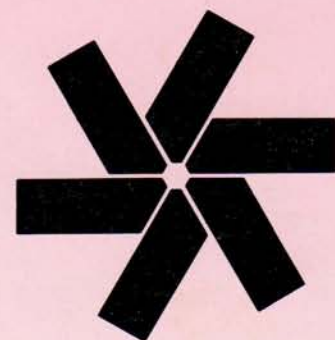


The Official **ZENITH** /Heath Computer Users Magazine

REMark®



April 1989

**Writing Applications for
Microsoft Windows
See Page 57**

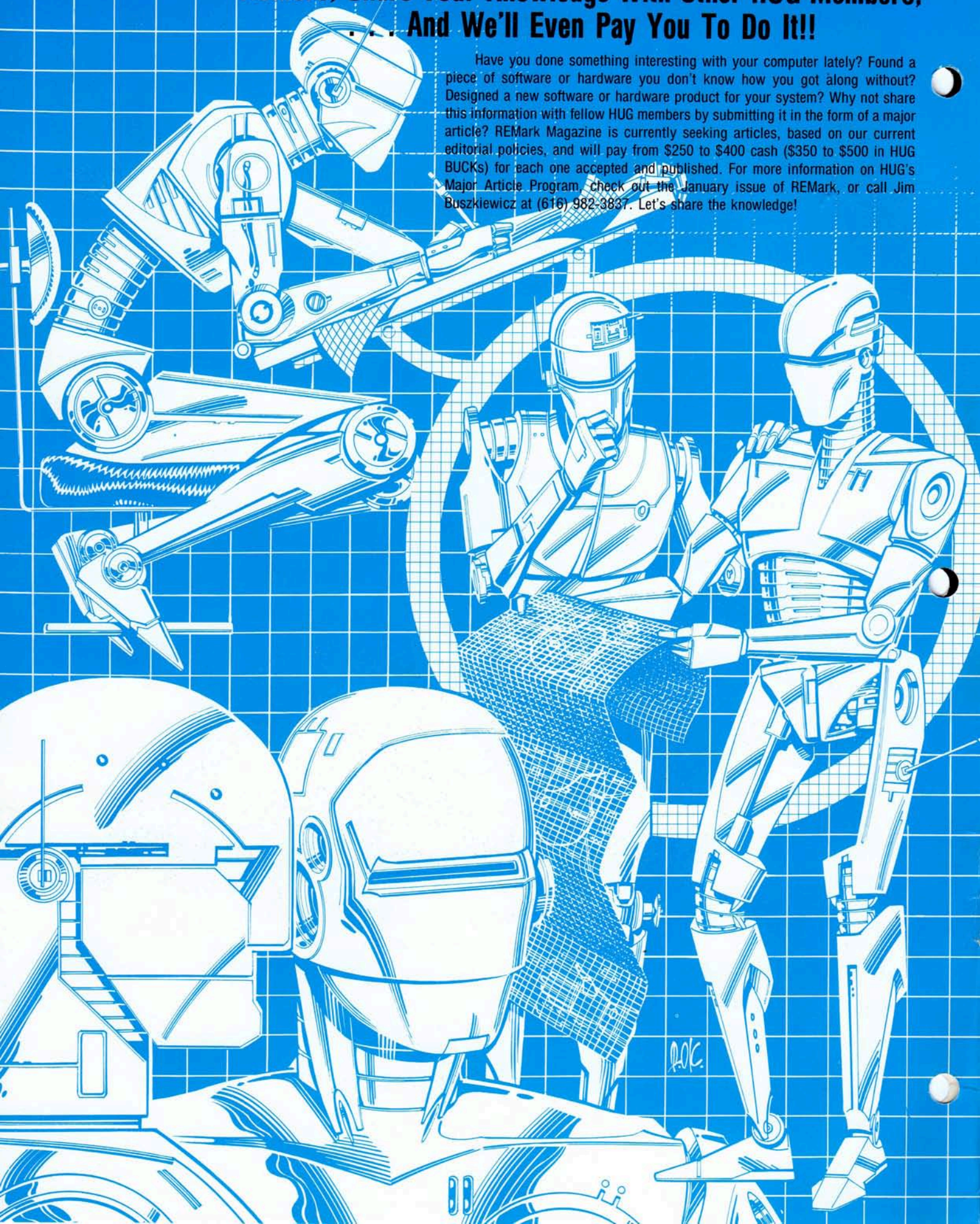


**Zork for the Z-100
See Page 33**

**Do You Need a Bigger Computer?
See Page 41**

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

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





Volume 10, Issue 4 • April 1989

On the Cover

You can have up to six floppies on your system. The CompatiCard adds 4 additional floppies to your MS-DOS system. See Page 35 for details.

