dobb's* have similar praise for **The Software Toolworks' $49.95** full featured **'C' compiler** for CP/M® and HDOS with:

- I/O redirection
- command line expansion
- execution trace and profile
- initializers
- Macro-80 compatability
- ROMable code
- and much more!

"We bought and evaluated over $1500 worth of 'C' compilers . . . C/80 is the one we use."

— Dr Bruce E. Wampler
Aspen Software
author of *"Grammatik"*

In reviews published worldwide the amazing **$49.95 C/80** from **The Software Toolworks** has consistently scored at or near the top — even when compared with compilers costing ten times as much!

The optional **C/80 MATHPAK** adds 32-bit floats and longs to the C/80 3.0 compiler. Includes I/O and transcendental function library all for only **$29.95!**

C/80 is only one of 41 great programs each **under sixty bucks.** Includes: LISP, Ratfor, assemblers and over 30 other CP/M® and MSDOS programs.

For your **free** catalog contact:

The Software Toolworks
15233 Ventura Blvd., Suite 1118,
Sherman Oaks, CA 91403 or call 818/986-4885 today!

CP/M is a registered trademark of Digital Research had my Z-133 RGB monitor repaired under warranty: they replaced the power supply and main circuit board. It still occasionally has problems with a narrowing display, etc. I expect the power supply to die again.

2. On Z-151 and IBM compatibility (W. Adney, "On The Leading Edge," REMark, June 1985, p. 55):

   a. The IBM AT has a different BIOS; indeed, the addresses of the serial ports have been changed, etc.

   b. I have three programs which run 100% on the IBM PC, but will not run on the Z-151:

      • First (and least important) the Sierra game Ulysses will not load on the Z-151. The computer simply refuses to recognize that there is a program on the disk. The Norton utilities will not read the disk either, but CopyWrite reports 96 bad sectors on the disk — I suspect this is the copy protection mode. I tried the same disk on the IBM PC, with no problem at all . . . works 100%.

      • Second, the University of Waterloo Pascal compiler — used extensively here at Ohio State — does not run on the Z-151. I am told that it looks for specific IBM ROM BIOS code which the Zenith does not have. Too bad, since this is a nice compiler, too. Runs 100% on True-Blue PCs.

      • Third, Ohio State has just finished writing a computer-assisted instruction package called Paragon. This has two parts: a "Student" program for instruction and an "Author" program for faculty. The Author program will load with a great deal of noise from the disk drives (no noise on the IBM!), but the Student program will not load at all. The Z-151 simply begins reading the disk, stops after about 3 seconds and is fully locked up. The system cannot be rebooted from the keyboard and the Zenith internal ROM cannot be accessed from the keyboard. I have loaded the "Diskwatch" utility program from PC magazine (June 11, 1985 issue); it reports "Disk error: Error Unknown" (disaster!) and "Disk Error: Illegal Sector Requested" when the program loads. Is the Zenith disk controller IC that different from the IBM? [This program loads the UCSD p-system under MS-DOS — odd but true.]

3. Occasionally, the system will "hang" when I'm using the ROM monitor programs. Twice it has frozen during the memory test when beginning the 4000: segment. The system cannot be rebooted from the keyboard nor from the hardware reset button I added; the power must be recycled. Immediately after that, I've rerun the same tests without a problem. Yesterday, the computer froze completely when I attempted to return to a program by issuing the "G" [go] command in response to the -> prompt. Usually, this works fine — but not always. Is there a bug in the Zenith ROM? This machine was shipped December 1984.

I will appreciate any answers from your readers or your technical staff.

Sincerely,

Richard G. Anderson
2606 Swansea Road
Columbus, OH 43221

recover from a

FATAL ERROR

in just sixty seconds with Rescue & Repair.

At last, there is a fast, simple way to recover data lost because of "Read Only" errors and operator mistakes. In fact, Rescue & Repair will allow you to retrieve your "lost" data after *any* error (except power failure or major technical malfunction), even if the machine has been warm booted or *reset*. Reviewers have been unanimous in their praise. **Interface Age** wrote *"The name says it all. . . The 66-page manual is excellent."* **H-SCOOP** *calls Rescue & Repair "a powerful and very useful program. . . It is extremely easy to use."* **Sextant** calls it *"a real life saver,"* and states that *"Rescue & Repair will recover files lost due to practically anything."* Rescue & Repair is available for MS, PC and Z-DOS* machines for $75 and most CP/M** machines for $55. To order, or for additional information contact:

21st CENTURY DATA, INC.

