

REMark®

\$2.50

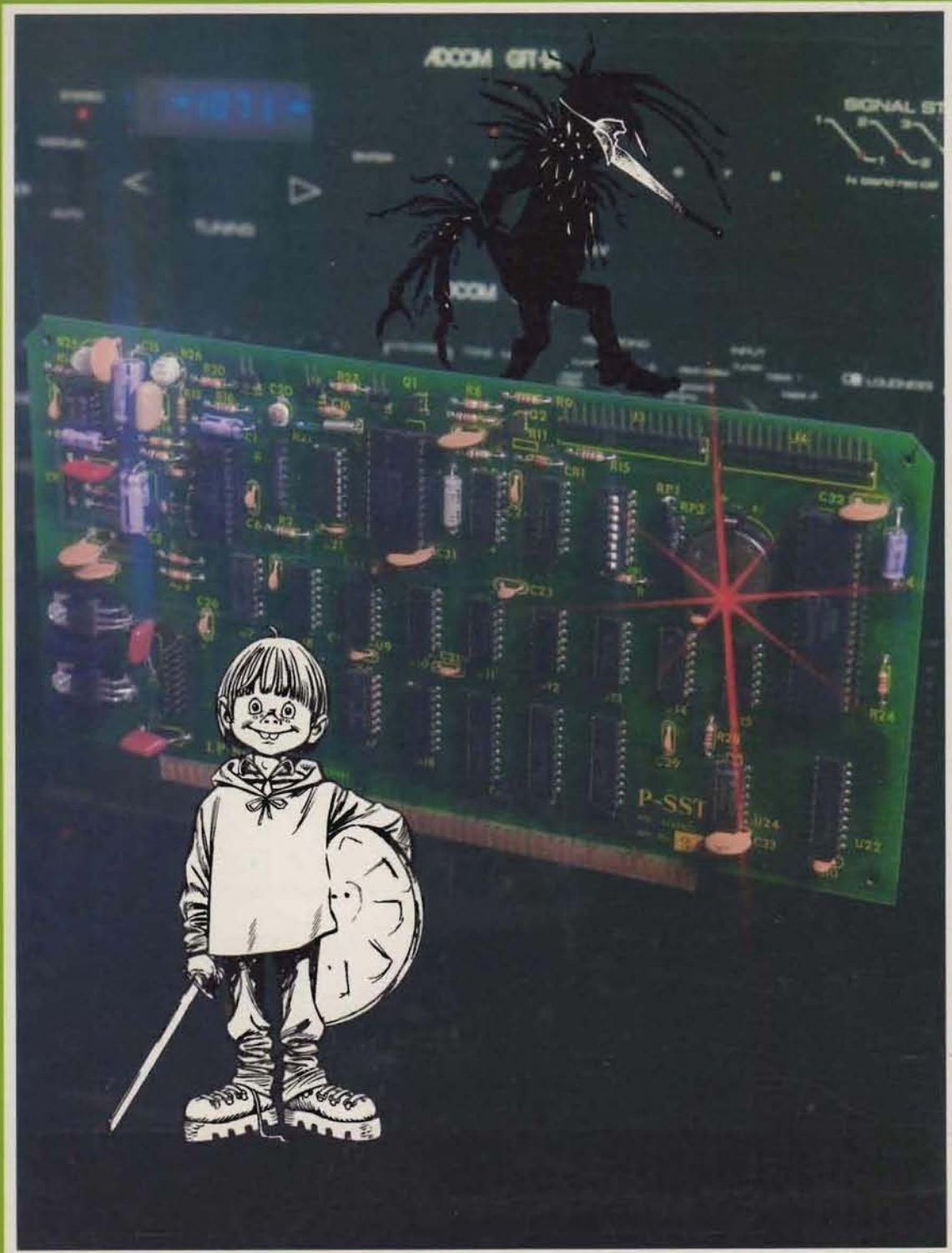
Volume 6, Issue 5 • May 1985

P/N 885-2064 Issue 64

Official magazine for users of



computer equipment.



RESOLUTION, SPEED, AND SILENCE

Superior Graphics and A Pseudo Disk for \$495.

That's right, silence! Because when you plug the Interactive Graphics Controller/Pseudo Disk into your H/Z89 or H/Z19-H8 system you not only get graphics that rival most 16 bit computers, you also get a high speed Pseudo Disk that does the work of another disk drive in your system, only much faster, and without all of that noise!

This upgrade stretches the horizons of your system into the future with High Resolution Graphics, Pseudo Disk Storage, 2 Trackball/Joystick Interfaces, and 2 Parallel Printer Ports, all for one very competitive price.

Copy your CP/M or HDOS files to the Pseudo Disk and cut execution times as much as 80% over mechanical drives with disk I/O intensive programs. Files are even retained when you cold boot, and HDOS users can sysgen it as the boot device!

Take a good look at these features:

- 3 pages of graphics, 640 by 250 pixels each
- Mix text and graphics with independent scrolling
- A remarkably easy-to-use high level driver that gives ANY language the power of fast graphics
- Unmatched drawing speeds result from an advanced design which requires no handshaking or wait states
- Use interlace mode and double the screen resolution to 640 by 500 pixels (requires Super19 or Ultra ROM)
- Plug in up to 2 trackballs or joysticks for interactive applications (provisions for light pen)
- 2 Centronics parallel printer ports with full support available
- An alternate character set ROM socket to double the number of characters your screen can display (Script character set available)
- Source codes available
- European 50Hz compatible
- Mounts professionally and easily into your H/Z89 or H/Z19-H8 and no existing features are sacrificed
- 90 day limited warranty.



Call or write NOW so we can rush our complete information package to you!

COMPARE	Northwest Digital Graphics-Plus	Cleveland Codonics Imaginator	SigmaSoft and Systems
Normal Pixel Resolution	512 x 250	504 x 247	640 x 250
Interlaced Pixel Resolution	n/a	n/a	640 x 500
Graphics Memory Size	16k	16k	64k/128k*
Number of Graphics Pages	1	1	3
Pseudo Disk Memory Size	n/a	n/a	64k/128k*/256k*
Super19/Ultra ROM Compatible	No	No	Yes
Vertical Graphics Scrolling	No	No	Yes
Type of Interface	Serial	Serial	Parallel
Independent Power Supply	No	No	Yes
Graphics Input Device Ports	n/a	n/a	2
Parallel Printer Ports	1*	n/a	2
Price/Performance Ratio	Good	Better	Best

*Option at additional charge.



"Support is the most important feature."

4488 Spring Valley #107, Dallas, TX 75234, (214) 392-1025

Copyright 1984 by SigmaSoft and Systems

The Interactive Graphics Controller is a trademark of SigmaSoft and Systems. H/Z89, H/Z19, and HDOS are trademarks of Heath Corp., Benton Harbor, MI. CP/M is a trademark of Digital Research, Inc., Pacific Grove, CA. Super19 is a trademark of Extended Technology Systems, Bensalem, PA. Ultra ROM is a trademark of Software Wizardry, St. Charles, MO. Codonics and Imaginator are trademarks of Cleveland Codonics, Inc., Cleveland, OH. Graphics-Plus is a trademark of Northwest Digital Systems, Seattle, WA.

Save up to 40% this month on Computer Accessories



SAVE \$180
while supplies last.
Hurry!

at Heath/Zenith Computers & Electronics

US Robotics • Hayes Compatible

300/1200 Baud

Auto-Answer MODEM

\$269

Cable Included.
Reg. \$449

Features full and half duplex operation, direct connection to standard phone jack, auto answer, originate and answer modes.

Epson RX-80 PRINTER

\$230

Reg. \$300

Doublestrike, Elite, and Italics. Prints 100 characters per second, has low-cost disposable printhead, quiet operation.

SURGE PROTECTOR

NOW **\$40**

Reg. \$50

Protect valuable documents against power loss with new Heathkit Surge Protector. Features unique "system powerup" of all peripherals when you turn on your computer. Also guards against line surges and voltage transients. 12 outlets.

HEATHKIT X-Y PLOTTER

NOW **\$315**

Reg. \$350

Easy-to-use, digital plotter draws fast, accurate graphics, operates on simple commands. Compatible with Lotus 1-2-3 and other graphics software packages. High resolution, clear lines, light weight. Red, green, blue and black pens available.

SUPER PRICE



DSDD NASHUA DISKETTES

Now **\$14.95**

per box of 10

For IBM, Apple, Zenith, Commodore, Atari and more. Lifetime replacement guarantee. *No limit.*



"SIDEKICK" Software

Now **\$33.88**

Reg. \$50

"InfoWorld" product-of-the-year! Desk management package has calculator, phone dialer, appointment calendar, and text editor. For IBM-PC or compatibles.



Protected DISK HOLDER

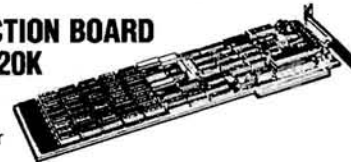
Lockable high-impact plastic holder keeps dust and fingers away from 70 disks.

Now **\$12.88** Reg. \$20

64K MULTI-FUNCTION BOARD
Expandable to 320K

Now **\$239**

Multi-function "Tecmar Captain" expansion board has 64K memory, serial and parallel ports, battery backup. 24 programs included.



IBM-PC COMPATIBLE COMPUTERS with 128K MEMORY

from **\$1599**
Includes \$570 in free software with computer.

Prices good through May 31, 1985.



NATIONAL HUG MEMBERS SAVE 10% ADDITIONAL OFF THESE PRICES



data systems

Phone orders accepted.

Your TOTAL SERVICE computer center

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

Heath ZENITH®

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 Latham
(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

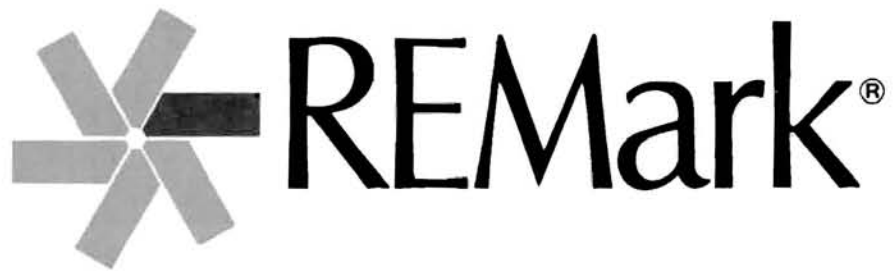
Although it is a policy to check material placed in REMark for accuracy, HUG offers no warranty, either expressed or implied, and is not responsible for any losses due to the use of any material in this magazine.

Articles submitted by users and published in REMark, which describe hardware modifications, are not supported by Heathkit Electronic Centers or Heath Technical Consultation.

HUG is provided as a service to its members for the purpose of fostering the exchange of ideas to enhance their usage of Heath equipment. As such, little or no evaluation of the programs or products advertised in REMark, the Software Catalog, or other HUG publications is performed by Heath Company, in general and HUG, in particular. The prospective user is hereby put on notice that the programs may contain faults, the consequence of which Heath Company, in general, and HUG, in particular, cannot be held responsible. The prospective user is, by virtue of obtaining and using these programs, assuming full risk for all consequences.

REMark is a registered trademark of the Heath/Zenith Users' Group, St. Joseph, Michigan.

Copyright © 1984, Heath/Zenith Users' Group



Buggin' HUG

..... 8

Sequential Files — Part 3

David E. Warnick 11

**Adding Sound And Speech
To The Z-100**

Tom Huber 14

CONVERT!

Raymond Dotson 18

The FORTRAN Formula — 4

Dick Stanley 22

**Displaying Text On The
Z-100 PCs — Part 1**

Mark J. Foster 28

Spiral Magic On The H/Z100

Gary Cramblitt 30

**COLOR — A Screen Color Setup
Program For The H/Z-100**

Glenn F. Roberts 35

HUG Price List

..... 41

HUG New Products	42
Excursions Into FORTRAN	
<i>Hank Lotz</i>	51
The Sometimes Useful H-89 Front Panel Monitor	
<i>Charles E. Horn</i>	57
HUG Club Update	58
Spooldisk 89	
<i>Peter Ruber</i>	59
SPREADSHEET Corner — Part 9	
<i>H.W. Bauman</i>	64
Meltdown Imminent	
<i>John W. Rogers</i>	70
Loading Assembled User Routines In Microsoft BASIC Under CP/M	
<i>Robert S. Wrathall</i>	72
The Elusive Escape Codes In Pascal	
<i>Paul W. Simmons</i>	74
Making Z-100 WordStar Faster — Part 2	
<i>Pat Swayne</i>	80

Index of Advertisers

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

Abe Dweck Programs	13
Analytical Products	13
CDR Systems, Inc.	45,73
Clarkson University	69
DelSoft	21
First Capitol Computers	27
Graphic Design Systems, Inc.	71
Guardian Data Systems	17
HME, Inc.	71
HUG Conference	9,29,69
Hilgraeve, Inc.	46
KC Computers	84
MDG & Associates	21
Newline Software	6,7
Norcom	71
Northwest Digital Systems	63
Paul Herman	58
Ross Custom Electronics	69
S&K Technology, Inc.	62
Secured Computer Systems	40
Sigma Soft and Systems	2
Software Toolworks	45,62
Solus Company	58
Sunflower Software, Inc.	79
Veritechnology	3
Barry Watzman	47-50
Wizard Software House	40

On The Cover: Shown is the multi-function P-SST Card by Software Wizardry. Look for the review on Page 14. (Photo by Jory Klopp, Heath Graphic Design.)

APPOINTMENT CALENDAR
Version 2.1 for MSDOS/ZDOS
Copyright 1985 by Newline Software

APPOINTMENT CALENDAR is a ZBASIC program designed especially for the H/Z-100 computer. **APPOINTMENT CALENDAR** will let you create, store, edit, display, and print a calendar of important events and times for appointments, schedules, and special occasions. It directly supports the Epson MX, RX, and FX series of printers, the Heath H-14 and H-25, Diablo 630, and MPI 99/150 printers. Virtually any other dot matrix or letter quality printer is supported by a standard ASCII version of the calendar printout.

APPOINTMENT CALENDAR is very easy to setup and use. It is fully menu driven and the program automatically provides default input entries and verifies user inputs to insure correct entry of data. The program is tailored to the Z-100 and takes advantage of the display and keyboard features built-in the Z-100. The function keys are used for command selection, and a command label appears above the function keys in the 25th line to make operation simple and straight-forward.

If you have important dates, birthdays, anniversaries, special events and you want to keep track of them on your computer, then **APPOINTMENT CALENDAR** is for you.

For Z-100, 128k memory minimum, Z-BASIC **\$29.95**

Z-TOUR 1000
Copyright 1984 by John E. Miles
Produced by Newline Software

Z-TOUR 1000 is an auto race 'card game' that simulates 'real life driving situations'. **Z-TOUR 1000** is a computer adaptation of the familiar Mile Borne card game. A full screen color display showing all cards and the playing board in vivid colors, quick speed, and complete implementation of all game rules, including COUP FOURRE and optional game extension make **Z-TOUR 1000** an interesting and fun game for adults and children.

The object of the game is to accumulate 10,000 points during several hands of play. During each hand, you and the computer will each try to travel exactly 700 miles, or exactly 1000 miles in the case of an extension. By drawing cards from the deck, you and the computer will be trying to slow each other down by placing road hazards in the way while you advance to the finish line.

In order to proceed, you must have a green light. If you see a stop sign, you must stop. You must obey the speed limit signs. If you have an accident, you must have your car repaired. If you run out of gas, you must fill your tank. If you have a flat tire, you must install a spare tire. The card deck contains cards for all the hazards, remedies, and mileage. How well you do is based on the luck of the draw and the skill with which you play your hand against the computer.

Z-TOUR 1000 is a challenging, cerebral computer game. If you're tired of the fast-paced video arcade style games that tax your manual dexterity rather than your game playing skills, you'll enjoy **Z-TOUR 1000**.

Requires: H/Z100 computer with color video ram installed, a color monitor, ZDOS or MSDOS, and ZBASIC

For Color Z-100: **\$29.95**

BOXES & SWITCHES
Two Games for the Price of One.

BOXES
Copyright 1985 by Terry Wlk
Produced by Newline Software

BOXES is a pixel graphics game program for the Z-100 with or without the color option. Perhaps you know the game by other names such as "Dots" or "Dot-to-Dot". It is a game of both skill and chance. You can play against another person or the computer. Briefly, the game is composed of a grid of dots. Each player completes a turn by drawing a line between two adjacent dots, with the intent of completing a box or making a move that does not allow your opponent to complete a box on his turn. Whenever you complete a box, it is marked as your's and you get another turn. The object of the game is to create more boxes than your opponent. **BOXES** is designed to be easy to learn and operate for game players of any age.

SWITCHES
Copyright 1985 by Terry Wlk
Produced by Newline Software

SWITCHES is an original pixel graphics puzzle program. The puzzle is composed of different rectangular parts. **SWITCHES** starts off with all parts set to one color called the "original color". The object of **SWITCHES** is to switch all of the puzzle parts from the original color to another color called the "switch color". You do that by selecting one rectangular part to switch to the opposite color. You can switch any part's color any number of times and in any sequence you like. However, the catch and the challenge is that in switching a part's color all adjoining parts will also switch color. The problem is in figuring the sequence of switches to solve the puzzle. You can try to solve the puzzle yourself, or watch the computer try to solve it. The number of puzzles is virtually limitless! This one will really get you thinking.

Program Requirements:

H/Z100 Computer with Z-DOS or MS-DOS 2.0, 128k of memory minimum, one or more disk drives, Color (optional). A Printer is optional for the SWITCH program. Both **BOXES** and **SWITCHES** are included

BOXES & SWITCHES \$29.95

STECH
Technical Analysis Program
Copyright 1985 by Newline Software

Ever put your money in the right place at the wrong time? Really frustrating isn't it? How do the "Big" guys do it? Part of their secret is knowing what to buy. The rest is knowing when. What you need is some help in making that critical timing decision.

STECH is an easy menu-driven program that opens up a broad array of technical analyses to look at several types of investments. Predicting future prices, making commodity decisions, looking at the impact of inflation, making a graph of price moves, are all part of this program. Each routine lets you enter your own data and it does the rest. You have the following choices available to you:

- Cash Flow Analysis
- Regression Analysis
- Exponential Smoothing
- Moving Averages
- Commodities Analysis
- Investment Return After Inflation
- Scattergram Program

Have you ever asked yourself, "Is this really worth it?" Now you have a variety of tools to help answer that question. No more tedious number-crunching. Sit back and do what you always wished you could do—focus on what this all means. In addition, the program comes with complete documentation to explain each routine and how it can be used. So your IRA earns 10% each year? What will it be worth in 20 years after inflation? **STECH** will answer that and much more.

What's so unusual about this software? First, it was designed with the user in mind. This little guy is not only friendly, he's also accurate and FAST. Why wait? You don't have to be a computer wiz, just do the following:

- Get a copy of **STECH**
- "Boot" up your system
- Start making more money . . .

So why wait? At this price you can afford to make a last decision!

For Z-100: 128k minimum, MS-DOS, Z-BASIC **\$99.00**
For Z-150, IBM-PC: 128k minimum, MS-DOS or PC-DOS **\$99.00**
For DEC Rainbow: MS-DOS or CP/M-86/80, MBASIC **\$99.00**
For DEC DECmate II or DECmate III: CP/M, MBASIC **\$99.00**

SFUND
Fundamental Stock Analysis Program
Copyright 1985 by Newline Software

Are all your stocks "In-The-Money"? Want to know the "Big" guys' secrets? If you aren't satisfied with your selections, you need to look at how other traders make theirs. Some would tell you that "It's all in the timing." That's only half-true. Knowing what to buy is the other half. This program is designed to help you make more money by choosing the right stock.

This easy menu-driven software not only performs a variety of fundamental analyses, it also has complete documentation to explain what all this stuff means! Here is a way to avoid tedious number-crunching and, instead, focus your gray-matter on the bigger issue—Which one is best? Use any source of "balance sheet" data and this program will really do a job for you.

You got a "hot tip" and want to check it out? You provide the program with some raw information and it calculates the following:

Liquidity	Leverage
Current Ratio	Debt Ratio
Liquidity Ratio	Times Interest Earned
Quick ("Acid") Ratio	Long-Term Debt to Assets

Activity	
Inventory Turnover	Return on Fixed Assets
Fixed Assets Turnover	Return on Capital
Total Assets Turnover	Equity Ratio
Return on Sales	Net Income to Net Worth
Return on Total Assets	Predicted Share Price
Return on Net Worth	

Wait!! There's much more . . . This program allows you to also compare up to THREE stocks simultaneously. This could mean looking at three possible choices in one stock family or looking at a single stock over three periods of time.

Now you have a way to make all these numbers come into focus to help you make a better decision. Why have you waited so long? Probably because you didn't want to spend hundreds of dollars on software, only to find yourself tied to someone's electronic database. Sometimes that's okay—sometimes not. Here's a way to have a choice in the matter.

For Z-100: 128k minimum, MS-DOS, Z-BASIC **\$ 99.00**
For Z-150, IBM-PC: 128k minimum, MS-DOS or PC-DOS **\$ 99.00**
For DEC Rainbow: MS-DOS or CP/M-86/80, MBASIC **\$ 99.00**
For DEC DECmate II or DECmate III: CP/M, MBASIC **\$ 99.00**

VEHICLE FLEET MANAGEMENT
Copyright 1984 by T.R. York Consulting
Produced by Newline Software

VEHICLE FLEET MANAGEMENT is designed to keep track of a fleet of up to 100 vehicles. This Menu Driven group of programs allows you to store information on fuel consumption and will calculate miles per gallon on each fuel purchase as well as vehicle and fleet averages. You can also keep track of purchases for parts, maintenance and insurance. Track vehicle maintenance schedules to insure your vehicles receive proper and timely service and maintenance. **VEHICLE FLEET MANAGEMENT** will produce reports on the computer display or printer to show you the status of each vehicle and total expenses for the entire fleet. Now that the IRS is requiring more accurate records, **VEHICLE FLEET MANAGEMENT** is a necessity for the small business owner with several vehicles. You'll have the data at your fingertips when next year's tax time arrives!

System Features:

- Each vehicle is assigned a Vehicle Number from 1 to 100. You enter initial data for each vehicle including: vehicle number, license number, current mileage, mileage between service, last service mileage, mileage between periodic maintenance and last periodic mileage.
- A Forms Generator prints out forms for the vehicle operator to log fuel, mileage, and expenses. The data is then entered into the program for computer record keeping, calculation and reporting. Data entries can also be corrected in the event of a data entry error.
- Screen displays and printed reports show:
 - Fuel data: Record Number, Vehicle Number, Date, Gallons, Price/Gallon, Miles since last fill-up, Miles/Gallon, Cost/Mile, Total Vehicle Cost.
 - Other data: Record Number, Vehicle Number, Date, Description of Expense, Cost/Mile, Total Vehicle Cost.
 - Summary for the Fleet: Total Miles, Total Gallons of Fuel, Average Miles/Gallon, Average Price/Gallon, Total Fuel Expenses, Total Other Expenses, Total of All Expenses, and Average Cost/Mile.
 - Individual vehicle summary displays show projected cost for 20,000 miles for each vehicle.

Vehicle Status shows: Vehicle Number, License Number, Miles to Next Service, Miles to Next Periodic Maintenance, and Total Cost to Date. A warning message is displayed if a vehicle is overdue for either service or periodic maintenance.

For H/Z-89, H/Z-90: CP/M, MBASIC **\$99.95**
For H/Z-100: CP/M-85, MBASIC **\$39.95**

NEWLINE SOFTWARE

P.O. Box 289
Tiverton, RI 02878

U.S. Checks Only; Shipping: U.S. \$3.00, Foreign \$10.00 - - C.D.D. \$3.00.

Finally, text processing that fits your computer to the letter.

Professional Text Processor Features:

Full Screen Editing
 Horizontal Scroll (256 col.)
 "OOPS" Key to Undo Errors
 Automatic File Backup
 Cut/Paste in Memory
 Cut Area to Disk File
 Insert File from Disk
 Global Search/Replace/Delete

Paragraph Formatting
 Paragraph Justification
 Adjustable TAB Stops
 On-Screen BOLD
 On-Screen Underline
 Auto Indent
 Auto Word Wrap
 Directory Display

Programmable Macro Keys
 Command Repeat Key
 TAB/Back TAB Commands
 Adjustable Margins
 Interactive Hyphenation
 Built-In Print Utility
 On-Screen HELP Displays
 Many, Many More . . .

Newline. Matched to the Zenith 100, 150 and the IBM-PC. If you have one of these systems, Newline has the right line of text processing software for you. Because Newline's Professional Text Processor (PTP) is matched to the keyboard and display characteristics of each of these computers.

There are no complicated key sequences to learn. And you'll be able to use labeled editing keys. Which means it's exceptionally easy to use.

New features for Newline's PTP. Even better, now PTP has more powerful text processing than ever before. There's everything from full screen text editing and on-screen bold, underline, paragraph fill and justification to cut and paste to configurable macro keys and much more. Plus, you'll be able to use our software with any printer.

Update from our old line. If you're presently using the Newline TxtPro software on your Zenith 100, now you can update to our more powerful version. It's called the PTP-100. If you currently have the ZDOS TxtPro, you can upgrade to ZDOS PTP-100. If you have the CP/M-85 TxtPro, you can upgrade to CP/M-86 PTP-100, but you'll also need to upgrade your system to CP/M-86. Just specify which one you have when you order. And if you return your old TxtPro disk to us with your order, you qualify for a special reduced price.

For Zenith 150 and IBM-PC users, there's the new PTP-PC. And it has all the same features as the PTP-100.

Software development editing. For programmers, Newline's PTP also offers auto indent and produces ASCII files for use with assemblers, interpreters and compilers. And that's not just different, it's unique.

Just give us the word. Newline's Professional Text Processor (PTP) is available right now. So place your order today. And get the text processing software that fits *your* system to the letter.

NAME _____
 ADDRESS _____
 CITY _____ STATE _____ ZIP _____

PTP-100 (for Z100) @ \$99. or PTP-100 (Z100 Upgrade) @ \$50.*

<input type="checkbox"/> CP/M-86 (replaces CP/M-85)	Qty _____	Amount _____
<input type="checkbox"/> ZDOS	_____	_____

*To qualify for Z100 upgrade price of \$50. you must return your TxtPro disk.

PTP-PC (for Z150 or IBM-PC) @ Special Introductory Price of \$149.

<input type="checkbox"/> CP/M-86	_____	_____
<input type="checkbox"/> MS-DOS or PC-DOS	_____	_____

RI Residents Add 6% Sales Tax
 Shipping (\$3. per program) _____
 TOTAL ENCLOSED _____

Payment in U.S. Funds Only • Allow 2 wks. for Personal Checks • CALL (401) 624-3322 FOR C.O.D. DELIVERIES

NEWLINE SOFTWARE
 P.O. Box 289 • Tiverton, RI 02878 • (401) 624-3322

BUGGIN' HUG

Painful BDOS Errors

Dear Editor:

I have been working on bulletin boards recently and was having a problem with erratic operation due to those painful "BDOS error" messages. I found the article by Pat Swayne in the February issue to be very helpful, but it did not quite fulfill my needs. Since other readers might have similar needs to mine, I thought I might pass this on.

What I needed was for CP/M to ignore bad sector error messages and keep going without requiring me to type a character on the keyboard. This was accomplished in a very crude, but effective manner by using HUG's DDEU to change track 0 sector 18 (19 on the 5-1/4" soft sector) byte 78 from a CD to a C3. I have found this to be handy in everyday use also.

I would point out that Pat's article was inaccurate in what happens if you do not type CTRL-C on a "Bad sector" error message. The error is ignored, not retried.

Thank you,

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

A Need To Interface Paper Tape Reader And High Speed Punch

Dear HUG:

I recently had the need to interface a paper tape reader and high speed punch to the Z-100 installed by the company I work for, for the purpose of editing, producing, storing in disk files, CNC machine tape programs. For this purpose, I used an excellent book recently published by McGraw-Hill entitled "Interfacing To S-100/IEEE 696 Microcomputers" by Sol Libes, Mark Garetz.

For any HUG members wishing to build peripherals that use the expansion slots in their S-100 card, the interfaces worked the first time. Covered in the book are serial, parallel and AD/DA converter interfacing.

Yours truly,

A.W. Inness
91 Southwalk Bay
Winnipeg, Manitoba
Canada

MPI Z-100 AP-PAK Screen Dump

Dear HUG:

I have an H-100 and the MPI-99 printer with their AP-PAK. There is a problem built into their screen dump program. It prints a dot

for each color, this makes the printout characters or graphics 2/3 longer than it should be.

When I contacted MPI about their screen dump program, they said they know about the problem and may start to work on a correction in six months, if enough users complained.

Is there anyone using the MPI AP-PAK that has solved or knows how to solve this problem?

Harold Stoiber
12810 Cherry Tree
New Berlin, WI 53151

ZD — A Sorted Directory For The H/Z-100

Dear HUG:

Jeff Kalis' "ZD — A Sorted Directory for the H/Z-100" (REMark, April 1984, pg. 14) is really super. I have it on all my working disks and call it, with the /c option, with the autoexec.bat file. It seems it could use an improvement, however, and the chore is beyond me. Specifically, once a title using the /t option has been given, it should be saved and inserted on each subsequent call of the directory. Anybody want to try it?

Sincerely,

David K. Wheeler
306 Winslow Street
Watertown, NY 13601

The H-89 And The Daisy Wheel Printer

Dear HUG:

I have recently purchased the Data Terminals and Communications (DTC) Style Writer Daisy Wheel printer, which is advertised in the Heathkit catalog and would like to relate my experiences in interfacing the printer with my H-89.

First of all, for those of you not familiar with the printer, a brief description follows:

Diablo(tm) 1640/1650 software compatible
(Magic Wand/SuperCalc(tm) compatible)
Intelligent buffer (35K characters)
Centronics Interface
Bidirectional printing (automatic)
10/12/15 or programmable pitch
Bold Face print mode
Selectable reprint of buffer
Super/Sub Script mode
And many more features too numerous for this letter

The printer is solidly built, the operating manual is adequate and ribbons/daisy wheels are readily available from Brother(tm) dealers.

Now the bad part for H-89 owners, in order to interface the DTC with our machine a Z89-11 Multi-Function I/O Card is required. So far so good, the Z89-11 manual states on page 4, "In the way of software, your Computer should be running the Heath Disk Operating System (HDOS) Version 2 or CP/M Version 2.2.03 (see catalog)." The bad part comes when you Boot up HDOS, load in your applications program (in my case ROOTS89) and select the



**INTERNATIONAL
HEATH/ZENITH USERS' GROUP CONFERENCE
Official Conference Registration Form**

**O'Hare Hyatt Regency
Rosemont, Illinois
August 9, 10, 11**

Name(s): _____

Company: _____
Address: _____
City: _____ State: _____ Zip: _____

Enclosed is \$25.00 for each of the individuals listed above to attend the International HUG Conference being held the weekend of August 9, 10 and 11, 1985. Please send tickets along with information regarding hotel reservations and transportation.

Amt. Enclosed: _____ No. Attending: _____

For Our Information:

Which Heath/Zenith computer do you now operate? _____

Are you a Non-User-Attendee? Yes No

Are you a computer related manufacturer? Yes No

If yes, would you like exhibit information? Yes No

Are you, or anyone in your party, interested in activities in or around the Chicago area other than the Conference? Yes No

If yes, please indicate any suggestions you may have: _____

Special Notice To Exhibitors:

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

For Your Information:

The \$25.00 you are paying for your reservation to the International HUG Conference entitles you to all functions of the Conference. Visitor tickets, for those of you simply attending the seminars and exhibits, are available for \$10.00. Visitor tickets do not include eligibility for prizes or food while attending the Conference.

Please send your completed registration form or suitable copy to:

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

Registration(s) must be post marked no later than July 31, 1985. Cancellation will not be accepted after this date.

PRINT routine from the menu. As you sit and wait for the printer to begin it's steady tap tap tap, the thought begins to creep up on you that there is something wrong. A quick call to Benton Harbor reveals that the manual is in error, and has been since it's initial printing. You should also ignore the statement in the Heathkit advertisement which states that the DTC Style Writer requires the HCA-53 parallel cable. You should instead purchase the HCA-14 parallel cable (for the H89 only). I was too eager to get my printer working to order another cable, so I rewired the HCA-53 in accordance with the following pin out:

H89 End	pin to	DTC End
1		1
3		2
5		3
7		4
9		5
11		6
13		7
23		16
24		16
21		11
19		10
17		9
15		8

I hope that this letter proves helpful to someone out there, and I also hope that someone may have written a device driver in HDOS for the Z89-11 Multi-Function I/O Board, and if so, would be willing to share their program with me. Thank you.

MSG David A. Zigler
109 Chapel Street, Apt. C
Clarksville, TN 37042

CROSSREF Program For The H100

Dear HUG:

I just received the August 1984 issue of REMark and quickly reviewed each article. I became interested in Mr. Dodgen's article — Inside Microsoft (TM) BASIC (MBASIC). I found it to be well written and to contain a detailed explanation of the subject matter. The functions provided by his CROSSREF program were just what I needed for my current program work. Unfortunately, the program was written for the H89 and I have an H100.

Using the explanation provided in Mr. Dodgen's article, I revised the program to operate on the H100. In addition, I corrected what appeared to be some typographical errors, and added a few enhancements:

1. Line 65000 — Set MP=25079 vs 28761
2. Line 65460 — Add the string designator (\$) to each variable V.
3. Line 65500 — Change line segment to:
IF VAL(LEFT\$(V\$(I),X1)) < VAL(LEFT\$(V\$(I+1),X2))
4. Add the following lines to strip quoted strings from the variable list:
 - a. Change line 65210 to end after the expression —
ELSE IF MM=14 THEN GOSUB 65400
 - b. Add line —
65212 IF MM=34 THEN GOSUB 65225
 - c. Add line —
65215 IF MM>64 AND MM<123 THEN GOSUB 65300
 - d. Add line —

```
65225 MP=MP+1:MM=PEEK(MP):IF MM<>34 GOTO
65225 ELSE RETURN
```

5. Modify line printer program steps as follows . . .
 - a. Change line segment in 65030 —
. . . INPUT "Do you want the output to go to a disk file (LPRINT) & to the printer";LP\$. . .
 - b. Change file name in 65030 from "LP:" to "LPRINT"
 - c. Change line 65525 to —
65525 NEXT:IF LP\$="Y" THEN PRINT:INPUT "Printer READY (Y/N)..";A\$:IF LEFT\$(A\$,1)<>"Y" GOTO 65530
ELSE POKE 3,158:WIDTH LPRINT 80:FOR I=1TO LV:
PRINT V\$(I): NEXT:POKE 3,149
 - d. Add line — 65530 CLOSE:END

I hope this information is helpful to other H/Z100 owners that may want to use Mr. Dodgen's program. Enjoy . . .

Yours very truly,

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

Gripes On Fortran: Ray Battalora's Problem

Dear HUG:

In response to Ray's concern with unnumbered Fortran program lines, I'd like to tell about my own experience with Fortran programming.

Recently, I have been converting a "path analysis mainframe Fortran program" into an SSS Supersoft Fortran program for my H100. Of course, I had a similar problem with line identification, but the other way around: I actually wrote line numbers on my program, creating an uncompileable program. I learned that I could not use the first six columns, reserved for the program statements. I found out that the ZDOS editor "edlin" assigns line numbers automatically and thus requires that the edited program be reread: "edlin XXXX.for" in order for the user to display the line numbers back e.g. "1,20L" will display the first 20 lines of the program "with the line numbers showing."

I hope this hint will help Ray.

Lucien Laforest
579 Boisjoli Street
Sherbrooke, Quebec
Canada J1J 3E8

AMSAT — Amateur Satellite Association

Dear HUG:

In volume 5, issue 8, August REMark page 65, we found a letter from D.C. Shoemaker asking for an OSCAR program.

The AMSAT is an international Amateur Satellite Association who publishes ORBIT and ASR Newsletters about Radio Amateur satellite activities. There is an Amsat Software Exchange department at the following address:

AMSAT Software Exchange
AMSAT Headquarters
P.O. Box 27
Washington, DC 20044

Vectored to Page 81

Sequential Files — Part 3

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



Thus far in this series, we've looked at the structure of sequential files, we've created them, and we've read the data stored in them. However, when we tried to reopen an existing file to add more data to it, we saw that all the existing data was lost. Of what use is a filing system which cannot be changed? Not very much, so this month we'll start by looking at the way we can add to or change an existing sequential file.

For those who would like to try the programs in this series, but who hate to type them and then hunt typing errors, I've prepared a disk with all the programs in the series. It's available in CP/M only, in Heath hard- or soft-sectored format, and will be copied on my H89. To get one, send \$5.00 to cover the cost of the disk and shipping to the address at the beginning of this article.

When we wish to add information to an existing sequential file, we can use one of two methods. The first is to read the entire file into memory (we'd have to use arrays), then add additional information to those arrays. Finally, we'd write the arrays back to a new sequential file. This is rather cumbersome, and on a large file, you can quickly run out of memory. I mentioned it at this point only so you'd know it can be done.

The second and the more practical method of adding information to a sequential file is the one we'll use here. It requires the use of a temporary file and requires that the following steps be performed.

1. Open a temporary file for writing
2. Open the file we wish to add information to for reading
3. Read each record from the old file and write it to the temporary file until we reach the end of the old file.
4. Proceed to write additional information to the temporary file
5. Close both files
6. Kill the original file
7. Rename the temporary file to the original file name

The program, complete with REMarks, begins like this.

```
2 ' **** MODSEQ.BAS
4 ' **** DAVID E. WARNICK
6 ' **** COPYRIGHT 1984
5000 'OPEN EXISTING FILE TO READ AS FILE #1
5010 OPEN "I",#1,"SEQFILE.DAT"
5020 'OPEN THE TEMPORARY FILE TO WRITE AS FILE #2
5030 OPEN "O",#2,"DATAFILE.TMP"
5500 'THE FOLLOWING LINES WILL READ A RECORD OF THE
      EXISTING FILE
5510 'THEN WRITE THAT RECORD TO THE TEMPORARY FILE
5520 'THIS WILL CONTINUE UNTIL ALL RECORDS HAVE BEEN
      COPIED
5530 'AT WHICH TIME THE "EOF" TEST WILL SEND US TO LINE
      6000
5540 FOR X=1 TO 100           'SET UP A LOOP
5550 IF EOF(1) THEN X=100:GOTO 5580 'TEST FOR END
                                  OF FILE
5560 INPUT #1,R$             'READ A RECORD
5570 PRINT #2,R$             'WRITE A RECORD
5580 NEXT X                   'CONTINUE TILL DONE
```

When we read and wrote the records above, we didn't need to know where fields began and ended. That information already existed in the file SEQFILE.DAT, so we merely passed it on to the new file. When all the existing records are copied to the new file, program execution will jump to line 6000. This is the place we'll write the instructions which permit us to add additional information to the file. They'll be the same as those lines which created the file.

```
6000 'ADD NEW INFORMATION TO THE TEMP FILE
6010 R$=""                   'MAKE SURE THERE'S NOTHING IN R$
6020 INPUT "LAST NAME ";A$   'ASK FOR LAST NAME
6030 IF A$="" GOTO 7000      'TYPE A RETURN TO EXIT
6040 R$=A$+"\\"             'ADD INFO TO THE RECORD
6050 INPUT "FIRST NAME ";A$ 'ASK FOR FIRST NAME
6060 R$=R$+A$+"\\"         'ADD FIRST NAME AND DELIMITER
6070 INPUT "MIDDLE INITIAL ";A$ 'ASK FOR MIDDLE INITIAL
6080 R$=R$+A$+"\\"         'ADD DATA TO RECORD
6090 INPUT "PHONE NUMBER ";A$ 'ASK FOR PHONE NUMBER
6100 R$=R$+A$              'ADD PHONE NUMBER
6110 PRINT #2,R$
6120 GOTO 6010              'GO DO IT AGAIN
```

This portion of the program lets us add to the information which has been copied into the temporary file. Go back and look at last month's article and the program WRITESEQ.BAS. This is essentially the same program. We've made it much shorter by combining the addition of a field delimiter in one instruction. Further, we've included our REMarks on the same line as the instruction they explain. These comments are very important. They enable us to see what we did and why we did it long after a program has been written. When we decide to modify our program in the future, a good set of comments will prove to be an invaluable aid. Don't take them too lightly as you write your own programs.

The first four steps we spelled out in the opening paragraphs of this article have been completed. Only steps five, six, and seven remain. Each is very short and simple. They'll begin at line 7000, as that's where line 6020 sends program execution when we type just a carriage return for the last name.

```
7000 'STEP 5 -- CLOSE BOTH FILES
7010 CLOSE #1
7020 CLOSE #2
```

Steps six and seven require some explanation as they introduce MBASIC commands we haven't used before. The first command is

```
KILL
```

It is used to remove a file from the disk. In our case, we'll remove the original data file so the new temporary file can replace it. You cannot KILL a file which is open. That's why we had to CLOSE #1. We could have closed and killed this file as soon as it was copied. However, if anything had caused premature program termination, the new file could have been lost. It is best to keep the old data around until the new data is ready to replace it.

The last step uses the command

```
NAME
```

Its purpose is to change the name of a file on the disk. It takes the form

```
NAME "oldfile" AS "newfile"
```

With this information, we can complete our program.

```
7030 'KILL THE OLD DATA FILE
7040 KILL "SEQFILE.DAT"
7050 'RENAME THE TEMPORARY FILE
7060 NAME "DATAFILE.TMP" AS "SEQFILE.DAT"
7070 END
```

That completes our program which adds additional records to an existing sequential file without losing any of the existing data. Enter MODSEQ.BAS including all numbered lines, presented thus far in this article, into your computer and run it.

The program will run long enough to copy all your existing entries, then it will ask for a "LAST NAME?" When this happens, add a few names and phone numbers. After a few entries, type the RETURN key when asked for a last name. This will terminate program execution and close the files. When the MBASIC prompt returns, your new file will be on the disk. You can run READSEQ.BAS which we wrote last month, and it will scroll all the old information which existed before this file update, then it will scroll the new information.

Further checks of the file can be made by exiting MBASIC to your operating system and looking at the disk directory, and by using

the TYPE or LIST commands to display the entire file on the CRT screen, or on your printer.

A third method exists for adding information to an existing sequential file. The steps required to use this method are

1. Open a new file for writing
2. Enter all new data
3. Close the new file
4. Open a temporary file for writing
5. Open the original file for reading
6. Read each record from the original file to the temporary file
7. Close the original file
8. Open the new file from step 1 for reading
9. Read each record from the new file to the temporary file
10. Close the new file
11. Close the temporary file
12. Kill the original file
13. Kill the new file
14. Rename the temporary file to the original file name

You can see that this method permits us to write the contents of two different files to a single file. When we do this, we must assure that the same conventions for field and record delimiters are used in each file. Otherwise, we'll have a file of data, but we won't be able to sort it out and use it.

I won't cover the programming steps for this process, as they should be rather obvious from the examples we've already shown. I presented the steps here as you should be aware that the method exists.

So, now we know how to make additions to an existing sequential file. But what about changes to the existing records, or the deletion of records from the file? We'll have to be able to make this type of change if the sequential file system is to be of any use to us. We're also going to have to be able to look up specific data and to sort the records of our file into the order in which we wish to use them.

To perform these tasks, we can no longer avoid reading our file into memory (RAM). We'll continue to use our phone number file, SEQFILE.DAT. In order to get the data from our disk file to the RAM inside our computer, we'll have to reserve space in that RAM for each item. We can reserve a space for each record, or better yet, a space for each field within each record. To do this, we'll use arrays. An array uses a single variable and subscripts it with a series of numbers to distinguish one element from another. Thus, we could use LN\$ for a last name array. When we add subscripts, LN(1) is the last name of the first record, LN\$(27) is the last name of the twenty-seventh record, and so forth.

We reserve space for an array by using the DIMension command. To make room for ten entries in the array A\$() we would write the program line

```
DIM A$(10)
```

For our purposes, we'll need arrays for each field. These are

```
last name      LN$( )
first name     NM$( )
middle initial MI$( )
phone number  PN$( )
```

Each array will be dimensioned to contain up to 100 elements, and they can all be dimensioned on the same program line. Our

sequential file utility program begins like this.

```
2 ' **** SEQUITIL.BAS
4 ' **** DAVID E. WARNICK
6 ' **** COPYRIGHT 1984
1000 DIM LNS(100),NMS(100),MIS(100),PNS(100)
```

We could have used a multi-dimensional array, such as A\$(4,100) to do the whole job with one variable. However, what's going on in the program may be less obvious to a novice at programming, and we don't want to lose anybody. Multi-dimensional arrays will be covered in a future article.