General

Customize Your Programming Language

Christopher A. Feuchter 53

Reader Service No.

Page No.

196	Covox, Inc.	6
197	DMA Technologies	4
104	FBE Research Co., Inc.	32
191	First Capitol Computer	4
105	First Capitol Computer	52
184	First Capitol Computer	55
193	First Capitol Computer	64
***	Heath Company	56
137	Jay Gold Software	9
136	Lindley Systems	51
117	Payload Computer	16
130	Quikdata, Inc.	28
121	Scottie Systems	6
195	Strategic Alliance, Inc.	64
153	Surplus Trading Company	6
194	Weltec	4

H/Z-100

Z-100 Survival Kit #3 *Paul F. Herman* 29

Zork for the Z-100 *Dan Gnagey* 33

H/Z-100 and PC Compatible

Xplore Your Batch Files . . . And Xplode Some Myths

Robert Brasfield 7

ENABLE — Part 16 *George P. Elwood* 23

On the Leading Edge *William M. Adney* 41

Resources

HUG Price List	2
Buggin' HUG	5
Bug Zapping	27
Classified Ads	31
H/Z Related Products	64

PC Compatibles

All models include the following series of computers: H/Z-130, 140, 150, 160, 170, 180, H/Z-200, and 300.

PC Compatible

POWERING UP *William M. Adney* 11

Softening the Hard Disk — Part II *John A. Negus* 17

Printing With Hewlett-Packard's DeskJet

Using WordPerfect *William N. Campbell* 21

A 1.44 MB Mini-Floppy for Your

Heath/Zenith PC Compatible *Mark R. Vansickle* 35

Getting Started With . . . Works *Alan Neibauer* 37

TK Solver Plus — Part 1 *Edwin G. Wiggins* 47

A Windows Small Memory Model Template

William S. Hall 57

HUG

Managing Editor Jim Buszkiewicz
(616) 982-3837

Software Engineer Pat Swayne
(616) 982-3463

Production Coordinator Lori Lerch
(616) 982-3794

Secretary Margaret Bacon
(616) 982-3463

HUG Bulletin Board (616) 982-3956

HUG Parts Ordering (616) 982-3463

Contributing Editor William M. Adney

Advertising Ruple's Advertising Service
Dept. REM, 240 Ward Avenue
P.O. Box 348
St. Joseph, MI 49085-0348
(616) 983-4550

Printer Imperial Printing
St. Joseph, MI

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

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

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

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 Heath/Zenith Computers & Electronics Center 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/Zenith 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 © 1989, Heath/Zenith Users' Group