P.O. Box 1139
Solana Beach, CA
92075
(619) 755-6218
California residents add 6% sales tax.

Z DOS is a trademark of Zenith Data Systems
CP/M is a trademark of Digital Research Corp

Quick 'P'

H89 PARALLEL PRINTER PORT

> RUN THE SWEET-P PLOTTER WITH YOUR H89 <

> USE EPSON PARALLEL GRAPHICS PRINTERS <

$69.50

$4.00 Shipping, Calif. residents add 7% Sales Tax
Allow 4-6 weeks for delivery

** PLUGS DIRECTLY INTO CURRENT SERIAL CARD
** COMPATIBLE WITH ALL HEATH SERIAL CARDS
** QUICK AND CLEAN INSTALLATION
** NO BIOS OR OTHER SOFTWARE MODIFICATIONS
** WORKS WITH ANYBODY'S DOS
** USES THE SAME PRINTER CABLE AS THE H/Z150

Custom Circuit Design

ME

MCGAFFEY ENGINEERING
1556 Halford Avenue
Suite 169
Santa Clara, CA 95051
(408) 926-2132

Continued from Page 33

DATA BASE MANAGEMENT SYSTEMS

HDOS

Part Number	Description of Product	Selling Price	Vol. Issue
885-1107-[37]	HDOS Data Base System H8/H89	30.00	23
885-1108-[37]	HDOS MBASIC Data Base Sys.	30.00	23
885-1109-[37]	HDOS Retriever ASM (3 Disks)	40.00	23
885-1110	HDOS Autofile (2 Disks)	30.00	23
885-1115-[37]	HDOS Navigational Program	20.00	25
885-8008	Farm Accounting System	45.00	30

CP/M

Part Number	Description of Product	Selling Price	Vol. Issue
885-1219-[37]	CP/M Navigational Program	20.00	31

AMATEUR RADIO

HDOS

Part Number	Description of Product	Selling Price	Vol. Issue
885-8016	Morse Code Transceiver Ver 2.0	20.00	42

CP/M

Part Number	Description of Product	Selling Price	Vol. Issue
885-1214-[37]	CP/M MBASIC Log Book (64k)	30.00	23
885-1234-[37]	CP/M Ham Help	20.00	49
885-1238-[37]	CP/M Ascirity	20.00	57
885-8020-[37]	CP/M RF Comp. Aided Design	30.00	44
885-8031-[37]	CP/M Morse Code Transceiver	20.00	57

COMMUNICATION

HDOS

Part Number	Description of Product	Selling Price	Vol. Issue
885-1122-[37]	HDOS MicroNET Connection	16.00	37

CP/M

Part Number	Description of Product	Selling Price	Vol. Issue
885-1207-[37]	CP/M TERM & HTOC	20.00	26
885-1224-[37]	CP/M MicroNET Connection	16.00	37
885-3003-[37]	CP/M ZTERM (Z100 Modem Pkg)	20.00	34
885-5004-37	CP/M-86 TERM86 and DSKED	20.00	56
885-5005-37	CP/M-86 16 Bit MicroNET Conn.	16.00	61
885-5006-37	CP/M-86 HUGPBBS	40.00	62
885-5007-37	CP/M-86 HUGPBBS Source List.	60.00	62
885-8005	MAPLE (Modem Appl. Effector)	35.00	29
885-8012-[37]	CP/M MAPLE (Modem Program)	35.00	34
885-8023-37	CP/M-85 MAPLE	35.00	45

MSDOS H/Z100 - H/Z150 PC

Part Number	Description of Product	Selling Price	Vol. Issue
885-3019-37	ZDOS 16 Bit MicroNET Connect	16.00	61
885-3027-37	MSDOS HUG PBBS	40.00	66
885-3028-37	MSDOS HUG PBBS Source Listing	60.00	66

MISCELLANEOUS

Part Number	Description of Product	Selling Price	Vol. Issue
885-0004	HUG Binder	5.75	
885-1221-[37]	Watzman ROM Source Code/Doc	30.00	33
885-4001	REMark Vol. I Issues 1-13	20.00	
885-4002	REMark Vol. II Issues 14-23	20.00	
885-4003	REMark Vol. III Issues 24-35	20.00	
885-4004	REMark Vol. IV Issues 36-47	20.00	
885-4005	REMark Vol. V Issues 48-59	25.00	
885-4500	HUG Software Catalog	9.75	
885-4600	Watzman/HUG ROM	45.00	41
885-4700	HUG Bulletin Board Handbook	5.00	50
885-3015-37	ZDOS Skyviews	20.00	55

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.