Now that the arrays are dimensioned, the next step is to read all the data from our file into those arrays. We'll read from our file using a FOR loop to copy a maximum of 100 records, and a counter (the variable CT) to keep track of how many records are in the array. Our program continues.

```
1010 OPEN "I",#1,"SEQFILE.DAT" 'OPEN FILE TO READ
1020 CT=1 'SET THE COUNTER
1030 FOR X=1 TO 100 'SET UP THE LOOP
1040 IF EOF(1) THEN X=100:GOTO 1180 'TEST FOR
    END-OF-FILE
1050 INPUT #1,R$ 'READ A RECORD
1060 'AT THIS POINT, WE'LL SEPARATE
1070 'THE RECORD INTO FIELDS, JUST AS
1080 'WE DID IN READSEQ.BAS
1090 A=INSTR(1,R$,"") 'FIND THE FIRST \
1100 B=INSTR(A+1,R$,"") 'FIND THE SECOND \
1110 C=INSTR(B+1,R$,"") 'FIND THE THIRD \
1120 LNS(CT)=LEFT$(R$,A-1) 'ARRAY VARIABLE =
    LAST NAME
1130 NMS(CT)=MID$(R$,A+1,B-A-1) 'ARRAY VARIABLE =
    FIRST NAME
1140 MIS(CT)=MID$(R$,B+1,C-B-1) 'ARRAY VARIABLE =
    MIDDLE INITIAL
1150 D=LEN(R$) 'LENGTH OF RECORD
1160 PNS(CT)=RIGHT$(R$,D-C) 'ARRAY VARIABLE =
    PHONE NUMBER
1170 CT=CT+1 'INCREMENT THE COUNTER
1180 NEXT X 'DO IT AGAIN
1190 CLOSE #1 'ALWAYS CLOSE FILES
    WHEN DONE
```

This part of the program read the contents of our file (up to 100 records) into the arrays we dimensioned. Line 1040 takes care of things if we run out of records before we reach 100. It sets the

loop control variable X to 100, so that line 1170 will set it to 101 and get us out of the loop. The final part of our program will print out to our printer (if you don't have a printer yet, substitute PRINT for LPRINT in the following program lines, and the output will go to the CRT screen) the contents of each element of the array to prove that everything worked as we planned it.

```
1200 FOR X=1 TO 100 'SET UP A LOOP
1210 LPRINT "ARRAY ELEMENT NUMBER ";X 'HEADER INFO
1220 LPRINT "NMS(";X;") = ";NMS(X) 'PRINT FIRST NAME
1230 LPRINT "MIS(";X;") = ";MIS(X) 'PRINT MIDDLE INITIAL
1240 LPRINT "LNS(";X;") = ";LNS(X) 'PRINT LAST NAME
1250 LPRINT "PNS(";X;") = ";PNS(X) 'PRINT PHONE NUMBER
1260 LPRINT 'PRINT A BLANK LINE
1270 NEXT X 'CONTINUE THROUGH LOOP
```

The LPRINT statements show how to mix quoted data with information stored in variables and arrays. We've seen how to take data from a sequential file and put it into an array. We proved that it worked by printing out the contents of the array.

Next month, we'll look at the operations required to change, sort, or delete that information. The operations will be similar to those used with random files last year. That will pave the way for information on handling files which are too big to fit into our available RAM, and to the conversion of sequential and random files to the opposite type.

See you next month. ✕

PIP—Parts Inventory Program A Fully Menu-Driven Parts Inventory Program

Automatic:

Purchase Orders + Posting of Items Received + Register
MORE! • Updates + Reports + Searches • MORE!

\$49.95

ABE DWECK PROGRAMS

PO Box 207, Carmel, New York 10512

Heath/Zenith CP/M Formats • NY residents add 6% sales tax.

EMULATE

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

Osborne 1	SSDD	Morrow MD2	SSDD	Cromemco	SSDD
Osborne 1	SSDD	Morrow MD3	DSDD	Cromemco	DSDD
Xerox 820	SSSD	Epson QX-10	DSDD	CDR 40TK	DSXD
Xerox 820	SSDD	Televideo 802	DSDD	CDR 80TK	DSXD
DEC V1180	SSDD	Actrix	SSDD	NEC 8001	SSDD
Ampro	SSDD	TRS80/Omikron	SSSD	Eagle II	SSDD
DEC Rainbow	SSDD	TRS80-4 CP/M	SSDD	Z100 40TK	DSDD

A universal format program will be supplied as a free update. The H37 version requires 64K of RAM and the use of a modified version of CP/M 2.2.03 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.

Must include your CP/M s/n when ordering.

For H37 with Heath CP/M \$59

**Limited Version For
CDR controller \$39**

Automatic Repeat

Simple plug-in installation of the REP2 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)

CDR Controllers At Discount!

For H89 FDC880H \$345

For H8 FDC8 Call For Quote

The Software Toolworks® - We Sell It At Discount!

Other Commercial Software at Discount - Call / Let Us Quote!

Downloading Service for many CP/M Formats - Only \$5/Disk + \$5/Order.
Customer Supplies Formatted Destination Disks.

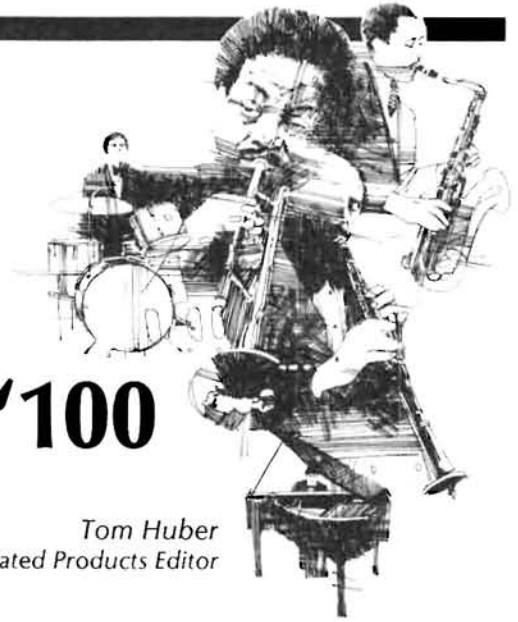
WE PAY POSTAGE! CA Residents Add 6% tax.
Call or Write For Catalog.

ANALYTICAL PRODUCTS 209/564-3687

20663 Ave 352

Woodlake, CA 93286

Adding Sound And Speech To The '100



Tom Huber
Related Products Editor

The P-SST card is a multifunction S-100 card for the H/Z-100 computer series from Software Wizardry. It can be adapted for use with other S-100-bus based computers, but the software for the card is especially suited to the H/Z-100 series.

Essentially, the P-SST "acronym" stands for Programmable Sound Speech and Time. From that, you are left open with all sorts of possibilities. And, by the time you finish reading this report, you may well decide that those possibilities are limited only by your imagination and programming skills. Fortunately, even the latter need not be a limiting factor if you consider obtaining some of the neat software packages available from Software Wizardry.

The time functions are controlled by an MM58167 clock/calendar IC that tracks time from a month at a step, down to thousandths of a second. It is crystal controlled, but may be "trimmed" through a variable capacitor for absolute accuracy. Since the clock does not depend upon an AC input reference, it will continue to run accurately with the on-board lithium battery for an advertised minimum of at least nine months. My best guess at this time is that the clock will continue to perform for at least a full year. Replacement lithium batteries are available from Software Wizardry for about \$1.50 each and are manufactured by several of the battery manufacturers. The Matsushita number is BR2320.

The clock chip has the capability of providing one of several modes for software interrupt operation. These include an "alarm" mode; a "heartbeat" clock tick interrupt signal for tenths of a second, seconds, minutes, hours, days (at midnight), weeks (at midnight Saturday), and months (at midnight on the last day of the month); and a "standby" mode. This last mode is similar to the alarm mode except, instead of providing a signal to the computer, the signal is sent to the outside world through a connector.

Since, I am not into applications that require these kinds of operations, I did not test these possible functions. However, there are a number of possible applications that come to mind, including automatic powering up of the computer for communications at a preset time (this would obviously require appropriate communications software and an autodial/auto-

answer modem. I would also strongly recommend a floppy disk drive that does not keep the heads "loaded" when the power is not on, or a rigid disk drive (autoboot enabled). One could also use this interrupt to automatically transmit information gathered from external sensors (perhaps temperature and humidity readings or other weather related information). At any rate, the possibilities are endless and the documentation is ample enough that you could use BASIC to set up the functions. Thanks to the built-in battery, you would not have to worry about power losses (unless they happened to take place when you called for the function to happen).

The most useful function for me was the support software for automatically logging the time and date upon powerup. Even with "DATETIME" (REMark Issue #40 — May, 1983) for Z-DOS or MS-DOS version 2, I still have to update the date and time function when I first power up a disk. Now all I do is run a short AUTOEXEC file which includes the CLOCK READ command line on booting up Z-DOS (MS-DOS) and my date and time is automatically set and displayed on the screen. The CLOCK.COM CHRONOLOGIC support program file occupies only 3,382 bytes (counting the AUTOEXEC.BAT file) on my disk, so my disk space is not hampered by adding the file. (The software is included with the card.)

However, one of my friends commented that the P-SST card was an awful expensive clock. It is indeed, except that I use more than the CLOCK READ function on this card.

One of the most enjoyable functions of the card is producing music from the sound section. In this section is the AY-3-8910, a three-voice sound chip, along with the Votrax SC01A speech synthesizer chip. While this product still does not give me total synthesizer control a la Moog, it comes in a nice second, good enough to have my computer and the P-SST card be one of the principals in a play (of the one-act roadshow variety) that I recently helped put on for my church.

The IC has three square wave sound generators and one pseudo-random (good enough for me) white noise generator (pulse-width square wave), the output of which can be combined with each sound generator. Each of the three output channels is fed through a digital-to-analog converter to a mixer which combines

the output to feed a one-watt power amplifier, providing both preamp and high-level audio output. For my application at home, I feed the output of my P-SST card directly to a 6 × 9-inch 8-ohm speaker. For the play, I used an external audio amplifier and two-way speaker system for more volume and a better tonal range.

There is one envelope generator that operates equally on all three channels, but it does provide a programmable attack/decay and cycle pattern control. Now naturally, I would like one for each channel, but you can't always have it all.

The amplitude is under direct program control for each of the channels, providing 16 volume levels. This number of levels, when applied in sequence, produces a very smooth attack or decay. You should not confuse the volume level with the envelope generator, since the volume of each channel is individually programmable. Programming the individual levels do take more work, but it is often worth it on a one-time basis.

The frequency range of each channel can be handled as a note or frequency, thanks to the software. Obviously, if you handle the frequency as a note on a scale, you have better control for music applications. If you handle the note as a frequency, then you have tight control from about 32 Hz to beyond the upper human hearing range. For music, the range is 8 octaves with middle C at the bottom end of the third octave. I have run into only a couple of scores that exceed this limit and then only at the bottom of the lowest scale. Since this is the feature I use most, I'll talk more about the music capabilities in a moment.

The voice synthesizer is the Votrax SCO1A IC and exhibits the usual tendency to reproduce German better than English. However, my limited experience with this product has been exceptionally good; the software for this chip is easier to use than earlier products I have used. The 64 phonemes give a wider variety of speech inflections than the 48 phonemes on some early synthesizer products from Votrax.

The 64 phonemes may be generated in any order and in three different pitches (no, it does not do a good "Tiny Tim" vibrato). Software Wizardry supplies an excellent text to phoneme conversion software program, so the most difficult job is done for you. All I have to do is interface the software with my programs and I'm off to the races (or exploring through dungeons with my faithful companion. With the proper software, text-only adventures are a lot more fun. Unfortunately, the voice quality doesn't quite approach that of WOPR in War Games or HAL in 2001, A Space Odyssey, but it is still very intelligible.

Incidentally, I mention these movies for comparison only. Until you have actually played with one of these cards, you really can't appreciate all of the power really available to you. You should also note that I wouldn't use a better synthesizer (like the Moog), even if it were available to me; I simply don't have the hours that it would take to properly program all the detail, nor do I believe that the job could be made simple through any magic supplied by Software Wizardry or anyone else. The job is simply too complex and what we, as humans, do with music and speech is, in my mind, beyond our current technology (and/or budget) to reproduce from something like the H/Z-100. If you disagree, I would like to hear from you (via Buggin' HUG).

I have used other synthesizers that have always required separate text and speech strings, if you want to echo the spoken material on the screen. With the supporting software, the two can be combined, producing clear English text while supporting the

synthesizer's occasional need for special phoneme coding.

This card is complex. It has taken me over two pages of WordStar text just to start describing this card's capabilities and I haven't described the two joystick (Atari switch-type) parallel input/output ports or the software support that is supplied for the price. For more technical information, you really need to contact the vendors and/or see this product in your local Heath/Zenith Computers and Electronic Centers. The card has just been added to the Heathkit catalog, so it should be available throughout the country. Last year at the HUG convention, Software Wizardry was showing off the card and I am told they will be back this year. You really have to see this thing perform to believe all it can do.

* * * * *

Obviously, I couldn't do justice to this card without rewriting the excellent manual that accompanies the product. The manual includes all the hardware and software information I think most people would ever want. It is complete with examples of how to "get inside" the card via software (including BASIC routines) and has enough technical information for assembly programmers to take full advantage of the card's capabilities.

It should be noted that the card does not need to be lonesome in your computer. More than one of these cards can be placed in a single computer; the base address of the card's ports are programmable. While I don't know of any software that supports two channels of three voices each, it sure would be a nice system to experiment around with. (I am told that LP Systems is designing the Z-150 version of this card with two of the sound chips, which can be either mixed or sent out separately for true stereo synthesized sound.)

If there were a way of controlling real time with this device, I would program at least two extra hours into each day, just to play with this card. I have had problems over the past several months tearing myself away from the pure fun involved in creating music (such as, "If I Were A Rich Man" from Fiddler On the Roof) to using all the fine support software that is included. If you think your spouse would divorce you over more hours spent on your computer, watch out. A fiddler on the roof I am with this card in my computer. It is very precarious up there on the roof.

As far as support software, I do need to mention the excellent demonstration material, including the very funny, "The Jabberwocky," spoken by the P-SST card with a proverbial tongue-in-cheek attitude.

I also need to mention the effect of increased clock speed (of the 8088) on the software. The card, as you might imagine, is usually unaffected by running a faster system clock. However, BASIC routines, which are dependent upon the system clock, will run fast, resulting in music that runs at the wrong tempo, but not the wrong pitch. This problem is alleviated by avoiding the BASIC foreground routines and running either background material only (which I prefer) or assembly language routines instead.

For the most part, the demonstration material is written for the MUSICA and MUSICK programs which play the scores in the background (entirely from the P-SST card) and draw either beautiful spiral art or kaleidoscope designs on the screen at the same time. The music samples include classics from the great composers including Rossini's "The William Tell Overture" to Dave Brubeck's "Take Five" and John William's "Star Wars" theme. Altogether, there are 18 music scores, all written as ASCII text

files, to be processed by the music programs. There are three text files, processed by the voice synthesis program. And finally, there is an arcade-type pong game that can demonstrate the use of joysticks on the computer (or can be used with keyboard keys if needed).

Unfortunately, the support software supplied with the card is for only a single voice. So where does one come up with the software for all three voices? Well, about three months ago, I received another package from Software Wizardry called the MAESTRO Music Support Package. This package provides the needed support for all three voices, along with 3-voice music subroutines for BASIC.

Even more important though, is the very powerful 3-voice music graphics editor. With this tool, programming the three separate channels is a snap (well, almost). It was the utility that I used to produce the music for the play in which my computer was featured. You should note here that this editor, although not necessarily big, requires at least 192k (mostly for buffer space).

Briefly, the music editor is divided into two sections: a main "menu" and an options section. For the most part, the options menu is called up only when some parameter of the system needs to be changed. The main menu contains the following routines:

- * Display the default disk's directory (unless otherwise specified) and read in a music score.
- * Play/step through a music score. In this mode, only one voice is played and the corresponding score for the voice is displayed graphically (yea, that's on a music staff!), which makes it very convenient for debugging (I'm not sure that's a good term to use here) and correcting mistakes (bad notes) in the score.
- * A play mode, in which the entire score is played while a selected voice is displayed graphically. Unfortunately, the display and music are played in two different time frames. With the speed-up kit installed and running at 7.5 Mhz, the score is displayed much quicker than the music is typically played. You may wonder how this is possible; the music is played in a "background" environment while the score is displayed in the computer's "foreground" environment; this process is somewhat similar to time sharing. Since some control is exercised by the foreground over the background environment, there are times when the music time frame is momentarily extended. While the documentation makes mention of this "problem," it is hardly noticeable, unless the music is extremely complex.
- * List the music score.
- * Enter the editor. In this mode, the commands available are: Append to the score; change a line of the score; delete a line of the score; empty the buffer (this is similar to the NEW command in BASIC); insert lines in the score; and list lines of the score. I'll mention more about this editor in a moment.
- * Finally, there is a write disk file operation that allows you to save the created/edited score on disk for later use in any program that can use it.

The options menu allows you to change a number of the system parameters.

- * Select console type. This function allows the use of either a color or monochrome display. In the color mode, colors are used very effectively to represent various elements of the musical staff. However, since I do not own a color monitor, I preferred to use the monochrome display mode and avoid the difficulty of attempting to "read" elements printed in the

"darker" colors.

- * Select starting line number. Because the buffer can hold a very large score, there is a tendency to get "lost." However, this option allows you to specify a line number from which the manual mode will start playing (see the previous list of routines).
- * Specify the displayed/processed voice (channel) number. The voices are numbered from 1 to 3, which represents the three channels of the music chip. Since the manual mode processes only one voice at a time, and since the program displays only one voice at a time, this option is needed to process/edit the various voices. I found that I needed to use this feature only occasionally, since I was able to "hunt and peck" my way through most scores without difficulty. I would suspect that if you use this program to create original music, you would want to use this feature often, particularly as you built the notes to play in the second and third voices.
- * Specify the time delay between screen displays. You may have noted my complaint that the screen "plays" through the score faster than the actual music. This option gives you some control over the display. But, and this is a big but, it only works well for music that is played at a fixed tempo. If you enter scores that use different tempos or make the use of a lot of ritardandos, then you will experience difficulty trying to balance the display against the music. Frankly, I did not dabble with this value, since I was more interested in the music than the display. Some of you may feel otherwise. I would be interested in reading your reasoning for wanting the display to march to the beat of the music.
- * You might be interested in examining only the music score itself and not have to listen to the sound each time you process a voice in the manual mode. There is an option to turn the music off (or on).
- * Obviously, there are times when you want to have your computer speak during music. This program allows you to enter (and display) text to accompany the music. Since there are times you don't want to hear the voice (it can be very distracting) while you are working on the score, you have the option of "telling" the voice synthesis to "shut up."
- * Likewise, the graphics generation can be turned off. This is most convenient when listening and working on a complex score, since the display of the graphics is a foreground operation and can detract from the timing of the music.

The editor itself is very easy to use. I have already told you the commands, which are invoked by pressing a function key, but I have not told you about the music notation used by the various software music processors that I examined along with the P-SST card. The notation is alphanumeric and nearly identical to that used by the IBM PC and PCjr. The PC has only one voice, while the PCjr has three voices similar to this card. With the advent of a lot of BASIC programs and particularly Zenith's introduction of a compatible (PC) GW-BASIC for the H/Z-100, it was particularly thoughtful of the people at Software Wizardry to use the same notation for the P-SST card. This makes it extremely easy to enter scores from the PC(jr) programs and adapt BASIC programs. (Now if only someone would patch GW-BASIC so that the PLAY, SOUND, COLOR, and SCREEN commands would translate correctly, we could run PC programs without modification. Hint. Hint.)

The BASIC routines included with this package require reserving high memory (half of the BASIC work segment), which unfortunately limits the amount of BASIC program space. Fortunately,

the Z-BASIC compiler has no memory space restriction (other than those imposed by the compilation process), and therefore, the music routines can be easily (relatively) linked with the BASIC code. I do not know if there are plans to include the three voice routines for the other compilers in the support package, which comes next in this review.

One of the real shortcomings to Zenith's software libraries for the H/Z-100 is the lack of good support libraries for the compilers, including FORTRAN, Pascal, and C language graphics routines. LP Systems, the hardware development arm of Software Wizardry, has come up with the answer in the form of the Development Libraries. Support is provided for Microsoft's Macro Assembler (MASM), FORTRAN compiler, Pascal compiler, and C compiler (the Lattice C compiler). In addition, Computer Innovations C86 C compiler is also supported. This support includes not only the P-SST routines, but a very good graphics library. Therefore, the support for the Microsoft BASIC library is not as broad, simply because the graphics routines are already built into Z-BASIC and the Z-BASIC compiler. (No Virginia, there is not a GW-BASIC compiler for the H/Z-100 yet.)

How good is this software? Both good and not so good. It appears to do the task, although not without some drawbacks.

One of the real "plusses" are the graphics routines for FORTRAN, Pascal, and the two C compilers. Not many programmers realize that FORTRAN is quite powerful, particularly when it comes to manipulating numeric calculations. By adding the power of the graphics routines to this compiler, LP Systems has given FORTRAN a badly needed "shot in the arm."

The excellent documentation (doesn't Software Wizardry usually do a good job?) mentions that it is possible other compilers, than those mentioned, might be used with the support libraries. However, this reviewer did not try any other compilers (none were available), but if they will interface with the Microsoft linking loader and LIB utility, they should work with this package.

Unfortunately, this software supports only a single voice at a time of the sound chip. I would prefer to be able to use all three voices and mix text at the same time. I think that with enough "tinkering" (which I am not prone to do because of time restrictions), one could effectively use this support package to produce all three voices simultaneously (I think — I am not sure at this point in time), similar to the support provided by MAESTRO. However, it will require a lot of work and experimentation. The documentation does not provide a great deal of support, other than the basics about the included code and how to call it from the various compilers. I think this is one area that could be improved.

However, the very badly needed graphics routines far outweigh the relatively weak music support from this package. The graphics alone make this package worth the price.

On a closing note (pun intended), I must say that the P-SST card and the two software support packages I had a chance to review continue to testify to the tremendous job done by the folks at Software Wizardry and LP Systems. I have enjoyed using and playing (with) both the software and the card.

Like all good computer products, the P-SST card will probably never be fully exploited, simply because you can do so much with it. And like all good computer products, you always wish for

just a little more capability. Most music requires more than three voices; in fact, most music scores start with at least four voices and often uses more. My one hope is that sometime, somewhere, somebody will produce a card with more than three voices and the software to drive it. In fact, my real dream is an interface to a true synthesizer, a la Moog, but the P-SST card will satisfy my immediate needs for quite some time. I think it will satisfy most of yours, too, if you are into computer-generated music and speech.

Note: Software Wizardry has told me that they are planning to release (in the near future) CP/M versions of some of this software for use on the 8085 side of the H/Z-100. Also in the works is an expanded P-SST card for the Z-150 PC family with six-voice sound.



Pricing:

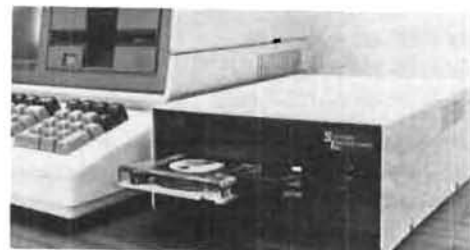
P-SST Card	\$395.00 (includes basic support software)
MAESTRO:	\$ 59.95
Develop. Libraries:	\$ 49.95 (includes routines for Microsoft BASIC, Microsoft FORTRAN, Microsoft Pascal, Microsoft (Lattice) C, and Computer Innovations C86 C Compilers)

Vendor:

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

Also available through most Heathkit Zenith Computer Stores.

GUARDIAN 25 25 MEGABYTE Z100 BACKUP



INTERFACES LIKE A FLOPPY

Can be used alone or as a Winchester backup

G25-Z1 Complete for Zenith Z100	\$1295.00
DC600 Certified Cartridge Tape	32.50
DC600-5 Cartridge Tape 5-pack	155.00

GUARDIAN DATA SYSTEMS

"Protecting your information investment"

44 STEDMAN ST., LOWELL, MA 01851

617-459-4449

CONVERT!

Raymond Dotson
214 S. Berkeley Blvd.
Goldsboro, NC 27530



CONVERT! A program that simulates artificial intelligence while converting between practically any two quantities. You correspond with the computer in plain English terms and receive its answers in a like manner.

Practically any conversion problem that can be stated in the following format will be answered.

HOW MANY ????? IN A ????? or
HOW MANY ????? IN 1 ????? or
HOW MANY ????? IN 47 ????? etc.

You may query in any combination of upper or lower case letters which will be converted to upper case automatically. The response will be similar to:

You will find N ????? in a ?????, or
The tables list N ????? in 1 ?????, or
My calculations show N ????? in 47 ?????, etc.

You may request the number of drops in a barrel, or the number of seconds in 23 years, or the number of decigrams in a myriagram, or the number of cents (or pennies) in 37 quarters, or the number of cubic-milliliters in 32.65 cubic-meters, etc. Some conversions between U.S. and Metric are also possible as the program is written. A more complete cross-conversion is easily added with an editor. Additional conversion strings may also be added for any other table you may need. Even unusual format questions will be answered such as: How many dollars in 213 quarters?

I have provided a separate data file containing the conversion information. This could well be contained in the main program,

however, I use a compiled version of CONVERT! and I am occasionally adding data conversion strings. In this manner, I am not required to recompile the main program after each change in the Data File.

CONVERT! digests your questions and determines the parameters and the quantities you have requested. It then searches its data file (DATA.CNV) for a parameter group containing those you requested. The conversion factor is calculated, also from data in the parameter group, and the answer is presented.

You may, at any time, request help by striking the "ESC" key when you are asked for a question. You will then receive a listing of all the parameters that are stored in the data file. You may peruse the entire parameter file or resume the conversion process at any time by again striking "ESC". Any two parameters contained in a single parameter group will be converted. Cross conversion between groups will be refused. An example would be attempting to CONVERT! between inches and cubic feet, obviously incompatible parameters.

Operation of the program is as follows: You ask a question such as "HOW MANY INCHES IN 3 FURLONGS?" (Question Mark NOT required). CONVERT! determines the third and last words to be parameters requiring conversion. The word or number string that is occupying the next-to-last position is taken to be the quantity of the second parameter. If "A" or "AN" is used, it is given the value of one (1). Any number used retains its value. The search routine finds a parameter group containing the two parameter words just determined. The group is digested starting at the first word, and all values between this word and the second

parameter word are multiplied together. This value is also multiplied by the quantity listed for the second parameter.

Regard the first parameter group in the file "DATA.CNV". You will find the string "INCH 12 FOOT 3 YARD 5.5 ROD 1 POLE 1 PERCH 40 FURLONG....". In our question above, CONVERT! identified this as being the parameter group containing both our requested parameters, INCH(es) and FURLONG(s). The numbers 12, 3, 5.5, 1, 1, and 40 were multiplied together and this product was multiplied by the quantity specified (3). The final product will be placed after the randomly selected response string (i.e. "The references indicate" and then followed by the latter part of the original question "inches in 3 furlongs.") This provides an answer of "The references indicate 23760 inches in 3 furlongs."

You will note in the same parameter group, that ROD, POLE, and PERCH are each separated by the number one (1). This is because they are like terms, each 5.5 yards in length. Therefore, they have multipliers of one (1). If you asked "How many rods in a perch," your answer would be "You will find 1 rod in a perch."

Since the computer can digest and "understand" your questions, although written in English, and since the answers randomly start with one of seven different responses, intelligence is simulated.

Program operation, although somewhat unusual in concept and execution, is nevertheless relatively straightforward. It is written for Zenith computers operating under CP/M 80, the eight-bit version 2.2.03, and using Zenith terminal codes and Microsoft BASIC. Conversion to other machines or BASIC varieties should offer no insurmountable obstacles. If you wish, you may follow along as program peculiarities are explained...

C\$ is the Escape code which is used with many other codes to designate console routines. UP\$ moves the cursor up one line. B1\$ moves it back one space without erasing the character moved over. Y\$ puts the terminal into direct cursor addressing mode. J\$ remembers the current cursor position, while K\$ is used to return to the remembered position. EL\$ clears the screen from the cursor to the bottom of the screen and EEL\$ clears the current line from the cursor position on. P\$ places the terminal into reverse video mode and Q0\$ returns it to normal. X1\$ activates the 25th (information) line and Y1\$ disables it. Likewise, the X4\$ and Y4\$ activates and deactivates the block cursor from the normal underline mode. BEEP\$ rings the terminal bell. CL\$ clears the screen and homes the cursor to the upper-left corner.

PEEK(11) samples the TIC counter for a seed to be used in the RANDOMIZE function, insuring a different starting number for each activation. The FND\$DEFINITION sets up a user defined function to permit placing the cursor at any point on the screen by specifying the "V" vertical and "H" horizontal locations. An example is used in line 500. Here we direct the cursor to line 8, column 28, turn on reverse video, print the program name, turn off reverse video and then proceed to show the program credits. Line 600 DIMensions a possible thirty parameter lines to be read in from the data file "DATA.CNV", then DEFines two variables, "T" and "V", as double precision and all the remainder as single precision. Line 700 provides a user defined round-off function to two decimal places and provides Q1\$ as a quotation mark (") to allow printing to the screen this normally unprintable character.

Line 900 reads the seven response strings into an array to be randomly selected later. The data file is then opened for input as

channel 1. Line 1000 reads into memory all the conversion strings contained in the file up to a maximum of thirty as DIMensioned in line 600. Lines beginning with an "*" are appended onto the previous line after removing the "*".

Lines 1300-1600 permit you to input a conversion question. The second part of line 1400 prints instructions on the 25th line telling you how to obtain HELP. Line 1600 requests your input and provides the first two words, "HOW MANY" so that your questions will all start in the proper manner. It then checks the first character of your input to see if it is the "ESCAPE" character. If so, the routine at line 5900 is called to print all known conversion parameters to the screen.

The routine at line 1800 first checks for and strips away ending punctuations from your question. The question is then converted to all upper-case characters. It is also converted to all lower-case and stored as Q1\$ and used later to formulate the computer's response. The question is then reprinted in the same position on the screen in all upper-case characters by line 2100. Lines 2200-2400 break the question into separate words, BIT\$(). Lines 2600-2700 check for the word "FEET" and convert any found to "FOOT" to simplify processing. Lines 2800-3000 remove the "ES" or "S" from all words with the exception of "MILES" for the same reason. The word "MILES" is stripped of the "S" only because if the "ES" were removed, the remaining term would be "MIL" which is also a correct term located in the same CONVersion group.

Lines 3100-3200 check the input parameters for "SQUARE", "CUBIC", and "FLUID" and, if found, add them to the next word with a connecting hyphen (-). Lines 3300-3400 check the next-to-last word or number string to insure that it is either "A", "AN", "ONE", or a numerical quantity. Anything else is refused.

The Search routine systematically checks each parameter group of the data file for a match with both input parameters. If no match is found, you are so informed. If the two parameters are found in the proper sequence, all is well. If the sequence is reversed, you are informed that the input format is unusual, however, the question will be accepted. An example of this would be if you asked "HOW MANY DOLLARS IN 24 QUARTERS." This is obviously an unusual request sequence, but the program will still answer in the proper manner.

The computational routine at line 4100 calculates the conversion factor by multiplying together all the numbers it finds between the two matched parameters and multiplying the product by the quantity you input for the final parameter.

Lines 5300-5500 assemble the computer's response by randomly selecting one of the seven response strings read into memory at line 900, appending the product obtained in the last routine and the last part of the lower-case version of your question.

The HELP routine at line 5800 first removes all the numeric strings from the data file and prints 6 parameter groups to the screen. The 25th line is then used to inform you that you may hit "ESC" to resume program operation, or any other key to display the next 6 parameter groups. This displays all the parameter groups known by the program (from the data file "DATA.CNV").

The ERROR trapping routine ends the program if you have no data file on the logged disk.

Speed of operation is quite rapid. Prompts are given on the 25th line to show options available. Accuracy is determined by the

conversion file data...garbage in, garbage out. I don't think I have fed it any garbage. If you disagree with any of my conversion factors, please let me know. I find CONVERT! to be a fun program and a superior method of conversion than any other I have encountered.

For those who dislike typing, if you will send a properly formatted Heath/Zenith hard or soft sector disk, I will copy both the program and data file to your disk for the handling fee of \$6.00. In addition, I will include a compiled version of the CONVERT! program (CONVERT!.COM). I use Zenith supplied CP/M-80 and Zenith disk drivers, 48 TPI only.

```
1 '
      INITIALIZATION ROUTINE and CREDITS
2 ON ERROR GOTO 68
3 C$=CHR$(27):UP$=C$+"A":B1$=C$+"D":Y$=C$+"Y":J$=C$+"j":
  K$=C$+"k":EL$=C$+"J":EEL$=C$+"K":P$=C$+"p":Q0$=C$+"q":
  X1$=C$+"x1":Y1$=C$+"y1":X4$=C$+"x4":Y4$=C$+"y4":
  BEEP$=CHR$(7):CL$=C$+"E":PRINT CL$
4 RANDOMIZE PEEK(11):
  DEF FND$(V,H)=Y$+CHR$(31+V)+CHR$(31+H):WIDTH 80:
  WIDTH LPRINT 80
5 PRINT FND$(8,28)P$=" CONVERSION UTILITY ="Q0$:
  PRINT FND$(10,27)"(C) 1984 by Raymond Dotson":
  PRINT TAB(30)"214 S. Berkeley Blvd.":
  PRINT TAB(30)"Goldsboro, NC 27530":
  PRINT TAB(33)"(919) 778-4112":
6 DIM CON$(30):DEFSNG A-Z:DEFDBL T,V
7 DEF FNR(X)=INT(X*1000+.5)*.001:QI$=CHR$(34)
8 '
      READ DATA
9 FOR I=1 TO 7:READ BEC$(I):NEXT:OPEN "I",1,"DATA.CNV":I=1
10 IF EOF(1) THEN NB=I-1:CLOSE:PRINT CL$:
  GOTO 13 ELSE LINE INPUT #1,CON$(I):
  IF LEFT$(CON$(I),1)!="*"
  THEN CON$(I-1)=CON$(I-1)+MID$(CON$(I),2):GOTO 10
11 I=I+1:GOTO 10
12 '
      ROUTINE TO INPUT QUESTION
13 PRINT FND$(4,12)
  "You may ask any question concerning conversions
  such as":PRINT:
  PRINT TAB(12)"How many inches in 2 yards":PRINT:
  PRINT TAB(12)
  "Use English words and I will answer in English words"
14 PRINT TAB(12)"after performing the required conversions."
15 PRINT J$X1$FND$(25,34)P$"ESC"Q0$ for HELP"K$FND$(14,1):
16 PRINT:PRINT:PRINT UP$BEEP$EEL$SPC(2)J$
  "Your Question? HOW MANY ":A$=INPUT$(1):
  IF A$=C$ THEN 59 ELSE IF A$=CHR$(13) THEN PRINT Y1$:
  PRINT:END ELSE PRINT A$::LINE INPUT Q$:Q$=A$+Q$
17 '
      ROUTINE TO PROCESS QUESTION
18 EN$=RIGHT$(Q$,1):IF EN$="?" OR EN$=","
  THEN Q$=LEFT$(Q$,LEN(Q$)-1)
19 FOR I=1 TO LEN(Q$):
  IF MID$(Q$,I,1)>="a" AND MID$(Q$,I,1)<="z"
  THEN MID$(Q$,I,1)=CHR$(ASC(MID$(Q$,I,1))-32)
20 NEXT:Q1$=Q$:FOR I=1 TO LEN(Q1$):
  IF MID$(Q1$,I,1)>="A" AND MID$(Q1$,I,1)<="Z"
  THEN MID$(Q1$,I,1)=CHR$(ASC(MID$(Q1$,I,1))+32)
21 NEXT:I=1:PRINT K$ HOW MANY "Q$"?
22 J=INSTR(1,Q$," "):IF J=0 THEN BIT$(I)=Q$:N=I:GOTO 25
23 BIT$(I)=LEFT$(Q$,J-1):Q$=MID$(Q$,J+1)
24 I=I+1:GOTO 22
25 FOR I=1 TO N
26 FOR U=1 TO LEN(BIT$(I)):IF MID$(BIT$(I),U,4)="FEET"
  THEN MID$(BIT$(I),U,4)="FOOT"
27 NEXT
```

```
28 IF BIT$(I)<>"MILES" AND RIGHT$(BIT$(I),2)="ES"
  THEN BIT$(I)=LEFT$(BIT$(I),LEN(BIT$(I))-2)
29 IF RIGHT$(BIT$(I),1)="S"
  THEN BIT$(I)=LEFT$(BIT$(I),LEN(BIT$(I))-1)
30 NEXT
31 IF BIT$(1)="SQUARE" OR BIT$(1)="CUBIC" OR BIT$(1)="FLUID"
  THEN BIT$(1)=BIT$(1)+"-"+BIT$(2):FOR I=2 TO N:
  BIT$(I)=BIT$(I+1):NEXT:N=N-1
32 IF BIT$(N-1)="SQUARE" OR BIT$(N-1)=
  "CUBIC" OR BIT$(N-1)="FLUID"
  THEN BIT$(N-1)=BIT$(N-1)+"-"+BIT$(N):N=N-1
33 V=VAL(BIT$(N-1))
34 IF V=0 AND BIT$(N-1)<>"A" AND BIT$(N-1)<>
  "AN" AND BIT$(N-1)<>"ONE" THEN PRINT TAB(14)
  "You must not spell your numbers! Use digits.":
  GOTO 16
35 '
```

```
ROUTINE TO SEARCH DATA
36 EXC=0:FOR I=1 TO NB
37 J=INSTR(1,CON$(I)," "+BIT$(1)):IF J=0 THEN 39
38 J1=INSTR(1,CON$(I)," "+BIT$(N)):IF J1>0 THEN 40
39 NEXT I:PRINT TAB(12)
  "I am not programmed to answer that question.":
  PRINT TAB(15)"or else the elements are dissimilar.":
  GOTO 16
40 IF J>J1 THEN SWAP J,J1:EXC=1:
  PRINT TAB(22)"Unusual format, however ..."
41 '
      ROUTINE TO COMPUTE VALUES
```

```
42 X=0:J=J+1:TOT=1
43 X=X+1:J=INSTR(J+1,CON$(I)," "):IF J>=J1 THEN 46
44 IF X/2=INT(X/2) THEN 43
45 VA=VAL(MID$(CON$(I),J)):IF VA THEN TOT=TOT*VA:GOTO 43
46 IF V THEN IF EXC=0 THEN TOT=TOT*V:GOTO 53
47 TA=TOT:IF EXC THEN TOT=1/TOT:IF V THEN TOT=V/TA
48 IF INT(TOT)<>1 THEN 53
49 Q0=INSTR(1,Q1$," ")
50 IF MID$(Q1$,Q0-1,1)="s"
  THEN Q1$=LEFT$(Q1$,Q0-2)+MID$(Q1$,Q0)
51 IF LEFT$(Q1$,Q0-1)="feet" THEN Q1$="foot"+MID$(Q1$,Q0)
52 '
      ROUTINE TO PRINT RESPONSES
```

```
53 PRINT EEL$TAB(14)BEC$(INT(RND(1)*7)+1):
54 IF TOT>100000 THEN PRINT TOT:Q1$="":GOTO 16
55 PRINT FNR(TOT):Q1$="":Q$=F$:GOTO 16
56 '
      RESPONSE DATA
```

```
57 DATA There are,You will find,My Data Banks show,
  The tables list,The references indicate,
  My calculations show,The charts show
58 '
      HELP ROUTINE
```

```
59 PRINT CL$Y1$FND$(4,20)"Parameters I currently understand:
  ":PRINT:PRINT:PRINT J$:
60 FOR I=1 TO NB:CT$=CON$(I)
61 J=INSTR(1,CT$," "):J1=INSTR(J+1,CT$," "):
  IF J1=0 THEN J1=LEN(CT$):PRINT MID$(CT$,J):GOTO 64
62 PRINT MID$(CT$,J,J1-J) " ":CT$=MID$(CT$,J1)
63 J=INSTR(1,CT$," "):CT$=MID$(CT$,J+1):GOTO 61
64 IF I/6=INT(I/6) THEN PRINT X4$X1$FND$(25,4)P$
  "ESC"Q0$ to resume"SPC(16)P$"Hit any Key!"B1$::
  A$=INPUT$(1):PRINT Y1$Y4$K$EL$Q0$::
  IF A$=C$ THEN PRINT CL$:GOTO 15
65 PRINT:NEXT:PRINT X4$X1$FND$(25,34)P$
  "Hit any Key!"B1$::A$=INPUT$(1):PRINT Y1$Y4$K$Q0$:
66 PRINT CL$:GOTO 15
67 '
      ERROR ROUTINE
```

```
68 IF ERL=9 THEN PRINT CL$FND$(6,27)QI$
  "DATA.CNV"QI$ was not found!":PRINT:PRINT:END
```

MIL 1000 INCH 12 FOOT 3 YARD 5.5 ROD 1 POLE 1 PERCH
 40 FURLONG 8 MILE 3 LEAGUE
 SQUARE-INCH 144 SQUARE-FOOT 9 SQUARE-YARD 30.25 SQUARE-ROD
 160 ACRE 640

- * SQUARE-MILE
 CUBIC-INCH 1728 CUBIC-FOOT 27 CUBIC-YARD 128 CORD
 DROP 60 TEASPOON 2.85714 CUBIC-INCH 1.05 TABLESPOON
 2 FLUID-OUNCE 4 GILL 2
- * CUP 2 PINT 2 QUART 4 GALLON 2 PECK 4 BUSHEL 3.94 BARREL
 GRAIN 27.34 DRAM 16 OUNCE 16 POUND 2000 TON
 MILLIMETER 10 CENTIMETER 10 DECIMETER 10 METER 10
 DECAMETER 10
- * HECTOMETER 10 KILOMETER 10 MYRIAMETER
 SQUARE-MILLIMETER 100 SQUARE-CENTIMETER 100
 SQUARE-DECIMETER 100
- * SQUARE-METER 100 SQUARE-DECAMETER 100
 SQUARE-HECTOMETER 100 SQUARE-KILOMETER
 SQUARE-METER 1 CENTIARE 100 ARE 100 HECTARE
 100 SQUARE-KILOMETER
 CUBIC-MILLIMETER 1000 CUBIC-CENTIMETER 1000
 CUBIC-DECIMETER 1000 CUBIC-METER
- MILLILITER 10 CENTILITER 10 DECILITER 10 LITER 10
 DECALITER 10
- * HECTOLITER 10 KILOLITER
 MILLIGRAM 10 CENTIGRAM 10 DECIGRAM 10 GRAM 10
 DECGRAM 10 HECTOGRAM 10
- * KILOGRAM 10 MYRIAGRAM 10 QUINTAL 1 TON
 SECOND 60 MINUTE 60 HOUR 24 DAY 30 MONTH 12 YEAR 10
 DECADE 2 SCORE 5
- * CENTURY 10 MILLENIUM
 SECOND 60 MINUTE 60 DEGREE 90 QUADRANT 4 CIRCLE
 CENTIMETER 2.54 INCH 36 YARD 1.0936 METER
 PENNY 1 PENNIE 1 CENT 5 NICKEL 2 DIME 2.5 QUARTER
 2 HALF 1 HALVE 2 DOLLAR



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 Pro-writer 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

MEDIA MASTER™

"Foreign" File Compatibility Through Disk-To-Disk Transfers On Your Z-100™ or Z-150™

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 breaks down the 5¼" disk format barrier by allowing your computer to READ, WRITE, and FORMAT over 70 different "foreign" disks.

MEDIA MASTER is easy to use. You just LOG IN your "foreign" disk prior to copying files to or from another floppy or hard disk. Through simple menu steps you can also display directories and type, print, or erase files.

To order, send check or money order for \$39.95 + \$2.50 S/H (Cal. residents please add 6%). To order COD (we pay all COD fees!), call our 24-HOUR TOLL FREE NUMBER: 800-824-7888, and ask for Operator 251. (Z-150 owners - please specify the IBM PC version). **Dealer Inquiries Invited**



4573 Heatherglen Ct.
 Moorpark, CA 93021
 Technical Questions? 805-529-5073

MEDIA MASTER supports the following disk formats:

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

Also available for the DEC Rainbow, Osborne, and Kaypro computers. Call or write for information.

The FORTRAN Formula-4

Dick Stanley
P.O. Box 9512
Alexandria, VA 22304

In the last article, we learned the basics of stock market technical analysis, and also some more about the FORMAT statement. We scratched the surface of using DO loops, and ended on the promise of beginning some real calculations this time. So we shall — but first, as they say, a few words about some details of FORTRAN that we shall need to know.

Quite An Array Of Things

You probably noticed in our previous discussion of stock market indicators that the point was made about the need to discern a trend from these, and not take one value of an indicator on a given day as being particularly meaningful in itself. This infers that we must keep track of the value of a certain indicator for a period of time, in order to see which way it is trending. Given the considerable ability of our computer to do this laborious clerical task, we would prefer to do this record-keeping with the machine, rather than with ruled paper and pencil.

We can certainly do that, but what is the easiest way? Each day's value of each indicator needs a name so that we can refer to it. We have already named the input variables, and those we will calculate, so perhaps we can simply add a serial digit to the variable name. That will work (so long as we shorten the name to keep to FORTRAN's limit of no more than 6 characters to a variable name). However, if we do things that way, it will become rather awkward to refer to the variables in repetitive processing, as we would have to refer to each by its proper name, rather than, say, the fifth one from the beginning.

There is an easier way. We can form an array of each variable. An array is simply an ordered set of data sharing the same symbolic name. Each element of the array is distinguished from the others by its place in the order, which is given by its subscript. This is really no different than the approach used by BASIC, and the notation looks just the same: FRED(3) is the third element in the array named FRED, which is a real variable in FORTRAN because it begins with a letter other than I through N.

Arrays may have up to three dimensions (subscripts) in FORTRAN, and the subscripts must be integers. They are separated by commas when there is more than one subscript. The subscripts may be calculated within the parentheses (except for division), but they may not themselves be subscripted. The following are valid array elements:

```
A(3,4)   FOO(2*J,7)   I(56)   FRED(1,2+N,5)   X(I*M-3).
```

The following are NOT valid array elements:

```
A(B(6),7)   FRED(1,2/N,6)   X(I(3+J)).
```

When you are using arrays, you must inform the compiler of your intent, so that it can allocate enough storage for the array when it compiles the program. This is not a problem with individual variables, because space is allocated as the variable is first encountered. However, the compiler is not clever enough to figure out how large you might wish to make the array, so it needs some help. In FORTRAN, this is done by using the DIMENSION or TYPE statements, for example:

```
DIMENSION FRED(6,6,9), TEST(15).  
REAL FOO(12)
```

The DIMENSION statement above creates a 3-dimensional array named FRED, with dimensions of 6, 6, and 9 for a total of 324 elements, and a one-dimensional array named TEST with 15 elements. The REAL type statement creates a one-dimensional array of twelve elements named FOO. There is no need to both dimension and type an array: declaring it in a TYPE statement is sufficient. The array elements are numbered from 1 to the number stated in the DIMENSION specification. The first elements in the arrays above are FRED(1,1,1) and TEST(1). There is no TEST(0).