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM		DESCRIPTION	PRICE
H8 - H/Z-89/90					
ACCOUNTING SYSTEM	885-8047-37	CPM		BUSINESS	20.00
ACTION GAMES	885-1220-[37]	CPM		GAME	20.00
ADVENTURE	885-1010	HDOS		GAME	10.00
ASCIRITY	885-1238-[37]	CPM		AMATEUR RADIO	20.00
AUTOFILE (Z80 ONLY)	885-1110	HDOS		DBMS	30.00
BHBASIC SUPPORT PACKAGE	885-1119-[37]	HDOS		UTILITY	20.00
CASTLE	885-8032-[37]	HDOS		ENTERTAINMENT	20.00
CHEAPCALC	885-1131-[37]	HDOS		SPREADSHEET	20.00
CHECKOFF	885-8010	HDOS		CHECKBOOK SOFTWARE	25.00
DEVICE DRIVERS	885-1105	HDOS		UTILITY	20.00
DISK UTILITIES	885-1213-[37]	CPM		UTILITY	20.00
DUNGEONS & DRAGONS	885-1093-[37]	HDOS		GAME	20.00
FLOATING POINT PACKAGE	885-1063	HDOS		UTILITY	18.00
GALACTIC WARRIORS	885-8009-[37]	HDOS		GAME	20.00
GALACTIC WARRIORS	885-8009-[37]	CPM		GAME	20.00
GAMES 1	885-1029-[37]	HDOS		GAMES	18.00
HARD SECTOR SUPPORT PACKAGE	885-1121	HDOS		UTILITY	30.00
HDOS PROGRAMMERS HELPER	885-8017	HDOS		UTILITY	16.00
HOME FINANCE	885-1070	HDOS		BUSINESS	18.00
HUG DISK DUPLICATION UTILITIES	885-1217-[37]	CPM		UTILITY	20.00
HUG SOFTWARE CATALOG	885-4500	VARIOUS		PRODUCTS THRU 1982	9.75
HUGMAN & MOVIE ANIMATION	885-1124	HDOS		ENTERTAINMENT	20.00
INFO. SYSTEM AND TEL. & MAIL SYSTEM	885-1108-[37]	HDOS		DBMS	30.00
LOGBOOK	885-1107-[37]	HDOS		AMATEUR RADIO	30.00
MAGBASE	885-1249-[37]	CPM		MAGAZINE DATABASE	25.00
MAPLE	885-8005	HDOS		COMMUNICATION	35.00
MAPLE	885-8012-[37]	CPM		COMMUNICATION	35.00
MICRONET CONNECTION	885-1122-[37]	HDOS		COMMUNICATION	16.00
MISCELLANEOUS UTILITIES	885-1089-[37]	HDOS		UTILITY	20.00
MORSE CODE TRANSCIEVER	885-8016	HDOS		AMATEUR RADIO	20.00
MORSE CODE TRANSCIEVER	885-8031-[37]	CPM		AMATEUR RADIO	20.00
PAGE EDITOR	885-1079-[37]	HDOS		UTILITY	25.00
PROGRAMS FOR PRINTERS	885-1082	HDOS		UTILITY	20.00
REMARK VOL 1 ISSUES 1-13	885-4001	N/A		1978 TO DECEMBER 1980	20.00
RUNOFF	885-1025	HDOS		TEXT PROCESSOR	35.00
SCICALC	885-8027	HDOS		UTILITY	20.00
SMALL BUSINESS PACKAGE	885-1071-[37]	HDOS		BUSINESS	75.00
SMALL-C COMPILER	885-1134	HDOS		LANGUAGE	30.00
SOFT SECTOR SUPPORT PACKAGE	885-1127-[37]	HDOS		UTILITY	20.00
STUDENT'S STATISTICS PACKAGE	885-8021	HDOS		EDUCATION	20.00
SUBMIT (Z80 ONLY)	885-8006	HDOS		UTILITY	20.00
TERM & HTOC	885-1207-[37]	CPM		COMMUNICATION & UTILITY	20.00
TINY BASIC COMPILER	885-1132-[37]	HDOS		LANGUAGE	25.00
TINY PASCAL	885-1086-[37]	HDOS		LANGUAGE	20.00
UDUMP	885-8004	HDOS		UTILITY	35.00
UTILITIES	885-1212-[37]	CPM		UTILITY	20.00
UTILITIES BY PS	885-1126	HDOS		UTILITY	20.00
VARIETY PACKAGE	885-1135-[37]	HDOS		UTILITY & GAMES	20.00
WHEW UTILITIES	885-1120-[37]	HDOS		UTILITY	20.00
XMET ROBOT X-ASSEMBLER	885-1229-[37]	CPM		UTILITY	20.00
Z80 ASSEMBLER	885-1078-[37]	HDOS		UTILITY	25.00
Z80 DEBUGGING TOOL (ALDT)	885-1116	HDOS		UTILITY	20.00

H8 - H/Z-89/90 - H/Z-100 (Not PC)

ADVENTURE	885-1222-[37]	CPM		GAME	10.00
BASIC-E	885-1215-[37]	CPM		LANGUAGE	20.00
CASSINO GAMES	885-1227-[37]	CPM		GAME	20.00
CHEAPCALC	885-1233-[37]	CPM		SPREADSHEET	20.00
CHECKOFF	885-8011-[37]	CPM		CHECKBOOK SOFTWARE	25.00
COPYDOS	885-1235-37	CPM		UTILITY	20.00
DISK DUMP & EDIT UTILITY	885-1225-[37]	CPM		UTILITY	30.00
DUNGEONS & DRAGONS	885-1209-[37]	CPM		GAMES	20.00
FAST ACTION GAMES	885-1228-[37]	CPM		GAME	20.00
FUN DISK I	885-1236-[37]	CPM		GAMES	20.00
FUN DISK II	885-1248-[37]	CPM		GAMES	35.00
GAMES DISK	885-1206-[37]	CPM		GAMES	20.00
GRADE	885-8036-[37]	CPM		GRADE BOOK	20.00
HRUN	885-1223-[37]	CPM		HDOS EMULATOR	40.00
HUG FILE MANAGER & UTILITIES	885-1246-[37]	CPM		UTILITY	20.00
HUG SOFTWARE CATALOG UPDATE #1	885-4501	VARIOUS		PRODUCTS 1983 THRU 1985	9.75
KEYMAP CPM-80	885-1230-[37]	CPM		UTILITY	20.00
MBASIC PAYROLL	885-1218-[37]	CPM		BUSINESS	60.00
MICRONET CONNECTION	885-1224-[37]	CPM		COMMUNICATION	16.00
NAVPROGSEVEN	885-1219-[37]	CPM		FLIGHT UTILITY	20.00
REMARK VOL 3 ISSUES 24-35	885-4003	N/A		1982	20.00
REMARK VOL 4 ISSUES 36-47	885-4004	N/A		1983	20.00
REMARK VOL 5 ISSUES 48-59	885-4005	N/A		1984	25.00
REMARK VOL 6 ISSUES 60-71	885-4006	N/A		1985	25.00
REMARK VOL 7 ISSUES 72-83	885-4007	N/A		1986	25.00
SEA BATTLE	885-1211-[37]	CPM		GAME	20.00
UTILITIES BY PS	885-1226-[37]	CPM		UTILITY	20.00
UTILITIES	885-1237-[37]	CPM		UTILITY	20.00