*

Heath/Zenith Users' Group

Western Regional HUG Conference

- LATEST TECHNOLOGY
- NEW SOFTWARE
- HEATH/ZENITH VENDORS
- GUEST SPEAKERS
- PANEL DISCUSSIONS
- DINNER/PRIZE DRAWING

[Advance Registration Rates Good Thru October 15th]

NOVEMBER 15 - 17, 1985

DINNER/PRIZE DRAWING $25.00
WITHOUT DINNER $7.50

— WIN A COMPUTER —

CONFERENCE OPENS FRIDAY AFTERNOON, NOVEMBER 15TH AND CLOSES AT NOON ON SUNDAY, NOVEMBER 17TH

AT... Disneyland Hotel ANAHEIM, CA.

The Official Hotel At The Magic Kingdom

MAKE CHECKS PAYABLE TO: **WESTERN REGIONAL HUG** CONFERENCE
Mail To: • 1555 N. Orange Grove Avenue, Pomona, CA 91767

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

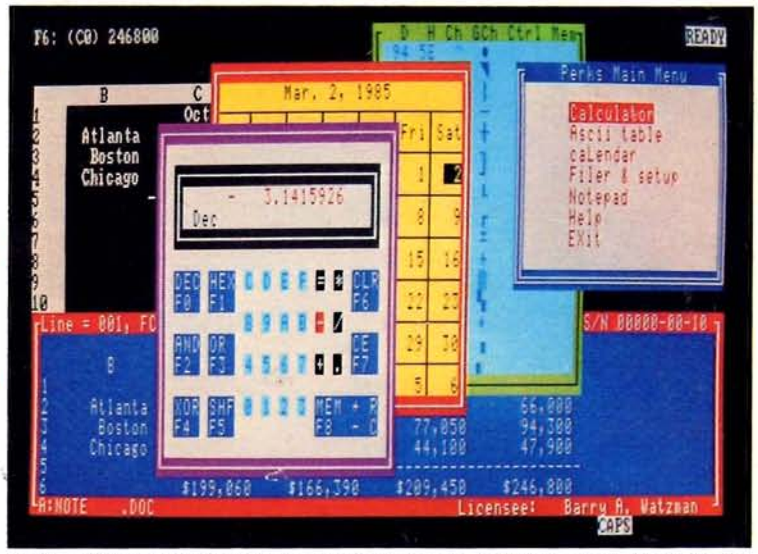
	NEW MEMBERSHIP RATES	RENEWAL RATES	
U.S. DOMESTIC	$20 ☐	$17 ☐	
FPO/APO & ALL OTHERS*	$35 ☐	$30 ☐	U.S. FUNDS

* Membership in France and Belgium is acquired through the local distributor at the prevailing rate.

If you have a Z-100, treat yourself to some

PERKS™

Perks is the background desktop utility program for Z-100's featuring full-color windows, a calculator, a notepad, an ASCII table, a Filer, a screen saver, alarm timers, context sensitive help screens and a calendar with an appointment schedule. Perks works with all Z-100's having 192K or more of memory and using the Z-DOS/MS-DOS 1.X or 2.X operating systems.



Shown above is an actual screen photo of Perks in operation. The Notepad window contains data "imported" from the Lotus 1-2-3 worksheet being prepared when Perks was activated.

Here are some actual unsolicited comments from customers:

"My heartiest congratulations and thanks for a fine product. I expect that Perks will be the best software dollar I have yet spent, and I have spent lots. Anyone with a Z-100 should run to buy Perks"

"I know I said this over the phone, however, once again I would like to thank you for a GREAT product."

"You're the best friend a Z-100 ever had! Mine loves Perks, and I want to thank you for the prompt filling of my order. Perks is everything and more than I thought it was going to be. The Notepad function alone is worth the price. And to think, I'd thought about buying a Programmer's Calculator from Hewlett Packard! Yours does everything I'd have used HP's for. Your Program is simply beautiful and elegant."

Perks is available at Heath/Zenith Computers & Electronics Centers, many independent Zenith Data Systems dealers or directly from:

BARRY A. WATZMAN Microcomputer Systems & Consulting

560 Sunset Road • Benton Harbor, Michigan 49022-7142 • (616) 925-3136

Heath/Zenith Users' Group

Hilltop Road
Saint Joseph, Michigan 49085

BULK RATE
U.S. Postage
PAID
Heath Users' Group

POSTMASTER: If undeliverable, please do not return.

P/N 885-2068