This is different from the situation in BASIC. In that language, an array can have up to 255 dimensions, and it can be declared in a limited way without specifically using the DIM statement. BASIC assumes that the first term in every array is numbered 0, however, unless the OPTION BASE statement has been used before the array is declared. I make these points because you will have to be a little careful in FORTRAN if you are already very familiar with array handling in BASIC. As a practical matter, the differences are minor. In over 20 years of programming, I can't recall ever needing an array of more than 3 dimensions. If you should, you can "fake" it with several smaller arrays. I find the "implicit definition" of arrays in BASIC to be troublesome: it always defines an array of size 10, and I always seem to need 11 terms! Better, in my view, to let the compiler make you define what you want in the first place. I happen to like having the first array term numbered 1, but that may be a historical prejudice.

You should remember that creating an array does not initialize the values of the elements to anything in particular. If it is important that they be of some value before you fill or read the array, you must initialize the array in your program. We'll show some of this later in the series.

Newer versions of FORTRAN (FORTRAN-77 and beyond) permit more latitude in specifying the declaration of arrays, to include the use of negative subscripts. If in doubt as to what your version

permits, check the specification manual that came with the compiler. In keeping with our policy of adhering to FORTRAN-66 standards, we shall limit our arrays to the standards described above.

What Good Are They?

We will find that our stock market data, both inputs and calculated values, can be easily organized and tracked if we define them to be arrays. Each subscript will represent a day, and we can readily extract elements from the past to calculate moving averages, draw graphs, or what have you. For ease of referring to the elements of each array, today will be element 1, yesterday will be element 2, the day before yesterday will be element 3, and so on.

How many days data shall we keep track of? Well, we will probably want to produce a graph in the future to see what trends may be developing, so it would be a good idea to keep a reasonable number of previous data points. As our CRT screens are 80 characters wide, and since any graphs will probably be displayed there with some space lost for labels, etc., let's choose to keep the past 60 days of data for each variable. We will need to remember to DIMENSION each variable as being of size 60, and each has one dimension.

If you quickly count up the variables we discussed last time, you will find there are 19 of them. We have just said that each will be represented as an array of 60 elements each, for a total of $19 \times 60 = 1140$ data elements kept in storage. Will they all fit in memory?

Let's take a look. We will make a matrix of the variables, see how many bytes are required to store one element of each, and then multiply the result by 60. That should tell us how much memory these arrays will occupy, and whether or not we have made a realistic choice for representing this data. Here is the matrix:

Variable	Type	Bytes of Storage Per Element
DJIA	REAL	4
VOLUME	INTEGER*4	4
ADVNC	INTEGER	2
DECLN	INTEGER	2
UNCHNG	INTEGER	2
UPVOL	INTEGER*4	4
DNVOL	INTEGER*4	4
HIGHS	INTEGER	2
LOWS	INTEGER	2
ADLINE	INTEGER*4	4
DJIPCT	REAL	4
ADPCT	REAL	4
ADVVOL	INTEGER*4	4
DECVOL	INTEGER*4	4
TRIN	REAL	4
HLDIF	INTEGER	2
DAY	INTEGER	2
MO	INTEGER	2
YR	INTEGER	2

Total		$58 \times 60 = 3480$ bytes storage.

We can certainly afford a little over 3K of storage in our system for these arrays, and they should cause no problem. Where did I find out how many bytes each type of variable used? Check your FORTRAN manual under "Storage Allocation" or "Data Representation", and you should find a table of some sort with the requisite data.

Adding A Little Control

Programming demands the ability to control the flow of operations. So far, the only way we have seen to do that in FORTRAN is with the DO loop. There are several other ways, and it will be helpful to look at a couple of those now, as we will need the capabilities in subsequent installments. First — and most infamous — is the GO TO statement. Misuse of the GO TO statement, probably more than any other factor, led to the development of structured programming (in which there are no GO TO statements). Nonetheless, the statement is useful if it is not abused.

There are three sorts of GO TO's in FORTRAN: unconditional, arithmetic, and assigned. We will quickly look at each one. Please note that GO TO is two words in FORTRAN, and not GOTO as used in BASIC.

The unconditional GO TO does precisely what its name implies. When the statement is encountered, control is transferred unconditionally to the place the GO TO points. Thus, the statement

```
GO TO 116
```

will transfer control to statement 116 each and every time it is encountered. No statements between this GO TO and 116 will be executed unless they can be reached by another path in the program. This is one of the dangers of the GO TO statement: it is easy to bypass whole segments of the program, leaving only "dead code." Another danger is that this statement makes it simple to create a rat's nest of branching through a program that no ordinary mortal can follow. We won't do that here, and you shouldn't do it anywhere else, either. Nevertheless, when you need to move quickly from here to there in a program, this is a sure-fire way to do it.

The computed GO TO allows you to transfer control depending on the value of a variable. It looks like this:

```
GO TO (k1, k2, ... kn), j
```

where k1 through kn are statement numbers, and j is a positive integer variable having a value between 1 and n. It works like this: if j=1, then control is passed to statement k1; if j=2, then control goes to k2, and so on up to kn. If j is greater than n or less than 1, control passes to the statement following the GO TO.

Confused? Look at this sequence of statements:

```
GO TO (100, 750, 6, 53), NUM
91 X = 37.539
```

Here is where control will be passed for various values of NUM:

```
NUM = 1      Next statement executed ==> 100
NUM = 2      Next statement executed ==> 750
NUM = 3      Next statement executed ==> 6
NUM = 4      Next statement executed ==> 53
NUM >= 5    Next statement executed ==> 91
NUM <= 0    Next statement executed ==> 91
```

That wasn't too bad, was it? You can already see that there is likely

to be some very handy uses for this sort of ability.

There is a third type of GOTO, the assigned GOTO. It is similar to the arithmetic GOTO in its functioning, and we will defer talking about it until a later installment.

What IF?

There must be a conditional program control capability if the language is to be useful, and FORTRAN also has this, in the form of the IF statement. There are two sorts of IF statements.

The arithmetic IF statement transfers control based on the value of an arithmetic expression. It looks like this:

```
IF(e) m1, m2, m3
```

where e is an arithmetic expression, and m1, m2, and m3 are statement numbers. It works very simply. The expression e is evaluated. If its value is negative, control is transferred to m1; if the expression is precisely zero, control transfers to m2; and if the expression is positive, control transfers to m3. Here's a practical example:

```
IF(N-7) 27, 105, 63
```

will transfer control to statement 27 for $N \leq 6$, to statement 105 for $N = 7$, and to statement 63 for $N > 8$.

FORTTRAN also supports a logical IF statement, which takes the form

```
IF(u) statement
```

where u is a logical expression and statement is an executable FORTRAN statement other than DO. This is similar to the IF in BASIC. An example might be:

```
IF(N=J) A(J)=3.0*FOX(N)
```

When the value of N equals the value of J, the value of A(J) will be replaced by three times the value of FOX(N). If N does not equal J, nothing will be done to the value of A(J).

There is no IF..THEN..ELSE construct in FORTRAN-66, as there is in BASIC. This feature is found in FORTRAN-77 and higher versions, where it is called the block IF. In most of our compilers, we won't have block IFs, so we will discuss means to improve the handling of IF statements as we go. If you have a FORTRAN compiler that implements block IFs, not to worry: the things you learned in BASIC, Pascal or dBASE II about setting up structured IF statements apply.

Saving To Disk

This project involves a lot of numerical data entry. Obviously, we won't want to reenter the data each time we want to look for trends. We certainly don't want to have to reenter 59 days worth of data to compare them to the data for today that was just entered. We want to be able to save what has been done in a disk file. Doing this isn't hard, but it isn't obvious from the documentation.

Before you can write or read a file, you must open it. There is a couple of ways to do that, but the way that makes the most sense in a CP/M or MS-DOS environment is as follows:

1. Open the file using the OPEN subroutine:

```
CALL OPEN(LUN, filename, drive)
```

where: LUN=logical unit number (6 through 10 are disk

drives); filename is the file name you want to use, less the drive designator; and drive is an integer which represents the drive you want the file on, with 0 being the currently logged drive, 1 being Drive A:, 2 being Drive B:, and so on. The filename must be enclosed in single quotation marks.

2. If the file exists, the OPEN subroutine will open it for reading or writing. If it does not exist, the OPEN subroutine will create it on the drive indicated. CAUTION: if you write to an existing sequential file, the previous contents of the file will be deleted and replaced with what you just wrote.
3. You can read or write to a file by using the statements:

```
READ(LUN, ERR=k1, END=k2) k
```

```
WRITE(LUN, ERR=k1, END=k2) k
```

where LUN is the Logical Unit Number you are writing to, k1 is the statement where control will transfer in the event of an I/O error, k2 is the number of the statement where control will be transferred when the End-Of-File is encountered, and k is a list of the variables to be read or written, separated by commas. This type of I/O is called Unformatted disk I/O. It is also possible to format the output to the disk, but more about that later. This will do fine for now.

Putting it all together, to write a file named STOCK.DAT on drive B: containing today's values of the DJIA and the Trader's Index, TRIN, we would write

```
CALL OPEN(6, 'STOCK.DAT', 2)
WRITE(6, ERR=1000, END=1050) DJIA(1), TRIN(1)
```

(statements 1000 and 1050 point to error and end-of-file handling routines that we won't worry about here).

To write the values of those quantities for the past 60 days, we would write something like this:

```
DIMENSION DJIA(60), TRIN(60)
:
:
:
CALL OPEN(6, 'STOCK.DAT', 2)
DO 400 J=1,60
400 WRITE(6, ERR=1000, END=1050) DJIA(J), TRIN(J)
```

Since we had to open the file to read or write, you might think that we should close it when we are finished. We do, but not quite in the way you might think. Remember, FORTRAN grew up when magnetic tape was the primary means of mass off-line data storage, so its syntax reflects that fact (sort of like the TTY: device in CP/M that recalls the days when a teletype was the "standard" operator I/O mechanism). To close a file on Logical Unit Number u when we are done with it, the command is ENDFILE u. To close the file on LUN u and immediately reopen it (handy when you want to be sure everything has been written to the file, but you need to work with it immediately), the command is REWIND u.

There is also a BACKSPACE u command in FORTRAN. On a magnetic tape drive, it moves the tape backwards one record. This is meaningless on a microcomputer with disk drives, however, so this command is usually not implemented, and will return an error if you use it.

What Now?

Let's see if this system can really calculate some data for us. Get

out the listing for STOXIN.FOR that we wrote in the last installment of this series, and modify it so that your listing looks like that in Figure 1. Then compile the resultant program and execute it. Using the same data that we input in the last installment, your output should look like the screen in Figure 2. Try using different data to get the feel of how the program operates, and what goes wrong when bad data is entered.

Today's Stock Statistics	
12/12/84	
Percentage change in the DJIA:	.5%
Today's Advance-Decline Line value:	-152.
Percentage change in the A/D line:	-7.6%
High-Low Differential:	-24.
TRader's INdex:	.64

Figure 2. Screen image of the output from the compiled program CALC.FOR, for the same data as shown in Figure 4 of the last installment of this series.

Examine the program listing. Notice that the method of clearing the screen has changed. There is a FORMAT statement used here that you haven't seen before: the A specification is used. This is one of a type of formats called Hollerith Conversions, and we'll learn more about it shortly. Notice how nicely the screen clears now that we are able to send the proper ESCape code to it, as compared to listing 24 blank lines as we did last time.

If you look a little further, you will see that wherever I wanted to print an apostrophe (single quote), I had to enter two apostrophes in the FORMAT statement. This is because the compiler has no way to know when it finds the second apostrophe that it has not found the end of the FORMAT field unless you double the apostrophe. To see what happens without doubling, recompile the program with the second apostrophe removed, and notice what manner of message you receive from the compiler.

There are some strange goings-on in the calculation section of the program that we just wrote. We seem to go to a lot of trouble to calculate TRIN, and we use some function not spoken of previously, that is apparently called FLOAT. Right! The variables that are required to be divided to calculate TRIN are all integers. If we perform the division, the answer is not what we expected, because there is no decimal part — that is discarded in integer division. We must first translate these integers into real, floating-point representations of the same numbers before we perform this division, so that the answer is meaningful. FLOAT does this for us. Although it is not documented in many FORTRAN manuals, it is almost always there, and it is used to transform an integer value into a floating-point value so that we can operate on it using non-integer arithmetic. Remove the FLOATs from the program, recompile it, and notice the value you now have for TRIN. It is zero, which is "obviously" not correct. It isn't what you expect or want, but it is correct, because that is the result when you use integer division.

What's Going On Here?

Be careful when entering your program. While debugging the program for this article, I couldn't get the value of the High-Low Differential, HLDIF, to print out. The compiler kept giving a **DT** error, which means that the data type specified by the FORMAT statement doesn't match the data type of the variable. I

had gone to some trouble to explicitly declare HLDIF as an integer variable, so this puzzled me, because the FORMAT specification was I5. After much chasing of gremlins, I noticed that HLDIF was the last variable listed in the INTEGER declaration, and it extended beyond Column 72 on the terminal. You'll remember (I hope) from the beginning of this series that everything beyond Column 72 winds up in the Great Bit Bucket in the Sky, and my declaration of HLDIF was no different. Moving it to a location where the compiler could read the entire variable name made all the difference. If you have similar troubles, check the length of your lines.

What's Next?

In this article, you have learned about program control statements, disk file operations, and mixed-mode (integer and real) calculations. We have gone into the use and structure of arrays in some detail, and begun to use them in our program. In the next installment, we will fill the data arrays with data, and show you how to calculate moving averages using data stored in arrays. We will save our data and results on disk, so that they can be easily used in subsequent programs. Until then, experiment with the program as it is. See what happens when you put in both good and bad data, and try out various combinations of I/O specifications to see what happens to the screen presentation when you do so.

A Quick Thanks

To all those who have written me about this series, many thanks. Your enthusiasm and suggestions are a great help. I have not yet been able to answer all your letters, but I'm working on it, and hope to have them all done before you read this. Several of you asked how to plot a graph in FORTRAN; stay with us, because we're going to do just that with some of the data we calculate in this series. We'll do it assuming a plain vanilla H89, so the results should be applicable to several other terminals, as well. In the meantime, keep those cards and letters coming!

Figure 1. Program listing for program CALC.FOR, built from the base of STOXIN.FOR which was prepared in the last installment.

```

C
C PROGRAM STOXIN
C This program inputs stock market data for analysis
C
C Declare variables
INTEGER MO, DAY, YR, ADVNC, DECLN, UNCHNG, HIGHS, LOWS, ESC, CLRSN
INTEGER*4 VOLUME, ADVVOL, DECVOL
INTEGER ADLINE(60), HLDIF
REAL DJIA(60), ADPCT
C
C Set up FORMAT specifications for I/O
FORMAT(1X)
FORMAT(19X, 'Stock Market Technical Analysis System')
FORMAT(' Enter the date as MM/DD/YY: ')
FORMAT(12,1X,12,1X,12)
FORMAT(' Enter the data indicated: ')
FORMAT(' Closing Dow Jones Industrial Average: ')
FORMAT(F8.2)
FORMAT(' Volume of shares: ')
FORMAT(110)
FORMAT(' Advancing Volume: ')
FORMAT(' Declining Volume: ')
FORMAT(' Number of Issues Advancing: ')
FORMAT(I5)
FORMAT(' Number of Issues Declining: ')
FORMAT(' Number of Issues Unchanged: ')
100
110
111
112
113
114
115
116
117
118
119
120
121
122
123

```

```

124 FORMAT(' Number of New Highs: ')
125 FORMAT(' Number of New Lows: ')
126 FORMAT(' +',.65X,I2,'/',I2,'/',I2)
127 FORMAT(' +',.65X,FB.2)
128 FORMAT(' +',.63X,I10)
129 FORMAT(' +',.68X,I5)
130 FORMAT(' +',.2A1)
C
C Clear screen
ESC = 27
CLRSCN = 69
WRITE(1,130) ESC, CLRSCN
C
C Get data. print value input at right side of screen on same line
WRITE(1,110)
WRITE(1,111)
READ(1,112) MO, DAY, YR
WRITE(1,126) MO, DAY, YR
WRITE(1,113)
WRITE(1,114)
READ(1,115) DJIA(1)
WRITE(1,127) DJIA(1)
WRITE(1,116)
READ(1,117) VOLUME
WRITE(1,128) VOLUME
WRITE(1,118)
READ(1,117) ADVVOL
WRITE(1,128) ADVVOL
WRITE(1,119)
READ(1,117) DECVOL
WRITE(1,128) DECVOL
WRITE(1,120)
READ(1,121) ADVNC
WRITE(1,129) ADVNC
WRITE(1,122)
READ(1,121) DECLN
WRITE(1,129) DECLN
WRITE(1,123)
READ(1,121) UNCHNG
WRITE(1,129) UNCHNG
WRITE(1,124)
READ(1,121) HIGHS
WRITE(1,129) HIGHS
WRITE(1,125)
READ(1,121) LOWS
WRITE(1,129) LOWS
End of STOXIN Original version
C
C Program CALC
C This program calculates technical stock indicators from
C data input from the keyboard
C
C Advance-Decline Line
ADLINE(1) = ADLINE(2) + ADVNC - DECLN
C
C Dow Jones Industrial Average percentage change
(this line fakes a value for DJIA(2) which we haven't entered)
DJIA(2) = DJIA(1) - 5.66
DJIPCT = (DJIA(1) - DJIA(2)) * 100.0 / DJIA(2)

```

```

C Advance-Decline percentage change
ADPCT = (ADVNC - DECLN) * 100.0 / (ADVNC + DECLN + UNCHNG)
C
C Trader's Index
TADV = FLOAT(ADVNC)
TDEC = FLOAT(DECLN)
TUPV = FLOAT(UPVOL)
TDNV = FLOAT(DNVOL)
TRIN = (TADV/TDEC)/(TUPV/TDNV)
C
C High-Low differential
HLDIF = HIGHS - LOWS
C
C Calculation finished. Set up formats to display results
150 FORMAT(' Today's Advance-Decline Line value: ',I8)
151 FORMAT(' Percentage change in the DJIA: ',F6.1,'%')
152 FORMAT(' Percentage change in the A/D line: ',F6.1,'%')
155 FORMAT(' Trader's Index: ',F6.2)
156 FORMAT(' High-Low Differential: ',I5)
157 FORMAT(27X, 'TODAY'S STOCK STATISTICS')
158 FORMAT(36X, I2, '/', I2, '/', I2)
C
C Print the results
WRITE(1,130) ESC, CLRSCN
WRITE(1,100)
WRITE(1,157)
WRITE(1,158) MO, DAY, YR
WRITE(1,100)
WRITE(1,100)
WRITE(1,151) DJIPCT
WRITE(1,150) ADLINE(1)
WRITE(1,152) ADPCT
WRITE(1,156) HLDIF
WRITE(1,155) TRIN
C
C END

```



**EXPLORE
NEW WORLDS
WITH
HUG
GAME
SOFTWARE**





High Powered Ammunition

The pros at First Capitol Computer know what you want—high powered ammunition for your Zenith. So we've engineered a series of the highest quality hard drives available for your Zenith 150 PC. If you're a power PC user, and only have a floppy based system, you're probably wishing for the power of a fast hard disk. Or maybe you have the standard 11 Mb drive and want something larger. Like 36 Mb. Use the full capabilities of powerful programs like Ashton Tate's D-Base III™ and Framework™; Lotus 1-2-3™ and Symphony™, and Micropro Wordstar™.

A Full Range Of Calibres

Our drive kits come in a size for every need; the standard 11 Mb, a serious 20 Mb, and a full bore 36 Mb.*

Our drives will allow autoboot from the hard disk, and require no permanent modifications to your existing hardware, and no changes to your operating system software—we've got everything needed in ROM on the controller board, customized for the Zenith PC.

Technical Wizardry

Our drive kits are engineered by Software Wizardry, the Zenith experts the experts consult, and are available exclusively through First Capitol Computer. You can count on leading-edge performance; quiet, smooth, and fast operation that will speed you through your most serious software.

All Hard Disks Are Not Created Equal

A lot of people are selling hard disks at a lot of prices—know what you are getting! Rapidly improving technology has resulted in the dumping of obsolete drives on the market. Not at First Capitol Computer!

Our drive kits use the latest state-of-the-art drives, with the newest media, head designs, fast access time, and low power consumption. You can be assured of the highest quality, equal to Zenith's original manufacturing standards.

Firepower

Load A High Calibre Winchester In Your Zenith

Value Priced

Our prices are some of the most competitive around, and when you consider our latest state-of-the-art components, First Capitol's hard disk upgrades are the best value on the market for your Z-150 PC series.

All You Need To Know

You don't have to be a technical wizard to install our systems—our wizards have done that for you! You get the drive, controller card, all necessary cables, and complete instructions.

Available Models:	List Price
SWI-WIN150-11 11 megabyte* internal hard disk kit	\$ 895.00
SWI-WIN150-20 20 megabyte* internal hard disk kit	\$1195.00
SWI-WIN150-36 36 megabyte* internal hard disk kit	\$1595.00

Available direct from First Capitol Computer. Please add \$2 minimum (or 2%, whichever is greater) for shipping and handling. If shipped to a Missouri address, please add appropriate sales tax.



First
Capitol
Computer

First Capitol Computer is a division of Software Wizardry, Inc.

1106 First Capitol Drive
St. Charles, MO 63301

Express Order Line (orders only please)

1-(800)-TO-BUY-IT (1-800-862-8948)

(314) 946-1968 (technical info)

*drive sizes are unformatted capacities

Displaying Text On The Z-100 PCs Part I

Mark J. Foster
Senior Systems Engineer
Systems Software Engineering
Zenith Data Systems Corporation

Welcome once again! Last month's column focused on setting up the various video modes and scroll modes used in the Z-150 and Z-160. To follow up on this theme, this month we'll begin to look at different methods of displaying characters on the screen.

Displaying Characters

When creating any program for a computer, you must make a tradeoff between portability (the ability to run the program on a wide variety of computer systems) and performance (speed and/or flexibility). There is no question that you can achieve maximum portability by using the operating system to perform Input/Output for you. Unfortunately, operating systems are, by their nature, designed for maximum portability, and consequently, usually offer you little flexibility. In addition, the "overhead" of most operating systems is very substantial, which means that your program will display characters slowly. Other notable disadvantages are that the DOS itself frequently will not allow access to color or other special video attributes.

Another approach to performing I/O (and one which is done by most of the major PC-compatible programs on the market today) is to let your program perform its I/O by communicating directly with the video hardware. This allows your program to achieve maximum performance, at the expense of poor portability. One major advantage of computers which are compatible with the IBM-PC is that they can take advantage of these fast, flexible programs (since they present a similar hardware interface). Unfortunately, this approach requires a lot of work on the part of the programmer, and even simple programs require a great deal of development in order to create the I/O drivers. Despite the disadvantages of this scheme, the ultimate control your program can have over the hardware is persuasive enough that we will cover some direct I/O in future columns.

The last approach is the one we will focus on this month; displaying characters by using the ROM BIOS which is built in to each Z-150 and Z-160. This approach combines some of each of the advantages listed above, by improving portability as compared to direct I/O, and improving speed and flexibility as compared to

DOS I/O. Because of the wide variety of video modes and scroll modes used in the Z-100 PCs, it may seem that displaying text is a complex process. Actually, the drivers in the ROM vastly simplify this task, by providing access to the hardware using a predefined set of entry points. As mentioned last month, the video entry point is interrupt 10H.

To display characters on the screen, you must first decide what your program will do, since the ROM actually provides three different methods of displaying text. These three methods are as follows:

1. Display character with attribute (color), with an optional repeat count, on an arbitrary page. The cursor is not advanced.
2. Display a character as above, but use the existing screen attribute at the screen position where the character will appear. The cursor is not advanced.
3. Display character in "terminal mode." In this mode, the existing screen attribute is used. In addition, the characters CR (Carriage Return - 0DH), LF (Line Feed - 0AH), BS (BackSpace - 08H), and BELL (beep - 07H) are processed as commands, rather than text characters. For these characters, the following operations occur:
CR: Move cursor to start of existing line
LF: Move cursor down one line
BS: Move cursor backward one space, unless it is already at the start of the line.
BELL: Beep the speaker for 1/6 second. In this mode, the cursor is advanced automatically. If the cursor wraps around at the end of a line, it will be placed at the start of the next line. If this happens when the cursor is already on the last line, the screen will be scrolled up one line to make room for new information.

Each of the above display methods are applicable for different purposes, but each is used in a similar fashion. Like the previous discussions of video modes, etc, the technique used to take advantage of the ROM's facilities is straightforward. First, load

the CPU registers with the parameters which are required for the call, then load AH with the function to be performed, then finally perform an Interrupt 10H. The parameters used for each of the functions above is as follows:

1. Display character with attribute:
 AH = 09H
 AL = Character to be displayed
 BH = Page number (normally 0)
 CX = Repeat count (normally 1)
 BL = Attribute, as follows:
 --- For video modes 0-3 ---
 Bit 7: Flash this character
 Bit 6: Background Red bit
 Bit 5: Background Green bit
 Bit 4: Background Blue bit
 Bit 3: Intensity bit (1 = bright)
 Bit 2: Foreground Red bit
 Bit 1: Foreground Green bit
 Bit 0: Foreground Blue bit
 --- For video modes 4-5 ---
 Value of 0: Black
 Value of 1: Cyan
 Value of 2: Magenta
 Value of 3: White
 --- For video mode 6 ---
 Value of 1: White
2. Display character with existing attribute:
 AH = 0AH
 AL = Character to be displayed
 BH = Page number (normally 0)
 CX = Repeat count (normally 1)
3. Display character in terminal mode:
 AH = 0EH
 AL = Character to be displayed
 --- If in video modes 4-6 ---
 BL: Attribute to use when displaying character

Now don't be too overwhelmed by this table! Until next month, for instance, you can always use a "1" in place of the repeat count, and a "0" wherever a page number is required. Just decide what you want to do, and take it a step at a time. For instance, to display characters using Function 0EH in text mode (video modes 0-3), you just put the character in AL, place a 0EH in AH, and then perform an interrupt 10H. For example, to say "Hi" using this technique, your program would be:

```
MOV AL,48H           ;Get the "H" from "Hi"
MOV AH,0EH           ;Set AH to display-character mode
INT 10H              ;Display the H!
MOV AL,69H           ;Get the "i" from "Hi"
INT 10H              ;Display the i! (AH was still set)
RET                  ;Finished with the program
```

(Once again, if you enter this program in DEBUG, don't enter the "H" after numbers).

The most complex area above is the use of attributes. Attributes are just any "special" features you want to display a character with, such as color, flash, etc. In text mode, the "color" bits (Red-Green-Blue above) are combined to form the eight possible colors which can be displayed for both foreground and background. As an example, consider the following attribute combinations:

07H - Normal (White-on-black) characters

70H - Reverse (black-on-white) characters
 1FH - Display bright-white characters on a blue background.
 87H - Display flashing normal characters.

For now, just pick one of the values shown above. Next month, we'll go into more detail on how to take better advantage of the screen attributes.

One additional point must be mentioned. Before displaying information, you've first got to tell the ROM where you want it to be displayed. You do this by calling the "Set-Cursor-Position" function, which is defined as follows:

```
Set Cursor Position (AH = 02H)
BH = Page number (normally 0)
DH = Line number (from 0-24)
DL = Horizontal position (from 0-79)
```

Therefore, to display "Hi" at the middle of the screen, set the cursor position to the middle of the screen before displaying anything. You can do this by placing the following code at the start of the program shown above:

```
MOV BH,0             ;Set page number to 0 (normal)
MOV DH,12            ;Point to the middle line
MOV DL,39            ;...and the middle of that line!
MOV AH,02H          ;Set AH to set-cursor-position
INT 10H              ;Set the cursor position
```

Note that we have been using function 0EH to display characters on the screen, since it is the simplest method available. The other video functions are more complex, primarily because they do not advance the cursor to the next screen position. As a result, if you display "Hi", as above, without changing the screen position, the "i" will overwrite the "H", and you'll display "i" instead! The trick to using functions 09H and 0AH then, is to advance the screen position before displaying each character. Experiment with this for now, because I've run out of space for this month!

Next Month

Next month, I'll go into more detail on displaying characters, including details on attributes and the use of multiple pages. In the future, I'll be covering more advanced areas, such as defining your own character sets for graphics modes, etc! Until then, experiment!



Plan your summer vacation
NOW!
Attend the 4th Annual
HUG International Conference
Chicago, O'Hare Hyatt Regency
August 9, 10, 11

Spiral Magic

On The H/Z100

Gary Cramblitt
 HQ 5th Signal Command
 PO box 138
 APO NY 09056

As children, we played with the Spirograph toy made by Kenner Toy Corporation. We spent many happy hours drawing spiral patterns on paper by inserting a colored pencil in a hole on the perimeter of a small disc, then rolling that disc around the inside of a larger ring. The rings and discs were ingeniously made in such a manner that our pencil always returned to the starting point, provided we went around the ring enough times. Then we would pick a new disc and/or ring and repeat the procedure in another color.

What we didn't realize, at the time, is that we were drawing a class of curves known as hypocycloids. The Turbo Pascal program in Listing 1 will produce hypocycloids on the screen of the H/Z100. For those who do not have a Pascal compiler, I've also provided the ZBASIC version in Listing 2. Figures 1 and 2 are samples from this program. This program not only will provide many hours of graphics fun, but it also illustrates techniques in coordinate conversion and graphics display on the H/Z100.

The Technique

The program begins by asking the user for the ratio of the radii of the two circles used to produce the cycloid. By expressing this as the ratio of two integers, the program guarantees that the cycloid will terminate in a reasonable number of rotations. The program next asks the user how many patterns to generate. In the simplest case, a single pattern is generated, i.e., a single hypocycloid is generated.

Hypocycloids are produced by the following equations:

$$x = (a-b)\cos(\theta) + b\cos(h\theta)$$

$$y = (a-b)\sin(\theta) - b\sin(h\theta)$$

where

- x,y = the x and y coordinates of a point on the hypocycloid
- a = radius of the fixed circle (the ring in the Spirograph toy)
- b = radius of the circle rolled around the interior of the ring
- h = (a-b)/b
- theta = angle between the x axis and the line connecting the centers of the two circles

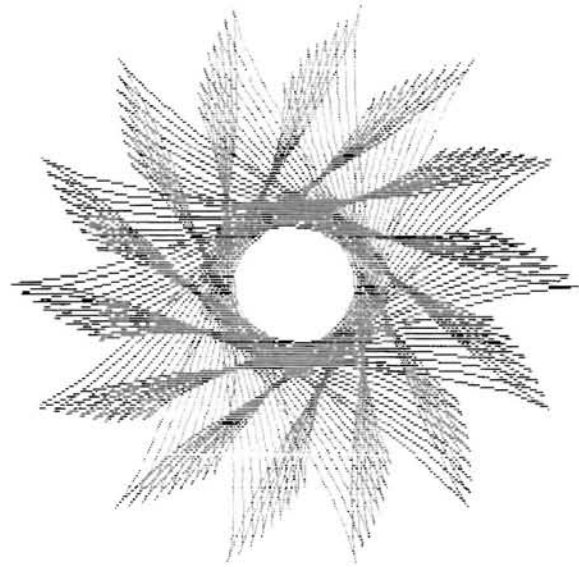


Figure 1. Sample hypocycloid produced with the following parameters. Ratio of radii: 14 to 5. Number of patterns: 9. Degrees rotated from previous pattern: 2. Starting fixed radius: 0.35. Ending radius: 0.25.

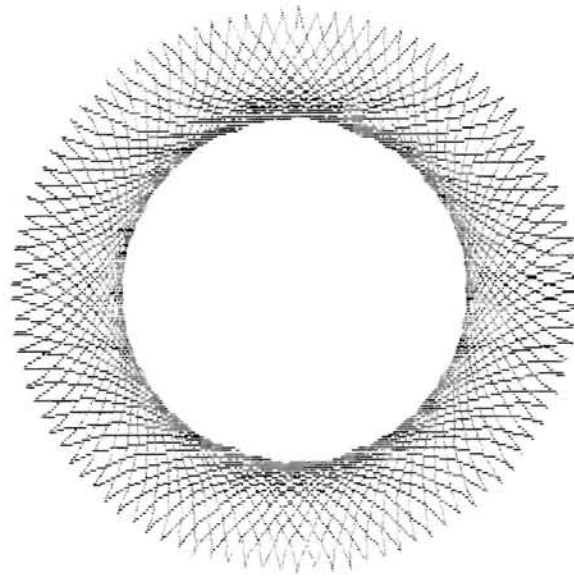


Figure 2. Another sample hypocycloid.

Procedure Hypocycloid generates a single hypocycloid on the screen. Figure 3 shows the generation of a cycloid in progress. In the diagram, we see that "a" is the radius of the fixed or nonmoving circle. "b" is the radius of the circle which is being "rolled" around the inside of the fixed circle. When the cycloid is begun, the point where the two circles touch is chosen as the point to be plotted (the point where our pencil would be if using a Spirograph). As the smaller circle is rolled around the interior of the fixed circle, this point moves in the characteristic hypocycloid curve. The position of the small circle is measured as the angle, theta, between the x axis and the line connecting the centers of the two circles, as shown in Figure 3. The procedure plots the hypocycloid starting with theta equal to zero and the x and y

coordinates are computed from the equations above. Then theta is incremented by a small interval, the new x and y coordinates are calculated, and a line is drawn from the old point to the new point. This process is repeated until the x and y coordinates return to the first point. If the theta intervals are sufficiently small, the resulting plot appears to curve smoothly on the screen of the H/Z100. In this program, intervals of 10 degrees between theta are chosen.

Procedure Hypocycloid generates x and y coordinates according to the coordinate scheme shown in Figure 4. In this scheme, the center of the screen is point (0,0). The x values may range from -0.5 to +0.5. The y values may range from -0.35 to +0.35. The range of y values is smaller because the screen of the H/Z100 is a rectangle and not a square. With this scheme, therefore, my screen is 1.0 units wide and approximately 0.7 units high. Parameter Aspect Ratio reflects this difference for my screen. Your screen may be different depending upon how you have the video adjusted. To determine what your aspect ratio is, simply measure the width and height of your screen. Your aspect ratio is then just the width divided by the height. Once the proper aspect ratio has been determined, the scheme in Figure 4 has the property that distances are uniform in both directions. By this, I mean that a distance of 0.1 units on the x axis has the same length as a distance of 0.1 units on the y axis. Stated in another way, if we plot a circle using this coordinate scheme, it will appear as a perfect circle on the screen, not as an ellipse.

Our next problem is to convert these x and y coordinates into the dot row and column numbers required by the H/Z100. The H/Z100 is capable of plotting 640 dots horizontally, and 225 dots vertically. Because I specifically chose not to plot on line 25 of the screen, this leaves only 216 dots vertically. Figure 5 shows a coordinate scheme adjusted to these dimensions, in which point (0,0) is in the upper left-hand corner of the screen. Functions ProjX and ProjY convert x and y values from the coordinate scheme in Figure 4 to the coordinate scheme in Figure 5.

Figure 1 was produced by plotting 9 hypocycloids. Each hypocycloid is rotated 2 degrees from the previous one. In addition, the radius of the fixed circle is decreased for each successive hypocycloid by a small amount. The radius of the rolling circle in each hypocycloid is adjusted so as to maintain the same ratio of fixed to rolling radii.

The program in Listings 1 and 2 will ask the user how many patterns or hypocycloids he wishes to plot. If more than one hypocycloid is to be plotted, then the program will ask how many degrees each hypocycloid is to be rotated from the previous hypocycloid. In addition, it will ask for the radius of the fixed circle of the first hypocycloid. It will then ask for the radius of the fixed circle of the last hypocycloid. These numbers are used to calculate the fixed and rolling circle radii of each successive hypocycloid, and also the angle each hypocycloid is to be rotated from the x axis. These parameters are then passed to procedure Hypocycloid to plot each one.

Details For The Pascal Program

The program in Listing 1 was written in Turbo Pascal version 2.00B. As much as practical, I have used standard Pascal as defined by Jensen and Wirth. Nevertheless, the following details will assist those attempting to implement the program in some other dialect of Pascal.

The function, KeyPressed, will return a True value whenever input is waiting from the keyboard.

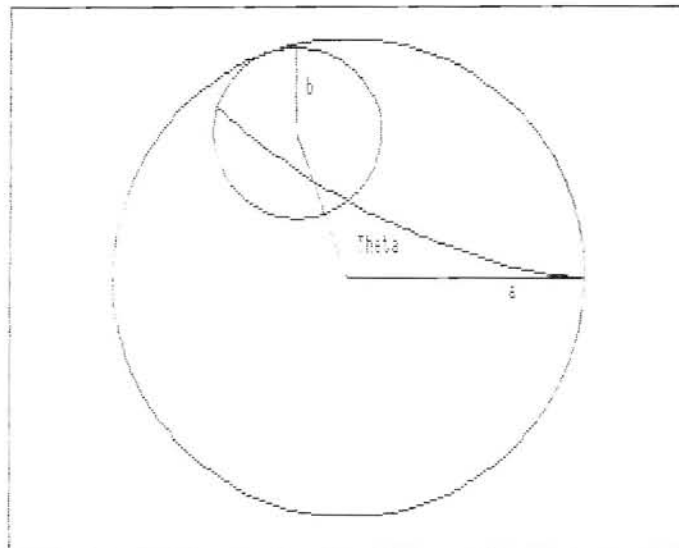


Figure 3. Generation of a hypocycloid. The hypocycloid is generated by following a point on the small circle as the circle is rotated around the inside of the large circle.

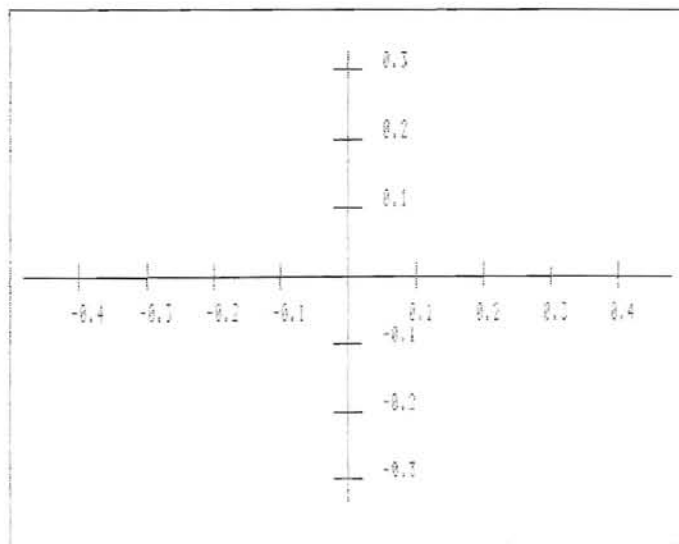


Figure 4. The coordinate scheme used by the Hypocycloid procedure.

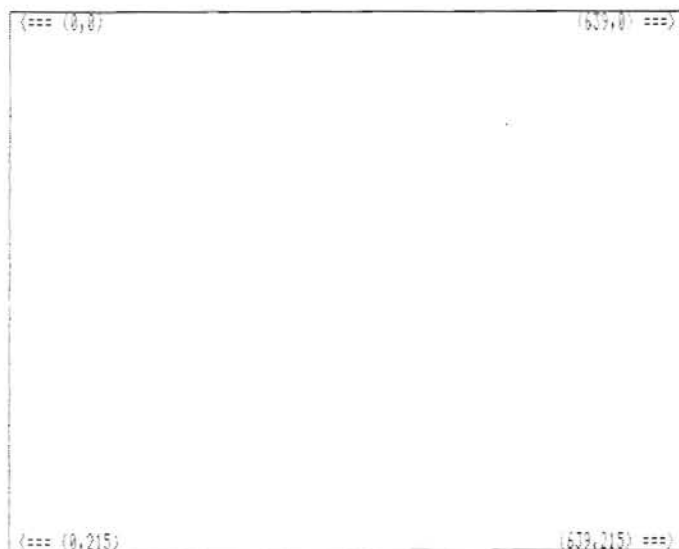


Figure 5. The coordinate scheme used by the H/Z100 ROM and ZBASIC.

Constants preceded by a dollar sign (\$) are hexadecimal constants.

The UpCase function converts its character argument to the upper case equivalent.

Procedure Point turns on a specified point in a specified color. It is highly Turbo Pascal specific. The algorithm is essentially that described by Randy Meyer in the May 1984 issue of REMark with corrections by Chuck Rogalo in the July 84 issue (Buggin' HUG). If your Pascal will not compile this routine, I strongly suggest that you consult those articles for the technique. Briefly, the algorithm in words is:

1. Calculate the row as:
$$\text{row} = ((Y_coordinate / 9) * 16) + (Y_coordinate \bmod 9)$$
2. Calculate the byte address as:
$$\text{byte_address} = (\text{row} * 128) + (X_coordinate / 8)$$
3. Calculate the bit address as:
$$\text{bit_address} = X_coordinate \bmod 8$$
4. Read and save the Video Control Register (VCR) at hex 0D8.
5. Turn off bit 7 and turn on bits 4 thru 6 of the VCR. Leave the other bits untouched. Bits are numbered 0 to 7 from least significant to most significant.
6. Read the byte at memory location base address C000 hex and offset byte_address. If blue is to be turned on, set the bit at bit_address. If blue is not to be turned on, then turn the bit off. Store the byte back at its original location.
7. Repeat step 6 for red at base address D000 hex and for green at E000 hex.
8. Restore the VCR to its original settings.

Within the Point routine, the "Shl" operator shifts a 16 bit integer n bits to the left with no wraparound. The "Shr" function shifts bits to the right. The "Port" array is defined by Turbo Pascal to enable direct access to IO ports. It either reads an 8-bit quantity from a specified port number if used on the right side of an equals sign, or it sends an 8-bit quantity to the port if used on the left side of the equals. In a similar manner, the "Mem" array permits direct access to memory when given a segment and offset address. Turbo Pascal permits the "And" and "Or" operators to be used with integers, in which case, it performs a bitwise logical And or logical Or of the operands.

Procedure Line draws a line from point (X1,Y1) to point (X2,Y2) in the specified color. The algorithm is based on the Variable-Duty-Cycle algorithm presented by Timothy Stryker in the October 1981 issue of Byte Magazine. The basic technique is to assign a number called the Duty Master to two counters, one for the X direction, and one for the Y direction. Each of these counters is then decremented an amount determined by the slope of the line. When either counter falls below 0, then the corresponding coordinate is incremented by one and a point is plotted on the screen. Then the Duty Master is added to the counter. This continues until we get to point (X2,Y2). Since the routine is standard Pascal, it should not present any difficulties with other Pascal dialects. Just enter it as given.

Details For The ZBASIC Version

The ZBASIC version is fairly straightforward and corresponds closely to the Pascal version.

Functions PROJX and PROJY convert x,y coordinates from the scheme in Figure 4 to the dot row and column numbers required

by ZBASIC.

The main routine at 20010 initializes several variables, including ASPECTRATIO. Depending upon how your screen's video is adjusted, you may need to alter the value of this variable. The main routine then calls the subroutine at 11000 to plot each picture.

The routine at 11000 corresponds to Procedure MultiPattern in the Pascal version. It obtains the various parameters from the user, then calls the subroutine at 510 once for each hypocycloid to be plotted in the picture.

The routine at 510 corresponds to Procedure Hypocycloid in the Pascal version. It plots a complete hypocycloid given the radius of the fixed circle, A, the radius of the rolling circle, B, the interval in radians between points to be plotted, DELTA, and the angle in radians to rotate the hypocycloid from the x axis, STARTANG. It uses functions PROJX and PROJY to convert coordinates from the scheme in Figure 4 to the dot row and column required by the ZBASIC LINE command.

Enhancements

Although the code is there to support colors, the programs in Listings 1 and 2 do not use color. An obvious enhancement, therefore, is to provide the user with control over the colors of each hypocycloid.

Hypocycloids are generated by rolling a circle around the inside of a fixed circle. If we rotate the circle around the outside of the fixed circle, then a class of cycloids called epicycloids is produced. The equations for these curves are:

$$\begin{aligned}x &= (a+b)*\cos(\text{theta}) - b*\cos(g*\text{theta}) \\y &= (a+b)*\sin(\text{theta}) - b*\sin(g*\text{theta})\end{aligned}$$

where

$$g = (a+b)/b$$

and all other variables are as before. It would be fun to modify the program to enable plotting of epicycloids, instead of hypocycloids or possibly both in the same picture.

For additional curves and formulas that might be adaptable to this program, see Reference 3. The possibilities for generating pleasing displays on the computer are limited only by the imagination of the programmer.

References

1. Jensen, Kathleen and Wirth, Niklaus. Pascal User Manual and Report, Second Edition. Springer-Verlag, 1978.
2. Microsoft ZBASIC (Z-DOS). Zenith Data Systems, 1982.
3. Selby, Samuel M., Editor-in-chief. CRC Standard Mathematical Tables. The Chemical Rubber Co., 1971.
3. Turbo Pascal Reference Manual. Borland International, Inc., 1984.