Price List

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
X-REFERENCE UTILITIES FOR MBASIC	885-1231-[37]	CPM	UTILITY	20.00
ZTERM	885-3003-[37]	CPM	COMMUNICATION	20.00

H/Z-100 (Not PC) Only

ACCOUNTING SYSTEM	885-8048-37	MSDOS	BUSINESS	20.00
CALC	885-8043-37	MSDOS	UTILITY	20.00
CARDCAT	885-3021-37	MSDOS	BUSINESS	20.00
CHEAPCALC	885-3006-37	MSDOS	SPREADSHEET	20.00
CHECKBOOK MANAGER	885-3013-37	MSDOS	BUSINESS	20.00
CP/EMULATOR	885-3007-37	MSDOS	CPM EMULATOR	20.00
DBZ	885-8034-37	MSDOS	DBMS	25.00
ETCHDUMP	885-3005-37	MSDOS	UTILITY	20.00
EZPLOT II	885-3049-37	MSDOS	PRINTER PLOTTING UTILITY	25.00
GAMES CONTEST PACKAGE	885-3017-37	MSDOS	GAMES	25.00
GAMES PACKAGE II	885-3044-37	MSDOS	GAMES	25.00
GRAPHICS	885-3031-37	MSDOS	ENTERTAINMENT	20.00
HELPSCREEN	885-3039-37	MSDOS	UTILITY	20.00
HUG BACKGROUND PRINT SPOOLER	885-1247-37	CPM	UTILITY	20.00
KEYMAC	885-3046-37	MSDOS	UTILITY	20.00
KEYMAP	885-3010-37	MSDOS	UTILITY	20.00
KEYMAP CPM-85	885-1245-37	CPM	UTILITY	20.00
MAPLE	885-8023-37	CPM	COMMUNICATION	35.00
MATHFLASH	885-8030-37	MSDOS	EDUCATION	20.00
ORBITS	885-8041-37	MSDOS	EDUCATION	25.00
POKER PARTY	885-8042-37	MSDOS	ENTERTAINMENT	20.00
SCICALC	885-8028-37	MSDOS	UTILITY	20.00
SKYVIEWS	885-3015-37	MSDOS	ASTRONOMY UTILITY	20.00
SMALL-C COMPILER	885-3026-37	MSDOS	LANGUAGE	30.00
SPELLS	885-3035-37	MSDOS	SPELLING CHECKER	20.00
SPREADSHEET CONTEST PACKAGE	885-3018-37	MSDOS	VARIOUS SPREADSHEETS	25.00
TREE-ID	885-3036-37	MSDOS	TREE IDENTIFIER	20.00
USEFUL PROGRAMS I	885-3022-37	MSDOS	UTILITIES	30.00
UTILITIES	885-3008-37	MSDOS	UTILITY	20.00
ZBASIC DUNGEONS & DRAGONS	885-3009-37	MSDOS	GAME	20.00
ZBASIC GRAPHIC GAMES	885-3004-37	MSDOS	GAMES	20.00
ZBASIC GAMES	885-3011-37	MSDOS	GAMES	20.00
ZPC II	885-3037-37	MSDOS	PC EMULATOR	60.00
ZPC UPGRADE DISK	885-3042-37	MSDOS	UTILITY	20.00