About the Author

Gary Cramblitt has worked with computers since 1977. In 1981 he built an H89 kit, seeking to understand the electronics behind his work. Specializing in user-friendly software interfaces and programmer tools, he is currently employed by IIT Research Institute in Worms, Germany.


```

100 DEF FNPROJX%(X) = CINT((X + .5) * XDOTS)
200 DEF FNPROJY%(Y) = CINT(YDOTS - (Y*ASPECTRATIO+ 5)*YDOTS)
300 GOTO 20010
500 REM PLOT HYPOCYCLOID ON THE SCREEN.  REQUIRES
    FOLLOWING PARAMETERS:
501 REM A =          RADIUS OF FIXED CIRCLE
502 REM B =          RADIUS OF ROLLING CIRCLE
503 REM DELTA =      INTERVAL TO USE BETWEEN POINTS OF THE CURVES
504 REM STARTANG =   THE ANGLE OF THE FIRST POINT OF THE CYCLOID
510 H = (A-B)/B
511 THETA = 0
520 ROTATECOS = COS(-STARTANG)
530 ROTATESIN = SIN(-STARTANG)
540 OLDX = FNPROJX%(A*COS(STARTANG))
550 OLDY = FNPROJY%(A*SIN(STARTANG))
560 STOPX = OLDX
570 STOPY = OLDY
571 X = 0
572 Y = 0
580 WHILE ((X <> STOPX) OR (Y <> STOPY))
590   THETA = THETA + DELTA
600   XPT = (A-B)*COS(THETA) + B*COS(H*THETA)
610   YPT = (A-B)*SIN(THETA) - B*SIN(H*THETA)
620   X = FNPROJX%(XPT*ROTATECOS + YPT*ROTATESIN)
630   Y = FNPROJY%(-XPT*ROTATESIN + YPT*ROTATECOS)
640   LINE (OLDX,OLDY)-(X,Y)
650   OLDX = X
660   OLDY = Y
680 WEND
690 RETURN
10000 REM OBTAIN PARAMETERS FROM USER AND DRAW THE HYPOCYCLOIDS
11000 PRINT
    " Ratio of Fixed To Rotating Circles in the pattern."
11010 INPUT " Express as a ratio of integers
    (EXAMPLE: 14,5)";FINT%,MINT%
11020 RATIO = FINT%/MINT%

```

```

11030 INPUT " Number of Patterns (EXAMPLE: 1)";PATTERNS%
11035 ROTATION = 0
11040 IF PATTERNS% > 1 THEN INPUT
    " Degrees each pattern should be rotated from the
    previous pattern";ROTATION
11060 INPUT " Starting Radius of the fixed circle
    (EXAMPLE: 0.35)";FIXEDR
11090 STARTANG = 0
11100 CHANGER = 1
11105 LASTR = FIXEDR
11110 IF PATTERNS% > 1 THEN INPUT
    " Ending radius of the fixed circle.";LASTR
11120 CLS
11130 IF PATTERNS% > 1
    THEN CHANGER = (FIXEDR-LASTR)/(PATTERNS%-1)
12000 WHILE FIXEDR >= LASTR
12010   A = FIXEDR
12020   B = A/RATIO
12030   GOSUB 510
12040   FIXEDR = FIXEDR - CHANGER
12050   STARTANG = STARTANG + ROTATION/180*PI
12090 WEND
12100 RETURN
20000 REM MAIN ROUTINE
20010 XDOTS = 640
20020 YDOTS = 216
20030 ASPECTRATIO = 1.4
20040 PI = 3.14159
20050 DELTA = 10/180*PI
20060 ANS$ = "Y"
20070 WHILE (ANS$ = "Y" OR ANS$ = "y")
20080   GOSUB 11000
20090   LOCATE 1,1
20100   INPUT "Do Another";ANS$
20110 WEND
50000 STOP

```

```

{ Spiro.pas
  Spirograph program for the Z100.  This program will generate
  hypocycloids on the screen of the Z100. }
{C-} { This Turbo Pascal compiler option required in order for the
  KeyPressed function to properly work.  (KeyPressed returns True
  value when input is waiting from the keyboard.) }

Const
  BlueSeg = $0C000;
  RedSeg  = $0D000;
  GreenSeg = $0E000;
  Black   = 0;
  Blue    = 1;
  Red     = 2;
  Magenta = 3;
  Green   = 4;
  Cyan    = 5;
  Yellow  = 6;
  White   = 7;
  Xdots  = 640; { Number of horizontal dots on the Z100 screen }
  Ydots  = 216; { 24 lines worth.  Don't plot on line 25. }
  AspectRatio = 1.4; { My screen is approximately 7 inches by 5 inches,
  yours may be different }

Var
  Ans: Char;
  Voff: Array[0..2] Of Integer; { Offsets to video memory }

Procedure Point (X,Y,Color: Integer);
{ This procedure turns on the point at X,Y with the specified color.
  X ranges from 0 to 639, Y from 0 to 215.  Point (0,0) is in upper left
  corner of screen. }
Const
  VCR = $0D8;
  VRAMenable = $07F;
  SingleColor = $070;

Var
  J: Integer;
  CurrentReg: Byte;
  Row: Integer;
  CharByte: Integer;
  BitAddress: Integer;
  ByteAddress: Integer;
  VByte: Byte;
  BitByte: Byte;

Begin
  If ((X >= 0) And (X < XDots) And (Y >= 0) And (Y < YDots)) Then Begin
    Row := ((Y Div 9) Shl 4) + (Y Mod 9);
    CharByte := X Div 8;
    BitAddress := X Mod 8;
    CurrentReg := (Row*128) + CharByte;
    Port[VCR] := CurrentReg And VRAMenable Or SingleColor;
    BitByte := $80 Shl BitAddress;
    For J := 0 To 2 Do Begin

```

```

Vbyte := Mem[VOff[J]:ByteAddress];
If (Color And (1 Shl J)) <> 0 Then Vbyte := Vbyte Or BitByte
Else Vbyte := Vbyte And Not BitByte;
Mem[VOff[J]:ByteAddress] := Vbyte;
End;
Port[VCR] := CurrentReg;
End;
End;

Procedure Line (X1, Y1, X2, Y2, Color: Integer);
{ This routine draws a line from (X1,Y1) to (X2,Y2) in the specified color.
The algorithm is based on the Variable-Duty-Cycle algorithm presented by
Timothy Stryker in the October 1981 Byte. }
Var
X, Y, DeltaX, DeltaY, DutyMaster, XCnt, YCnt: Integer;
StepX, StepY: Integer;
DoPlot: Boolean;
Begin
StepX := 1; If X2 < X1 Then StepX := -1;
StepY := 1; If Y2 < Y1 Then StepY := -1;
2 - X1; If DeltaX < 0 Then DeltaX := -DeltaX;
DeltaY := Y2 - Y1; If DeltaY < 0 Then DeltaY := -DeltaY;
DutyMaster := DeltaX; If DeltaY > DeltaX Then DutyMaster := DeltaY;
XCnt := DutyMaster; YCnt := DutyMaster;
X := X1; Y := Y1;
Point (X,Y,Color);
While ((X <> X2) Or (Y <> Y2)) Do Begin
DoPlot := False;
XCnt := XCnt - DeltaX;
If XCnt < 0 Then Begin
DoPlot := True;
X := X + StepX;
XCnt := XCnt + DutyMaster;
End;
YCnt := YCnt - DeltaY;
If YCnt < 0 Then Begin
DoPlot := True;
Y := Y + StepY;
YCnt := YCnt + DutyMaster;
End;
If DoPlot Then Point (X,Y,Color);
End;
End;

Function ProjX (X: Real): Integer;
{ Projects a coordinate point onto the Z100 dot plane. The Z100 screen is
viewed as a window 1.0 units wide by .7 units high. The center is point
(0,0). Given an X coordinate for such a window, this routine returns the
corresponding Z100 dot column. }
Begin ProjX := Round((X+0.5)*XDots); End;

Function ProjY (Y: Real): Integer;
{ Projects a coordinate point onto the Z100 dot plane. The Z100 screen is
viewed as a window 1.0 units wide by .7 units high. The center is point
(0,0). Given a Y coordinate for such a window, this routine returns the
corresponding Z100 dot row. }
Begin ProjY := YDots - 1 - Round((Y*AspectRatio+0.5)*YDots); End;

Procedure Hypocycloid (a, b, Delta, StartAng: Real);

```

```

{ This routine generates a hypocycloid on the screen of the Z100. The
generated hypocycloid is based on the following formulas:

```

$$x = (a-b)\cos(\theta) + b\cos(h*\theta)$$

$$y = (a-b)\sin(\theta) - b\sin(h*\theta)$$

where

$$h = (a-b)/b$$

a = radius of fixed circle

b = radius of rolling circle

theta = angle between the x axis and the line connecting the two circles' centers

Points along the hypocycloid are calculated at intervals of Delta radians. The first point is at StartAng from the x axis. The cycloid is generated until the cusps overlap, i.e. we get back to the starting point. }

```

Var
h, Theta, XPt, YPt: Real;
RotateSin, RotateCos: Real;
OldX, OldY, X, Y: Integer;
StopX, StopY: Integer;
Begin
h := (a-b)/b;
Theta := 0.0;
RotateCos := Cos(-StartAng);
RotateSin := Sin(-StartAng);
OldX := ProjX(a*cos(Theta) + b*cos(h*Theta));
OldY := ProjY(a*sin(Theta) - b*sin(h*Theta));
StopX := OldX; StopY := OldY;
Repeat
{ Calculate the next point Delta radians from last point }
Theta := Theta + Delta;
XPt := (a-b)*cos(Theta) + b*cos(h*Theta);
YPt := (a-b)*sin(Theta) - b*sin(h*Theta);
{ Rotate the cycloid StartAng radians and project it onto the screen
of the Z100. }
X := ProjX( XPt*RotateCos + YPt*RotateSin);
Y := ProjY(-XPt*RotateSin + YPt*RotateCos);
{ Draw a line from the last point to the new point. }
Line (OldX,OldY,X,Y,White);
OldX := X; OldY := Y;
Until ((OldX = StopX) And (OldY = StopY));
End;

Procedure MultiPattern;
Var
Patterns: Integer; { Number of Patterns to draw }
Fint: Integer;
Mint: Integer;
Rotation: Real;
Ratio: Real;
FixedR: Real;
{ Degr to rotate each cycloid from the previous }
{ Ratio of Fixed to Moving Radii }
{ Fixed circle radius }

```



COLOR

A Screen Color Setup Program For The H/Z-100

*Glenn F. Roberts, Ph.D.
12048 Greywing Square, #C-3
Reston, VA 22091*

This article presents a simple program that allows you to set the foreground and background text colors on the Heath/Zenith 100 computer. With this program you can select any of the sixty-four possible combinations of text and screen background colors. The program shows you samples of all of these combinations on your screen and allows you to select the one most pleasing to you. While this program is written primarily for the H/Z-100 computer, I give some suggestions at the end of the article on how to modify it for use on the H/Z-150 series of IBM compatible computers.

This program was modeled after the COLOR program supplied by Columbia Data Products with its IBM-PC compatible computer systems. The program has two modes of operation: a setup mode and a recall mode. Setup mode is used to select a new color combination and recall mode is used to recall the previously selected combination. You specify the setup mode by append-

look on the default disk for a file called 'COLOR.COM'. This could cause you problems if you rename the program or attempt to call it from another disk. If you want it to always look on a certain disk, you must change the first byte in FCBREC. Change it to 1 for A:, 2 for B:, etc. If you want to call the program something other than COLOR, you must change the file name specified in FCBREC to be the same as whatever name you choose.

A second comment, related to the first, has to do with using COLOR under MS-DOS 2.0. If you are making use of subdirectories, then you have to be a little careful when using the /C switch to save a new color combination. This switch will only work if a copy of COLOR.COM exists in your present subdirectory.

If you own an H/Z-150 series computer, you should be able to modify this program for use on your machine. The easiest way to do this is to make use of the ANSI console driver. The ANSI driver is a device driver program that causes the console to respond to the ANSI escape sequences. These escape sequences are different from those recognized by the H/Z-100, but conform to standards specified by the American National Standards Institute. These sequences are described in the MS-DOS 2.0 Programmer's Utility Pack, as well as in the IBM DOS 2.0 Technical Reference. You'll have to change the escape sequences embedded in the various text messages, as well as the routines GOTO_XY and READ_KEY. You'll also have to install the ANSI driver by copying the file ANSI.SYS to your boot disk and adding the following line to your CONFIG.SYS file:

```
DEVICE=ANSI.SYS
```

If you're like me, you enjoy taking programs that appear in REMark and other magazines and customizing them to your own liking. In case you haven't already thought of at least half a dozen

ways to improve COLOR, I've prepared a list of some fun additions and modifications to the program which you might like to try.

1. Warn the user if he or she attempts to select a combination in which the foreground color is the same as the background color, or better yet disallow these combinations altogether.
2. Verify that the user has the red and blue color RAM chips installed by sending the sequence ESC i 0, and interpreting the sequence the console sends back.
3. Modify the program to allow the foreground and background colors to be specified directly on the command line.
4. Expand the program to a generalized console configuration program that allows you to control other console features, such as key click, auto repeat, etc.
5. The program currently sets the colors to white on black if the user hits control-c when in the setup mode. Add code to explicitly handle control-c interrupts and restore the colors to what they were before the program was executed.
6. Add a /Nosave switch to allow the user to change the current color settings without affecting the values permanently saved in COLOR.COM.
7. On MS-DOS 2.0 systems make use of the new DOS functions, such as function 37H to get the ASCII switch character and 3DH to open a file. Be sure to add a call to 30H to check the version of MS-DOS.

```

BLUE EQU '1' ; sequences
RED EQU '2'
MAGENTA EQU '3'
GREEN EQU '4'
CYAN EQU '5'
YELLOW EQU '6'
WHITE EQU '7'

UP_ARROW EQU 'A' ; <ESC> A = up arrow key
DOWN_ARROW EQU 'B' ; <ESC> B = down arrow key
RIGHT_ARROW EQU 'C' ; <ESC> C = right arrow key
LEFT_ARROW EQU 'D' ; <ESC> D = left arrow key

DOSI_TERM EQU 020H ; Normal terminate to DOS
DOSI_FUNC EQU 021H ; Call a dos function
DOSF_CONOUT EQU 2 ; Output a single character
DOSF_DRCINE EQU 8 ; Input a char. (no echo)
DOSF_OUTSTR EQU 9 ; Output a string
DOSF_OPFILE EQU 15 ; Open a file
DOSF_CLFILE EQU 16 ; Close a file
DOSF_SEQWRITE EQU 21 ; Sequential write
DOSF_SDIOA EQU 26 ; Set disk I/O address

NCOLORS EQU 8 ; Number of colors possible
CURSOR EQU '><' ; Cursor pattern
PHD_DIOA EQU 080H ; Command line info.
; --- Macros
;
DISPLAY MACRO STRING
MOV DX,OFFSET STRING
MOV AH,DOSF_OUTSTR
INT DOSI_FUNC
ENDM

;
READ_KBD MACRO
MOV AH,DOSF_DRCINE
INT DOSI_FUNC
ENDM

;
DISPLAY_CHAR MACRO CHARACTER
MOV DL,CHARACTER
MOV AH,DOSF_CONOUT
INT DOSI_FUNC
ENDM

; --- Code starts here
ORG 100H

BEGIN: JMP MAIN ; Jump to main program
;
; --- Message to set up new color and clear screen
;
MSG_CLRSC DB ESC,'m'
FORE_COL DB WHITE ; NOTE: program changes
BACK_COL DB BLACK ; these values in the
RECLEAN EQU $-100H ; COLOR.COM file.
DB ESC,'E','$'

```



```

JNZ BADWRITE ; yes, jump
;
MOV AH,DOSF_CLFILE ; no, then close
LEA DX,FCBREC ; the file
INT DOSI_FUNC
;
CLC ; and return with
RET ; no error
BADWRITE:
STC ; flag as error
RET ; and return
SAVE_COLORS ENDP

CODE ENDS

END BEGIN
mov al, [bx]
cmp ah, 'a' ;Less than a?
jl cmky21 ;If so, don't capitalize.
cmp ah, 'z'+1 ;More than z?
jns cmky21
and ah, 1370 ;Capitalize the letter.
cmky21: cmp ah, al
je cmky3
jg cmky2y
jmp cmky41 ;Fail if ah precedes al alphabetically.
cmky2y: jmp cmky6 ;Not this keyword - try the next.
cmky3: inc bx ;We match here, how 'bout next char?
mov al, [bx]
cmp al, '$' ;End of keyword?
jne cmky3x
jmp cmky7 ;Succeed.
cmky3x: mov dl, al ;Save al'

```



**ILLUSTRATOR
GRAPHICS DESIGN PACKAGE**
by Wizard Software House
79 MARSHALL STREET
PROVIDENCE, RI 02909
(401) 331-5034 (617) 881-2543

Box & Box Fill **Circle & Circle Fill**
Diamond & Diamond Fill **Ellipse & Ellipse Fill**
Triangles & Triangle Fill **Turtle Mode**
16 x 16 Font Generator **26 User Defined Macros**



**Get a World of
Graphics Power**

Unlimited Colors
Rubber Band Mode
User Defined Line Styles
Irregular Area Fills
Easy Single Key Commands
Full Interlace Operation
640 x (225, 400, 480)

Relocatable Graphic Images
Support for
Most Dot Printers
Easy Interface to User Programs

Be a HERO
with your
Computer
Buy One
NOW!

\$89.95 for ZDOS or HDOS + Imaginator

Name _____
Address _____
City/State _____
Zip _____

Zdos Hdas



HEATH/ZENITH 88, 89, 90 PERIPHERALS

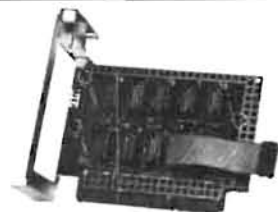
16K RAM EXPANSION CARD

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



2 PORT SERIAL CARD I/O 3 PORT PARALLEL

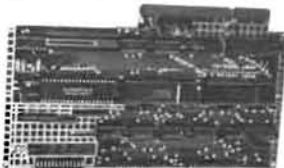
"...not your typical vanilla-flavored serial and parallel interface..."

Your H/Z 88, 89, 90 can now directly connect and operate EPSON, IDS, ANADIX, 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

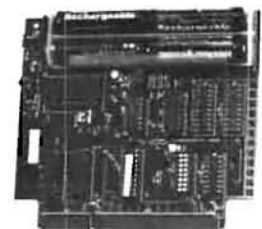


REAL TIME CLOCK

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

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

ZENITH
data systems
SERVICE CENTER

HUG Price List

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

Part Number	Description of Product	Selling Price	Vol. Issue	Part Number	Description of Product	Selling Price	Vol. Issue	Part Number	Description of Product	Selling Price	Vol. Issue
HDOS HARDCOPY SOFTWARE								PROGRAMMING LANGUAGES			
885-1008	Volume I Documentation	9.00		885-1082	Programs for Printers H8/H89	20.00		HDOS			
885-1013	Volume II Documentation	12.00		885-1083-[37]	Disk XVII Misc H8/H89	20.00	11	885-1038-[37]	Wise on Disk H8/H89	18.00	
885-1015	Volume III Documentation	9.00		885-1089-[37]	Disk XVIII Misc H8/H89	20.00	20	885-1042-[37]	PILOT on Disk H8/H89	19.00	
885-1037	Volume IV Documentation	12.00	8	885-1090-[37]	Disk XIX Utilities H8/H89	20.00	22	885-1059	FOCAL-8 H8/H89 Disk	25.00	13
885-1058	Volume V Documentation	12.00		885-1092-[37]	Relocating Debug Tool H8/H89	30.00	14	885-1078-[37]	HDOS Z80 Assembler	25.00	21
MISCELLANEOUS HDOS COLLECTIONS								885-1085	PILOT Documentation	9.00	
885-1032	Disk V H8/H89	18.00	8	885-1098	H8 Color Graphics ASM	20.00	19	885-1086-[37]	Tiny HDOS PASCAL H8/H89	20.00	13
885-1044-[37]	Disk VI H8/H89	18.00		885-1099	H8 Color Graphics Tiny PASCAL	20.00	19	885-1094	HDOS Fig-Forth H8/H89	40.00	18
885-1064-[37]	Disk IX H8/H89 Disk	18.00		885-1105	HDOS Device Drivers H8/H89	20.00	24	885-1132-[37]	HDOS Tiny BASIC Compiler	20.00	59
885-1066-[37]	Disk X H8/H89	18.00	10	885-1116	HDOS Z80 Debugging Tool	20.00	27	885-1134	HDOS SMALL-C Compiler	30.00	63
885-1069	Disk XIII Misc H8/H89	18.00		885-1119-[37]	BHBASIC Support	20.00	29	CP/M			
GAMES								885-1208-[37]	CP/M Fig-Forth H8/H89 2 Disks	40.00	18
HDOS				885-1120-[37]	HDOS 'WHEW' Utilities	20.00	33	885-1215-[37]	CP/M BASIC-E	20.00	26
885-1010	Adventure Disk H8/H89	10.00	4	885-1121	HDOS Hard Sec Sup Pkg 2 Disks	30.00	37	BUSINESS, FINANCE AND EDUCATION			
885-1029-[37]	Disk II Games 1 H8/H89	18.00	8	885-1123	XMET Robot & Cross Assembler	20.00	40	HDOS			
885-1030-[37]	Disk III Games 2 H8/H89	18.00	8	885-1126	HDOS Utilities by PS:	20.00	42	885-1047	Stocks H8/H89 Disk	18.00	
885-1031	Disk IV MUSIC H8 Only	20.00	25	885-1127-[37]	HDOS Soft Sector Support Pkg	30.00	45	885-1048	Personal Account H8/H89 Disk	18.00	
885-1067-[37]	Disk XI H8/H19/H89 Games	18.00	12	885-1128-[37]	HDOS DISKVIEW	16.00	46	885-1049	Income Tax Records H8/H89 Disk	18.00	
885-1068	Disk XII MBASIC Graphic Games	18.00	10	885-1129-[37]	HDOS CVT Color Video Terminal	20.00	46	885-1055-[37]	MBASIC Inventory Disk H8/H89	30.00	
885-1088-[37]	Disk XVII MBASIC Graph Games	20.00	14	885-8001	SE (Screen Editor)	25.00	28	885-1056	MBASIC Mail List	30.00	
885-1093-[37]	D&D H8/H89 Disk	20.00	16	885-8003	BHTOMB	25.00	28	885-1070	Disk XIV Home Fin H8/H89	18.00	
885-1096-[37]	MBASIC Action Games H8/H89	20.00	18	885-8004	UDUMP	35.00	28	885-1071-[37]	MBASIC SmbusPk H8/H19/H89	75.00	17
885-1103	Sea Battle HDOS H19/H8/H89	20.00	20	885-8006	HDOS SUBMIT	20.00	31	885-1091-[37]	Grade/Score Keeping H8/H89	30.00	14
885-1111-[37]	HDOS MBASIC Games H8/H89	20.00	23	885-8007	EZITRANS	30.00	30	885-1097-[37]	MBASIC Quiz Disk H8/H89	20.00	18
885-1112-[37]	HDOS Graphic Games H8/H89	20.00	23	885-8015	HDOS TEXTSET Formatter	30.00	42	885-1118-[37]	MBASIC Payroll	60.00	30
885-1113-[37]	HDOS Action Games H8/H89	20.00	23	885-8017	HDOS Programmers Helper	16.00	42	885-1131-[37]	HDOS CHEAPCALC	20.00	47
885-1114	H8 Color Raiders & Goop	20.00	23	885-8024	HDOS BHBASIC Utilities Disk	16.00	46	885-8010	HDOS CHECKOFF	25.00	32
885-1124	HUGMAN & Movie Animation Pkg	20.00	41	CP/M				885-8021	HDOS Student's Statistics Pkg	20.00	44
885-1125	MAZEMADNESS	20.00	41	885-1210-[37]	CP/M ED (same as 885-1022)	20.00	20	885-8027	HDOS SCICALC	20.00	50
885-1130	Star Battle	20.00	47	885-1212-[37]	CP/M Utilities H8/H89	20.00	21	CP/M			
885-1133-[37]	HDOS Games Collection I	20.00	59	885-1213-[37]	CP/M Disk Utilities H8/H89	20.00	22	885-1218-[37]	CP/M MBASIC Payroll	60.00	31
885-8009-[37]	HDOS & CP/M Galactic Warrior	20.00	32	885-1217-[37]	HUG Disk Duplication Utilities	20.00	26	885-1233-[37]	CP/M CHEAPCALC	20.00	47
885-8022	HDOS SHAPES	16.00	45	885-1223-[37]	HRUN HDOS Emulator 3 Disks	40.00	37	885-1239-[37]	Spread Sht. Contest Disk I	20.00	
885-8026	HDOS Space Drop	16.00	49	885-1225-[37]	CP/M Disk Dump & Edit Utility	30.00	40	885-1240-[37]	Spread Sht. Contest Disk II	20.00	
885-8032-[37]	HDOS Castle	20.00	59	885-1226-[37]	CP/M Utilities by PS	20.00	40	885-1241-[37]	Spread Sht. Contest Disk III	20.00	
CP/M				885-1229-[37]	XMET Robot Cross Assembler	20.00	40	885-1242-[37]	Spread Sht. Contest Disk IV	20.00	
885-1206-[37]	CP/M Games Disk	20.00	11	885-1230-[37]	CP/M Function Key Mapper	20.00	42	885-1243-[37]	Spread Sht. Contest Disk V	20.00	
885-1209-[37]	CP/M MBASIC D&D	20.00	19	885-1231-[37]	Cross Ref Utilities for MBASIC	20.00	43	885-1244-[37]	Spread Sht. Contest Disk VI	20.00	
885-1211-[37]	CP/M Sea Battle	20.00	20	885-1232-[37]	CP/M Color Video Terminal	20.00	46	885-8011-[37]	CP/M CHECKOFF	25.00	32
885-1220-[37]	CP/M Action Games	20.00	32	885-1235-[37]	CP/M COPYDOS	20.00	54	ZDOS			
885-1222-[37]	CP/M Adventure	10.00	35	885-1237-[37]	CP/M Utilities	20.00	55	885-3006-37	ZDOS CHEAPCALC	20.00	47
885-1227-[37]	CP/M Casino Games	20.00	38	885-1245-37	CP/M 85 KEYMAP	20.00	63	885-3013-37	ZDOS Checkbook Manager	20.00	54
885-1228-[37]	CP/M Fast Action Games	20.00	39	885-5001-37	CP/M 86 KEYMAP	20.00	51	885-3018-37	ZDOS Contest Spreadsheet Disk	25.00	58
885-1236-[37]	CP/M Fun Disk I	20.00	55	885-5002-37	CP/M 86 HUG Editor	20.00	52	885-8028-37	ZDOS SCICALC	20.00	50
ZDOS				885-5003-37	CP/M 86 Utilities by PS:	20.00	54	885-8030-37	ZDOS MATHFLASH	20.00	55
885-3004-37	ZDOS ZBASIC Graphic Games	20.00	37	885-8018-[37]	CP/M FAST EDDY & BIG EDDY	20.00	43	DATA BASE MANAGEMENT SYSTEMS			
885-3009-37	ZDOS ZBASIC D&D	20.00	50	885-8019-[37]	DOCUMAT and DOCULIST	20.00	43	HDOS			
885-3011-37	ZDOS ZBASIC Games Disk	20.00	52	885-8025-37	CP/M 85/86 FAST EDDY	20.00	49	885-1107-[37]	HDOS Data Base System H8/H89	30.00	23
885-3017-37	ZDOS Contest Games Disk	25.00	58	ZDOS				885-1108-[37]	HDOS MBASIC Data Base Sys.	30.00	23
UTILITIES				885-3005-37	ZDOS ETCHDUMP	20.00	39	885-1109-[37]	HDOS Retriever ASM (3 Disks)	40.00	23
HDOS				885-3007-37	ZDOS CP/Emulator	20.00	47	885-1110	HDOS Autofile (2 Disks)	30.00	23
885-1022-[37]	HUG Editor (ED) Disk H8/H89	20.00	20	885-3008-37	ZDOS Utilities	20.00	47	885-1115-[37]	HDOS Navigational Program	20.00	25
885-1025	Runoff Disk H8/H89	35.00		885-3010-37	ZDOS KEYMAP	20.00	51	885-8008	Farm Accounting System	45.00	30
885-1060-[37]	Disk VII H8/H89	18.00		885-3022-37	ZDOS/MSDOS Useful Programs I	30.00	63	CP/M			
885-1061	TMI Load H8 ONLY Disk	18.00		885-3023-37	ZDOS/MSDOS EZPLOT	20.00	63	885-1219-[37]	CP/M Navigational Program	20.00	31
885-1062-[37]	Disk VIII H8/H89 (2 Disks)	25.00		885-8029-37	ZDOS FAST EDDY	20.00	53	PC/IBM COMPATIBLE			
885-1063	Floating Point Disk H8/H89	18.00		H/Z100 ZDOS - H/Z150 MSDOS				885-6001-37	MSDOS Keymapper	20.00	59
885-1065	Fix Point Package H8/H89 Disk	18.00	10	885-3012-37§§	ZDOS HUG Editor	20.00	52	885-6002-37	CP/Emulator II & ZEmulator	20.00	59
885-1075	HDOS Support Package H8/H89	60.00		885-3014-37§§	ZDOS/MSDOS Utilities II	20.00	54	885-8033-37	MSDOS Fast Edit	20.00	62
885-1077	TXTCON/BASCON H8/H89	18.00		885-3016-37§	ZDOS/MSDOS Adventure	10.00	57				
885-1079-[37]	HDOS Page Editor	25.00	15	885-3020-37§§	MSDOS HUG Menu System	20.00	62	§ All program files run on both			
885-1080	EDITX H8/H19/H89 Disk	20.00		885-3021-37§§	ZDOS/MSDOS Cardcat	20.00	63	§§ Program files run partially on both			

HUG Price List

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

Part Number	Description of Product	Selling Price	Vol. Issue	Part Number	Description of Product	Selling Price	Vol. Issue	Part Number	Description of Product	Selling Price	Vol. Issue
HDOS HARDCOPY SOFTWARE								PROGRAMMING LANGUAGES			
885-1008	Volume I Documentation	9.00		885-1082	Programs for Printers H8/H89	20.00		HDOS			
885-1013	Volume II Documentation	12.00		885-1083-[37]	Disk XVI Misc H8/H89	20.00	11	885-1038-[37]	Wise on Disk H8/H89	18.00	
885-1015	Volume III Documentation	9.00		885-1089-[37]	Disk XVII Misc H8/H89	20.00	20	885-1042-[37]	PILOT on Disk H8/H89	19.00	
885-1037	Volume IV Documentation	12.00	8	885-1090-[37]	Disk XIX Utilities H8/H89	20.00	22	885-1059	FOCAL-8 H8/H89 Disk	25.00	13
885-1058	Volume V Documentation	12.00		885-1092-[37]	Relocating Debug Tool H8/H89	30.00	14	885-1078-[37]	HDOS Z80 Assembler	25.00	21
MISCELLANEOUS HDOS COLLECTIONS								885-1085	PILOT Documentation	9.00	
885-1032	Disk V H8/H89	18.00	8	885-1098	H8 Color Graphics ASM	20.00	19	885-1086-[37]	Tiny HDOS PASCAL H8/H89	20.00	13
885-1044-[37]	Disk VI H8/H89	18.00		885-1099	H8 Color Graphics Tiny PASCAL	20.00	19	885-1094	HDOS Fig-Forth H8/H89	40.00	18
885-1064-[37]	Disk IX H8/H89 Disk	18.00		885-1105	HDOS Device Drivers H8/H89	20.00	24	885-1132-[37]	HDOS Tiny BASIC Compiler	20.00	59
885-1066-[37]	Disk X H8/H89	18.00	10	885-1116	HDOS Z80 Debugging Tool	20.00	27	885-1134	HDOS SMALL-C Compiler	30.00	63
885-1069	Disk XIII Misc H8/H89	18.00		885-1119-[37]	BHBASIC Support	20.00	29	CP/M			
GAMES								885-1208-[37]	CP/M Fig-Forth H8/H89 2 Disks	40.00	18
HDOS				885-1120-[37]	HDOS 'WHEW' Utilities	20.00	33	885-1215-[37]	CP/M BASIC-E	20.00	26
885-1010	Adventure Disk H8/H89	10.00	4	885-1121	HDOS Hard Sec Sup Pkg 2 Disks	30.00	37	BUSINESS, FINANCE AND EDUCATION			
885-1029-[37]	Disk II Games 1 H8/H89	18.00	8	885-1123	XMET Robot & Cross Assembler	20.00	40	HDOS			
885-1030-[37]	Disk III Games 2 H8/H89	18.00	8	885-1126	HDOS Utilities by PS:	20.00	42	885-1047	Stocks H8/H89 Disk	18.00	
885-1031	Disk IV MUSIC H8 Only	20.00	25	885-1127-[37]	HDOS Soft Sector Support Pkg	30.00	45	885-1048	Personal Account H8/H89 Disk	18.00	
885-1067-[37]	Disk XI H8/H19/H89 Games	18.00	12	885-1128-[37]	HDOS DISKVIEW	16.00	46	885-1049	Income Tax Records H8/H89 Disk	18.00	
885-1068	Disk XII MBASIC Graphic Games	18.00	10	885-1129-[37]	HDOS CVT Color Video Terminal	20.00	46	885-1055-[37]	MBASIC Inventory Disk H8/H89	30.00	
885-1088-[37]	Disk XVII MBASIC Graph Games	20.00	14	885-8001	SE (Screen Editor)	25.00	28	885-1056	MBASIC Mail List	30.00	
885-1093-[37]	D&D H8/H89 Disk	20.00	16	885-8003	BHTOMB	25.00	28	885-1070	Disk XIV Home Fin H8/H89	18.00	
885-1096-[37]	MBASIC Action Games H8/H89	20.00	18	885-8004	UDUMP	35.00	28	885-1071-[37]	MBASIC SmBusPk H8/H19/H89	75.00	17
885-1103	Sea Battle HDOS H19/H8/H89	20.00	20	885-8006	HDOS SUBMIT	20.00	31	885-1091-[37]	Grade/Score Keeping H8/H89	30.00	14
885-1111-[37]	HDOS MBASIC Games H8/H89	20.00	23	885-8007	EZITRANS.	30.00	30	885-1097-[37]	MBASIC Quiz Disk H8/H89	20.00	18
885-1112-[37]	HDOS Graphic Games H8/H89	20.00	23	885-8015	HDOS TEXTSET Formatter	30.00	42	885-1118-[37]	MBASIC Payroll	60.00	30
885-1113-[37]	HDOS Action Games H8/H89	20.00	23	885-8017	HDOS Programmers Helper	16.00	42	885-1131-[37]	HDOS CHEAPCALC	20.00	47
885-1114	H8 Color Raiders & Goop	20.00	23	885-8024	HDOS BHBASIC Utilities Disk	16.00	46	885-8010	HDOS CHECKOFF	25.00	32
885-1124	HUGMAN & Movie Animation Pkg	20.00	41	CP/M				885-8021	HDOS Student's Statistics Pkg	20.00	44
885-1125	MAZEMADNESS	20.00	41	885-1210-[37]	CP/M ED (same as 885-1022)	20.00	20	885-8027	HDOS SCICALC	20.00	50
885-1130	Star Battle	20.00	47	885-1212-[37]	CP/M Utilities H8/H89	20.00	21	CP/M			
885-1133-[37]	HDOS Games Collection I	20.00	59	885-1213-[37]	CP/M Disk Utilities H8/H89	20.00	22	885-1218-[37]	CP/M MBASIC Payroll	60.00	31
885-8009-[37]	HDOS & CP/M Galactic Warrior	20.00	32	885-1217-[37]	HUG Disk Duplication Utilities	20.00	26	885-1233-[37]	CP/M CHEAPCALC	20.00	47
885-8022	HDOS SHAPES	16.00	45	885-1223-[37]	HRUN HDOS Emulator 3 Disks	40.00	37	885-1239-[37]	Spread Sht. Contest Disk I	20.00	
885-8026	HDOS Space Drop	16.00	49	885-1225-[37]	CP/M Disk Dump & Edit Utility	30.00	40	885-1240-[37]	Spread Sht. Contest Disk II	20.00	
885-8032-[37]	HDOS Castle	20.00	59	885-1226-[37]	CP/M Utilities by PS:	20.00	40	885-1241-[37]	Spread Sht. Contest Disk III	20.00	
CP/M				885-1229-[37]	XMET Robot Cross Assembler	20.00	40	885-1242-[37]	Spread Sht. Contest Disk IV	20.00	
885-1206-[37]	CP/M Games Disk	20.00	11	885-1230-[37]	CP/M Function Key Mapper	20.00	42	885-1243-[37]	Spread Sht. Contest Disk V	20.00	
885-1209-[37]	CP/M MBASIC D&D	20.00	19	885-1231-[37]	Cross Ref Utilities for MBASIC	20.00	43	885-1244-[37]	Spread Sht. Contest Disk VI	20.00	
885-1211-[37]	CP/M Sea Battle	20.00	20	885-1232-[37]	CP/M Color Video Terminal	20.00	46	885-8011-[37]	CP/M CHECKOFF	25.00	32
885-1220-[37]	CP/M Action Games	20.00	32	885-1235-[37]	CP/M COPYDOS	20.00	54	ZDOS			
885-1222-[37]	CP/M Adventure	10.00	35	885-1237-[37]	CP/M UTILITIES	20.00	55	885-3006-37	ZDOS CHEAPCALC	20.00	47
885-1227-[37]	CP/M Casino Games	20.00	38	885-1245-37	CP/M 85 KEYMAP	20.00	63	885-3013-37	ZDOS Checkbook Manager	20.00	54
885-1228-[37]	CP/M Fast Action Games	20.00	39	885-5001-37	CP/M 86 KEYMAP	20.00	51	885-3018-37	ZDOS Contest Spreadsheet Disk	25.00	58
885-1236-[37]	CP/M Fun Disk I	20.00	55	885-5002-37	CP/M 86 HUG Editor	20.00	52	885-8028-37	ZDOS SCICALC	20.00	50
ZDOS				885-5003-37	CP/M 86 UTILITIES by PS:	20.00	54	885-8030-37	ZDOS MATHFLASH	20.00	55
885-3004-37	ZDOS ZBASIC Graphic Games	20.00	37	885-8018-[37]	CP/M FAST EDDY & BIG EDDY	20.00	43	DATA BASE MANAGEMENT SYSTEMS			
885-3009-37	ZDOS ZBASIC D&D	20.00	50	885-8019-[37]	DOCUMAT and DOCULIST	20.00	43	HDOS			
885-3011-37	ZDOS ZBASIC Games Disk	20.00	52	885-8025-37	CP/M 85/86 FAST EDDY	20.00	49	885-1107-[37]	HDOS Data Base System H8/H89	30.00	23
885-3017-37	ZDOS Contest Games Disk	25.00	58	ZDOS				885-1108-[37]	HDOS MBASIC Data Base Sys.	30.00	23
UTILITIES								885-1109-[37]	HDOS Retriever ASM (3 Disks)	40.00	23
HDOS				885-3005-37	ZDOS ETCHDUMP	20.00	39	885-1110	HDOS Autofile (2 Disks)	30.00	23
885-1022-[37]	HUG Editor (ED) Disk H8/H89	20.00	20	885-3007-37	ZDOS CP/Emulator	20.00	47	885-1115-[37]	HDOS Navigational Program	20.00	25
885-1025	Runoff Disk H8/H89	35.00		885-3008-37	ZDOS Utilities	20.00	47	885-8008	Farm Accounting System	45.00	30
885-1060-[37]	Disk VII H8/H89	18.00		885-3010-37	ZDOS KEYMAP	20.00	51	CP/M			
885-1061	TMI Load H8 ONLY Disk	18.00		885-3022-37	ZDOS/MSDOS Useful Programs I	30.00	63	885-1219-[37]	CP/M Navigational Program	20.00	31
885-1062-[37]	Disk VIII H8/H89 (2 Disks)	25.00		885-3023-37	ZDOS/MSDOS EZPLOT	20.00	63	PC/IBM COMPATIBLE			
885-1063	Floating Point Disk H8/H89	18.00		885-8029-37	ZDOS FAST EDDY	20.00	53	885-6001-37	MSDOS Keymapper	20.00	59
885-1065	Fix Point Package H8/H89 Disk	18.00	10					885-6002-37	CP/Emulator II & ZEmulator	20.00	59
885-1075	HDOS Support Package H8/H89	60.00						885-8033-37	MSDOS Fast Edit	20.00	62
885-1077	TXTCON/BASCION H8/H89	18.00									
885-1079-[37]	HDOS Page Editor	25.00	15								
885-1080	EDITX H8/H19/H89 Disk	20.00									

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

PC/IBM COMPATIBLE



HUG NEW PRODUCTS

at the CP/M command level.

SYSINFO — This is a handy little program for “hackers” that tells you the following memory addresses in your system: The highest memory address, the BIOS warm boot entry, the BDOS entry, the CCP entry, and the system stack address. It warns you if you have not used MOVCPM to match your CP/M to the amount of memory you have.

LONGREL.LIB — These are the assembly macros that were presented in the REMark article “Long Relative Addressing for the 8080”. They provide a way to write position independent code for the 8080 processor. They are intended for use with the Digital Research MAC assembler.

SEARCH.ASM — This is the source code for a practical application for the LONGREL macros. This program adds a Search command to the DDT debugging tool.

SEARCH.BIN — The assembled SEARCH program, which can be loaded into and run from any location in memory while you are using DDT.

TABLE C Rating: (0),(1),(3),(10)

885-1246-37 CP/M HUG File Manager And Utilities \$20.00

Introduction: This disk contains a CP/M version of the HUG File Manager, and some miscellaneous utilities.

Requirements: An H8 computer with a Heath/Zenith terminal, an H/Z-89,90 computer, or an H/Z-100 computer and the appropriate version of 8-bit CP/M, and at least 48k of memory (128k on Z-100).

This disk contains the following files:

README	.DOC	SYSINFO	.ASM
HFM	.COM	LONGREL	.LIB
HFM	.ASM	SEARCH	.ASM
SYSINFO	.COM	SEARCH	.BIN

Program Author: P. Swayne, HUG

HFM — This is a CP/M program based on the original HUG File Manager. It is a menu driven utility that lets you copy files (including copying to other user numbers), delete files, type and print files, list files in hexadecimal, and sort or unsort the directory display. HFM displays all of the files (or as many as will fit) from the selected disk and user number on the screen, and highlights one of them. You can move the indicator (the highlighted entry) using the arrow keys on your keyboard to select a file to operate on, or flag several files for multiple copy or delete operations. Commands are executed by moving a second indicator (using Space or Back Space) to select a command and then pressing Return. You can also execute a command by pressing its first letter. (This method of command selection is modeled after Multiplan (tm) or Microsoft Word (tm).)

HFM allows you to page through screens of files if there are too many to fit on the screen, or to specify via wildcards which files will appear on the screen. You can change disks, change the logged drive, and change the logged user number from HFM, and you can copy files from the current user number to any other user number, including numbers 16-31, which are not supported

885-3024-37 MS-DOS/Z-DOS 8080 To 8088 Translator \$20.00

Introduction: The programs on this disk can translate the assembly source code for CP/M programs into 8088 assembly language, for operation under MS-DOS or Z-DOS. They have a feature not found in similar programs, the ability to prompt the user to make decisions that will optimize the code during the translation process.

Requirements: These programs will run on either a Z-100 or Z-150 series computer, and MS-DOS or Z-DOS with at least 128k of memory.

The following programs are on the disk:

README	.DOC
ICT	.DOC
ICT	.COM
ICT	.ASM
ICT2	.COM
ICT2	.ASM
WSCON	.A80

Program Author: P. Swayne, HUG

Program Content: ICT (Interactive Code Translator) is a program that translates CP/M 8080 assembly language code into MS-DOS 8088 assembly language code. During the translation process, ICT watches for opcodes that do not translate exactly into 8088 equivalents, and prompts you to make decisions on how you want them translated. For example, to translate the 8080 code PUSH PSW exactly requires the following lines of 8088 code:

LAHF
 XCHG AL, AH
 PUSH AX
 XCHG AL, AH

However, if the program you are translating only needs the accumulator saved, a simple PUSH AX instruction will suffice to translate PUSH PSW. ICT allows you to select from 3 options on a menu whenever it encounters a PUSH PSW instruction. Other 8080 with optional translations are POP PSW, INX, DCX, DAD, and all jump instructions. By allowing you to optimize the program during translation, ICT can help you omit hundreds or even thousands of bytes of non-essential code and save hours of optimizing work after the translation. ICT can also be used in a non-interactive mode, in which it translates all codes to exact equivalents. In this mode, it is probably the fastest translator available.

ICT also translates the expression CALL BDOS into a valid MS-DOS system call. Because of this, programs need little or no editing after the translation to produce a program that will work under MS-DOS.

ICT2 is a special version of ICT that works only under MS-DOS version 2 and allows you to specify full path names for the input and output files.

WCON.A80 is the source code for the CP/M version of the WCON text to WordStar translation program. It is provided to give you practice in translating programs. After you translate it, make a few edits, and assemble it, you will have a working MS-DOS version of WCON.

TABLE C Rating: (10)

**885-3025-37 MS-DOS/Z-DOS
 Miscellaneous Utilities \$20.00**

Introduction: This disk contains a generous collection of utilities for MS-DOS and Z-DOS users.

Requirements: Most of the programs on this disk will run on either a Z-100 series computer or Z-100 PC series (Z-150 series)

computer, using the MS-DOS or Z-DOS operating system with at least 128k of memory. A few of the programs are specific to the Z-100 or Z-150 computers.

This disk contains the following programs:

README .DOC
 Z-100 Only Programs:
 FASTIO .COM SETFAST .ASM
 FASTIO .ASM FIXPRT .COM
 SETFAST .COM FIXPRT .ASM
 Z-150 Only Programs:
 SCRNCCLK .COM CLOCK .ASM
 SCRNCCLK .RED COLOR .COM
 SCRNCCLK .ASM COLOR .ASM
 CLOCK .COM
 Programs for both systems:
 HFM2 .COM HSORT .ASM
 HFM2 .ASM WCON .COM
 HFMJR .COM WCON .ASM
 HFMJR .ASM TAB2SPC .COM
 ATTRIB .COM TAB2SPC .ASM
 ATTRIB .ASM TYPER .COM
 REPRINT .COM TYPER .ASM
 REPRINT .ASM MACROS .ASM
 HSORT .COM

Program Authors:

REPRINT — R. A. Metz

The HFM programs were developed from the original HFM program — R. A. Metz and P. Swayne
 All other programs — P. Swayne, HUG

FASTIO — This program speeds up all non-disk input/output under MS-DOS or Z-DOS on a Z-100 computer. It can make word processors and editors scroll faster and print faster. With FASTIO installed, standard WordStar (tm) will work as fast as when patched as described in REMark, without the patches. If your word processor or editor seems to work slower under MS-DOS than under Z-DOS, it will work faster with FASTIO installed on either operating system. Even typing files to the screen or listing the disk directory works faster with FASTIO. FASTIO intercepts all I/O system calls 1 through 12, plus the MS-DOS version 2 Xenix compatible read and write calls (when they refer to standard I/O devices), and performs the operations itself as fast as the

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.

system will allow.

Some other advantages of FASTIO are that it provides the Xenix read and write calls (to standard I/O devices only) under Z-DOS, and it provides CP/M-like line editing (Control-X to erase a line, Control-R to repeat it, etc.) when DOS function 10 is used. Some disadvantages are that I/O cannot be redirected under MS-DOS 2 while it is active, and that printing files via PRINT.COM under MS-DOS 2 is slowed down. However, the SETFAST program, described below, can remove these disadvantages. For Z-100 only.

SETFAST — This program can disable and re-enable FASTIO once it has been installed. It can even be used while the MS-DOS 2 version of PRINT.COM is in the process of printing a file, to select faster printing or faster screen I/O as desired. For Z-100 only.

FIXPRT — If you are using version 2.13 or 2.15 of MS-DOS, printing may still be slower than it should be even with FASTIO installed. When you run this program, it temporarily installs a patch that makes these versions of MS-DOS print as fast as the later versions. For Z-100 only.

SCRNCLK — This program provides a digital clock display on your screen that is always there, even while you are running other programs. It is highly compatible with PC programs, and has been used with such programs as PC PALETTE and SIDEKICK. SCRNCLK.RED is the same as SCRNCLK.COM, except that the clock display is red instead of white. The .ASM file can be altered to make other colors. For Z-150 only. A Z-100 version of this program is available on HUG disk 885-3014-37.

CLOCK — This program can be used to disable and re-enable the SCRNCLK clock display.

COLOR — This program allows you to change the colors of the characters and background on the screen while you are in the text mode. It works like the GW-BASIC color command. For Z-150 only. A Z-100 version of this program is on HUG disk 885-3008-37.

The following programs are for both Z-100 and Z-150.

HFM2 — This is a new version of the HUG File Manager that was released on the HUG Menu System disk (885-3020-37). This version works like the original version, but does not require COMMAND.COM or a separate HEXLIST utility to work. It, therefore, is more efficient in systems with little memory. In addition, this version gives you the choice of printing files using PRINT.COM or directly. The original version always uses PRINT.COM. For MS-DOS version 2 only.

HFMJR — This is a junior version of the HUG File Managers for those who are using Z-DOS or MS-DOS version 1.25. It has all of the features of the original HUG File Manager except the ability to work with tree directories, to use PRINT.COM for printing files, or to change disk labels. For all versions of MS-DOS/Z-DOS.

ATTRIB — This program allows you to change the attributes on any file, so that you can patch system files, etc. It works like the FLAGS program that comes with the Programmer's Utility Pack. For MS-DOS version 2 only.

REPRINT — This program permits you to redirect output to the printer under MS-DOS version 2 in the same way that you normally can with screen output. A good use for this program is to run WordStar and have its printer output go to a disk. The result is a disk file with all of the printer controls in it for special effects

(underline, bold, etc), that can be then copied to a printer any number of times. For MS-DOS version 2 only.

HSORT — This program is an MS-DOS translation of the original HUG Sorter program that was originally released for CP/M and HDOS. It sorts lines of text rapidly into alphabetical order. It uses a Shell-Metzner sort routine. For MS-DOS or Z-DOS.

WCON — This is a translation of the original CP/M WCON program from HUG. It translates an ordinary text file into the format used by WordStar in its document mode. The program changes all spaces except for one space after words and two after periods to "soft spaces", and it changes all carriage return characters within paragraphs to "soft returns". The result is a file that can be processed by WordStar's reformat command. WCON can translate a file of any size, and is very fast. FOR MS-DOS or Z-DOS.

TAB2SPC — This program converts tabs in a text file to an equivalent amount of spaces, so that the appearance of the file is retained. This program can be used on a file before WCON, if it has tabs in it. For MS-DOS or Z-DOS.

TYPFR — This program is useful whenever you want to type a few lines and have them printed. It allows you to include any special codes to set up effects on your printer. For MS-DOS or Z-DOS.

MACROS.ASM — Required to assemble REPRINT.ASM.

TABLE C Rating: (0),(1),(2),(3),(10)

885-5008-37 CP/M-86 8080 To 8088 Translator And HFM \$20.00

Introduction: This disk contains a program that can translate the assembly source code for CP/M programs into 8088 assembly language, for operation under CP/M-86. It has a feature not found in similar programs, the ability to prompt the user to make decisions that will optimize the code during the translation process. This disk also contains a CP/M-86 version of the HUG File Manager.

Requirements: A Z-100 series computer, CP/M-86, and at least 128k of memory.

This disk contains the following programs:

README	.DOC	WCON	.ASM
ICT	.DOC	HFM	.CMD
ICT	.CMD	HFM	.A86
ICT	.A86		

Program Author: P. Swayne, HUG

ICT — This is the Interactive Code Translator that translates CP/M 8080 assembly source code into CP/M-86 8088 assembly source code. For a description of this program, see part no. 885-3024-37. This version of ICT translates CALL BDOS into INT 224, so that programs can be assembled to run under CP/M-86 with little or no work after the translation.

WCON.ASM — This is the source code for the CP/M version of the WCON text to WordStar translation program. It is provided to give you practice in translating programs. After you translate it, make a few edits, and assemble it, you will have a working

CP/M-86 version of WSCON.

HFM — This is a CP/M-86 program based on the original HUG File Manager. It is a menu driven utility that lets you copy files (including copying to other user numbers), delete files, type and print files, list files in hexadecimal, and sort or unsort the directory display. HFM displays all of the files (or as many as will fit) from the selected disk and user number on the screen, and highlights one of them. You can move the indicator (the highlighted entry) using the arrow keys on your keyboard to select a file to operate on, or flag several files for multiple copy or delete operations. Commands are executed by moving a second indicator (using Space or Back Space) to select a command and then pressing Return. You can also execute a command by pressing its first letter. (This method of command selection is modeled after Multiplan (tm) or Microsoft Word (tm).)

HFM allows you to page through screens of files if there are too many to fit on the screen, or to specify via wildcards which files will appear on the screen. You can change disks, change the logged drive, and change the logged user number from HFM, and you can copy files from the current user number to any other user number, including numbers 16-31, which are not supported at the CP/M command level.

TABLE C Rating: (0),(1),(3),(10)



The Software Toolworks presents:
THE COMPUTER CHEF™ SERIES

ALL RIGHT, COMPUTER CHEF, I DIDN'T GET TO THE MARKET TODAY. WHAT CAN I DO WITH A HEAD OF CABBAGE AND A CAN OF SPAM?

OH, THAT LOOKS NICE. LET'S SCALE IT FOR FIVE PEOPLE AND PRINT IT OUT.

HEY, MOM! HOW LONG DO YOU COOK SPAM FOR?

NOW WHERE'S MY RECIPE FOR CHOCOLATE MOUSE!

THE COMPUTER CHEF SERIES ALSO INCLUDES 'BEST OF WOK TALK' AND 'WHAT'S FOR DINNER'. IT'S AVAILABLE FOR CP/M, ZDOS AND HDOS AT MOST HEATH/ZENITH DEALERS. CALL OR WRITE FOR FREE AI PRODUCT CATALOG AND NEAREST DEALER.

A SOFT BUY IN A HARD WORLD!

The Computer Chef Series includes Computer Chef and Best of Wok Talk, \$29.95 each, and What's for Dinner, \$19.95. The programs store, scale and help choose from over 300 available recipes or from those you've entered. The Computer Chef Series from your local retailer or The Software Toolworks, 15233 Ventura Blvd., Suite 118, Sherman Oaks, CA 91403 (818) 986-4885.

CP/M is a registered trademark of Digital Research.

Controlled Data Recording Systems Inc.

AVAILABLE NOW THE FDC-H8 DOUBLE DENSITY 8" AND 5.25" CONTROLLER FOR THE H8 COMPUTER

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

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

PRICE \$495

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

ANNOUNCING THE MODIFIER

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

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

MODIFY 89 is set for the following disk types:

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

* = Double sided 5.25" drive required
S.S. = single sided, D.S. = double sided, S.D. = single density, D.D. = double density

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

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

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

ACCESS™

The State of the Art in Communications

- Talk to bulletin boards, information utilities, main frames or other micros
- Send and receive any kind of file—text, binary, .EXE, .COM, etc.
- Transfer files using a variety of protocols, including XMODEM protocol
- Use ACCESS to turn your computer into a communications robot
- Unattended auto-answer mode configures to caller's baud rate, and
- Depending on caller's password, grants unlimited, limited, or no access
- Logs password, duration, date and time of calls you receive
- Logs calls you place—recording number dialed, duration, date & time
- Compatible with all modems, even USR S100 and other board modems

Note: Certain features not included in 8 bit versions.

Start Communicating Right Away!

You can start communicating as soon as you pull ACCESS out of its package. We don't make you learn a strange new language of names and symbols the way most programs do. Our customers proclaim it—ACCESS is the easiest and fastest to use!

Your Own Communications Robot

ACCESS can dial, log on, send or receive, answer prompts, even make decisions. Sit back and relax while ACCESS does your work for you.

While you eat dinner or watch TV, have ACCESS call the local bulletin boards, pick up messages to you, drop off messages to others, and collect the classified ads for you to go through later.

Have ACCESS reduce your telephone and timesharing bills by making calls for you late at night, when rates are lowest. When you get up in the morning, your computer can be chock full of fresh information, just like your morning paper.

Instant Links with Other Micros

ACCESS gives you instant phone links with any other 8-or 16-bit microcomputer for fast, efficient file transfers or just to chat. Or you can direct-cable your computer to others.

And ACCESS answers calls and lets people with one of your passwords have either limited or unrestricted use of your system . . . including freedom to run programs other than ACCESS!

Make Those Tough Connections

Again and again our customers have thanked us for the power to talk with main frames, minis, and specialized networks, with emulators, microcontrollers, and development boards. And you'll thank us too, when ACCESS gets you into systems that defy people with other programs.

Don't Let Your Computer Act Dumb!

Other programs only let your computer act like a dumb terminal while you're on line, but ACCESS leaves the full power of your computer in your hands. Not only can you view your directories and files, you can run your other software—without leaving ACCESS or hanging up the phone.

What Makes ACCESS So Powerful?

To make ACCESS so easy to use, yet fast and flexible, took state-of-the-art programming and years of evolution. To achieve compact power, ACCESS is written in C, the language used to write the famous operating system UNIX. And ACCESS is interrupt-driven for ultimate speed with reliability.

How You Can Get ACCESS

Buy ACCESS from your local Heath or ZDS dealer. If there isn't a dealer near you, order it direct from Hilgraeve Inc.

ACCESS, H-8 or H/Z-89 with CP/M (64K req'd) \$49.95

ACCESS, H/Z-110 or 120 with ZDOS, CP/M-85, or -86 \$59.95

ACCESS, H/Z-150 or 160 with MSDOS or CP/M-86 \$69.95

Add \$3.00 shipping, MI residents add 4%

Hilgraeve Inc. P.O. Box 941 Monroe, MI 48161 (313) 243-0576



Z-100



SYSTEMS
SOFTWARE
NEWS

Issue Number 3, Benton Harbor, Michigan

Sidekick/Spotlight - Like Utility for Z-100

Perks™ Desktop Utility announced for Z-100

Benton Harbor - Barry Watzman has announced the availability of Perks™, a desktop utility program for the Zenith Data Systems Z-100 series of computers. Perks is a background utility which is similar in many respects to products such as Sidekick and Spotlight that have been available for the IBM-PC and PC clones since mid-1984, but which have not been available for the Z-100 series.

Once loaded, Perks remains dormant in the Z-100's memory until it is activated by pressing "Shift-Break" on the Z-100 keyboard; on activation, any of Perks' features may be used as desired; then by selecting the "exit" function or by pressing "Shift-Break" again, the user is restored to his original activity which is unaffected by the intervening use of the Perks utility.

Perks features include:

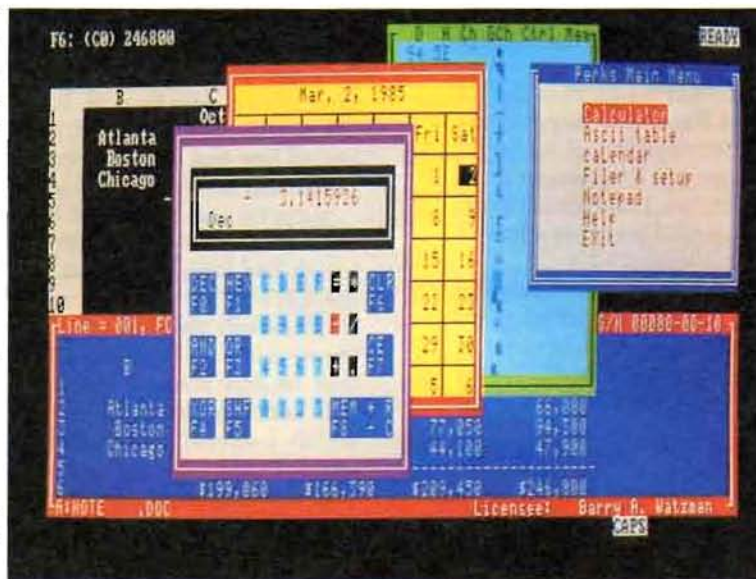
- A four-function calculator (with memory) featuring a 9-digit floating decimal display and the ability to perform operations in both decimal and hex including decimal/hex conversions. Although the calculator operates with a full floating decimal point, the numeric system uses only integer arithmetic for internal calculations, thus eliminating the rounding errors which are common in floating point systems.
- A Perpetual Calendar with an Associated Appointment Calendar - The Perpetual calendar accepts any date in the 20th century and displays a calendar for the month in which the date falls, with the date highlighted; by pressing a single

key, an appointment calendar for that date is displayed.

- An ASCII Table - an invaluable aid to programmers, the ASCII table gives the hex value, the decimal value, the character representation, the graphics character representation and the mnemonics (such as BEL, ETX, ACK, etc.) for all binary character values from 00 to 7F hex.

- Context Sensitive Help Screens - The Help key may be pressed at any time while using Perks, and Help information corresponding to the particular Perks functions in use at the time will be displayed.

- A Screen Saver - When Perks detects that the Z-100 is not actually being used (because neither a keyboard entry nor video display output has occurred for a user-specified time interval), Perks turns off the Z-100 video display screen to prevent screen damage caused by leaving a fixed display on the screen for long periods of time. Upon resumption of activity, Perks instantly restores the screen with no loss of data. The screen saver is active regardless of



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. Due to printing lead times, this photo was taken using a functional prototype which may differ in some respects from the final production version.

whether Perks is in its active or dormant state. The user may set the screen saver "inactivity threshold", the time period of inactivity which must pass to activate the screen saver, or turn the screen saver off entirely.

- A Notepad (text editor) similar in many respects to Wordstar's non-document mode. Although provided with only a small (about 4K) text buffer to minimize Perks' memory requirements, the notepad features on-screen video editing

Copyright (c) 1985, Barry A. Watzman, all rights reserved. Printed in the USA by Action Print, Louisville, Kentucky

Perks and Pane-Relief are trademarks of Barry A. Watzman. CP/M, MP/M-86, CP/M-86, CP/M Plus, Concurrent CP/M-86, MAC, SID and RMAC are trademarks of Digital Research, Inc. MS-DOS is a trademark of Microsoft, Inc. Z-100, Z-DOS are trademarks of Zenith Data Systems, Inc. IBM and IBM-PC are trademarks of International Business Machines, Inc. Sidekick is a trademark of Borland International, Inc. Spotlight is a trademark of Software Arts, Inc. WordStar is a trademark of MicroPro International, Inc. 1-2-3 is a trademark of Lotus Development Corp.

ADVERTISEMENT

including the following functions:

Cursor Movement:

Up/Down/Left/Right
Top-of-Screen/Bottom-of-Screen
Word-Left/Word-Right
Beginning-of-Note/End-of-Note

Block Functions:

Move/Copy/Delete/Print/Write

Scrolling:

Up Line/Page, Down Line/Page

String Operations:

String Search/Search and Replace

Modes:

Both Insert and Over-Write Modes

Other:

Reading of disk file into Notepad
Delete character under cursor
Delete character left of cursor
Insert/Delete Line

An "Import" feature is also supported which turns off all Perks windows without exiting Perks, allows the user to mark a block of text on the underlying screen (from the application being used when Perks was activated), and then

returns to the Perks Notepad with all text from the marked block having been inserted into the Notepad note at the cursor position.

• **Filer** - The filer permits access to many DOS functions including changing the directory path, erasing files and listing the directory of any drive.

All Perks features operate through windows managed by the PANE-RELIEF™ Window manager (see related story). When using Perks, it is not unusual to have three to five windows on the screen at once; windows may be moved around in real-time and in all four directions by the user; the size of the notepad window may also be varied in real time by the user. Perks saves the entire contents of the screen underlying it; upon exiting Perks, the screen is completely restored, including any color and/or bit mapped graphics which may have been present.

Perks is coded entirely in assembly language and uses less than 64K of memory (combined total for both code and data). Perks is available for Z-100's (e.g. 100/110/120 series; not for 150/160 series) with 192K or more of memory under the Z-DOS 1.X and MS-DOS 2.X operating systems. A version for the Watzman implementation of the CP/M-86 operating system on the Z-100 will also be available at a later date. Perks is priced at \$99.97. A signed license agreement is required with each order.

Watzman Operating Systems For Z-100 Enhanced

Barry Watzman has announced the availability of greatly enhanced implementations of Digital Research's CP/M Plus, CP/M-86 and MP/M-86 operating systems. The enhanced versions have been shipped to all new customers who ordered the Watzman implementation of these operating systems subsequent to Feb. 15th, and have also been made available to customers who had purchased the Watzman implementations of these operating systems previously.

Among the enhancements to the CP/M Plus BIOS in the 1.2 release are the addition of a Screen Saver, full support for 96-tpi floppy disk drives, and the conversion of all serial and parallel peripheral devices to interrupt driven operation with full FIFO buffering in both directions. Enhanced utilities and source code for the BIOS (not provided in earlier releases) are also included in the version 1.2 release.

The changes to both CP/M-86 and MP/M-86 (reflected in the 2.X versions of the BIOS and XIOS, respectively) are much more significant, as many of the features present in the initial release of CP/M-Plus were added to CP/M-86 and MP/M-86 for the first time. Among the additional features included are:

- Built-in Memory disk
- Built-in Screen Saver
- Interrupt-driven FIFO-buffered serial & parallel peripherals

- Full 96-tpi support
- Drive Search Mask (cuts boot time up to 85%)
- New Disk Drivers (up to 300% faster)
- XON-XOFF support

In addition, both CP/M-86 and MP/M-86 now come with improved utilities offering additional functionality over previous versions. For example the Configuration program now allows selection of Memory Disk size, Cursor Type, Cursor Blinking on/off, drive search mask, wrap/discard at end of line and Screen saver threshold, none of which were supported in earlier versions.

Surprise was expressed by some customers that such an extensive update was offered for the MP/M-86 package, which had been discontinued for over seven months, or for the CP/M-Plus package, which was discontinued by Digital Research about 60 days prior to the announcement of the upgrade. When questioned about this, Watzman noted that he has always assured his customers that "support will continue to be available for any software products which I develop and market, even if the products themselves should subsequently be discontinued". Watzman added that this would continue to be the case, and that his customers can look forward to continued consultation and maintenance services for their products.

CP/M-Plus Discontinued Amid Rave Reviews

On December 18, 1984, Digital Research unexpectedly discontinued CP/M Plus as a retail product. As a consequence of this, the Watzman implementation of CP/M-Plus for the Z-100 will be discontinued when present inventories are exhausted and additional supplies cannot be obtained. The CP/M Plus situation parallels what had happened previously with MP/M-86, which was also discontinued by Digital Research after a successful version of the product was offered for the Z-100. The only significant difference is that field

inventory levels for CP/M-Plus are significantly higher than those which existed for MP/M-86 (for which there was essentially no field inventory), thus permitting continued sales for a longer period following the discontinuance. Barry Watzman, who offered dual processor MP/M-86 for the Z-100, noted that when his MP/M-86 inventories were exhausted in July, 1984, there were between 100 and 200 customers who wanted the product and who had to be turned away in the 60 to 90 days following the discontinuance.

He estimated that field inventories of CP/M Plus would last into the 2nd or 3rd quarter of 1985, depending on sales and exactly how many copies were in the hands of distributors and retailers. The discontinuance of CP/M Plus came just as the product was upleveled to the 1.2 version of the BIOS (see related story) and as the product began getting excellent reviews and praise from customers who had purchased it. Writing in the January issue of the H-Scoop newsletter (#58), Henry Fale presented the following comparison of CP/M-85 and CP/M-Plus on the Z-100 (the task was a DBase II pack and index using 836 records of 165 bytes each):

	Zenith CP/M-85	Watzman CP/M-Plus
5" Floppies	46:42	10:57*
8" Floppies	19:20	8:55*
Winchester	9:26	4:19
Memory Disk	Not Supported	2:35

*These two results were measured but were omitted from the H-Scoop article cited.

Watzman noted that speed differences aside, there are many features present in CP/M-Plus which simply do not exist at all in CP/M-85, including the following:

Print Buffer	Video Screen Editing of
Memory Disk	Console Input
Drive Search Path	Time and Date Stamping
Screen Saver	Password Protection

Support for 4 ea. 5" & 8" floppies (vs. 2 ea w/CP/M-85)
 Support for 4 Winchester Partitions (vs. 2 w/CP/M-85)
 Support for Banked Memory
 Support for Partitions Up to 16Mb (vs. 8Mb Max w/CP/M-85)
 Enhanced & additional utilities
 Automatic Login of Removable Media
 Automatic paging of console output
 Support for Resident System Extensions

Ability to execute 16-Bit code imbedded within 8-bit programs
 MAC, SID, RMAC, LINK, LIB, XREF included (\$200.00 separately)
 On-Line HELP facility
 Console I/O to/from Disk Files
 Directory Hashing, BDOS LRU Deblocking
 Improved I/O Redirection
 Vastly improved documentation
 More powerful batch processor
 Support for files larger than 8Mb

Watzman acknowledged that because of its price (\$350.00), CP/M-Plus is not for everybody; however he recommends it highly to Z-100 owners who are using 8-bit software extensively. He noted that if a customer bought CP/M-85 and then purchased a memory-disk, screen saver, print spooler and DR Assembler Plus Tools (MAC, SID, RMAC, LINK, LIB, XREF) separately, the customer would have spent much more than the price of CP/M-Plus while having received only a tiny fraction of CP/M-Plus's features.

CP/M-86 "Installation Kit B" Price Increased to \$124.97

At the same time that Digital Research was discontinuing CP/M Plus through retail distribution, they were also announcing a \$40.00 price increase (from \$60.00 to \$100.00) on the IBM-PC version of CP/M-86. Since this package represents a major portion of Barry Watzman's CP/M-86 "Installation Kit B" for the Z-100, the subsequent announcement of a \$25.00 price increase for "Installation Kit B" (from \$99.97 to \$124.97) was hardly unexpected.

"Installation Kit B", on the market for about a year, is a product which includes Digital Research's IBM-PC version of CP/M-86 bundled with a diskette and manual developed by Mr. Watzman that converts it into a Z-100 version of CP/M-86.

Watzman noted that even with the price increase, "Installation Kit B" was still selling for less than half of Zenith's \$250.00 list price for their version of CP/M-86. Watzman cited the following features of his \$124.97 implementation of CP/M-86 as among those which are not present in the

\$250.00 Zenith version of CP/M-86:

- Screen Saver
- Memory Disk
- Extended Double Density 5" Formatting
- Extended Double Density 8" Formatting
- Support for four 8" floppies (vs. two in the Zenith version)
- Support for four 5" floppies (vs. two in the Zenith version)
- Support for four Winchester Partitions (vs. two in the Zenith version)
- Significantly faster floppy disk drivers
- Interrupt-driven FIFO-buffered (in both directions) Serial & Parallel ports
- Drive Search Mask for faster Bootup
- A higher level of on-going support

Watzman advised anyone considering the purchase of CP/M-86 for the Z-100 to "Compare the features and buy the best — even if it's cheaper."

PANE-RELIEF™ Window Manager to be Offered to Software Developers

Barry Watzman has announced that the PANE-RELIEF™ window manager, which he had planned to use in his now-shelved Z-100 implementation of Concurrent CP/M-86 with Windows (see Z-100 SSN issue number 2) and which is used in the Perks Desktop utility, will be licensed separately to software developers who wish to include windows in their own products.

PANE-RELIEF supports an unlimited number of Z-100 windows, each of which is totally independent of the others. Each window looks like an intelligent terminal to the application driving it, with all standard Z-19/Z-100 terminal functions available; however in addition to the standard intelligent terminal features, Pane-Relief also supports such window-specific features as window movement, window expansion

and contraction and window bordering, with or without text imbedded in the top and/or bottom border.

PANE-RELIEF is licensed in linkable Object form (only) and is available in three formats: one format suitable for linking under Digital Research 16-bit operating systems with Digital's LINK86 linker (DR-OS version), a second format suitable for linking under MS-DOS with Digital's LINK86 linker (MS-DOS version), and a 3rd format suitable for linking under MS-DOS with Microsoft's LINK linker. Licensing terms for PANE-RELIEF are subject to negotiation and will normally include both an up-front fee plus a royalty on each software item using PANE-RELIEF which is sold. For further information, software developers are encouraged to contact Mr. Watzman at (616) 925-3136.

How to Order the Products Described in This Publication

Please read these instructions to order the products described in this publication:

A single CPU license for Perks is \$99.97. There is no shipping charge on Perks for US or Canadian destinations. Perks is not copy protected, but it does display the licensee's name on the screen. Please print the personalization text (which must be the licensee's name abbreviated to 20 characters or less) in the place provided on the license agreement, which must be signed and returned with your order before it will be shipped.

CP/M-86 "Installation Kit B" is \$124.97 for a single CPU license. The license agreement, which is required before the product will be shipped, covers only the Watzman portion of the software; a separate Digital Research license is contained within the DR portion of the product, which you will receive sealed. There is a base shipping charge of \$5.00 for street addresses in the Continental US (48 states), see below.

CP/M-Plus is \$350.00 with a base shipping charge of \$6.00; all other terms are as for CP/M-86, above. As this is a discontinued product, potential customers are urged to verify availability prior to ordering.

Installation kits without the operating systems themselves are also available for CP/M-86, CP/M-Plus and MP/M-86 at \$75.00, \$115.00 and \$150.00, respectively. There is a base shipping charge of \$3.00 each on these items, and a license agreement is required.

The base shipping charge is for street addresses within the continental US (48 states) only. For Alaska, Hawaii and Canada, double this amount; for P.O. Boxes, add \$3.00. Michigan residents add 4% sales tax. Foreign customers should inquire as to shipping and export license requirements prior to ordering.

Software License Agreement

To: Mr. Barry Watzman
560 Sunset Rd.
Benton Harbor, Mi. 49022
(616) 925-3136

The undersigned hereby offers to license the following software product(s) subject to the terms and conditions stated herein:

- | | |
|--|---|
| <input type="checkbox"/> Perks Desktop Utility
(specify personalization text below) | <input type="checkbox"/> CP/M Plus Installation Kit |
| <input type="checkbox"/> CP/M-86 Installation Kit "B" | <input type="checkbox"/> MP/M-86 Installation Kit |
| | <input type="checkbox"/> Other _____ |

The licensee agrees that this software is copyrighted and that what is being purchased is a license to use the software, and not title to the software itself.

The licensee and the licensor agree that this license is perpetual unless it is canceled voluntarily by the licensee, or unless the licensee defaults on payment or violates the terms of this agreement, in which case the licensor may cancel it. In any case, it is agreed that all sales are final and that there will be no refund of the license fee once the sealed media pack has been opened. In the event of cancellation, the licensee agrees to destroy the original software supplied by the licensor and all copies of the original which have been made and to furnish the licensor with a notarized sworn statement certifying that such destruction has been accomplished.

The licensee agrees that the only warranty made by the licensor is to replace the documentation and/or media within 90 days of the license purchase if it is determined that the media and/or documentation is physically unreadable due to recording or printing errors. With regard to the contents of the software and documentation themselves, it is agreed that there is NO WARRANTY WHATSOEVER, EITHER EXPRESS OR IMPLIED, AND THAT THE LICENSOR SHALL NOT BE LIABLE FOR WARRANTIES OF FITNESS OF PURPOSE OR MERCHANTABILITY, NOR FOR INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES. IN THE EVENT THAT THIS DISCLAIMER IS DISALLOWED BY LAW, IT IS AGREED THAT THE LICENSOR'S LIABILITY IS LIMITED TO REFUND OF THE PURCHASE PRICE PAID UPON RETURN OF THE ORIGINAL PROGRAM DISK(S) AND THE DESTRUCTION OF ALL COPIES WHICH HAVE BEEN MADE. THE LICENSEE ALSO AGREES THAT THIS PRODUCT IS NOT INTENDED AS "CONSUMER GOODS" UNDER STATE OR FEDERAL WARRANTY LAWS.

The licensee agrees that under the terms of this license the software may be used only on a single computer owned by the licensee. The licensee agrees to take reasonable steps to protect the software from theft or use contrary to the terms of this agreement.

The licensee also understands that under this license he has the right to make copies of the software and/or documentation as required for his own use of the software, and to modify the software, but that he does not have the right to make this software available to anyone for use on a computer other than as specified above, and he agrees not to remove the copyright notices, personalization text and/or serial numbers from the software, listings or messages. The licensee owns any additions or adaptations which he may make to the software, but the licensee agrees that if the license is terminated he may not continue to use any part of the software provided by the licensor even if he has modified it.

The licensee understands that from time to time he may receive or have the option to purchase updates to this software or additional software offered to him by the licensor. The licensee agrees that any such software which he may license, purchase or receive from the licensor in the future as a consequence of this purchase will be covered by the terms of this agreement, unless a separate agreement is executed covering such software. The licensee also agrees that the licensor has no duty or obligation to supply any such updates or additional software.

The licensee understands that operating system installation kits consist of the software necessary to install the specified operating system(s), which are products of Digital Research Inc. of Pacific Grove, Calif. The licensee understands that while such installation kits may be purchased with the specified operating systems, the installation kit and the operating systems are separate products, and the licensee agrees that the licensor will not be responsible in any way for the operating systems themselves.

Licensee (print) _____ Address _____ Phone _____

Authorized Signature _____ Title _____ Date _____

Personalization Text for Perks (Licensee Name abbreviated to 20 characters). Please print:

Excursions Into FORTRAN

Hank Lotz
2024 Sampson Street
Pittsburgh, PA 15221

This article deals with Microsoft FORTRAN-80 under CP/M (2.2.03 or 2.2.04), but there is the hope that users of other FORTRAN versions and other systems might chance to pick up a stray programming idea or two from among the accompanying lines of code.

Judging from what I hear and read, I think there are more regular readers of REMark who have FORTRAN compilers, than might be guessed from the number of FORTRAN-oriented articles this magazine has published. And there is much for us to learn. Even if you've previously programmed in FORTRAN on large computers, a good deal of familiarization is necessary when you begin trying to do so on a microcomputer. I can think of a few programming aims that are either impossible, purely in FORTRAN, or else require some special manipulation to achieve. Sometimes certain characteristics of the CP/M system or of the FORTRAN language must be explored in order to produce desired program behavior. Even though I can't cover all of these "impossible" tasks in the present article, I was bearing them in mind when I chose the title, the mood of which could extend to sequels describing other "excursions", should the opportunity materialize. Whether it does or not, I would like to present here my work on a couple of program features for which I felt a relatively immediate need.

A Nice Program Feature

It is a great convenience to be able to pass parameters to a program, even to an interactive program, from the system command line. At a more appropriate time, you will be referred to Listing 1 where you can see the mechanics of how this is done. The "command line" is the string of characters typed at the console after a system prompt, for example:

```
A>EDIT OLDFILE NEWFILE
```

In this command line (or "command string") the Magic Wand editor, EDIT, is invoked, and the parameters passed to it are the file names on which it will operate. OLDFILE will be read from disk and, after editing it, EDIT will create NEWFILE. CP/M's editor, ED, also handles command string parameters, for example:

```
A>ED DISKFILE B:
```

Here a file named DISKFILE will be edited and the result written to disk B:. Notice that the second parameter this time is not a file name, but a disk drive name.

With these examples I'm attempting not only to illustrate the power of this usage of the command string, but also to point out how TWO parameter fields, as opposed to just one, can frequently be necessary or desirable after the program name. It is

often equally important to supply BOTH an input file name and an output file name to some applications programs.

Even numeric quantities can be passed as input to a program via the command line, because how you parse (analyze) the parameters once they are retrieved by your program is left to your discretion. (Space won't allow a detailed procedure, but I'll mention that for numeric quantities, some reliance on the FORTRAN "DECODE" statement after a call to ANLSTR, described below, can work wonders.) However, probably the most important use for this parameter-passing feature indeed remains the handling of disk file names. Later in this article, I will present a routine that parses CP/M disk file names, shaping them into the standard format required by the Microsoft FORTRAN "OPEN" subroutine. File names can also be obtained in "pre-parsed" form from the CP/M file control blocks (FCB's) at system addresses 5C hex and 6C hex, but my subroutine, called FPARSE, can be used to process a file name that was input interactively (during program execution), as well as one extracted from a system command string. Moreover, FPARSE does some error-checking not offered by the CP/M system, and it has been thoroughly tested. But file name parsing comes later; first we have to fetch the string, and we'll discuss that now.

Retrieving and Analyzing Command String Parameters

After I became familiar enough with the CP/M system to see what it does with such command strings, and to learn what I had to do to get at them from within my own programs, the next step was to come up with a subroutine that could be used in all (or most) FORTRAN programs requiring access to command string parameters. The criteria were dictated by need, common sense, and human engineering (okay, user-friendliness) considerations. My requirements were as follows:

1. The subroutine's output must be a character string and a description of its properties, so that a main program can handle it intelligently and easily. This should hold regardless of the nature of the string (whether it is a file name, numeric quantity, string of options, etc.)
2. The parameter fields must be allowed to appear anywhere in the command line, without the programmer's having to worry about where the user might type them or how many blanks he might insert between fields.
3. The subroutine must be able to handle two entered fields, but must also work if only one field is entered.
4. An error flag must be returned to the main program if almost any conceivable error occurs. MAIN can then take its own appropriate action.

5. It would be nice if so useful a subroutine could be made to work optionally from an interactive mode, not just from a system command line.

Well, the product of all this labor I dubbed subroutine ANLSTR, for “analyze string”. Listing 1 gives the source. Here’s a simple main program that, when run, demonstrates the subroutine pretty clearly:

```

PROGRAM MAIN
INTEGER*1 MODE, ISTR(80), L1, L2, L3, L4, NCHR1, NCHR2,
IERSTR
MODE=0
CALL ANLSTR(MODE, ISTR, L1, L2, L3, L4, NCHR1, NCHR2,
IERSTR)
IF (IERSTR.NE.0) GO TO 900
IF (NCHR1.EQ.0) GO TO 300
WRITE(1,100) (ISTR(I), I=L1, L2)
100 FORMAT(1H, 'THE FIRST FIELD IS: ',80A1)
IF (NCHR2.NE.0) GO TO 200
WRITE(1,150)
150 FORMAT(1H, 'THERE IS NO SECOND FIELD.')
GO TO 999
200 WRITE(1,250) (ISTR(I), I=L3, L4)
250 FORMAT(1H, 'THE SECOND FIELD IS: ',80A1)
GO TO 999
300 WRITE(1,350)
350 FORMAT(1H, 'NO PARAMETERS WERE ENTERED IN COMMAND
STRING.')
GO TO 999
900 WRITE(1,901)
901 FORMAT(1H,
'WE BRANCHED HERE BECAUSE AN ERROR WAS DETECTED.' /
1 1H,
'(WERE THERE TOO MANY FIELDS?
WAS THE COMMAND LINE TOO ',
2 'LONG?') /)
999 END

```

One example of a run of this demo program might be:

```
A>MAIN PARAMETER1 PARAMETER2
```

Now let’s take a look at the arguments of subroutine ANLSTR. As we discuss the arguments, follow along with the argument list in the CALL ANLSTR statement (the fourth line in above listing). The subroutine uses 9 arguments. The first one, MODE, we set to 0 in the third line of the main program. In this mode, subroutine ANLSTR will look for parameters passed from the user’s CP/M system command line. Whenever we let MODE be 0, then it is the ONLY argument we must set in the subroutine call; all the others are returned. After the return to MAIN, the next argument (the array ISTR) will be a character string containing the command line parameters, which in the above example were “PARAMETER1” and “PARAMETER2”. Using the PEEK function, ANLSTR takes this string from the CP/M system DMA area which begins at address 80 hex. (Here’s where you should glance at Listing 1 to see the PEEK function being used.) I have allowed up to 80 characters for ISTR. This provides more-than-adequate storage for the parameter portion of 1 full command line typed on the CRT.

I’ll interject here a reminder that we’re primarily discussing the arguments returned from ANLSTR, not trying to detail how ANLSTR works. I’m hopeful that the comments found in Listing 1 can obviate the need for too much of a blow-by-blow account here in the text.

The rest of the arguments tell us where in the ISTR array both of the parameter fields start and finish, and how long each field is.

Specifically, L1 and L2 are the starting and ending subscripts (respectively) within ISTR that define the boundaries of field 1. L3 and L4 do exactly the same thing for field 2; that is, ISTR(L3) holds the first character of field 2, and ISTR(L4) the last. The number of bytes contained in each of the fields is already calculated for us, and is available in 2 returned values, NCHR1 and NCHR2. In this discussion, let’s consider NCHR2 first. NCHR2 will be the number of bytes in field 2. If just NCHR2 is zero, then only field 1 is present. Now consider NCHR1. If NCHR1 equals zero, then NO nonblank parameters at all were entered by the user in the command string. (That’s because if the first field is null (NCHR1=0), the second must be null also, otherwise the second would BECOME the first, and thus the first-field byte count, NCHR1, would be NONzero.) After the call the error flag, IERSTR, should be the first thing tested. A zero value means no errors. If a value of 1 is returned in IERSTR, then more than 2 fields were entered or the 80-byte limit on ISTR was exceeded.

In programs you write utilizing ANLSTR, you can test NCHR1, NCHR2, and IERSTR and then initiate an interactive mode, if those tests show that the string your user entered was illegal for your purposes (or nonexistent). To “go interactive” you may, within the same program run, assign to MODE the new value 1 and call ANLSTR again, after doing a READ of ISTR. But mode 1 will be covered more thoroughly in a moment.

If you are up to typing in the subroutine, ANLSTR, and the above main program, run MAIN as shown above, using various different experimental strings for the two parameter fields and you will see how delightful the whole idea really is! Try it without any parameters. Try it with three parameters. (If you try long strings for parameters, your terminal should have wrap-around enabled so you can see all the bytes.)

The Interactive Approach

Now let’s consider how we can use ANLSTR for interactive input to a program. Just in case all this “interactive” business is confusing anyone, it’s time for a definition: When a program prints a message on a CRT to prompt a user who then supplies keyboard input, we have an example of what I mean by the “interactive mode”; the user is interacting with the program during its execution. End of definition.

When we use ANLSTR in this interactive mode, two principal changes must be made in our procedure. First, MODE must now be set to 1 (instead of 0) in the main program. When MODE is 1, ANLSTR knows not to look in the system buffer for the ISTR array values as it did before, but instead to expect ISTR to be passed to it by the main program. Second, therefore, we must indeed assign the ISTR array values in MAIN and pass them to the subroutine, which will then perform the analysis for us. We do this, in the main program, by prompting the user and utilizing a READ statement.

It doesn’t seem as though the need for such scrutiny of input should be nearly so acute here as in mode 0. After all, couldn’t we just read it in and use it directly with no analysis? The answer is yes, but if you already have the subroutine loaded with your program it isn’t going to take up any more memory space, so why not take advantage of some of its convenient attributes. The advantages are worth it: If any leading blanks get typed in, they will be ignored by looking at ISTR only from L1 to L2. Indeed, you, the programmer, can be confident there will NEVER be any blanks from ISTR(L1) to ISTR(L2) inclusive, and likewise, from ISTR(L3) to ISTR(L4).

For another advantage, since keyboard input of the interactive kind often calls for only ONE field per READ, we can check the integrity of the input by examining NCHR2. If NCHR2 isn't zero, we know the second field, thus detected, was caused by one or more embedded blanks in the string as typed.

Also, NCHR1 is the number of NONBLANK bytes input from the keyboard. This can be useful in weeding out inputs which are too long. As an example, we will check the byte count before entering FPARSE (which is discussed next). The main program that serves to demonstrate FPARSE in the paragraphs below, doubles as an example of this interactive ANLSTR mode.

Parsing A CP/M File Name

Listing 2 shows subroutine FPARSE. The arguments are described in the source comments. FPARSE was kept as a separate routine, so that ANLSTR could be summoned independently by any program, even to handle strings that aren't file names. But while ANLSTR can be used without FPARSE, I do not advise a CONVERSE practice. In other words, it is strongly recommended that an FPARSE call always be preceded by a call to ANLSTR to obtain the input; see NOTES 1, 2, & 3 in the comment lines of Listing 2. FPARSE functions identically regardless of where ANLSTR gets its input; therefore, as was promised, we'll try interactive input this time, to illustrate that facet of ANLSTR (mode 1).

Let me first address some minor points. The uppercase conversion in FPARSE is to satisfy CP/M's propensity for capital letters. I have taken asterisk, question mark, delete, and semi-colon to be illegal file name characters. Of course, the non-printing ASCII characters with decimal values less than 32 are also not allowed. Period and colon are legal only for their proper functional usages, not as part of a file name per se. If any readers have some kind of exotic ideas for file names, they can always change the code in their copies. Also, in certain program statements, I've used numeric constants for ASCII bytes. (I had reasons for doing that, which I won't go into.)

Here's a simple main program to demonstrate subroutine FPARSE. Running this demo program can shed a lot of light:

After you've typed in subroutine FPARSE and this demo main program, PARSE, compile and load them, and run PARSE. Don't type any parameters in the command string for this demo because we're in the "interactive mode". (If you do, they'll just be ignored.) The command line should be simply:

A>PARSE

When you are prompted, enter a legal file name. Try this several times. Some legal variations are:

```
B:WXYZ
WXYZ
WXYZ.
N:.X
A:ABCDEFGH.XYZ
```

Notice there are no embedded blanks in legal examples. CP/M doesn't allow them either in keyboard input. FPARSE will insert blanks for you only when they are needed to conform to the FORTRAN standard form for file names.

Next try some illegal entries:

```
B::ABC (Too many colons)
A:XYZ.XYZ. (Too many periods)
```

```

PROGRAM PARSE
INTEGER*1 MODE,ISTR(80),L1,L2,L3,L4,NCHR1,NCHR2,IERSTR
INTEGER*1 NMRAW(14),NAMF(11),IDRV,IER,I,L
GO TO 2050
2040 WRITE(1,2041)
2041 FORMAT(1H,'REJECTED BY ANLSTR, TRY AGAIN!')
2050 WRITE(1,2051)
2051 FORMAT(1HD,'(CONTROL-C STOPS); GIVE FILE NAME--')
READ(1,2070)ISTR
2070 FORMAT(80A1)
MODE=1
CALL ANLSTR(MODE,ISTR,L1,L2,L3,L4,NCHR1,NCHR2,IERSTR)
IF(NCHR1.GT.14)GO TO 2040
C
C (THE LONGEST LEGAL CP/M FILE NAME IS 14 BYTES, INCLUDING DRIVE NAME.)
C
C IF(NCHR1.LE.0)GO TO 2040
C IF(NCHR2.NE.0)GO TO 2040
C
C A TEST OF IERSTR WOULD BE SUPERFLUOUS IN THIS CASE; WE'VE ALREADY
C DISALLOWED MORE THAN 1 FIELD AND RULED OUT NCHR1 > 14.
C
C THIS INITIALIZATION PROVIDES TRAILING BLANKS NEEDED BY FPARSE:
C
DO 2200 I=1,14
NMRAW(I)=32
2200 CONTINUE
C
L=0
DO 2250 I=L1,L2
L=L+1
NMRAW(L)=ISTR(I)
2250 CONTINUE
CALL FPARSE(NMRAW,NAMF,IDRV,IER)
IF(IER.NE.0)GO TO 2500
WRITE(1,2300)NAMF,NAMF,IDRV
2300 FORMAT(1H,'THE PARSED FILE NAME IS: ',11A1,25X,'12345678123'/
1 1H,'CALL OPEN(6,NAMF,IDRV)" IS NOW EQUIVALENT TO:"CALL OPEN',
2 '(6,'',11A1,'',',.I2,')"'')
GO TO 2050
2500 WRITE(1,2501)
2501 FORMAT(1H,'FPARSE SAYS FILE NAME IS ILLEGAL, TRY AGAIN!')
GO TO 2050
9999 END

```

```

ABC DAT (Embedded blanks)
Q:X (Illegal drive)
ABCDEFGHI.XYZ (Too long)
NAME*.FOR (Illegal character)
B: (Drive name only, "null" file name)
B:. (Another "null" file name)

```

Closing Comments

It's always difficult trying to comprehend someone else's source program, but even if one can't or doesn't want to follow the sub-routines' inner workings, he or she still shouldn't be deprived of

their usefulness. The time you'll spend carefully typing in these subroutines will pay off, because you'll use them in so many of your programs. You can omit the comment lines, since they are available in these published listings. In their absence, though, you could include a reference to the issue and page of this article. The subroutines are now yours, as well as mine. Apply them!

About The Author

Hank Lotz, now self-employed, was previously employed by the Westinghouse Electric Corporation for over 25 years. There, he worked in the generator development engineering department as technician, computer operator, and programmer. Concurrently, his electrical engineering credits were earned at the University of Pittsburgh during evening classes. Hank has built many Heathkits, including an H-89. His other interests include writing, chess, and listening to classical music. He also enjoys playing the piano.

Listing 1

```

SUBROUTINE ANLSTR(MODE,ISTR,L1,L2,L3,L4,NCHR1,NCHR2,IERSTR)
C BY HANK LOTZ; MAY 3, 1984
C SUBROUTINE TO RETRIEVE & ANALYZE A STRING FROM THE CP/M SYSTEM DMA
C AREA WHEN USED IN "MODE 0", OR, TO ANALYZE A STRING PASSED FROM
C MAIN WHEN USED IN "MODE 1".
C NOTES:
C THERE ARE NO BLANKS BETWEEN THE L1 AND L2 VALUES (INCLUSIVE) RETURNED
C BY THIS SUBROUTINE, NOR BETWEEN L3 AND L4 INCLUSIVE. IF A FIELD IS
C ABSENT ITS LIMITING SUBSCRIPTS ARE RETURNED AS ZEROS. NCHR1 SHOULD BE
C USED TO CHECK FOR ABSENCE OF ALL FIELDS (NCHR1=0). THIS SUBROUTINE
C DIFFERENTIATES ONLY BETWEEN BLANKS AND NONBLANKS.
C MODE 0 INPUTS: MODE
C MODE 0 OUTPUTS: ISTR ARRAY, L1, L2, L3, L4, NCHR1, NCHR2, & IERSTR.
C MODE 1 INPUTS: MODE, & ISTR ARRAY
C MODE 1 OUTPUTS: L1, L2, L3, L4, NCHR1, NCHR2, & IERSTR.
C PARTIAL LIST AND DESCRIPTION OF VARIABLES:
C -----
C IERSTR --- ERROR FLAG. SET TO 1 IF MORE THAN 2 FIELDS, OR IF KNT IS
C MORE THAN 80 CHARACTERS, OR (IN MODE 1) IF ISTR(80) NOT
C BLANK. SET TO 0 IF ISTR ARRAY IS ALL BLANKS, OR IF KNT<2,
C OR IF THERE ARE FIELD(S) BUT NO ERRORS.
C ISTR --- ARRAY CONTAINING THE STRING TO BE ANALYZED.
C KBLK --- RUNNING COUNT OF CONTIGUOUS BLANKS BETWEEN LFST AND LLST.
C KNT --- COUNT OF CHARACTERS IN ISTR (INCL ALL BLANKS IF MODE 0,
C NOT INCL TRAILING BLANKS IF MODE 1).
C (THIS DOES INCL CP/M INITIAL DELIMITING BLANK FROM DMA.)
C L1 --- LOW 1ST-FIELD SUBSCRIPT OF ISTR ARRAY. (0 IF NONE OR ERROR)
C L2 --- HIGH 1ST-FIELD SUBSCRIPT OF ISTR ARRAY. (0 IF NONE OR ERROR)
C L3 --- LOW 2ND-FIELD SUBSCRIPT OF ISTR ARRAY. (0 IF NONE OR ERROR)
C L4 --- HIGH 2ND-FIELD SUBSCRIPT OF ISTR ARRAY. (0 IF NONE OR ERROR)
C LFST --- SUBSCRIPT OF VERY FIRST NONBLANK IN ENTIRE ISTR ARRAY.
C LLST --- SUBSCRIPT OF VERY LAST NONBLANK IN ENTIRE ISTR ARRAY.
C NBLK --- NUMBER OF BLANKS BETWEEN LFST AND LLST.
C NCHR --- TOTAL NUMBER OF NONBLANK CHARS IN ENTIRE ISTR ARRAY.
C NCHR1 --- NUMBER OF CHARS IN FIRST FIELD. (IF NONZERO, THERE ARE NO
C BLANKS IN FIRST FIELD). IF 0, ENTIRE ISTR ARRAY IS BLANK,
C OR KNT=0, OR IERSTR WILL NOT BE ZERO.
C NCHR2 --- NUMBER OF CHARS IN SECOND FIELD. (IF NONZERO, THERE ARE NO
C BLANKS IN SECOND FIELD). IF 0, SECOND FIELD IS ABSENT, OR
C IERSTR WILL NOT BE ZERO.
C NPOS --- NUMBER OF POSITIONS BETWEEN LFST AND LLST INCLUSIVE.
C*****
INTEGER*1 I,IERSTR,ISTR(80),J,KBLK,KNT,L1,L2,L3,L4,LFST,LLST
INTEGER*1 MODE,N,NBLK,NCHR,NCHR1,NCHR2,NPOS
IF (MODE.NE.0.AND.MODE.NE.1)GO TO 998
IERSTR=0
KBLK=0
KNT=0
L1=0
L2=0
L3=0
L4=0

```

```

260 LLST=I
NCHR=NCHR+1
300 CONTINUE
C NCHR IS COUNT OF NONBLANK CHARS IRRESPECTIVE OF THEIR POSITIONS.
C
C IF (NCHR.EQ.0)RETURN
NPOS=LLST-LFST+1
NBLK=NPOS-NCHR
IF (NBLK.EQ.0)GO TO 990
C
C WE NOW KNOW THERE'S AT LEAST 1 BLANK BETWEEN LFST AND LLST.
C CHECK THAT ALL BLANKS PRESENT ARE CONTIGUOUS (2 FIELDS ONLY).
C
N=1
DO 400 I=LFST,LLST
IF (ISTR(I).NE.32)GO TO (350,500).N
N=2
L3=I+1
KBLK=KBLK+1
IF (KBLK.EQ.NBLK)GO TO 600
GO TO 400
L2=I
400 CONTINUE
GO TO 998
C
C BLANKS NOT CONTIGUOUS. THERE ARE MORE THAN 2 FIELDS.
C
500 L2=0
L3=0

```

```

LFST=0
LLST=0
NCHR=0
NCHR1=0
NCHR2=0
IF(MODE.EQ.1)GO TO 110
C*****
C CODE FOR MODE 0 ONLY:
C
KNT=PEEK('Z'80)
DO 100 I=1,80
ISTR(I)=PEEK('Z'80'+I)
100 CONTINUE
GO TO 200
C
C END SPECIAL MODE 0 CODE.
C*****
C CODE FOR MODE 1 ONLY:
C
C MODE = 1 (STRING "ISTR" WAS PASSED FROM MAIN).
C
C VERIFYING LAST BYTE AS BLANK SHOWS NO WORD WAS TRUNCATED DURING
C CONSOLE INPUT. ALSO, IT ALLOWS US TO SHIFT WHOLE ARRAY TO THE RIGHT
C 1 SUBSCRIPT & PUT A BLANK IN ISTR(1). MAKING IT COMPATIBLE WITH MODE
C 0 (WHERE ISTR COMES FROM DMA AREA). WE THEN COMPUTE A "KNT", & WE
C CAN USE THE SAME ALGORITHM AS FOR MODE 0 (EXCEPT FOR PEEK SECTION).
C
110 IF(ISTR(80).NE.32)GO TO 999
J=81
DO 130 I=1,79
J=J-1
ISTR(J)=ISTR(J-1)
130 CONTINUE
ISTR(1)=32
KNT=81
DO 150 I=1,80
KNT=KNT-1
IF(ISTR(KNT).NE.32)GO TO 200
150 CONTINUE
C
C END SPECIAL MODE 1 CODE.
C*****
C CODE COMMON TO BOTH MODES:
C
C THE COUNT (KNT) MUST BE GREATER THAN 1 AND LESS THAN 81 FOR ANY VALID
C STRING. LEADING BLANKS AND TRAILING BLANKS IN ISTR WILL BE IGNORED.
C
200 IF(KNT.LT.0)GO TO 998
IF(KNT.GE.1.AND.ISTR(1).NE.32)GO TO 998
IF(KNT.LT.2)RETURN
IF(KNT.GT.80)GO TO 999
N=1
DO 300 I=2,KNT
IF(ISTR(I).NE.32)GO TO (250,260),N
GO TO 300
C
C STATEMENT 250 CAN BE EXECUTED ONLY ONCE, SINCE N WILL BE SET TO 2.
C
250 LFST=I
N=2

```

```

GO TO 999
C
C THERE ARE 2 FIELDS.
C
600 L1=LFST
L4=LLST
NCHR1=L2-L1+1
NCHR2=L4-L3+1
RETURN
C
C THERE IS 1 FIELD ONLY.
C
990 L1=LFST
L2=LLST
NCHR1=NCHR
RETURN
C
C SYSTEM FAILURE OR BUG.
C
998 STOPANLSTR
C
C ERROR RETURN.
C
999 IERSTR=1
RETURN
END

```

Listing 2

```

SUBROUTINE FPARSE(NMRAW,NAMF,IDRV,IER)
C
C BY HANK LOTZ; MAY 10, 1984
C
C SUBROUTINE TO PARSE A DISK FILE NAME, PUTTING IT INTO THE FORMAT
C SUITABLE FOR FORTRAN FILE ACCESS, AND SUPPLYING A DRIVE PARAMETER.
C
C*****
C ONLY NMRAW (THE UNPROCESSED FILE NAME) IS PASSED FROM MAIN PROGRAM.
C ALL OTHER ARGUMENTS ARE RETURNED FROM THIS SUBROUTINE.
C*****
C
C FPARSE HANDLES NMRAW (FROM MAIN) BASED ON THESE ITEMS:
C
1) REJECTS CHARACTERS "?" " " " " " " AND "DEL"
2) ALLOWS NO MORE THAN 1 COLON, & NO MORE THAN 1 PERIOD.
3) ONLY A SINGLE LETTER MAY PRECEDE A COLON.
4) MORE THAN 8 NAME CHARS BEFORE "." SETS ERROR FLAG, AS
DOES ANY NONBLANK BEYOND THE 3RD POSITION AFTER "."
C
NOTE 1 -- NMRAW IS ASSUMED BY FPARSE TO BE LEFT-JUSTIFIED. (IF
IT IS NOT LEFT-JUSTIFIED, LEADING BLANKS WILL BE
TAKEN AS PART OF THE FILE NAME.)
NOTE 2 -- NMRAW CAN EASILY BE LEFT-JUSTIFIED WHEN PROCESSED BY
SUBROUTINE ANLSTR BEFORE THE FPARSE CALL.
NOTE 3 -- ANLSTR REJECTS EMBEDDED BLANKS WITHIN A FIELD; FPARSE
WOULD TREAT EMBEDDED BLANKS AS OTHER CHARS.
C
C

```


C PARTIAL LIST AND DESCRIPTION OF VARIABLES:

```
C C-----  
C ICOLON --- NUMBER OF COLONS IN NMRW STRING  
C IDRV --- DISK DRIVE PARAMETER TO BE RETURNED TO MAIN, 0=DEFAULT DRV,  
C 1=DRV A., 2=DRV B., ETC. MAX. DRIVE NUMBER IS 16. (P.).  
C IER --- ERROR-RETURN FLAG; IER = 1 SIGNALS ERROR IN INPUT STRING.  
C IPER --- NUMBER OF PERIODS IN NMRW STRING.  
C IPSUB --- SUBSCRIPT OF PERIOD IN NMRW; (ZERO IF NO PERIOD).  
C ISTRT --- THE SUBSCRIPT OF NMRW WHERE THE FILE NAME PROPER BEGINS.  
C ITRAIL --- THAT SUBSCRIPT OF NMRW WHICH IMMEDIATELY FOLLOWS THE 8TH  
C PLACE IN THE FILE NAME PROPER (WILL BE EITHER 9 OR 11).  
C NAMF --- THE FORMATTED FILE NAME TO BE RETURNED TO MAIN.  
C NMRW --- THIS RAW FILE NAME ARRAY AS PASSED FROM MAIN MUST BE  
C EXACTLY 14 BYTES WIDE. PROGRAMMER MUST CHECK THIS IN MAIN.  
C FOR NAMES SHORTER THAN 14 CHARS, NMRW MUST BE PADDED WITH  
C BLANKS TO RIGHT, IN MAIN.  
C NOTE: IN THE CURRENT VERSION, THE NMRW PASSED FROM MAIN  
C GETS CHANGED TO UPPER CASE BY THE SUBROUTINE.  
C*****  
C INTEGER*1 I,ICOLON,IDRV,IBR,IPER,IPSUB,ISTRT,ITRAIL,J,L,NAMF(11)  
C INTEGER*1 NMRW(14)  
C  
C INITIALIZATIONS:  
C DO 110 I=1,11  
C NAMF(I)=32  
C CONTINUE  
110 ICOLON=0  
C IER=0  
C IDRV=0  
C IPER=0  
C IPSUB=0  
C ISTRT=1  
C ITRAIL=9  
C  
C FIRST, CONVERT TO UPPERCASE CHARACTERS.  
C DO 150 I=1,14  
C IF(NMRW(I).GE.97.AND.NMRW(I).LE.122)NMRW(I)=NMRW(I)-32  
150 CONTINUE  
C  
C NOW LOOK FOR ILLEGAL CHARACTERS.  
C DO 160 I=1,14  
C IF(NMRW(I).EQ.42)GO TO 900  
C IF(NMRW(I).EQ.59)GO TO 900  
C IF(NMRW(I).EQ.63)GO TO 900  
C IF(NMRW(I).EQ.127)GO TO 900  
C IF(NMRW(I).LT.32)GO TO 900  
C IF(NMRW(I).EQ.58)ICOLON=ICOLON+1  
C IF(NMRW(I).EQ.46)IPER=IPER+1  
C IF(NMRW(I).EQ.46)IPSUB=I  
C CONTINUE  
160 IF(ICOLON.GT.1)GO TO 900  
C IF(IPER.GT.1)GO TO 900  
C  
C IF WE GET HERE ALL BYTES ARE LEGAL & MAX OF 1 COLON, MAX OF 1 PERIOD.  
C  
C IF(ICOLON.NE.1)GO TO 200  
C  
C COLON EXISTS: USER HAS SPECIFIED DRIVE. FILE NAME STARTS AT NMRW(3).  
C TEST FOR PROPER SYNTAX, THEN SET DRIVE (IDRV):
```

```
IF(NMRW(2).NE.58)GO TO 900  
IF(NMRW(1).LT.65.OR.NMRW(1).GT.80)GO TO 900  
IDRV=NMRW(1)-64  
ISTRT=3  
ITRAIL=11  
C  
C WE NOW ENTER 200 WITH IDRV, ITRAIL, AND ISTRT CORRECTLY SET.  
C  
C 200 IF(IPER.EQ.1)GO TO 300  
C  
C NO PERIOD EXISTS, THEREFORE ONLY BLANKS MAY OCCUPY NMRW(ITRAIL) THRU  
C NMRW(14). (WE USE ONLY THE LEFTMOST 8 PLACES FOR A VALID NAME.)  
C  
C DO 230 I=ITRAIL,14  
C IF(NMRW(I).NE.32)GO TO 900  
C CONTINUE  
230  
C  
C REQUIREMENT FOR "ONLY BLANKS" (STATED JUST ABOVE) IS MET.  
C  
C ISTRT=ISTRT-1  
C DO 250 I=1,8  
C J=ISTRT+I  
C NAMF(I)=NMRW(J)  
C CONTINUE  
C GO TO 500  
250  
C  
C A PERIOD EXISTS (AT IPSUB).  
C CHECK FOR NO MORE THAN 8 PLACES IN FILE NAME PROPER:  
C  
C 300 IF(IPSUB.GT.ITRAIL)GO TO 900  
C J=IPSUB+4  
C  
C IF J>14, THE FOLLOWING TEST IS NOT NEEDED (MAX POSSIBLE J IS 15).  
C TEST GIVES ERROR IF NONBLANK OCCURS AFTER 3RD FILE-TYPE POSITION.  
C  
C IF(J.GT.14)GO TO 350  
C DO 330 I=J,14  
C IF(NMRW(I).NE.32)GO TO 900  
C CONTINUE  
330  
C  
C (NOTE: IPSUB ISN'T LESS THAN ISTRT, & IPSUB=ISTRT IS A SPECIAL CASE.)  
C  
C L=0  
C IF(IPSUB.EQ.ISTRT)GO TO 450  
C J=IPSUB-1  
C DO 400 I=ISTRT,J  
C L=L+1  
C NAMF(L)=NMRW(I)  
C CONTINUE  
400  
C 450 NAMF(9)=NMRW(IPSUB+1)  
C NAMF(10)=NMRW(IPSUB+2)  
C NAMF(11)=NMRW(IPSUB+3)  
C DO 500 I=1,11  
C IF(NAMF(I).NE.32)RETURN  
C CONTINUE  
500  
C IER=1  
C RETURN  
C END
```



The Sometimes Useful H-89 Front Panel Monitor

Charles E. Horn, PE
Horn Engineering Associates
1714 Patricia Lane
Garland, TX 75042

If you built your H-89 from a kit, you might barely remember running the test for the H-17 disk rotation speed, which had to be set to permit booting your system for the first time. You might also have run the memory and computing tests as described in the operation manual. Very likely, you ran these tests once and then never ran them again. They were run from the built-in front panel monitor that, even though primitive, can be very useful. It will permit entry of programs into memory and provides a means to jump to the program for execution. The computing test does just that, using the boot ROM console output routine to print the output data. The memory and speed tests simply jump to boot ROM routines for execution.

For an exercise in use of the monitor, suppose that you have a non-Heath boot ROM for which source code is not available. You might like to disassemble and study it, using DDT under CP/M, but you can not just dump or list this code because RAM was swapped into the ROM space when CP/M was booted. (This does not happen under HDOS). Our first solution to this problem was to use the monitor to manually enter a program that would move the ROM code into a memory area that would not be affected when CP/M was booted. (A cold boot only affects the memory locations that CP/M uses to install itself). Then, after booting CP/M, we used DDT to disassemble the relocated ROM code and to direct it to the printer, using the CTRL-P and -L(saddr),(eaddr) functions. In a 64K system, our choice of memory locations was A000H to A7FFH, representing the 2K ROM code from 0000H to 07FFH. We then realized that an easier way to install the move code would be to write a program under CP/M to move the ROM code, assemble it, then use DDT to install it at a safe location in memory. The system could then be reset and the monitor could simply be used to jump to our installed program for execution.

For those who would like to try this procedure, the move program is listed below. With this code, the system will appear to crash but the ROM code will have been moved to A000H. CP/M can then be booted and DDT can be called in to read the relocated code. Note the ORG at 9F80H, just below the space allocated for the relocated ROM code.

```
;
; MOVEROM.ASM
; A Program to move 2K of ROM code from 0000H to A000H.
;
ORG 9F80H ;where DDT will install
          ;..MOVEROM.HEX
LXI B,0800H ;no of bytes to move (2K)
LXI D,0000H ;move from this address
LXI H,0A000H ;..to this address
LOOP: LDAX D ;get a byte
      MOV M,A ;move it
      INX H ;bump destination pointer
      INX D ;bump source pointer
      DCX B ;decrement byte count
```

```
MOV A,H ;get high byte of destination
          address
CPI 0A8H ;next address after end (A800H)?
JNZ LOOP ;continue if not
HALT: JMP HALT ;hang here 'till reset
;
; NOTE: A HLT instruction will not execute from the monitor;
; hence, we just use the looping HALT.
;
END
;
```

Your assembler will create the MOVEROM.HEX file. Do not LOAD this file. The COM file is not required for this procedure. Now, under DDT, issue the following commands:

```
-IMOVEROM.HEX <RET>
-R <RET>
-CTRL-C
```

Then, reset the system (SHIFT-RESET). At the H: prompt, issue the command:

```
H: G(o) 237200 <RET>
```

Note that the monitor will accept addresses only in split-octal; single bytes in octal. The above address is equivalent to 9F80H, where DDT loaded our move program. This is the point at which the system will appear to lock up. It has executed the move program, almost instantly, and is hanging in the HALT loop. Reset the system, then boot CP/M. Call in DDT and issue the command:

```
-LA000 <RET>
```

The first 11 lines of the moved code should appear. The first instruction should be a JMP to some address within the ROM range of 0000H to 07FFH. The entire disassembled ROM code can now be viewed, using the command:

```
-LA000,A7FF <RET>
```

This disassembled code can be sent to the LST: device (printer) by preceding the above command with a CTRL-P. If a disk file of this code is desired, the code can be moved to the bottom of the TPA with the commands:

```
-MA000,A7FF,0100 <RET>
-CTRL-C.
```

Then, from the A> prompt, issue the command:

```
SAVE B BOOTROM.COM <RET>.
```

This file can be called back into DDT at any time and the code can be moved back to A000, if desired. (We picked this address to make address translation easy.) It can even be disassembled with a genuine disassembler, such as the one on HUG Utility Disk P/N 885-1212, which will create ASM source code with labels. However, the ORG of this code will be at 0100H; not the true ROM origin at 0000H.

If you would prefer to enter the move code directly from the monitor, here it is. This code is equivalent to the assembly language code shown above. The nnn values are the old values in memory that are displayed by the monitor. They are replaced by our program.

```

H: S(ubstitut) 237200 <RET> ;start at 9F80H
237200 nnn 001 <SPACE> ;LXI B,0800H
237201 nnn 000 <SPACE>
237202 nnn 010 <SPACE>
237203 nnn 021 <SPACE> ;LXI D,0000H
237204 nnn 000 <SPACE>
237205 nnn 000 <SPACE>
237206 nnn 041 <SPACE> ;LXI H,A000H
237207 nnn 000 <SPACE>
237210 nnn 240 <SPACE>
237211 nnn 032 <SPACE> ;LDAX D
237212 nnn 167 <SPACE> ;MOV M,A
237213 nnn 043 <SPACE> ;INX H
237214 nnn 023 <SPACE> ;INX D
237215 nnn 013 <SPACE> ;DCX B
237216 nnn 174 <SPACE> ;MOV A,H
237217 nnn 376 <SPACE> ;CPI DABH
237220 nnn 250 <SPACE>
237221 nnn 302 <SPACE> ;JNZ LOOP
237222 nnn 211 <SPACE>
237223 nnn 237 <SPACE>
237224 nnn 303 <SPACE> ;JMP HALT
237225 nnn 224 <SPACE>
237226 nnn 237 <RET>
H: G(o) 237200 <RET>
<SHIFT-RESET>
H: B(oot)

```

finis



Updated local HUG club information

New Club: **WCHUG (West Coast HUG)**
 c/o Van Christopher
 10784 Magnolia Avenue #2H
 Santee, CA 92071
 (619) 449-7298
 Contact Person: Van Christopher
 Monthly Newsletter - 12 issues for \$15.00
 Software Library and ZDS Bulletin Board
 Notebook

The **NWFHUG (Northwest Florida HUG)** has a new address: 38 Berwick Circle, Shalimar, FL 32579. New contact person and president is Allan White. He may be reached at (904) 651-2970.

New address for the **Lehigh Valley HUG** is c/o Bob Kendi, Computing Center, Mart Library 8-B, Lehigh University, Bethlehem, PA 18015.

A-SQR-HUG, Ann Arbor, Michigan will now meet the last Monday of each month.

CONNHUG (Connecticut HUG) has a new mailing address as follows: c/o Bob Conlon, 71 Pardee, Bristol, CT 06010. They meet the first Wednesday each month at 7:00 pm at the Greater Hartford Chapter of the American Red Cross.

HUG of Central Florida Computer Society now has a membership of 20. They meet every fourth Wednesday at Kane Furniture at the intersection of 436 and Red Bug Road. Phone number correction: (305) 644-6847.

Welcome to

It's Great!

DOODLER

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 208
 Safety Harbor, FL 33572

Specification Sheet available on Request

RECIPES

TAXES

ADDRESS BOOK

ORDERS

CUSTOMER LISTS

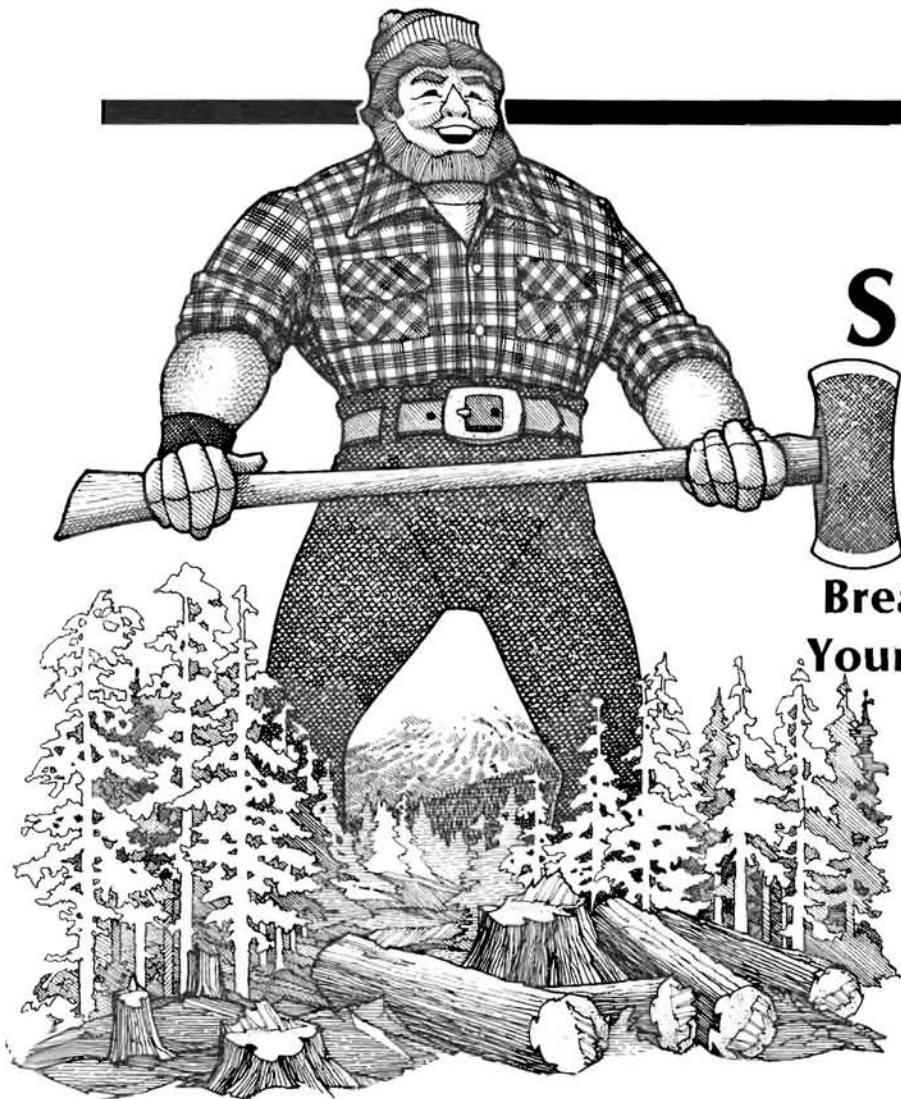
THE FINAL FILE

ADD
 SORT
 BROWSE
 EDIT
 SEARCH
 PRINT REPORTS

ONE PROGRAM FOR ALL YOUR FILE NEEDS! DESIGN CUSTOM DATA ENTRY SCREENS, PRINTER FORMATS, LABELS, MAIL LISTS, SEARCH AND SORT. FUNCTION KEYS USED FOR ILLUSTRATED SINGLE STROKE COMMANDS. SIMPLE TO USE WITH SOPHISTICATED RESULTS. REQUIRES ZDOS, 128K RAM, RECOMMEND DUAL DISC DRIVES FOR BEST USE.

\$59.95

SOLUS CO. Box 134R, Kula, HI 96790



Spooldisk 89

**Break The RAM Barrier & Turn
Your H-89 Into A 448K Monster!**

*Peter Ruber
P.O. Box 502
Oakdale, NY 11769*

The ads are certainly seductive! Sleek color photos of sexy, streamlined Z-100's and Z-150's sporting hi-res graphics, lots of RAM, turbo-charged microprocessors executing your commands at lightning speeds! And, of course, all the pricey software you could ever want — but seldom need.

For many of us, computers have become Electronic Mistresses in our lives; and computer manufacturers, in an effort to survive the industry shake-out, are trying to condition us like the auto-makers: trade up before you become an embarrassment to your computer-club colleagues.

I still think Heath/Zenith missed the boat with the H-89. Except for the Apple, which has undergone several organ transplants and face-lifts since 1977, the H-89 has survived Heath's meager efforts to up-grade the machine's versatility. Fortunately, for all of us, a kind of peripheral industry has emerged over the last few years that realized the awesome potential of the H-89's open architecture, and has created a number of enhancements that are both useful and challenging.

I recently acquired two pieces of hardware that will blow your mind (as one of my teenage sons would paraphrase it) without going into cardiac arrest. Because of the complexity and features of these products, I will review them in two parts and give you some hints on how to install and use them. If I seem overly enthusiastic, it is because these products are first-rate; and because the support I've received so far has been quick and helpful.

In combination, these products will allow you to add 384K of RAM for a total of 448K to your 64K H-89. They are the new SigmaSoft and Systems' "Interactive Graphics Controller," which sports high-resolution interlace mode graphics with 640 X 500 line resolution with 256K RAM workspace, plus parallel interfacing for two printers and two Atari-compatible joystick or trackball controllers — and, yes, even a light-pen.

The other new item is a novel printer-buffer device from FBE Research Company, Inc. called SPOOLDISK 89.

Since my computer time is usually limited to a few hours each night, I often lose most of my session when I have to print out several long documents. Even printing out the Assembly Language listings or Source Code files from a HUG program disk can take an hour or more. Naturally, the computer is tied up spitting out data to the printer that it resembles a blinking neon light until the job is done. I used Digital Research's DESPOOL program for a week or so, but the effort it required to digest the cryptic instructions to a reasonably simplistic level took more effort than it was worth. While I was afforded the privilege of placing a document into an unused area of RAM, output to the printer was halted each time I pecked at the keyboard on another project. It became a Print-Or-Not-To-Print game of Cat-and-Mouse, and it was simpler to dump DESPOOL into my Trivial Nonsense file.

I almost bought a printer buffer unit when I opted for SPOOLDISK 89. The advantages were obvious: it came on a single card I could mount inside the H-89, and the \$295.00 price tag for 128K of

RAM was a decidedly superior bargain to a printer buffer that was the same price with only 64k.

When SPOOLDISK 89 arrived less than three weeks after my order, I discovered that I received a lot more power and versatility than I had been expecting.

The FBE SPOOLDISK 89 is a microprocessor-based 128K byte Electronic Disk Emulator, a Printer Interface and Printer Spooler rolled into one. It uses an Intel 8031 stand-alone high-performance single chip microprocessor that operates with an 11 Mhz clock. Most instructions are executed in just over one microsecond. As a Disk Emulator, it has all the characteristics of a floppy disk drive, except that it has no moving parts and track and sector accessing is instantaneous. The disk is organized as 16 tracks (0-15) of 32 sectors (0-31) of 256K bytes for a total capacity of 131,072 bytes of RAM, plus 2K bytes of Program Memory (PROM). As such, you can INIT and SYSGEN it under HDOS (as well as CP/M) just like a disk. PIP your favorite text editor onto SPOOLDISK 89 and enjoy more than 120K bytes of work space — nearly four times the overhead you would have if you used the H-89's RAM.

But the real fun begins when you've loaded SPOOLDISK 89 with your work-in-progress or any program or document that requires hard copy. You type a one-letter command, followed by the filename, and within seconds your printer starts grinding away. Then you RESET the computer, BOOT up any new system disk containing a spreadsheet, data base or another word processing program, and continue your work. When you reset the H-89, the 8031 CPU on SPOOLDISK 89 takes over the printer I/O responsibilities. It's like having a second computer.

When SPOOLDISK 89 has completed its chores, you can PIP or COPY the files you've just printed to any logical disk drive on your system. To reload SPOOLDISK 89 with additional programs or documents, you enter the COMMAND MENU and instruct the computer to initialize and clear the RAM. As a safety feature, a screen prompt instructs you that the initialization process will destroy your files, and you are requested to answer YES or No. If you answer affirmatively, the initialization process is over by the time your finger releases the RETURN key.

As a printer interface, SPOOLDISK 89 is available in either Parallel or Serial configuration. If you choose the Parallel version (which I did), you can easily modify the board for Serial communications in less than one-half hour by installing the FBE Research SA1 RS232 Adapter, which provides for RTS/DTR Handshaking and 300-1200-9600 Baud for a modest \$15.00.

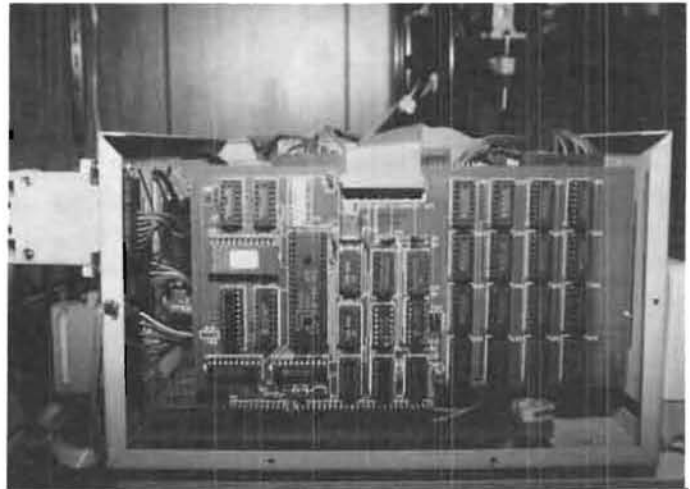
Installing Spooldisk 89

This is almost painless; and possibly disappointing to those who like to swing on their wire jungles occasionally. You have a choice of plugging the SPOOLDISK 89 card into any of the three right-hand bus expansion slots. It is fully compatible with the H/Z-17 and H/Z-37 Disk Controllers, as well as the H-88-3 Triple Serial Interface Card (or equivalent), as it uses I/O Ports 327Q and 337Q which are unused registers of the 8250 ACE's on the H-88-3.

If you are using only the H/Z-17 hard-sectored disk Controller Card and the H-88-3 Serial Card, then plug SPOOLDISK 89 into slot P504, and experience a lot more versatility. Conversely, you can plug it into P506 if you only use the H/Z-37 soft-sectored Controller. But if you're a die-hard who insists on using both types of Disk Controller Cards, you can invest in SLOT 4.

SLOT 4 is a neat little bus expander from FBE that piggy-backs into the right-hand bus and gives you a fourth slot — P517/P518, which shares its I/O lines with P504. SLOT 4 is available for \$47.50 — or at a give-away price of \$25.00, if ordered at the same time as SPOOLDISK 89. It comes with its own separate 16-page Technical Manual, and can be ordered in "straight-through" or "switched" configuration.

Switching I/O usage between P517 and P504 is accomplished under software control (Source Code drivers for HDOS and CP/M are included in the Technical Manual, including instructions on how to incorporate them into your Operating System), or by means of an externally mounted Toggle Switch that can be wired to SLOT 4. I, personally, prefer the Toggle Switch approach. It's quicker and infinitely more fun to drill holes in computer cases.



The FBE SPOOLDISK 89 shown installed in the Microflash M-89 Expansion Box. It is equally at home in the H/Z-89.

As a precautionary measure, FBE urges you not to install either SPOOLDISK 89 or SLOT 4 without first making certain that you have the power up-grade in your H-89. This is standard in the newer H-89A's and comes with the H/Z-37 Controller package. On the remote chance that your unit doesn't have the power upgrade, full instructions and a parts list are included for this purpose.

Bear in mind, however, that if you choose the Parallel version of SPOOLDISK 89, you must remove any Serial cable connected to your printer.

Operating Spooldisk 89

SPOOLDISK 89 comes with an impressive software support package for both HDOS and version 2.2.03 (or 2.2.04) of the Heath/Zenith implementation of CP/M. In reference to the latter, you receive two disks: the first contains all the necessary utilities to modify your BIOS so that CP/M will recognize SPOOLDISK 89 as a logical device on your system, plus the .COM files that are the Operating System and Device Drivers for the card. The Technical Manual guides you clearly on the necessary steps on how to incorporate this enhanced BIOS on your system disk. For those less familiar with CP/M, it is suggested that you keep your Heath/Zenith CP/M manual close at hand.

For those using CP/M versions where the BIOS cannot be replaced (as with the Magnolia Microsystems implementation), or with systems where the BIOS has been replaced (e.g. the

Livingston Logic Labs BIOS-80), a special utility called BNG80.COM will allow you to use SPOOLDISK 89 with your system. Software hackers will not be disappointed either. ASM files have been included to permit the alteration of your BIOS through the MAKEBIOS procedure so that it is specifically tailored to your system. This second CP/M support disk from FBE Research also contains BIOS-2/4.SYS, which is the standard SPOOLDISK 89 BIOS modified to allow H/Z-17 disk operations with CPU speeds of 2 or 4 Mhz. This BIOS checks the CPU speed each time an H/Z-17 drive is accessed and adjusts the timing constants according to the CPU speed.

Finally, if you have need to duplicate a lot of hard-sectored disks (such as in a small software house), a DUPL will allow you to transfer an entire H/Z-17 disk to SPOOLDISK 89 in about 18 seconds, and produce and verify a duplicated disk in about 36 seconds.

All HDOS support files are contained on one disk. There are 14 files, including ASM and REL files if you wish to optimize SPOOLDISK 89 for any special purpose.

To get started under HDOS, it is suggested that you create a System Volume disk to which you transfer the Q.ABS, SETP.ABS, LP.DVD, SS.DVD and PROLOGUE.SYS files. While the basic SPOOLDISK 89 software support package is optimized for the H/Z-17 controller, you will have no problems using it with the H/Z-37, which I use for most of my work, except when I have to PIP hard-sectored software to a soft-sectored format.

When you set up your SPOOLDISK 89 SYSTEM VOLUME with these files, make certain that you have no other active device drivers on it except for SY.DVD and DK.DVD. The first order of business is to activate the printer device driver (LP.DVD) by invoking the SETP program. Respond <YES> when you are asked if you want to enable the Queued Printing Device Driver. Then set the printer parameters for form and page length, and your left and right paper margins. Type <YES> in response to recording these settings on your disk.

You have eight driver options to work with (LP0-LP7). Each can contain a different set of printer initialization codes for normal, compressed, enhanced, expanded, emboldened and italic printing modes. Each LP driver option can be changed at any time by using the SETP program. RESET the computer after setting or changing any printer parameters. When enabled, the printer drivers simulate tabs and formfeeds with spaces and linefeeds, and control the parameters you have created. If you do not enable the printer drivers, all copy is passed straight through without regard to format, margins or skip over perforations.

You are now ready to go to work. When you BOOT up with your SPOOLDISK 89 System Volume, all device drivers load automatically and the Disk Emulator (SS:) is initialized. Under CP/M, initialization consists of writing a blank directory. And under HDOS, empty-disk versions of the DIRECT.SYS, RGT.SYS and GRT.SYS files are created. A fresh label sector is also written.

If you have a document requiring hard copy, PIP that file to device SS:, type <Q> to enter the COMMAND MODE, and select your options. For Queued printing, type <Q>, followed by the "filename" and SPOOLDISK 89 automatically initiates printing. If you have a series of files larger than the 127K workspace available under the Queued printing method, you have the option of invoking the FIFO mode (FIRST IN FIRST OUT). Here your available workspace is reduced to 64K, but SPOOLDISK 89 now acts as a buffer through which your files are passed.

As buffer space becomes available — based on the speed of your printer — the files to be printed are continuously fed to the buffer area until exhausted.

When you're ready to print out additional files, type <Q> to return to the COMMAND MODE; then type <I> to initialize the SPOOLDISK 89 surface. This deletes your files and clears the work space.

Optionally, you may elect (under CP/M) to SYSGEN SS: and use its RAM space to assemble ASM or ACM files, compile BASIC programs with lightning speed, or use SPOOLDISK 89 to SYSGEN disks for you. The SYSGEN capability is not available under HDOS at this time.

There is virtually no difference between the HDOS and CP/M operations, except that in the COMMAND MODE you have option <P> to enable your printer driver and to set up the printer parameters. Under HDOS, this is handled by the SETP program.

Because the CP/M BIOS supplied by FBE incorporates the SPOOLDISK 89 as a fully operational disk drive, your RAM capacity is reduced to about 118K. There is one reserved track of 8K bytes for the Operating System, and the directory, which may contain up to 64 entries, will consume another 2K bytes.

Under HDOS, SPOOLDISK 89 becomes just another device to be reckoned with, and it is simply referred to as SS:. Work area available under HDOS conventions is 488 sectors.

But with CP/M, it is always the last drive of the system counting both real and imaginary drives. If your system contains the H/Z-17 Controller, SPOOLDISK 89 will be Drive D:, since there are three possible real and/or imaginary H/Z-17 drives (A:, B: and C:).

From the COMMAND MENU under both Operating Systems you can interrupt, resume or cancel any printing in progress by a one-letter command; and you can read the directory with the CAT, DIR and STAT commands. It is important to remember that if you have used SPOOLDISK 89 as your work area for any original copy or programming, you should PIP those files to your system volume or working disk before you create a hard copy listing. While I mentioned earlier that the SPOOLDISK 89 memory cannot be wiped out even by Resetting the computer and Booting up a new disk, I have accidentally shut down the H-89 in a momentary blunder and destroyed a file or two.

In Serial configuration, SPOOLDISK 89 is available for a variety of popular printers, namely all models of the Epson, plus specific models from NEC, Prowriter, C. Itoh, Okidata and Centronics. Since new printer-specific drivers are added periodically, it is advisable to contact FBE for additional information. An LP.ASM file is included for any custom printer driver applications. The limited Centronics compatible Parallel configuration should handle almost any dot-matrix or letter quality printer without problems. Ribbon connector pinouts are supplied with the manual.

To illustrate the care that has gone into this product, there is a two-post Terminal Block that enables you to connect an LED to act as a monitor and activity indicator when the board is in use. Unlike the purring noise a disk drive emits when reading or writing from a disk, the silence from SPOOLDISK 89 can be disconcerting, especially when it's compiling a program or assembling a source file. The LED can be mounted on the keyboard cover,

above and to the right of the numeric keypad; or even on the H-89's face-plate, just below the internal drive. The positive and negative leads from the LED can be connected to a pair of wire leads terminating in a two-hole socket which you connect to TB4. This allows you a quick disconnect if you ever have to remove SPOOLDISK 89 from the CPU board. There is also a second 2-pin header marked TP1, which allows you to reset the board in the event you get an error message on BOOTing up the computer that advises you that SPOOLDISK 89 is not present, or if you wish to clear the contents from the system prompt. A momentary contact switch mounted on the front panel (or elsewhere) with a two-hole connector on TB1 is all you need.

Documentation

The 35-page Technical Manual you receive with SPOOLDISK 89 is printed offset from a letter quality printer, which includes one page of parts identification and a four-page schematic. This I would have preferred to see as one folded sheet, but the legends are large and professional and easy to follow. The pages of the manual are punched for a three-hole binder.

The copy is terse and to the point, and most features are covered with reasonable clarity. More space is devoted to setting up under CP/M than HDOS because of the complexity of installing a new BIOS. I would have liked to see a better rearrangement of some of the paragraphs so that the user is led through a logical progression of steps from installation to usage. All of the details that you need to know are there and a couple of careful readings of the manual will get you going. Nitpicking aside, a couple of minor questions that I had were promptly answered by Dave Brockman by return mail.

The SPOOLDISK 89 board is a high-caliber product, with all the frills: silk-screening and solder-masking, and all IC's are in sockets for easy servicing. It comes in a heat-sealed plastic sleeve to protect it during shipping, as do the three software disks. The product is fully tested and burned in prior to shipping and comes with the full 90-day repair or replacement warranty. More importantly, FBE has established a reputation over the last few years for providing prompt delivery and full support for all of its products.

About the only feature missing is multiple copy printing of a single file. Some word processing has this built-in feature, while text editors generally do not. However, it is a simple matter of typing the "Q" command, followed by the <filename> of the document you wish to print, and repeating this sequence for as many copies as are needed.

Due to recent price reductions in chips, the price of SPOOLDISK 89 has been lowered to \$295.00. The printer cable is an additional \$24.00, and you must specify Serial or Parallel version and the type of printer you are using. You will feel comfortable using this board in less than a hour; and within a few days you'll be using it as automatically as you do your disk drive or printer, and wonder how you ever got along without it. It has become one of the most useful additions to my arsenal of H-89 peripherals.

For more information, write to:

FBE Research Company, Inc.
P.O. Box 68234
Seattle, WA 98168

(206) 246-9815 (6pm-10pm PST)



"C/80... the best software buy in America!" —MICROSYSTEMS

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 compatibility
- 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"

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.

WATCHWORD

The word processor and full screen editor for the Z100 and ZDOS.

FRIENDLY - FAST - FLEXIBLE

- See subscripts, superscripts, underlining and boldface directly on the screen
- Create your own fonts and special characters
- Remap any key and configure to your printer
- Written in native 8088 assembly language for fast response

Other features: centering, formatting, microjustification, arbitrary line length, automatic horizontal scrolling, split screen, macros, and color.

Demo available at all Heathkit Electronics Centers.

For additional details, send a SASE.

Demo Disk: \$3.00
WatchWord with
Manual: \$100.00

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

ATTN: Steve Robbins

MORE BENEFITS. LESS COST.



Integrated Text and Graphics so you'll have the best features of a text terminal and a graphics terminal combined in one efficient package. It's the most versatile terminal available today.

Dual Plane Memory so you can combine the planes to create a four-level gray scale. And you can overlay the planes to compare one set of data or graphs with another.

Selectable Resolution so you can pick the right resolution for each application. Choose 1024 or 512 horizontal and 500 or 250 vertical or any combination of these.

128K dedicated display memory so you can work more quickly. Store up to 8 full-screen graphic images or 75 pages of text in the terminal.

Integrated graphics operations so you can work more efficiently. Pan, zoom, area move, arc drawing, vector erase, and many other graphics commands make complex images a snap.

3 complete set-up configurations in non-volatile memory, so you can concentrate on your work, not on memorizing set-up routines. Plain English set-up, of course.

DEC/Tektronix emulation so the GP-29 is compatible with your system's software. The GP-29 is compatible with the VT-100/VT-220, Tektronix 4010/4014, and other terminals.

Variable text formats so you can select the appropriate display: 80 or 132 columns by 24 or 49 lines.

Multiple character sets so you've got many convenient choices: VT100, VT220 and math/Greek, and an optional 128-character soft font.

All keys programmable so you're not artificially limited: customize to meet any application, then store your customized keyboard in non-volatile memory.

Much more including transparent mode, dual bi-directional ports, graphics/text printer support, optional graphics input mouse, and other features to help you work faster and more efficiently.

And with all these benefits, the complete GP-29 terminal costs only \$1,695.

It's the best cost/benefit ratio in the business.

If you currently own a Zenith Z29 terminal, you can upgrade it to full GP-29 capability for only \$995.

GRAPHICS-PLUS GP-29

DEALER INQUIRIES INVITED



Northwest Digital Systems
P.O. Box 15288
Seattle, WA 98115
(206) 524-0014

SPREADSHEET Corner

PERSONAL FINANCIAL ACCOUNTS
 ACCOUNTS—CHECKS DEPOSITS BALANCE
 THIS PERIOD *****
 JUNE *CARRYOVER: **CHECKS* \$245.77 4367.49 \$179.02 58.00 20.00
 1988 *FROM LAST: **CRA CARDS 58.00 58.00 58.00 58.00 58.00

 CHECK REGISTER

CEL: DATE DESCRIPTION ACCT. CODE CHECK DEPOSIT RUNNING BALANCE MORTGAGE PAYMENT

CEL:	DATE	DESCRIPTION	ACCT. CODE	CHECK	DEPOSIT	RUNNING BALANCE	MORTGAGE PAYMENT
				585.42		436.60	58.00
					5167.49	3404.49	58.00
				5189.54		5114.95	58.00
				545.77		5169.22	58.00
				5245.98		1574.24	5245.98
					5780.00	4701.74	58.00
				535.76		5267.98	58.00
				545.76		5223.60	58.00

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



This month I would like to give some practical experience in the use of most of the LOTUS 1-2-3 Data Management Functions. I have chosen a rather easy application that I hope you readers will be able to adapt to a useful program for yourselves. I am going to call it "BANKING RECORDS SYSTEM" (could be called a ledger). I have set it up for a small business, but it could be easily modified for a personal application for your yearly income tax records. I have tried to make it versatile and applicable to your needs. As always, I will write the article in a tutorial format.

This BANKING RECORDS SYSTEM will consist of three modules (parts) as follows:

1. Data Input Module — contains the raw input information (data).
2. Data Query Module — provides for extracts and analyses of input data.
3. Data Table Module — provides for a summary of the data as needed.

You will find that I am using names to describe the modules, functions and formulas that will closely match the 1-2-3 terminology to help with the learning process of using the terms over and over to teach them. I am going to assume that you have been working with the previous "SPREADSHEET Corner" articles with 1-2-3 and thus you know how to move around and create a worksheet, as well as how to input data into the worksheet.

The Banking Records System that I am helping you create is an expanded check-book register for recording checks and deposits using the usual check number, date, name, and amount references. However, in addition I will provide for you to categorize entries by two different code references, plus an optional memo reference. These references will be used to extract, correlate and summarize the data entries. I know that this will not be clear to you at this point, but you will understand what I have in mind as we pro-

ceed. (I have to make it look a little difficult to impress you, don't I? Ha!)

First, I will help you create the Input Data Module. See Figure 1 for my "SPREADSHEET Preparation Form." Did you re-

member that YOU MUST prepare one for EACH of the three modules? Start the Data Input Module by entering the labels and the dividing lines as shown. I have put my suggested column widths at the top of Figure 1.

	A	B	C	D	E	F	G	H
	6	9	15	6	6	10	17	
1	***CORNER COMPUTER SUPPLIES—BANKING RECORDS SYSTEM***							
2								
3	DATA INPUT MODULE				SUM:		723.44	
4	*****							
5	trans#	date	name	code#1	code#2	amount	memo	
6	-----							
7	1001	8401.03	Steiner	wages	disks	-800.00	Expense-Payroll	
8	1002	8401.03	Phillips	wages	forms	-770.00	Expense-Payroll	
9	1003	8401.03	Williams	wages	paper	-550.00	Expense-Payroll	
10	1004	8401.03	Anderson	wages	ribbon	-780.00	Expense-Payroll	
11	6001	8401.05	Applesauce	whsl	forms	2250.00	Sales-Wholesale	
12	6002	8401.06	Morsecode	retl	disks	2334.28	Sales-Retail	
13	6003	8401.06	Morsecode	retl	forms	1228.96	Sales-Retail	
14	6004	8401.06	Morsecode	retl	ribbon	669.32	Sales-Retail	
15	1005	8401.09	Times	ganda	ribbon	-1000.00	Expense-Advertisel	
16	1006	8401.09	Times	ganda	paper	-800.00	Expense-Advertisel	
17	1007	8401.12	Federated	matls	disks	-770.00	Expense-Materials	
18	1008	8401.13	Federated	matls	forms	-550.00	Expense-Materials	
19	1009	8401.13	PCsupplies	matls	ribbon	-700.00	Expense-Materials	
20	1010	8401.16	PCsupplies	matls	paper	-600.00	Expense-Materials	
21	1011	8401.18	Steiner	wages	disks	-800.00	Expense-Payroll	
22	1012	8401.18	Phillips	wages	forms	-770.00	Expense-Payroll	
23	1013	8401.18	Williams	wages	paper	-550.00	Expense-Payroll	
24	1014	8401.18	Anderson	wages	ribbon	-780.00	Expense-Payroll	
25	1015	8401.20	Westcoast	matls	ribbon	-460.00	Expense-Materials	
26	6005	8401.23	Applesauce	whsl	disks	2360.00	Sales-Wholesale	
27	6006	8401.23	Atarian	whsl	ribbon	1185.88	Sales-Wholesale	
28	6007	8401.23	Atarian	whsl	paper	1375.00	Sales-Wholesale	
XX								
XX								
XX								
XX								
400	***WORKSHEET LIMIT FOR DATA & FORMULAS & FORMATTING***							

Figure 1

IMPORTANT! Because of the limited space for Figure 1, I am not able to show you how I have set a "flag" at the bottom of the module. When you enter your data starting at row 9 and continue down the module, it is necessary to have a method of displaying a "flag" to show that you have reached the bottom limit that was provided for the system. This will explain itself as we proceed. At this time, you probably do not know how many entries you will need, but you will want to calculate a running balance of all entries and be able to format the data. I looked at last year's bank records and found that my monthly average was 30 entries, so 12 months times the 30 equals 360. Therefore, to provide for some growth, I set a temporary limit for the worksheet at row 400. Another element to consider is the size of working memory (RAM) that you have for your system. Also, the larger the worksheet, the longer the program will take to recalculate. This is not much of a problem because we are talking about 20 or 30 seconds at most.

Therefore, I will go to (function key "F5" on Z-100, remember?) cell A400 and enter a label, such as the following:

```
***WORKSHEET LIMIT FOR DATA & FORMULAS & FORMATTING***
```

This label will be your "flag", if you reach it while entering data into the worksheet, that will tell you to add more space. This will require a NEED to reformat the worksheet and revise some formulas. I will discuss this more later. I hope that this will cause you to give thought before deciding on row 100, 500 or what!

The Data Input Module requires only one formula to total the entries. I put the label "SUM:" in cell E5 and the following formula in cell F5:

```
@SUM(F9..F400)
```

Since most of our Data Input Module has numeric data that should be expressed in dollars and cents format, I think this would be a good time to format the entire worksheet to express numbers with two decimal places. I am sure that you know the following command:

```
/Worksheet Global Format Fixed 2, and Return.
```

Cell F5 should now show 0.00. I want the 'trans#' column (column "A") to express its data as integers so I used the following command:

```
/Range Format Fixed 0, Return, A9..A400, and Return.
```

I need to use a few more formatting commands so that the other two modules will work with this Data Input Module. I would like you to try something different than what I have shown you previously. I will assign NAMES to various ranges in this module! There are two schools of thought as to whether it is easier to use "range names" or "range CellX..CellY." I will let you decide which you prefer after you try both methods for a period of time. Maybe at the completion of this assignment you can make the decision!

I will give the ENTIRE data input range the name "INPUT". That makes sense, I am sure you will agree. So, go to cell A7 and give the following command:

```
/Range Name Create INPUT, Return, A7..A400, and Return.
```

It is IMPORTANT to define the INPUT range as starting with row 7 rather than row 9 so that the labels, such as trans#, date, name, etc. are included as part of the input range so that the Data Query functions will work with this module. This will clear up when I

explain Data Query. Also, NOTE that I defined the INPUT range down to row 400! Thus, I hope that you can see that if your data entries would extend your worksheet beyond (below) row 400, that this data would NOT BE in the INPUT range! This is one of the reasons why the INPUT range and the formula in F5 would have to be revised to enlarge our worksheet size.

Next, I must give names to the columns under each label heading. You will see why later in this article. I hope that you will bear with me on this. Go to cell A7 and give the following command:

```
/Range Name Labels Down A7..G7, and Return.
```

This command assigns the cell A8 the name "trans#", cell B8 the name "date", cell C8 the name "name", etc. I will use these names in the Data Query Module. This completes the Data Input Module and it is ready for input data entries. I hope that you have been SAVING your work periodically as you were working! I do not believe that I should have to keep reminding you, but it sometimes helps all of us. I use the following command:

```
/File Save BNKRCD84, and Return.
```

I have prepared this assignment for a fictitious and improbable small business. To keep the template simple, I have used only three expense categories:

1. wages
2. matls (Materials)
3. ganda (General and Administrative)

and I have two income categories (code#1):

1. retl (Retail Sales)
2. whls (Wholesale Sales)

I have further decided that the business will have four products (code#2):

1. disks
2. forms
3. paper
4. ribbon

Of course, you would choose these to meet your needs! For this example, I would like to have you go along with my data so that we can compare results as we proceed.

Let's enter the data into the Data Input Module. Go to cell A9 and enter '1001'. This column will have the check and deposit references. I started the check numbers with 1001 and the deposit numbers with 6001. In cell B9, I have entered the date '8401.03'. I will explain why I have used this date number. The first two digits (84) represent the year 1984, the next two digits represent the month January, and the two place decimal represents the day of the month. Did I hear you ask why did he do this? There are many ways to enter dates into computer templates. In fact, 1-2-3 has special functions for representing dates. However, I am SET in my ways and feel that the decimal number system that I am suggesting has several advantages! First, the dates behave logically; that is, January 1, 1984 (8401.01) is greater than December 31, 1983 (8312.31). This will simplify the process of searching for all items (fields in database language) earlier or later than a desired date. Do you see this? If not, you will as we go along with this example. The use of the two decimal places to indicate the day of the month improves the readability in my mind. This numbering system lends itself to a method of calculating the number of days between two dates. (I will show you this method in the Stock

Market assignment that we will be doing in another article using Lookup Tables.) Of course, you might want to use the 1-2-3 built-in date functions for this capability, BUT those functions are NOT as convenient for many data management purposes unless some Keyboard Macros are used. I will show you Keyboard Macros in a future article. For now, I will not use them. I would like to have you work with my present method.

After entering the date, go to cell C9 and enter 'Steiner'. In cell D9, enter 'wages', cell E9 enter "disks", cell F9 enter '-400' (I will use -400 rather than (400) for this assignment.) I think that I hear you asking why does he use lowercase label and codes. It is just something that I always do when I work with database programs. There is no special reason other than it tells me this is a database program. At my age, I am set in my ways!

Before I proceed, I would like to have you consider some important data entry principles that we have established. First, consistency is a MUST when entering data into the trans#, date, name columns and CRUCIAL in the 'code#1' and 'code#2' columns. Once we start lowercase letters for the code references, we MUST stick with them! ALWAYS use the SAME code names! Inadvertent trailing spaces MUST BE avoided. "disks" and "disks" will be treated as different names as we use them for data processing. Figure 1 shows the complete set of data that will be used for this assignment. Please enter it exactly as shown! You can experiment with your Data Input Module and data AFTER you get this assignment completed and working. A couple of other notes. I have listed expense items as negative numbers. Some readers might prefer to maintain separate records (ledgers) for expense and income items. Then, all amounts in each column could be entered as positive numbers. I would usually do it that way, but I wanted to present this assignment in a non-accounting fashion; so, I mixed the expense and income items to show that it still is a workable method. Because I mixed them, I had to use different reference number series for each. (Checks 1001 and up. Deposits 6001 and up. They must not overlap!) It is important that with the mixed items to MAKE SURE that the references will always be different. This will allow the items to be "extracted" or what, as we will discover later in this article.

The 'memo' column "G" is available for additional information about the entries. Again, it is important to be consistent how the column is used. For instance, if we wanted to sort the data items later. An example would be to use the field "Sales" to list all the Sales items. We will clear this up as we use this. How about SAVING again? Do you remember the command?

/File Save BNKRCD84, and Return.

If you have been watching, the worksheet has been displaying a running balance in cell F5. If your entries agree with mine, your total should read 723.44! If you do not have this total, recheck your entries. Otherwise, this Module does not do anything unusual. Let's put it to work!

Some of the LOTUS 1-2-3's special functions will become useful with the Data Query Module that I will work with next. Refer to Figure 2 for my "SPREADSHEET Preparation Form." Don't forget I WANT YOU to do your own for practice to really learn how to do them and find out why they are very important. Again, I have suggested the column widths across the top and the labels. This is

an easy form to type, so go ahead and complete your Data Query template.

Now, I will create some additional range names that we will find useful. First, I will name the rectangle (I am sure all you readers know that is a quadrilateral having all its angles right angles. Right? Ha!) defined by cells H5..N6 as CRITERION. Here is the command that I used:

/Range Name Create CRITERION, Return, H5..N6, and Return.

Next, I named the rectangle H9..N400 as OUTPUT with another similar command as above:

/Range Name Create OUTPUT, Return, H9..N400, and Return. 1.

Now, I want to format column "H" so that its numbers will be integers using the following command:

/Range Format Fixed 0, Return, H1..N400, and Return.

And, format the rectangle H6..N6 in text format with the following command:

/Range Format Text H6..N6, and Return.

This will display any formulas that we may enter in this area in text form rather than as calculated values. This is a good method to use so that you can check your work as you go.

Last, go to cell M7 and enter the following formula:

@DSUM(INPUT,5,CRITERION).

Did you notice that we have a new formula? LOTUS 1-2-3 has a set of data base functions that can be used to generate statistics about a database. These functions have the following general form:

@function name(Input range,offset,Criterion range).

The offset is the column number of the field to be used in the database, starting with zero for the leftmost column. They work just about the way their corresponding 1-2-3 functions (without the "D"), except they use as their list of values the fields of

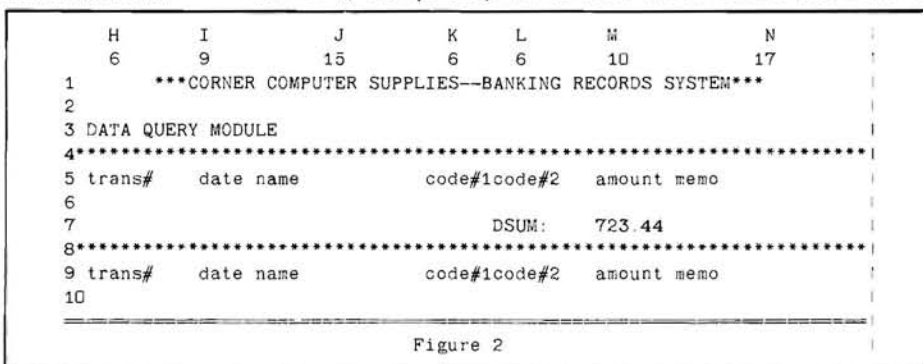


Figure 2

records in the Input range that fit the criteria in the Criterion range. I know that you will not find them easy to understand now, but I believe that by usage the function will become clear. Thus, the function general form is:

@DSUM(input,offset,criterion).

Now, if you will check back in this article, you will see that I defined the range — INPUT. Do you see why I used that name? I also defined a range — CRITERION. Again, can you see the choice for the name? I believe it is always a good idea to keep the names relative to the project. It makes them easier to remember

and help to learn them by repetition. The offset is column "M", which is the fifth column from the left labeled "amount". I will discuss the other "@D" functions when we need them in the future.

If you have created the Data Query Module correctly, cell M7 should have the same value as cell F5, which for this example is 723.44. Do you have this total? If not, go back and recheck your entries in this module. It is good template design to include such a check for your work! Maybe I should remind you to SAVE again.

I am going to present the LOTUS 1-2-3 Data Query functions next, but I should define the Data Query input, output, and criterion ranges first. You may wonder why we must do this, but it will become clear to you as we use these functions. They are easy because we have decided on the range names already. I have listed the Data Query commands below:

1. /Data Query Input INPUT, and Return.
This command assigns the range A7..G400, which we have previously named 'INPUT' as the Data Query input range.
2. Criterion CRITERION, and Return.
If you are following along with your computer on this assignment, you will find that you are still in the /Data Query submenu, so this and the following commands do not require reentering the /Data Query portion of the command! This command assigns the previously named range 'CRITERION' (H5..N6) as our Data Query Criterion range.
3. Output OUTPUT, and Return.
This time from the same /Data Query submenu, I have assigned the previously named range 'OUTPUT' (H9..N400) as the Data Query Output range.
4. Quit, and Return.
This command allows us to leave the /Data Query menu.

Next, I would like to have you learn some of the LOTUS 1-2-3 Data Query functions. In general, the Data Query functions permit us to select items from the input range (Remember what this range is?) by entering various match criteria (fancy word?) in the criterion range (Remember what this range is?). As I have stated many times before, the best way to learn is by doing. So, I will go over some Data Query function examples with you. I will explain the function and use the function with an example.

Let's search the BANKING RECORDS for all the entries that pertain to "disks". Go to cell L6 and enter 'disks'. MAKE SURE that the entry is EXACTLY the same way that was used when the data entry was made! NOTE! That a new SUM (DSUM) appears in cell M7. This is the result from the database statistical sum function and it is the total of all the entry amounts with the 'code#2' reference-'disks'. Now enter the following command:

```
/Data Query Find.
```

The screen will display the Data Input with the first entry highlighted. NOTE that this entry has 'disks' in the code#2 column. Press the cursor down arrow and note that the next entry with 'disks' in the code#2 column is now highlighted. Continue to use the cursor down and up keys until you are finished examining all the entries with 'disks' in the code#2 column. Press the HOME key and return to the first 'disks' record. Press the ESC key to return to the Query screen with the pointer back at cell L6. To return to the worksheet give the command Quit.

Now try searching for the entries that pertain to 'disks' and 'wages'. Enter 'wages' in cell K6 and leave 'disks' in L6. Note the

new total in cell M7. The Data Query Find function has already been defined, so press the Function Key "F7" (Z-100) for Query. How many entries did you find that meet the double criteria? Try different criteria until you really know this function. Press the ESC key to return to the worksheet.

Next, I would like to have you try the Data Query Extract function, which I think you will find even more useful than Find because it will provide you with a method to make separate lists of selected items from your input data. If you still do not have 'wages' in cell K6 and 'disks' in cell L6, put them back. Now use the following command:

```
/Data Query Extract
```

You will now see two rows displayed in the output section of the Query screen that meet the double criteria. Do you? The total of the extracted amounts will appear in cell M7. To return to the worksheet, give the command Quit.

Let's erase the cells K6..L6 with the following command:

```
/Range Erase K6 L6, and Return
```

Enter 'Morsecode' in cell J6 to extract all of the items for 'Morsecode'. The last Data Query function that we used was Extract, so press Function Key "F7" (Query) and you will have the Morsecode items displayed and totaled in M7. Do you find this?

I would like to demonstrate another Data Query function application that is very useful. This time I will enter a formula into the criterion area. Do you remember where this area is? As my example, I will extract all the items greater than \$1200. Erase cell J6 and enter the following formula into cell M6:

```
+amount>1200.
```

Don't forget the plus sign is required by 1-2-3 to specify a formula, not a label. I know that you know this, but I like to teach by repetition so that you will not forget these instructions. Press "F7" (Why?) and the items meeting the criteria will be displayed and their total will be in cell M7. How many items did you find displayed? Check the Input Data to see if all entries are shown. You can check your work easily.

Let's try another, to extract and total all the expense items, enter the following formula:

```
+amount<=0
```

Do you see why I used this? I am sure that you can. I will not use the space to explain it. Again, refer to the Input Data and see if all items are displayed and the correct total shown. I could use another Extract function to test that all expense items were entered as negative numbers. I will enter the following formula in cell H6 and erase cell M6:

```
+trans#<=6000
```

Did you remember that all expenses were paid by checks whose numbers were 1001 up to 6000, but not over 6000? Press "F7" again. Now check the extracted item display and make sure that all the check reference entries have negative amounts. Do they?

Here is another usage for this valuable function. I will extract all expense items paid between January 6 and January 16. I will leave the above formula in cell H6 and enter the following formula in cell I6 (letter "I"):

```
+date>8401.05#AND#date<8401.17
```

Press "F7". Did you remember that with 1-2-3 we must use the "#" symbol on each side of the logical operator — AND? Did your display and total come out correctly? Did you check it?

I could further limit my criteria for the Extract by entering 'paper' in cell L6 and pressing "F6". Did it work correctly? Do you know why? I would like to remind you that you CAN ENTER formulas in the Criterion range, because we assigned names to the columns someplace back in this article to the columns in the Data Input Module with the following command:

```
/Range Labels Down.
```

Go back and review where we did this!

Now, I would like to have you try at least one example by yourself. I would like to have you extract some items from the memo column by moving the cursor pointer to cell N6 and enter the appropriate match criteria. Can you do it? It might take a couple of tries, but I am sure that you can do it! Try other uses for this function, and there are a lot of them that we have not done, until you feel that you really know this important function!

I hope that after you have completed the various Extracted entries that you can see that you can save or print any/all of those items to separate Print Files, and that they could easily be inserted into reports or other word processing files. The command would be as follows:

```
/Print File
```

If you cannot, do not worry. I am going to go into Print Files in detail in a future article. I just thought that some of you readers might be ahead of the rest of us. I asked for cards about your equipment, such as printers that I would like to know about before I get into 1-2-3 printing commands!

The best part of the Banking Records System is yet to come! I will complete the Data Table Module with you which will provide summaries for all the data entries. Before I proceed, erase whatever you may still have entered into the criterion range H6..N6. You know how, don't you? I gave the command back away in this article. It is a good time to review anyway.

Now, I will begin the construction of the Data Table Module. Figure 3 shows my "SPREADSHEET Preparation Form." You did not forget that we always do this first? Enter the labels and divider lines per the Form. BE SURE that when you enter the code names that they are EXACTLY the way they appear in the other two modules! If you were creating your own Banking Records, you would continue to enter each of your code#1 categories down column "P". It is also VERY IMPORTANT that ALL of the possible code#1 entries be included in column "P". Enter ALL code#2 entries in row 7 starting at cell Q7. AGAIN, please make SURE that these entries match the Data Input Module labels! You can choose whether you would like the labels right or left aligned, BUT the spelling and capitalization along with inadvertent leading or trailing spaces MUST BE THE SAME as you used previously! If this were your form, you would continue your entries for all code#2 items along row 7.

The Data Table Module requires some simple formulas that we

now all know. Put the following formula in cell V8:

```
@SUM(Q8..T8).
```

Copy this formula to cells V9..V12 with the following command:

```
/Copy V8, Return, V9..V12, and Return.
```

Use the following formula in cell Q14:

```
@SUM(Q8..Q12).
```

Copy this formula to cells R14..V14 with the following command:

```
/Copy Q14, Return, R14..V14, and Return.
```

Go to cell P7 and enter the following absolute reference:

```
+$M$7.
```

If you do not remember what "absolute" means, go back and review previous "SPREADSHEET Corner" articles and/or use the 1-2-3 HELP screens and manual. I will not explain why it is needed, because I want to make SURE that you REVIEW this subject. It will be very important in future articles! The Data Table Module should now look like Figure 3 and if you have done everything correctly, cell P7 should have the same result as cell M7 and F5 — 923.44! This is our automatic check. BE SURE to SAVE your work.

The Data Table Module will demonstrate the LOTUS 1-2-3 data-

	0	P	Q	R	S	T	U	V
	8	8	9	9	9	9	3	11
1		***CORNER COMPUTER SUPPLIES---BANKING RECORDS SYSTEM***						
2								
3		DATA TABLE MODULE						
4		*****						
5								
6			CODE#2					
7		723.44	disks	forms	paper	ribbon		TOTALS
8		wages	-1600.00	-1540.00	-1100.00	-1560.00		-5800.00
9		matls	-770.00	-550.00	-600.00	-1160.00		-3080.00
10	CODE#1	ganda	0.00	0.00	-800.00	-1000.00		-1800.00
11		ret1	2334.28	1228.96	0.00	669.32		4232.56
12		whls1	2360.00	2250.00	1375.00	1185.88		7170.88
13								
14		TOTALS:	2324.28	1388.96	-1125.00	-1864.80		723.44

Figure 3

base POWER! The results will appear faster than the moves that I will be trying to explain. Therefore, you are really going to have to watch what we do carefully! Before I start, recheck that you have erased every entry in cells H6..N6. I have given the command for this previously. This is IMPORTANT!

Move the cursor pointer to cell P7 and enter the following command:

```
/Data Table 2.
```

A prompt will ask for Table Range. Enter P7..T12. NOTE what area this includes. The next prompt will ask for Input cell 1. Enter K6. And the last prompt will ask for Input cell 2. Enter L6. The message "WAIT" will appear in the upper right corner of the screen. After about 15 seconds, the complete Data Table Summary will appear with the subtotals for all combinations of code#1 and code#2 entries. For an automatic check of our results, you should find that cell V14 should contain the same value as cell P7 — 923.44. If they are not the same, you have error of omission,

duplication or misspelling (including inadvertent spaces) when you entered code#1 or code#2 entries in the Data Table Module. Recheck carefully until you obtain the correct results.

Also, check the cell V14 value with cell F5. Again, they should be the same. If not, the Data Table Module does not match the total of the entries in the Data Input Module. The usual reason for this error is some entry was left in cells H6..N6. Erase these cells again as previously instructed. When cell V14 value equals cell F5 and cell M7, other data manipulations are possible using the Banking Records System. For an example, I will display the first half of January 1984 in the Data Table. Go to cell I6 and enter the following formula:

```
+date-8401.16.
```

Since I have previously defined the Data Table function, press Function Key "F8" (TABLE), wait a few seconds and the Data Table will display the new summary for the first half of January. Did you get the results? That covers a lot of WHAT IF'S... in a hurry, RIGHT? Now that you have the Banking Records System set up, there are many more possibilities for its use with the 1-2-3 features. This article is pretty long, so I would like you readers to try some by yourselves. I suggest that you try Data Sort with the memo column. Look Data Sort up in the 1-2-3 Manual. I have described a database system that is applicable to your needs. After you have tried the template, please try to adapt it to your application!

Oh yes, I want to quickly explain what you need to do if you reach the row 400 of this template with the input data (or whatever size you have decided on). To extend the capacity, you must insert extra rows. The easy way to do this is to move the cursor pointer

to cell A400 and enter the following command (I am going to change to 500 rows.):

```
/Worksheet Input Row A400..A500, and Return.
```

This Insert Row command will automatically redefine the named ranges INPUT, OUTPUT, and CRITERION and also change the formula in cell F5. All the database functions will work correctly. However, the other ranges to put columns in integer format must be reformatted. Look back in this article for the ranges A9..A400 and H9..H400 and change the 400 to 500. WARNING! Inserting rows within the top 8 rows or using the ?Move command to shift areas of the template can cause a lot of errors to occur in most functions and formulas. SO, experiment carefully and always keep backup files!

HAPPY SPREADSHEETING!!!



Plan your summer vacation 
NOW!
 Attend the 4th Annual
HUG International Conference
 Chicago, O'Hare Hyatt Regency
 August 9, 10, 11 

CLARKSON SOFTWARE \$49

High quality software developed for Clarkson's Z-100 program. Packages at \$49 each for Z-100, Z-150 and IBM PC compatible machines. Institution rates available.

GALAHAD: Word processor, Zeditor, Lancelot speller and language analyzer.

NARCISSUS: Authoring system for quizzes, essay tests, writing aids, language drill.

SMALL-C: Version 2.1 running under MS-DOS (ZDOS).

TERMINAL EMULATOR: Emulates a Tektronix 4010 graphics terminal, the H19, Z19 and VT52. (Available Z-100 only.)

GRAPHICS LIBRARIES: Graphics dumps for Z-100, and MS-set of graphics subroutines for Fortran and Pascal. (Available Z-100 only.)



Contact
 Professor David W. Bray
 Clarkson University
 (315) 268-6455

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 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 XON/XOFF and READY/BUSY Protocols

NO SPECIAL SOFTWARE REQUIRED: Transfer disk files (Intel Hex Format) to EPROM with your system's line printer or modem software. Transfer EPROM contents to disk file in Intel Hex Format 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	27C16	8741*	8748*
2516	2732	27128A	2815	8742*	8749*
2532	2732A	27256	2816		8751*
2564	2764	68764	2817	X2212*	8755*

*Requires low cost personality adapter

SOFTWARE AVAILABLE FOR: **

CP/M systems on 8" SSSD - many 5 1/4 formats
 TRS-80 Model I & III TRSDOS
 Heath HS/H89 HDOS & CP/M
 ZENITH Z90 & Z100 CP/M - Z100 ZDOS
 KAYPRO II/IV
 IBM PC-DOS - many MS-DOS systems

IntelliBurner Programmer with Software \$299.00
 RS-232 Interconnect Cable 12.00
 IntelliBurner PC Board, EPROM, Plans & Software 99.00

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
DumBurner II Bare PC Board, Plans & Software	59
16K/32K DumBurner for 24 Pin EPROMs with software	149
16K/32K DumBurner PC Board, Plans & Software	39



ROSS CUSTOM ELECTRONICS

1307 Darlene Way-Suite A12
 Boulder City, Nevada 89005

PHONE (702) 293-7426

!! 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 Base Billing, \$1.00 * accepted, Foreign Orders add required postage. Specialty Environment and Media Requirements, MS-DOS 2.10, see TM Heath/Zenith, CP/M a TM Digital Research, TRS-80 Model III a TM Tandy Corp, IBM PC and PC-DOS are TM IBM, MS-DOS a TM Microsoft



Meltdown Imminent

John W. Rogers

Bored, with nothing to do? Why not fire up the nuclear reactor and produce a little power? Or maybe a little meltdown will suit you better. If nothing else, it should certainly liven things up a bit. Sure, you say, and where am I going to come up with a nuclear reactor anyway?

Well, have I got a deal for you. All you need is your trusty H/Z-100 computer and you're all set for Reactor-100. This program is not really a game in the strictest sense, but more of a simulation of a nuclear power plant. Using the superb graphics capabilities of the 100, this simulation tests your good judgement and common sense (a little luck doesn't hurt either).

David B. Wadsworth is the author of Reactor-100 which is written in compiled Z-BASIC. Compiled Z-BASIC programs require that you have version 1.2 or above of the monitor ROM installed in your H/Z-100 computer. You will also need the color RAM option installed on your video board, as many of the schematics and displays in the program are displayed in high resolution color graphics. You can check which version of the monitor ROM is in your machine by typing a "V" from the boot prompt. If your machine is set to autoboot, you will have to abort the autoboot sequence by hitting the DELETE key immediately after the machine beeps the second time upon power-up. The program also supports the P-SST board and light pen both sold by Software Wizardry. The P-SST board is used to generate sound effects and the light pen can be used to select commands and help information from the computer screen.

For the nuclear power neophytes out there, a little background in nuclear power plant operation is probably in order.

When a radioactive material is bombarded with neutrons, nuclear fission takes place. Some substances, such as uranium 238, provide enough of their own neutrons to free other neutrons, thus producing a self-sustaining reaction (a chain reaction). As a neutron is knocked free of its parent atom, the energy that formerly held it in place is released. This energy is harnessed in a nuclear reactor in the form of heat.

In all reactors, a structure of fissionable material is arranged in such a manner so as to produce a controlled chain reaction. This reaction becomes self-sustaining when it is creating enough neutrons to

support the continued fissioning of the fuel. One problem is that the neutrons travel so fast that they tend to leave the reactor without reacting with other neutrons. To slow down these particles, a moderating material is needed. This material acts as a buffer which slows the neutrons to thermal speeds where they will do some good in supporting a continued reaction.

The pressurized light water reactor is one of the most widely used types of nuclear reactors. In this reactor, a fuel, such as uranium 235, is arranged in a geometric pattern which promotes fission and also provides flow around the fuel elements of normal (but very pure) water. The water acts as a heat removal medium and also as the moderating material. The water is maintained at very high pressure to prevent boiling, and is recirculated through a heat exchanger.

The reactor modeled in this simulation is of this type and is very similar to those used in a standard nuclear power plant.

The object of the simulation is to bring the nuclear plant on line and produce power without any unfortunate mishaps. There is a training mode where the systems will not be breaking down, providing you with the opportunity to get the feel of controlling the reactor without having to worry about what will go wrong next. After you have become comfortable with running the reactor, you can run under the real mode in which random failures occur which must be dealt with in a proper manner. Failure to properly deal with any problems or operating the reactor too close to the limits can result in anything from equipment damage to a complete meltdown of the reactor. Naturally, this is to be avoided.

In case this all seems too simple to you, remember that while the simulation is running the demand on power from your plant is varying. You are not only expected to provide enough power to meet these demands, but it would be nice if you could produce enough to sell some excess into the power line grid. Also, you must monitor the cooling systems and occasionally replenish their levels when they run too low. After running the simulation for a period of time, you shut down the plant for general maintenance. At this time, you are given a damage report of how the plant is faring and the maintenance is automatically performed. When you decide to terminate the game, your performance is rated and recorded to disk. This figure is used to

calculate an average for your performance over all the sessions you run.

As I mentioned previously, the simulation uses the high-resolution color graphics mode of the Z-100. A schematic of the power plant is shown on the screen and two menus, one showing the status of the various power plant systems, and another consisting of warning signals. The schematic and the menus make extensive use of color to reflect the condition of the various systems and controls. If something is approaching some critical value, the color of an indicator changes to yellow, and if the problem becomes of critical proportions, the indicator goes to red. This feature makes it very easy to quickly scan the screen for any problems. On the schematic, color is used to show the operating position of the valves and the condition of the various pumps.

When I first opened the manual for this program, I was immediately impressed by the neatness and organization of the layout. It gives the appearance that somebody has devoted a little time in trying to make the program documentation clear and easy to interpret. The manual appears to have been typed on some type of letter quality printer and it comes in one of Software Wizardry's very nice vinyl binders.

Although the layout and content of the documentation is very good, the program itself seems, at first, a bit overwhelming. But after about four or five sessions of using the training mode and bringing the reactor online, using a guide provided in the manual, I could do it without the help of that guide. After a few more sessions, I started using the real mode of operation and the simulation got really interesting. All I will say is that my neighbor who works at a nuclear plant near here does not want me to even get close to the real plant. Now I am getting pretty good at this, and I only destroy the neighborhood about two times out of five.

I enjoy this program very much and highly recommend it to anyone looking for something other than the average game program. As I mentioned, the documentation is excellent and the program makes good use of the Z-100's high-resolution color graphics capabilities. If you think that you may be interested in the program, you obtain it from:

Software Wizardry
122 Yankee Drive
St. Charles, MO 63301
(314) 946-1968



IMPROVED GRAPHICS for H19 & H89

G-Prom is a new character generator that

- Improves screen graphics resolution 2½ times.
- Improves formation of 23 ASCII characters.
- Is a plug-in replacement for the original C.G. ROM.

Only \$19.95 with shipping, documentation, instructions, and demo program listing.

NORCOM

9630 Hayes
Overland Park, KS 66212

SPIKE PROTECTION

PROTECT COMPUTERS, VCR's, STEREOS,
AND OTHER EQUIPMENT

Attractive 6-outlet power strip!

3-way protection from spikes
Clamps voltage to 135 VAC

AVOID DAMAGE TO YOUR EQUIPMENT & FILES -

MODEL

101 Basic 3-Way Protection.....	\$29.95
102 w/LED monitor circuit added.....	\$39.95**
103 w/filter added to Model 101.....	\$49.95
104 w/LED circuit added to Model 103..	\$62.95

**Recommended for most users

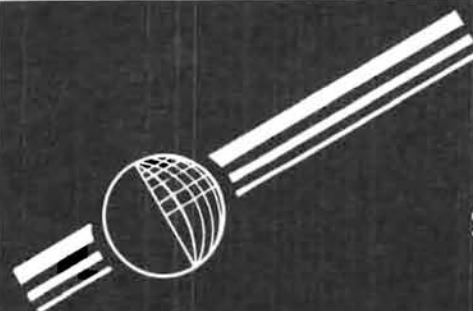
ALSO PROTECT DIRECT-CONNECT MODEMS..... \$15.00

FABRIC COVERS FOR H/Z COMPUTERS..... \$10.00up

FABRIC COVERS FOR PRINTERS..(Inquire).. \$ 6.00up

SHIPPING/HANDLING \$3. Checks/money orders only.
Order shipped after check clears.

HME, Inc., 751 Westcott Dr., Wetumpka, AL 36092



Graphic
Design
Systems
Inc.

Z-100

MEMORY BOARD

STARTING AT 379.00 (64K INSTALLED)

- SPECIALLY DESIGNED FOR Z-100
- UP TO 1MB CAPACITY
- USES 64K OR 256K CHIPS
- PARITY GENERATION / DETECTION

CONTACT: Graphic Design Systems Inc.
1020 ORIOLE LANE
MARIETTA GA. 30067
(404) 973-8471

Loading Assembled User Routines In Microsoft BASIC Under CP/M

Robert S. Wrathall
113 E. Riviera Dr.
Tempe, AZ 85282

Introduction

One of the greater liabilities of using CP/M is the difficulty of adding user generated IO. Device drivers are built into the BIOS.SYS program, and adding a new driver necessitates re-writing the BIOS. For all but the most undaunted system programmers, this is a real chore. For most of the "Rest Of Us", we would like to do something quick and dirty to do the job, and we mostly program in BASIC or FORTRAN. Hence, we need a way of quickly and easily generating specialized assembly language routines for special functions and IO, and linking them to the higher level language.

(It should be mentioned here, that one of the advantages of HDOS is its relative independence of IO difficulty. It is possible to quickly and easily write some sophisticated IO routines and incorporate them directly into the operating system as needed. See the very excellent article by Dallas, Lamb and Jorgenson entitled "The HDOS Device Driver Programmer's Guide" (REMark 20, September 1981.)

In Microsoft BASIC, the capability is given in the form of the USER function. If the user can get an assembled routine into memory, then he/she can access that routine by calling the USER function. In the typical user function call, a pointer is stored in the Z80 processor registers which point to the storage location of a variable. This information can be used in passing data from the user function routine to the BASIC program.

The methods of loading such user routines tend to be awkward. A typical technique is to incorporate the user routine as a data definition for a text variable. The function VARPTR will return the pointer to the text string which can be decoded for a USER function call. In this method, the routine is limited to 255 bytes and must be position independent code.

A second method is to use DDT to load BASIC and the user routines at the locations where they belong, and then branching to the start of BASIC and running. This method does not have the severe limitations of those imposed by the text string method, but it is a little awkward. It can probably be facilitated by the use of the XSUB and SUBMIT utilities, however.

Method

An alternative method for loading user routines is to load directly from BASIC. This method is direct and allows for the usage of user routines of any length and non-relocatable code. To do this, a short routine must be included in the BASIC code which loads user routines into memory above at memory locations above BASIC. This routine allows the programmer to load user function, ad lib., from the disk and to POKE them into memory.

The random access mode of disk IO can read blocks of data from a .COM file and store them into a string variable. These data can then be POKE'd into the memory locations where they are to be used.

The real trick is to generate the .COM file in such a way that it can be readily loaded by the basic routine. This is done in the M80 macro assembler and L80 linker. The M80 macro assembler creates a .REL file and the linker converts that file to a .COM file and links it together with any routines from libraries which are specified. One of the options of the M80 macro assembler is the .PHASE command. This command allows the assembly of code stored at the memory location 100H, but meant to run at some other location. When the file is saved by the L80 linker, the location 100H is assumed the start of the linked file. With the .PHASE command, the routine is not runnable at location 100H because all memory calls are referenced to the .PHASE location. In order to run this routine, it must be moved in memory to the location specified by the .PHASE command. An example of such a user assembled routine is shown below.

```
START EQU 'STARTING ADDRESS'  
      .PHASE START  
      {PROGRAM LINES}  
      END START
```

The END statement adds three bytes to the beginning of the .COM file. This is a JMP statement to the start of the program followed by the two bytes of the start address, which in this example is START. Hence, the user routine will reside at 'STARTING ADDRESS' with three bytes added on by the END statement,

which simply JMP to 'STARTING ADDRESS'. These three bytes can be used by the BASIC routine to determine the load address of that file and to place it at that location.

Using this technique, it is possible to write some fairly sophisticated interrupt-driven IO routines for devices to be accessed by a BASIC program, where the true number crunching and logic can be performed.

When this routine is implemented, care must be taken not to overwrite the BIOS.SYS which resides in high memory. The start of BIOS can be obtained from memory locations 0001 and 0002. Also, care must be taken not to overwrite the user routines by BASIC operations. This can be avoided by running BASIC with the /M:<highest memory location> switch in the command line where the highest memory location is below the user routines.

Listing Of BASIC Program

```

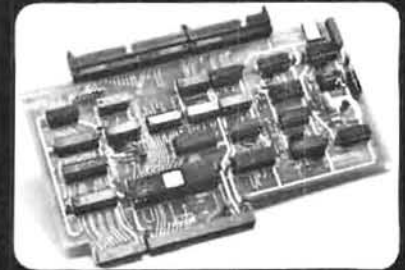
10 REM***** LDUSER.BAS - R.S. WRATHALL 3/8/84 *****
11 REM* THIS ROUTINE LOADS A USER FILE INTO MEMORY FROM
    A COM FILE *
12 REM* AND LOADS IT IN POSITION TO RUN. IT MUST BE A
    COM FILE WITH *
13 REM* ITS START ADDRESS LOADED INTO THE 2ND AND 3RD
    BYTES. THIS *
14 REM* CAN BE GENERATED IN THE M80 MACRO ASSEMBLER
    WITH THE PHASE *
15 REM*   START   EQU -STARTING ADDRESS- *
16 REM*   .PHASE   START *
17 REM*   (PROGRAM) *
18 REM*   END START *
20 REM* THE FIRST LINE OF (PROGRAM) MUST BE THE BEGINNING
    OR A BRANCH *
21 REM* TO THE REAL START ADDRESS OF THE PROGRAM *
22 REM* .....
100 DEFINT I,J:REM DEFINE 'I,J' AS INTEGER
110 IZ=0:REM ZERO "ZERO COUNTER"
120 OPEN "R",1,"USER3.COM":
    REM OPEN FILE 'USER3.COM' AS A RANDOM ACCESS FILE
130 FIELD #1,128 AS A$:REM SET UP A$ AS A BUFFER
135 J=0:REM ZERO BLOCK COUNTER
140 J=J+1:REM INCREMENT BLOCK COUNTER
150 GET #1,J:
    REM GET A BLOCK OF DATA FROM RANDOM FILE AND PLACE IN A$
160 IF J=1 THEN GOSUB 300:REM FIND START OF ROUTINE
190 IF EOF(1) THEN GOTO 270:REM LOOK FOR END OF FILE
200 FOR I=0 TO 127:REM START POKING THE BLOCK
210 IF MID$(A$,I+1,1)=CHR$(0) THEN IZ=IZ+1 ELSE IZ=0:
    REM LOOK FOR 0'S FOR END
220 IF IZ>20 THEN 270:REM IF TWENTY ZERO'S,
    THEN END (MAY NOT NEED THESE LINES)
225 REM POKE DATA OFFSET BY THREE TO SKIP OVER
    JUMP STATEMENT IN COM FILE
230 POKE IADDR+I+128*(J-1)-3,CVI(MID$(A$,I+1,1)+CHR$(0)):
    REM POKE
235 REM OPTIONAL PRINT STATEMENT TO SEE WHAT IS GOING IN
240 PRINT HEX$(IADDR+I+128*(J-1)-3);" ";
    HEX$(CVI(MID$(A$,I+1,1)+CHR$(0)));" ";
250 NEXT I:REM END BLOCK DUMP
260 GOTO 140:REM LOOP FOR NEXT BLOCK
270 CLOSE:
    REM FROM END OF FILE STATEMENT, CLOSE RANDOM FILE
280 END
290 REM SUBROUTINE TO DETERMINE LOAD ADDRESS OF COM FILE
300 IADDR=CVI(MID$(A$,2,2)):
    REM   FIND ADDRESS TO START LOAD
310 PRINT "ADDRESS OF LOAD = ";HEX$(IADDR):
    REM AND PRINT IT
320 RETURN

```

✱

1 CONTROLLER

FOR 8"
& 5.25"
DRIVES



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

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

PRICED AT \$395

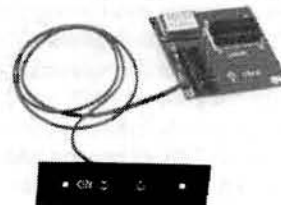
Includes controller board CP/M boot prom, I/O decoder prom, hardware/software manuals BIOS source listing. HDOS driver now available for \$50.00.

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



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

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.

**SPEND LESS TIME WAITING
FOR YOUR COMPUTER**

Contact your local Heath/Zenith dealer to buy this cost effective Z100 enhancement, or contact:



Controlled Data Recording Systems, Inc.
7210 Clairemont Mesa Blvd.
San Diego, Ca. 92111
Telephone (619) 560-1272

The Elusive Escape Codes In PASCAL

*Paul W. Simmons
6409 Glenbard Road
Burke, VA 22015*

Early in February, I ordered a Low-Profile Z-100 computer, a color monitor, and a host of software. I then began to wait, wait, wait, and wait. In about three weeks, in the midst of painting the kitchen yellow, I got a call. The voice on the phone said, "Your computer is in." With the finesse of a professional quarterback, I passed off the paint roller to my wife, leaped over the paint cans and brushes, opened the door and sped away in the family van. On the way to the store, I prioritized the work to be done on the Z-100. First, convert a financial management package to the Z-100. Next, write a bowling league secretary-treasurer program. Then, learn the programming language Pascal. Better yet, write the bowling league program in Pascal.

At the store, I picked up the computer and software, but there was no monitor. "The monitor will be here in a month or so," the salesman said. Now what? What good is a computer without a monitor? Then I had a brilliant idea: maybe I could borrow Dirck's monitor. He bought an All-In-One Z-100 computer and a color monitor in December. That meant he had a green and white built-in monitor, and an extra color monitor just gathering dust. I thought, "Dirck's easy. He'll surely let me borrow his dusty old monitor." So I borrowed a quarter from the salesman (after all, who can afford a phone call after buying a Z-100), and called Dirck. "Dirck!", I said, "My computer is in, but the color monitor won't show up for another month or so." Dirck responded with, "Who is this?" After I settled down, I eloquently pleaded my case and in the process totally humiliated myself. I also got the color monitor.

The conversion of the financial management program went smoothly. Wow, the Z-100 is fast! Now for Pascal and the bowling program. Since I knew nothing about Pascal, I started by doing some studying. I devoured several texts on Pascal and the Microsoft Pascal manuals that came with the computer. The Microsoft manuals are tough, but I managed to retain at least 1% of what I read.

At the crack of dawn one Saturday morning in April, I started to write the bowling league data entry program. The plan was to duplicate the Captain's Record (the form that the two competing teams record their bowlers' scores upon) on the screen. The program would prompt the user for input by moving to the appropriate locations on the form. After a data value is entered the cursor would automatically move to the next input location on the form. The form would be displayed in blue on the screen and

the user's response would be displayed in white. Using this scenario, it is clear at a glance what text is user input and what text is part of the form.

I completed the Warnier-Orr diagrams (a modern version of the flow chart for diagramming structured code) and began coding. Then things came to a complete stop. How do you change colors and control the cursor in Pascal? Obviously, it must be one of the things I missed or forgot when reading the Pascal manual.

So I scoured the manuals one more time, but still found no answer. When Zenith documents something they do it up royally, and the solution is probably somewhere in the twenty-one Z-100 manuals on my bookshelf. But by the time I worked my way through them all, I'd be so old and senile I'd have forgotten what I was looking for. Then there was a great flash (an explosion of insight, you might say). Many printers are controlled by escape sequences (the escape character followed by one or more command and data characters); maybe the color monitor was also controlled that way. Sure enough, in the TM-100 Technical Manual on page 10-38, there they were, all kinds of escape sequences doing everything I wanted to do. The only difference was that Zenith terms them escape codes. I immediately began writing a series of test programs. I thought all you would need to do would be to send the escape sequence to the monitor with the WRITE(x) procedure in Pascal and the desired action would be performed. Wrong, wrong, and wrong! Some things worked, but any command that positioned the cursor failed. Now what?

Back to studying the Pascal manual. By now beads of sweat were trickling off my forehead. My wife was quipping repeatedly, "Where's the color?" Finally, the answer was found in Appendix A of the Pascal Compiler User's Manual. It turns out the DOSXQQ function allows a Pascal program to access the Z-DOS function requests described in Appendix I of the Z-DOS manual. These functions allow the programmer to pass escape sequences to the monitor.

Another battery of test programs were written. With each test program I learned over and over again what the textbooks mean when they say Pascal is a strictly typed language. What a shock to a hacker like myself who grew up with FORTRAN, BASIC, and Assembler. Once I had learned how to move the cursor at will and change colors, I began coding the bowling program Cap-

tain's Report again, happy as a clam. However, two nagging thoughts continued to recur: 1) maybe there is a better way and 2) how many other people are there floundering around, attempting to do these things. Several hours later the Captain's Report was complete. My wife passed by the computer as she frequently does to see if my eyes have completely bugged out. She saw the routine running in living color and rewarded me with a hug. "That's it!" I said, "I'll write an article for HUG. Certainly there is a Z-100 Pascal programmer out there that knows about these things. Also, I'll find out if there is a set of Pascal procedures to use the high resolution graphics capabilities of the Z-100, you know, things like plot, draw, circle, move, and so on." Send a note if you know how to do this or better yet, send an article to HUG.

Eight procedures and one function were written to issue escape codes to Z-DOS. These are contained in Listing 1. They have the ability to perform all the necessary cursor functions and many other things, like erasing and editing functions, changing the mode of operation, etc. Listing 1 is in the proper form to be included in your Pascal program immediately before the definition of your procedures. I maintain Listing 1 in a file on my Pascal diskette named ESCAPE.PAS and include it into my program by inserting the following line in the source code prior to the procedure definitions: [include: 'ESCAPE.PAS'].

Listing 2 is a demonstration program using the procedures defined in Listing 1. This code will no doubt appear amusing to you experienced Pascal programmers. It's the best I can do at the moment. Go ahead and laugh, I keep myself in stitches most of the time. Briefly, this program puts a form on the screen with blue letters on a red background. The user then types in her or his name and address. The cursor automatically moves to the next input location on the form. Once all input is entered the bottom portion of the screen is changed to blue and the input data are printed out in white characters. I realize this is not an earth-shattering routine, but it does exemplify the utility of using the procedures in Listing 1.

Before I begin a detailed description of Listing 1, let's review the screen coordinate system used by the Z-100. The top-left corner of the screen has a coordinate of row=1 and column=1. Column values increase up to 80 as we proceed across the screen from left to right. The row values increase to 25 as we proceed down the screen. Row 25 is a special row on the Z-100. It does not scroll, therefore, permanent information such as the function key definitions are frequently displayed there.

The following is a description of Listing 1. The only defense I have for this code is it works. That is, it works on my computer. Certainly what's good for me is good for you, too?

The first line of the listing defines the function DOSXQQ. The directive EXTERN identifies DOSXQQ function as residing in another loaded module. Thus, only the DOSXQQ function heading is needed in Listing 1 followed by the word EXTERN. Details of how to use the DOSXQQ function are contained on page 119 of the Microsoft Pascal Compiler User's Guide. The DOSXQQ Pascal function apparently provides an interface to the Z-DOS functions defined beginning on page I.5, Appendix I, Volume II of the Z-DOS Software Documentation.

Procedure CLEAR. [ESC E]

This procedure clears the screen and homes the cursor (moves the cursor to the upper-left corner of the screen). There are no parameters to worry about and the procedure is invoked by sim-

ply including the word CLEAR in your program. This is done on line 57 of Listing 2. Procedure CLEAR operates as follows: First, the function DOSXQQ is called with parameters 2 and 27. The 2 selects Z-DOS function 2 (video output) and the 27 is the ordinal value of the escape character (the position of the escape character in the collating sequence). Then function DOSXQQ is called a second time with the parameters 2 and WRD('E'). As with the previous call, the 2 selects the video output Z-DOS function. The expression WRD('E') supplies the ordinal value of the character 'E' in word format to DOSXQQ. As a result of these calls to DOSXQQ, the sequence 'ESC E' is transmitted to Z-DOS and Z-DOS responds by clearing the screen and homing the cursor. That's all there is to it. The 'x' values returned by the two DOSXQQ calls are unimportant and are ignored.

Procedure LOCATE. [ESC Y ROW COL]

Procedure LOCATE positions the cursor at a specified row and column on the monitor's screen. Thus, a subsequent read or write will take place at that location on the screen. Two value parameters are passed to the LOCATE procedure; they are ROW, which may range from 1 to 25, and COL, which may range from 1 to 80.

Listing 1 shows that LOCATE works as follows: First, the sequence 'ESC Y' is passed to Z-DOS via the DOSXQQ function in a manner similar to that used in the procedure CLEAR. Next, the row and column positions are passed to Z-DOS. (An example of how this is done is on line 58 of Listing 2.) ROW and COL are integer values and must be converted to type word before the DOSXQQ function will accept them. This is accomplished by using the WRD function (e.g. the expression WRD(ROW) generates an equivalent word value for the integer ROW). Then an offset of 31 must be added to each value. Add 31! Why? Well, it turns out that Z-DOS actually expects the ROW and COL values to be ASCII characters. The ordinal value of the character reflects the row or column value. The first printable ASCII character was chosen as the first row and column value. This is the blank character and has ordinal value of 32. Therefore, an offset of 31 must be added to the row and column values so they will reflect the correct ASCII character. You don't need to worry about all of this since the LOCATE procedure does it all for you.

In this case, the 'x' values returned by the DOSXQQ function are unimportant and are ignored.

Procedure LOCATE will allow you to write on the 25th line of the monitor. As mentioned earlier, this line does not scroll; thus it is a good place to display permanent instructions. Before the 25th line may be written to, it must be enabled by calling procedure ESCAPE__x, which will be discussed a little later.

Procedure POSITION. [ESC n ROW COL]

The procedure POSITION returns the current row and column position of the cursor. An example of the use of the POSITION procedure is shown on line 70 of Listing 2. In this example, the position of the cursor is found after the word 'NAME:' is printed. This position is then saved for later repositioning of the cursor.

As Listing 1 shows, procedure POSITION uses Z-DOS function 2 (video output) to issue the sequence 'ESC n' and it uses Z-DOS function 6 (direct console I/O) to retrieve the response. The response is returned in the form 'ESC Y ROW COL'. Lines 41 and 42 issue the 'ESC n' sequence as we have seen before and line 43 retrieves the first character of the response from Z-DOS. Note,

when we wish to retrieve characters from Z-DOS we use the same function DOSXQQ with the first parameter set to 6 and the second parameter set to 255. In this case, the value of 'x' is significant. It is the value returned by Z-DOS. Thus, when line 43 is executed, 'x' is set to 27 (the ordinal value of the escape character) by the DOSXQQ function. Then, when the next line is executed, 'x' is set to 89 (the ordinal value for 'Y') by the DOSXQQ function. The POSITION procedure reads past these values to get to the row and column values. On line 45, DOSXQQ is called to retrieve the row value of the cursor. It is returned in 'x' as type byte. Also, it contains an offset of 31 similar to the LOCATE procedure. The ORD function is used to convert 'x' to integer. Then 31 is subtracted and the result is set equal to the parameter variable ROW. The column position is processed similarly and set to the parameter variable COL.

Procedure ESCAPE [ESC COMMAND]

Procedure ESCAPE may be used to perform a variety of different functions. The value parameter COMMAND, of type character, is supplied by the user to select the desired result. On line 91 of Listing 2 the character 'J' is used to clear the screen from the current cursor position to the bottom of the screen. Below is a list of some of the escape codes and their purposes. This information was taken from pages 10.38 to 10.41 of TM-100 Technical Manual.

Cursor Functions

ESC A Cursor up
ESC B Cursor down
ESC C Cursor right
ESC D Cursor left
ESC H Cursor home
ESC I Cursor index
ESC j Save cursor position
ESC k Set cursor to previously saved position

Erasing And Editing (I have not tested most of these.)

ESC E Clear display and home cursor
ESC J Erase to end of page
ESC K Erase to end of line
ESC L Insert line
ESC M Delete line
ESC N Delete character
ESC O Exit insert character mode
ESC @ Enter insert character mode
ESC b Erase to beginning of display
ESC l Erase entire line
ESC o Erase to beginning of line

Modes Of Operation

ESC F Enter graphics mode
ESC G Exit graphics mode
ESC = Enter alternate keypad mode
ESC > Exit alternate keypad mode
ESC p Enter reverse video mode
ESC q Exit reverse video mode
ESC t Enter keypad shifted mode
ESC u Exit keypad shifted mode

Additional Functions

ESC [Keyboard enable
ESC] Keyboard disable
ESC v Wrap-around at end of line

ESC w Discard at end of line
ESC c Key click
ESC z Reset to power-up configuration

The detailed definition of these escape codes begins on page 10.42 of TM-100 Technical Manual.

The code for procedure ESCAPE, shown in Listing 1, is fairly straightforward. First, the escape character (ordinal value 27) is sent via DOSXQQ function to Z-DOS followed by the character COMMAND supplied as a parameter by the user. After this, Z-DOS performs the selected escape function. Note, calling procedure ESCAPE with a COMMAND value of 'E' has the same result as calling procedure CLEAR.

Procedure TERMINAL__TYPE [ESC i 0 BANKS VRAM]

This procedure may be used to determine the number of banks of screen memory that are present and the size of the screen ram. If three banks of screen memory are present, you can assume a color monitor is being used. The TERMINAL__TYPE procedure issues the escape sequence 'ESC i 0' and Z-DOS will respond with the following:

ESC i E BANKS VRAM

Where BANKS equals:
1 = one bank of VRAM
3 = three banks of VRAM

Where VRAM equals:
A = 32k byte VRAM
B = 64k byte VRAM

The TERMINAL__TYPE procedure is used on line 52 of Listing 2 to determine if a color monitor is being used.

The TERMINAL__TYPE procedure performs similarly to the POSITION procedure. First, the escape sequence 'ESC i 0' is issued to Z-DOS via the DOSXQQ function. In response, Z-DOS issues the sequence 'ESC i E BANKS VRAM'. TERMINAL__TYPE procedure ignores the values 'ESC i E' returned and converts the BANKS and VRAM values into characters, returning them to the user as parameter variables.

Function VT52 [ESC Z]

The VT52 function is used to determine if your monitor may function as a VT52. VT52 is a Boolean function and could be used as follows: IF VT52 THEN tube:= 'VT52' ELSE tube:= 'not VT52'. Function VT52 performs as follows: First, the escape sequence 'ESC Z' is issued to Z-DOS via the DOSXQQ function. Then if the monitor can perform as a VT52, the sequence 'ESC / K' is issued by Z-DOS. The DOSXQQ function is used to return this sequence. It is then tested to see if a '/' K' was returned. If so, VT52 is set to true; otherwise, it is set to false.

Procedure COLOR [ESC m FORE BACK]

This procedure is used to change the foreground and background colors of the monitor. There are eight colors available on the Z-100. Lines 56 and 60 of Listing 2 illustrate the use of the COLOR procedure. Colors are identified by ASCII characters '0' through '7' as defined below:

Where FORE and BACK equal:
0 = Black
1 = Blue
2 = Red
3 = Magenta

- 4 = Green
- 5 = Cyan
- 6 = Yellow
- 7 = White

For example, if you wish to have blue characters on a red background, use the statement: COLOR('1','2').

The COLOR procedure works by issuing to Z-DOS the escape sequence 'ESC m FORE BACK'. Of course, this is accomplished by using the DOSXQQ function, what else.

Procedure ESCAPE__x [ESC x COMMAND]

The procedure ESCAPE__x is used to perform a variety of functions based upon the value of the single parameter COMMAND. The following table is taken from the TM-100 Technical Manual. Page 10.39 lists the available COMMAND values with their respective meanings:

ESC x COMMAND Set modes

Where COMMAND equals:

- 1 = Enable 25th line
- 2 = No key click
- 4 = Block cursor
- 5 = Cursor off
- 6 = Keypad shifted
- 7 = Enter alternate keypad mode
- 8 = Auto line feed on receipt of line feed
- 9 = Auto CR on receipt of line feed
- ; = Nonblinking cursor
- < = Disable keyboard auto repeat

- ? = Enable key expansion
- @ = Enable event driven (key up/down) mode

An example of using the ESCAPE__x procedure is on line 80 of Listing 2. In this case, the cursor was changed to a block. Note, COMMAND is a character value parameter.

Procedure ESCAPE__y [ESC y COMMAND]

This procedure performs the opposite function as the ESCAPE__x procedure above. For example, line 101 of Listing 2 returns the cursor to an underline character. The Listing of available command characters and their meanings (from TM-100 Technical Manual page 10.40) are listed below:

ESC y COMMAND Reset modes

Where COMMAND equals:

- 1 = Disable 25th line
- 2 = Enable key click
- 4 = Underscore cursor
- 5 = Cursor on
- 6 = Keypad unshifted
- 7 = Exit alternate keypad mode
- 8 = No auto line feed
- 9 = No auto CR
- ; = Blinking cursor
- < = Enable keyboard auto repeat
- ? = Disable key expansion
- @ = Disable event driven (key up/down) mode

Well, that's that. Type these listings in and see how they work.

```

1: {
2: {
3: { * LISTING 1 *
4: {
5: {
6:
7: function DOSXQQ(COMMAND,PARAMETER:word):byte;extern;
8: {Reference: Pascal Compiler User's Guide, page 119}
9:
10: {Escape Code References: Technical Manual (TM-100) pages 10-38, 10-51}
11:
12: procedure CLEAR; {ESC E - clears the screen}
13:
14: var x: byte;
15:
16: begin
17: x:=DOSXQQ(2,27); {ESC }
18: x:=DOSXQQ(2,wrld('E')); { E }
19: end;
20:
21: procedure LOCATE(ROW, COL: integer);
22: { Moves the cursor to location ROW and COL on the screen }
23: {ESC Y ROW COL - Direct Cursor Addressing}
24:
25: var x: byte;
26:
27: begin
28: x:=DOSXQQ(2,27); {ESC }
29: x:=DOSXQQ(2,wrld('Y')); { Y }
30: x:=DOSXQQ(2,31+wrld(ROW)); { ROW }
31: x:=DOSXQQ(2,31+wrld(COL)); { COL }
32: end;
33:
34: procedure POSITION (var ROW, COL: integer);
35: { stores the cursor position in ROW and COL. }
36: {ESC n - returns ESC Y ROW COL}
37:
38: var x: byte;
39:
40: begin
41: x:=DOSXQQ(2,27); {ESC }
42: x:=DOSXQQ(2,wrld('n')); { n }
43: x:=DOSXQQ(6,255); {ESC }
44: x:=DOSXQQ(6,255); { Y }
45: x:=DOSXQQ(6,255); { ROW }
46: ROW := ord(x) - 31;
47: x:=DOSXQQ(6,255);
48: COL := ord(x) - 31;
49: end;
50:
51: procedure ESCAPE (COMMAND: char);
52: {ESC character - for use with any of the escape sequences}
53: {that don't return a value}
54:
55: var x: byte;
56:
57: begin
58: x:=DOSXQQ(2,27);
59: x:=DOSXQQ(2,wrld(COMMAND))

```

```

60: end;
61:
62: procedure TERMINAL_TYPE (var BANKS, VRAM: char);
63: {ESC i 0 - Identify Zenith Terminal Type}
64:
65: var x: byte;
66:
67: begin
68:   x:=DOSXQQ(2,27); {ESC }
69:   x:=DOSXQQ(2,wrdd('i')); { i }
70:   x:=DOSXQQ(2,wrdd('0')); { 0 }
71:   x := DOSXQQ(6,255); x := DOSXQQ(6,255); x := DOSXQQ(6,255); {ESC i E }
72:   {BANKS and VRAM are computed next}
73:   BANKS :=chr(ord(DOSXQQ(6,255))); VRAM := chr(ord(DOSXQQ(6,255)));
74: end;
75:
76: function VT52: boolean;
77:
78: var x: byte;
79:   a,b: byte;
80:
81: begin
82:   VT52 := false;
83:   x := DOSXQQ(2,27); x := DOSXQQ(2,wrdd('Z')); {ESC Z }
84:   x := DOSXQQ(6,255); {skip past the escape} {ESC }
85:   a := DOSXQQ(6,255); b := DOSXQQ(6,255); { ? ? }
86:   { If the characters / and K are returned it's a VT52 }
87:   if (a=wrdd('/') and (b=wrdd('K'))) then VT52 := true;
88: end;
89:
90: procedure COLOR(FORE,BACK: char);
91: { Change the foreground and background colors. }
92: { ESC m FORE BACK }
93:
94: var x : byte;
95:
96: begin
97:   x := DOSXQQ(2,27); x := DOSXQQ(2,wrdd('m')); {ESC m }
98:   x := DOSXQQ(2,wrdd(FORE)); x := DOSXQQ(2,wrdd(BACK)); {FORE BACK }
99: end;
100:
101: procedure ESCAPE_x(COMMAND: char);
102: { ESC x COMMAND - Set Modes }
103:
104: var x: byte;
105:
106: begin
107:   {ESC x COMMAND }
108:   x:=DOSXQQ(2,27); x:=DOSXQQ(2,wrdd('x')); x:=DOSXQQ(2,wrdd(COMMAND))
109: end;
110:
111: procedure ESCAPE_y(COMMAND: char);
112: { ESC y - Reset Modes }
113:
114: var x: byte;
115:
116: begin
117:   {ESC y COMMAND }
118:   x:=DOSXQQ(2,27); x:=DOSXQQ(2,wrdd('y')); x:=DOSXQQ(2,wrdd(COMMAND))
119: end;
1:
2:
3:
4:
5:
6:
7: program ESCAPE_CODE_EXAMPLES (input, output);
8:
9: { This program demonstrates the use of the escape code procedures }
10: { defined in Listing 1. First, a form to input name and address }
11: { is printed on the screen. Then, the user is prompted for input }
12: { and finally, the input entered is printed on the screen. Wow! }
13:
14: const
15:   black = '0'; { Define the colors with meaningful names. }
16:   blue = '1'; { These constants will be used with }
17:   red = '2'; { COLOR procedure. }
18:   magenta = '3';
19:   green = '4';
20:   cyan = '5';
21:   yellow = '6';
22:   white = '7';
23: type
24:   axis = (x, y); {Axis and entity are used as}
25:   entity = (name, street, city, state, zip); {indexes to the pos array }
26:
27: var
28:   BANKS, VRAM: char; { Used by the TERMINAL_TYPE procedure}
29:   { The pos array contains the position of the cursor to prompt the user}
30:   { for input.
31:   pos: array [entity, axis] of integer;
32:   color_monitor: boolean; {Set true if color monitor in use}
33:   name_data, street_data: lstring(30); {Lstring variables are used }
34:   city_data: lstring(20); {to input name and address. }
35:   state_data: lstring(2); { }
36:   zip_data: lstring(5); { }
37:
38: {$include 'escape.pas'} {Obtain the escape procedures in Listing 1}
39:
40: procedure DASHES (count: integer);
41: { DASHES prints a line of '_' COUNT long }
42:
43: var i: integer;
44:
45: begin
46:   for i:=1 to count do write('_');
47: end;
48:
49: begin
50:
51: {Determine if a color monitor is being used.}
52:   TERMINAL_TYPE(BANKS, VRAM);
53:   if BANKS='3' then color_monitor:=true else color_monitor:=false;
54:
55: {Change the color and clear the screen.}
56:   if color_monitor then COLOR(white, red);
57:   CLEAR;
58:   LOCATE (2, 30); write('ESCAPE CODE EXAMPLES');
59:

```

```

60: if color_monitor then COLOR(blue, red);
61: LOCATE(4, 23); write('This program merely reads a name');
62: LOCATE(5, 23); write('and address and then prints them. ');
63:
64: { Print the form on the upper portion of the screen }
65: if color_monitor then COLOR(yellow, red);
66:
67: { Save the position of the cursor after printing NAME: }
68: { This pos will be used later to prompt the user for input. }
69: LOCATE(7, 20); write('NAME: ');
70: POSITION(pos[name, y], pos[name, x]); DASHES(30);
71: LOCATE(9, 20); write('STREET: ');
72: POSITION(pos[street, y], pos[street, x]); DASHES(30);
73: LOCATE(11, 20); write('CITY: ');
74: POSITION(pos[city, y], pos[city, x]); DASHES(20);
75: write('STATE: '); POSITION(pos[state, y], pos[state, x]); DASHES(2);
76: LOCATE(13, 20); write('ZIP: ');
77: POSITION(pos[zip, y], pos[zip, x]); DASHES(5);
78:
79: {Change to block cursor.}
80: ESCAPE_x('4');
81:
82: { Prompt the user for input. }
83: LOCATE(pos[name, y], pos[name, x]); readln(name_data);
84: LOCATE(pos[street, y], pos[street, x]); readln(street_data);
85: LOCATE(pos[city, y], pos[city, x]); readln(city_data);
86: LOCATE(pos[state, y], pos[state, x]); readln(state_data);
87: LOCATE(pos[zip, y], pos[zip, x]); readln(zip_data);
88:
89: {Change the color of the bottom of the screen.}
90: LOCATE(16, 1); if color_monitor then COLOR(white, blue);
91: ESCAPE('J');
92:
93: { Print out the user's input. }
94: LOCATE(17, 20); write(name_data);
95: LOCATE(18, 20); write(street_data);
96: LOCATE(19, 20); write(city_data);
97: LOCATE(19, 20+ord(city_data.len)); write(' ', state_data);
98: LOCATE(20, 20); write(zip_data);
99:
100: {Return the cursor to a underline.}
101: ESCAPE_y('4')
102: end.

```



NEW for your Z-100 Series Computer, the **INFORMER** is always in memory ready to perform its various functions at your beck and call. Just press a function key and you have:

- * Appointment Scheduler and Calendar with automatic insertion of repetitive events like birthdays and anniversaries
- * Address Book with phone numbers
- * Scratch Pad
- * Calculator
- * Alarm Clocks (2)
- * Automatic Phone Dialer
- * ASCII Conversion Table
- * English/Metric Conversion Table
- * Up to 32 Special Tables and Charts

When you are finished with the **INFORMER**, return to whatever program you interrupted and pick up where you left off.

You can also print any of the **INFORMER'S** screens as well as labels from your Address Book.

The **INFORMER** IS ONLY \$69.95. **ORDER TODAY**

Requires: MSDOS version 2 or higher
64-K Video RAM Parts

The INFORMER

SUNFLOWER SOFTWARE, INC. (913) 631-1333
13915 Midland Drive, Dept. R5, Shawnee, KS 66216

SUNFLOWER SOFTWARE, INC.
13915 Midland Drive, Dept. R5
Shawnee, KS 66216 (913) 631-1333

- Please, send me a free catalog
 H8 H/Z89 H/Z100

NAME _____
ADDRESS _____ APT _____
CITY _____ STATE _____ ZIP _____

- VISA ACCT # _____
 M/C EXP DATE _____

ITEM	FORMAT	PRICE

KANSAS 4% TAX _____
Shipping/Handling 2.00
Amount Enclosed _____

Making Z-100 WordStar Faster (Part 2)

Pat Swayne
HUG Software Engineer

In last February's issue of REMark, I presented an article containing patches to make the screen output (scrolling, etc.) of the Z-100 version of WordStar faster. Since then I have learned that my WordStar version 3.3 is not the release that is being shipped now. If your version signs on with a "Restricted Rights Legend" when you first start it, the patch on page 32 of the first article will not work with your version. There is only one change for the newer version. The patch at address 55C7 must instead be made at address 55E6. Below is the complete patch with the change noted. The patch should be made using DEBUG.

```
-NWS.COM
-L
-E55E6
xxxx:55E6 50.C3      (This address was 55C7)
-E2BA
xxxx:02BA 00.EB 90.24 C3.C3 00.EB 90.2E C3.C3
xxxx:02C0 00.EB 90.31
-E2E0
xxxx:02E0 00.9A 00.03 00.00 00.40 00.00 00.75 00.03 00.B0
xxxx:02E8 00.00 00.C3 00.B0 00.FF 00.C3 00.9A 00.06 00.00
xxxx:02F0 00.40 00.00 00.C3 00.9C 00.FC 00.9A 00.19 00.00
xxxx:02F8 00.01 00.FE 00.9D 00.C3
-W
```

As mentioned in the article, this patch disables the built-in function key processing in WordStar, so you must use KEYMAP (HUG part no. 885-3010-37) if you want the function and keypad keys to work after the patch. If you already made the patch from the first article, start over with an unpatch copy and redo the patch. And, as I said in the first article, DO NOT make the patch if the old values at each address are not what is shown here. The only exceptions to this rule are the patches at addresses 292 and 29B from the first article. The old numbers will probably be zeros for your version of WordStar, instead of what is shown.

Some customers have asked me if there is a way to shorten the time that the "Restricted Rights Legend" message appears on the screen when word starts. You can shorten the time to nearly nothing with this patch:

```
-E558F
xxxx:558F 09.01
-E5593
xxxx:5593 FF.01 FF.00
-W
```

There is an additional patch you should make to this version of WordStar if you are going to use it with KEYMAP. It prevents WordStar from erasing the 25th line or altering the keypad key responses.

```
-E587F
xxxx:587F 7A.45
xxxx:5880 1B.24
```

Making WordStar Print Faster

One customer discovered that WordStar prints slower when run under MS-DOS version 2 than when it is run under Z-DOS. The following patch will make it print as fast as possible under either operating system. This patch is valid for all versions of WordStar released for the Z-100 by Heath/Zenith.

```
-E7DD
xxxx:07DD 8A.9A D0.0C B4.00
xxxx:07E0 05.40 CD.00 21.90
-W
```

If you use WordStar's simultaneous printing feature, you may want to use the following patch instead of the one above. It gives WordStar the ability to check the printer's status, and makes simultaneous printing a little faster. Before you make this patch, you should run Install and make sure the WordStar printer options are set the way you want them. If you make any changes, write them back to WordStar and then restart Install. Select "Specify Printer Name" from the main menu, and select "User Defined Printer" from the printer menu (even if you previously selected another printer). Select option E, Drivers, from the user defined printer menu, and select option C, User Supplied Subroutine, from the driver menu. Then press HOME twice to get back to the main menu, and make the changes to WordStar. Now, you can make the following patch with DEBUG:

```
-E7F0
xxxx:07F0 E4.B8 05.00 90.02 24.9A 01.4B 34.00 01.40 F9.00
xxxx:07F8 74.80 01.E4 C3.80 F8.75 C3.01 E6.F9 04.C3 90.9A
xxxx:0800 C3.0C E4.00 05.40 90.00 24.C3
-E811
xxxx:0811 90.EB F8.DD C3.C3 90.EB 90.E9
-W
```

If you do not know how to use DEBUG to make patches, refer to the first WordStar article in the February issue, or to your MS-DOS manual.



```

LastR:      Real:      { When fixed circle is this small, then stop }
Changer:    Real:      { Diff between fixed circle radii of cycloids }
StartAng:   Real:      { Radians to rotate the cycloid from x axis }
Begin
{ Obtain parameters for the cycloid from the user. }
WriteLn (' Ratio of Fixed to Rotating Circles in the pattern. ');
Write (' Express as a ratio of integers (Default is 14 5)> ');
Fint := 14;
Mint := 5;
ReadLn (Fint,Mint);
Ratio := Fint/Mint;
Write (' Number of Patterns (Default is 1)? ');
Patterns := 1;
ReadLn (Patterns);
Rotation := 0.0;
If Patterns > 1 Then Begin
  Write (' Degrees each pattern should be rotated ');
  Write (' from the previous pattern? ');
  ReadLn (Rotation);
End;
Write (' Starting radius of the fixed circle (Default is 0.35)? ');
FixedR := 0.35;
ReadLn (FixedR);
LastR := FixedR;
If Patterns > 1 Then Begin
  Write (' Ending radius of the fixed circle (Default is ',FixedR,')? ');
  ReadLn (LastR);
End;
ClrScr; { Clear the screen }
StartAng := 0.0;
Changer := 1.0;
If Patterns > 1 Then Changer := (FixedR - LastR)/(Patterns-1);
Repeat
{ Generate the hypocycloid making up the current pattern. }
Hypocycloid (FixedR,FixedR/Ratio,10/180*Pi,StartAng);
{ Calculate the radius of the fixed circle for the next pattern. }
FixedR := FixedR - Changer;
{ Calculate the angle that the hypocycloid is rotated from the X axis. }
StartAng := StartAng + Rotation/180*Pi;
Until ((FixedR < LastR) Or KeyPressed);
End;
Begin
{ Initialize the segment offsets for video memory on the Z100. }
VOff[0] := BlueSeg; VOff[1] := RedSeg; VOff[2] := GreenSeg;
Repeat
MultiPattern: { Do a plot }
If KeyPressed Then ReadLn: { Clear keyboard buffer }
Write ('Do another?');
ReadLn (Ans);
Ans := UpCase(Ans);
Until Ans <> 'Y';
End.

```



MPI Sprint Printer

Dear HUG:

I was impressed with your write up of the MPI Sprint Printer, which I recently purchased complete with the AP-PAK programs. In addition, I acquired the Z89-11 Multiple I/O board.

When I connected the cable and board, performed the CPM configure operation, I expected the printer to start printing ... WRONG!!

Investigation revealed that the cable P/N 134-1186 is definitely not wired correctly at the DB-25 end. Here are the steps to reconfigure the cable end and get you up and running.

- A. Remove and discard the 2 nylon pin plugs from the DB-25 connector.
- B. Identify pin (1) of P605 (end of -1186 that plugs into the I/O board — white wire).
- C. Using a pin removal tool for the DB-25 connector, relocate DB-25 pins 1 thru 11 in the connector so as to be the same as P-605. (Pin 1 to pin 1, etc.)
- D. Place wire from P-605-13 in unused hole.
- E. Place wire from P-605-15 in DB-25 pin 15.
- F. Insure connections are correct.
- G. Reconfigure for parallel printer, high signal polarity and it is ready to run.

While on the subject of the Sprint, the font program, style 64 will not work with Software Toolworks' text. The program (text) does not recognize the .&& commands. Is there any assistance you can

give me or advise me whom to contact.

Sincerely,

Frank Houle
 302 Blairmore Blvd.
 Orange Park, FL 32073

Improvements For Blackjack

Dear HUG:

Here's an improvement on the "Blackjack" program given in Section 7 of EC1110, Microsoft BASIC Education Course.

Every time you play the game, as written, the hands (or cards) follow the same sequence which you eventually memorize. This is because the RND(X) function will create the same sequence of "pseudo" random numbers each time it runs unless the "seed" number X is different.

Add the RANDOMIZE function as given on page 7-8 of the BASIC-80 (CP/M) software (Microsoft BASIC) manual.

Put the following line in the Blackjack program:

```
10015 RANDOMIZE PEEK(11)
```

This reseeds the program every time it runs, so you get a different sequence. The chance of reseeding with the same number is 1 out of 256! Also, whenever the program reshuffles the deck, the Random Generator is reseeded. Memory cell 11 changes numbers from 0 to 255 and then repeats continually. One cycle takes less than a second.

I use an H-89A computer with 64K RAM and CP/M Microsoft

BASIC Interpreter. I really get a lot out of every issue of REMark. Have cancelled all other magazines!! Keep up the good work.

Yours very truly,

Milton P. Klintworth
2990 Shawnee Lane
Drayton Plains, MI 48020

Biorhythm Modification

Dear HUG:

In reference to David Warnick's article on "Standards For Terminal Control" (Issue 46, again). I offer a convenience modification to his Biorhythm program. As currently written, his program is fixed at a 90-day chart. By adding one line and revising two others, selection of any length chart is allowed.

Add line 990:

```
990 INPUT;  
    " Hello! Type in the number of days for the chart";N
```

Revise lines 5000 and 6000:

```
5000 FOR X=1 TO N          'FOR A SELECTED CHART LENGTH  
6000 NEXT X              'BACK TO LINE 5000 TIL N-DAY CHART  
                          IS COMPLETE
```

Of course, the remark in line 6000 has no bearing on the program, but it was changed just to follow through with the revision. Hope some of the HUGgies will appreciate this little mod, although I'm sure many have already experimented with it and have come up with this, or many other innovations.

Regards,

E.F. (Frank) Van Nostrand
2415 Perry Avenue
Edgewood, MD 21040

Needs Assistance For Occupational Therapy

Dear HUG:

Our Occupational Therapy Department is researching the use of a computer as a treatment modality. We service acute and day hospital psychiatric populations, inpatient and outpatient physically disabled persons, including a 36 bed rehabilitation unit and pediatrics.

We would appreciate any information you could share with us regarding the use of a computer with these populations. Areas of interest might be word processing, cognitive retraining, vocational training, problem solving and perceptual motor skills or any other areas.

Any information or references pertaining to program guidelines, considerations in selecting a computer for patient treatment and descriptive information on software would be most appreciated.

Thank you for your assistance.

Sincerely,

Jennifer Lubarsky
Lutheran General Hospital
Occupational Therapy Department
1775 W. Dempster Street
Park Ridge, IL 60068

An Omission From October 1984

Dear HUG,

Although we were pleased to discover you published our letter in the October '84 issue of REMark (p. 5), we were dismayed that you decided to omit the table we furnished with the letter. This is because much of the information listed in the table was not given in the letter. Information that we think is of value to HUGgies, such as which languages and compilers we tried and actual program run times. We were first aware that the table was not published in REMark when we got a telephone call from a guy who was interested in the BASIC program run time. We think this is only the first inquiry, since benchmark programs are of great interest to computerists. This is demonstrated by the number of letters and articles in BYTE about benchmark programs.

An additional problem is that we appear irresponsible for not reporting which languages were used and the actual results.

So, we ask that you reconsider your decision and publish the table in a future REMark. A copy of the table is enclosed.

Sincerely,

Richard and Janet Hirsch
470 Belleview
Webster Groves, MO 63119

Prime Number Program Run on a Z-100

Language	Version	Code Size (Bytes)	Execution Time (sec)	
			Normal Clock Speed:	Fast
Assembler	MASM, V1.07 c. 81,82	17280	8	5
C	Optm. CI-C86, V2.10A, c. 84	13056	12.5	8
C	CI-C86, V1.33, c. 81,82,83	13414	16	10.5
Pascal	Turbo, V2.00B, c. 83,84	9602	16	10.5
Pascal	IBM, V1.00, c. 81	33024	80	52.5
ZBASIC	Compiled, V5.4, c. 82,83	20736	14	9.5
ZBASIC	Interpreted, Rev 1.0, c. 82	[512]	2128	1395

Fortran Using Line Numbers

Dear HUG,

In the October '84 issue of REMark, Ray Battalora complained that Heath/Zenith's Fortran compiler reports errors using line numbers which don't necessarily relate to the physical line numbers in the source file. He asked if there is any way to get a line numbered listing of a Fortran program for debugging purposes. The answer is yes; the Fortran compiler always gives the user the option of obtaining a numbered listing during the first pass of compilation. This file is the third file prompted for (or the third file specified on the command line invocation) and has the default extension ".LST". If you want a "hard copy," you can simply specify PRN: for this file. The default is NUL: (i.e. the listing file is discarded).

Nearly all Fortran compilers work this way. Admittedly, this approach can be a pain, since the only way to be guaranteed of having a correct line numbered listing is to generate a new one every time the program is changed and recompiled.

Fortran compilers tend to be implemented this way for historical reasons. Fortran was originally designed for use on large main-frame computers with high speed line printers. The primary

means of obtaining a listing of your program was often to read in the deck of punched cards containing your Fortran source code and appropriate Job Control Language for invoking the compiler. You would then wait an hour or two and come back and pick up a nice thick listing, complete with line numbers, errors, a cross reference, and the amount billed to your account. It may come as a surprise to some HUGgies, but a large amount of data processing is still carried on this way today!

Sincerely,

Glenn F. Roberts, Ph.D.
12048 Greywing Square, #C-3
Reston, VA 22091

Problems With ZBASIC Graphic Programs

Dear HUG:

We recently received the MSDOS 2.13 release. It is a very nice upgrade to MSDOS and seems to work pretty well. Unfortunately, we experienced some problems with ZBASIC graphic programs that we did not have with the ZDOS 1.1 release.

The problem is that after using the ESCAPE z instruction which is sometimes used to reset the terminal, it is no more possible to execute any graphic instruction. It seems that the Video RAM is not enabled. To illustrate the bug, I enclosed a sample program

which runs perfectly under ZDOS 1.0 and does not draw any line after the first ESCAPE z instruction under MSDOS 2.13.

This problem is solved by enabling again video RAM after an ESCAPE z with the instruction:

```
OUT $HDB,&H7B
```

followed by PRINT CHR\$(27)+"x1" to enable also the 25th line.

```
10 CLS
20 FOR I=10 TO 100 STEP 2
30 FOR J=100 TO 600 STEP 10
40 LINE(I,I)-(J,J)
50 NEXT J
60 PRINT CHR$(27)+"z"
70 NEXT I
```

Best regards from the GUFIH.

Dr. Bernard Pidoux
37, bd Saint-Jacques
75014 Paris



Vectorized from Page 41

Part Number	Description of Product	Selling Price	Vol. Issue	Part Number	Description of Product	Selling Price	Vol. Issue	Part Number	Description of Product	Selling Price	Vol. Issue
AMATEUR RADIO				CP/M				MISCELLANEOUS			
HDOS				885-1207-[37]	CP/M TERM & HTOC	20.00	26	885-0004	HUG Binder	5.75	
885-8016	Morse Code Transceiver Ver 2.0	20.00	42	885-1224-[37]	CP/M MicroNET Connection	16.00	37	885-1221-[37]	Watzman ROM Source Code/Doc	30.00	33
CP/M				885-3003-[37]	CP/M ZTERM (Z100 Modem Pkg)	20.00	34	885-4001	REMark Vol. I Issues 1-13	20.00	
885-1214-[37]	CP/M MBASIC Log Book (64k)	30.00	23	885-5004-37	CP/M 86 TERM06 and DSKED	20.00	56	885-4002	REMark Vol. II Issues 14-23	20.00	
885-1234-[37]	CP/M Ham Help	20.00	49	885-5005-37	CP/M 86 16 Bit MicroNET Conn.	16.00	61	885-4003	REMark Vol. III Issues 24-35	20.00	
885-1238-[37]	CP/M ASCIRITY	20.00	57	885-5006-37	CP/M 86 HUGPBBS	40.00	62	885-4004	REMark Vol. IV Issues 36-47	20.00	
885-8020-[37]	CP/M RF Comp. Aided Design	30.00	44	885-5007-37	CP/M 86 HUGPBBS Source List	60.00	62	885-4005	REMark Vol. V Issues 48-59	25.00	
885-8031-[37]	CP/M Morse Code Transceiver	20.00	57	885-8005	MAPLE (Modem Appl. Effector)	35.00	29	885-4500	HUG Software Catalog	9.75	
				885-8012-[37]	CP/M MAPLE (Modem Program)	35.00	34	885-4600	Watzman/HUG ROM	45.00	41
				885-8023-37	CP/M 85 MAPLE	35.00	45	885-4700	HUG Bulletin Board Handbook	5.00	50
				ZDOS				885-3015-37	ZDOS SKYVIEWS	20.00	55
				885-3019-37	ZDOS 16 Bit MicroNET Connect.	16.00	61				
COMMUNICATION											
HDOS											
885-1122-[37]	HDOS MicroNET Connection	16.00	37								

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

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 _____

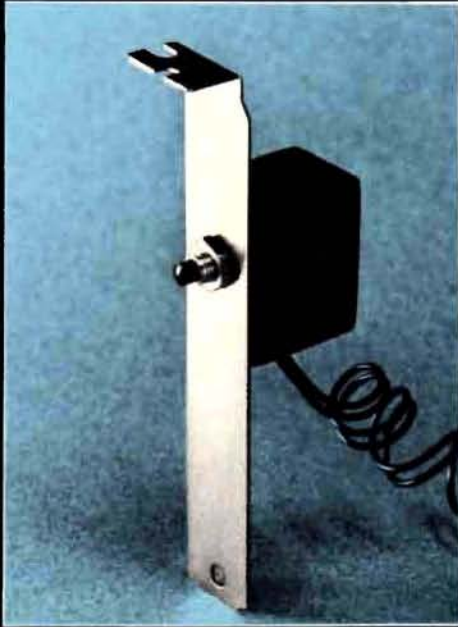
Zip _____

REMEMBER - ENCLOSE CHECK OR MONEY ORDER

CHECK THE APPROPRIATE BOX AND RETURN TO HUG

	NEW MEMBERSHIP RATES	RENEWAL RATES
U.S. DOMESTIC	\$20 <input type="checkbox"/>	\$17 <input type="checkbox"/>
FPO/APO & ALL OTHERS*	\$35 <input type="checkbox"/>	\$30 <input type="checkbox"/> U.S. FUNDS

* Membership in France and Belgium is acquired through the local distributor at the prevailing rate.



KC COMPUTERS
8033 Sunset Blvd., Suite 819,
Los Angeles, CA 90046
(213) 858-7763

YOUR Z-150 PC'S POWER SWITCH IS **NOT** A RESET BUTTON

RESET-Z IS!

No more need to power down
your Z-150 when it hangs up!

INTRODUCTORY PRICE **\$29.95**

- Electronic system reset that safely simulates power-on reset condition.
- Eliminates wear and tear on your power supply, boards and drives.
- Prevents possible hard disk damage.
- 5 minute installation requires only a screwdriver; EASY step by step instructions require no soldering or drilling.
- Installs in rear panel and requires no slot.
- Introductory price- \$29.95 including shipping & handling.

For use on Z/H-150 and 160 computers only
Dealer inquiries invited.
California residents add \$1.95 sales tax.
Personal checks—allow 2 weeks to clear.



Hilltop Road
Saint Joseph, Michigan 49085

BULK RATE
U.S. Postage
PAID
Heath Users' Group

POSTMASTER: If undeliverable,
please do not return.

P/N 885-2064