H/Z-100 and PC Compatibles

ADVENTURE	885-3016	MSDOS	GAME	10.00
ASSEMBLY LANGUAGE UTILITIES	885-8046	MSDOS	UTILITY	20.00
BOTH SIDES PRINTER UTILITY	885-3048	MSDOS	UTILITY	20.00
CXREF	885-3051	MSDOS	UTILITY	17.00
DEBUG SUPPORT UTILITIES	885-3038	MSDOS	UTILITY	20.00
DPATH	885-8039	MSDOS	UTILITY	20.00
HADES	885-3040	MSDOS	UTILITY	40.00
HELP	885-8040	MSDOS	CAI	25.00
HEPCAT	885-3045	MSDOS	UTILITY	35.00
HUG BACKGROUND PRINT SPOOLER	885-3029	MSDOS	UTILITY	20.00
HUG EDITOR	885-3012	MSDOS	TEXT PROCESSOR	20.00
HUG MENU SYSTEM	885-3020	MSDOS	UTILITY	20.00
HUG SOFTWARE CATALOG UPDATE #1	885-4501	VARIOUS	PROD 1983 THRU 1985	9.75
HUGMCP	885-3033	MSDOS	COMMUNICATION	40.00
HUGPBBS SOURCE LISTING	885-3028	MSDOS	COMMUNICATION	60.00
HUGPBBS	885-3027	MSDOS	COMMUNICATION	40.00
ICT 8080 TO 8088 TRANSLATOR	885-3024	MSDOS	UTILITY	20.00
MAGBASE	885-3050	VARIOUS	MAGAZINE DATABASE	25.00
MATT	885-8045	MSDOS	MATRIX UTILITY	20.00
MISCELLANEOUS UTILITIES	885-3025	MSDOS	UTILITIES	20.00
PS's PC & Z100 UTILITIES	885-3052	MSDOS	UTILITY	20.00
REMARK VOL 5 ISSUES 48-59	885-4005	N/A	1984	25.00
REMARK VOL 6 ISSUES 60-71	885-4006	N/A	1985	25.00
REMARK VOL 7 ISSUES 72-83	885-4007	N/A	1986	25.00
REMARK VOL 8 ISSUES 84-95	885-4008	N/A	1987	25.00
SCREEN DUMP	885-3043	MSDOS	UTILITY	30.00
UTILITIES II	885-3014	MSDOS	UTILITY	20.00
Z100 WORDSTAR CONNECTION	885-3047	MSDOS	UTILITY	20.00

PC Compatibles

ACCOUNTING SYSTEM	885-8049	MSDOS	BUSINESS	20.00
CARDCAT	885-6006	MSDOS	CATALOGING SYSTEM	20.00
CHEAPCALC	885-6004	MSDOS	SPREADSHEET	20.00
CP/EMULATOR II & ZEMULATOR	885-6002	MSDOS	CPM & Z100 EMULATORS	20.00
DUNGEONS & DRAGONS	885-6007	MSDOS	GAME	20.00
EZPLOT II	885-6013	MSDOS	PRINTER PLOTTING UTILITY	25.00
GRADE	885-8037	MSDOS	GRADE BOOK	20.00
HAM HELP	885-6010	MSDOS	AMATEUR RADIO	20.00
KEYMAP	885-6001	MSDOS	UTILITY	20.00
PS's PC UTILITIES	885-6011	MSDOS	UTILITIES	20.00
POWERING UP	885-4604	N/A	GUIDE TO USING PCS	12.00
SCREEN SAVER PLUS	885-6009	MSDOS	UTILITIES	20.00
SKYVIEWS	885-6005	MSDOS	ASTRONOMY UTILITY	20.00
TCSPELL	885-8044	MSDOS	SPELLING CHECKER	20.00
ULTRA RTTY	885-6012	MSDOS	AMATEUR RADIO	20.00

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

Magazines everywhere, and no way to reference the wealth of information they hold? Not anymore! Now there's **MAGBASE**; a database designed specifically for referencing magazine articles. Don't let those one-hundred-and-some back issues of REMark, or C Users Journal, or Veterinary Medicine, (or any magazine) gather dust, use **MAGBASE**, and find that article you read two years ago! **MAGBASE** is available for **MSDOS HUG P/N 885-3050** or **CP/M (P/N 885-1249-[27])**.

LAPTOP OWNERS . . . don't feel left out! All of HUG's MSDOS software is available on 3-1/2" micro-floppies too! When ordering, just add a "-80" to the 7-digit HUG part number. For the standard 5-1/4" floppy, just add a "-37".

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

ORDERING INFORMATION

For VISA and MasterCard phone orders, telephone the Heath Users' Group directly at (616) 982-3463. Have the part number(s), descriptions, and quantity ready for quick processing. By mail, send your order, plus 10% postage and handling (\$1.00 minimum charge, up to a maximum of \$5.00) to: Heath Users' Group, P.O. Box 217, Benton Harbor, MI 49022-0217. VISA and MasterCard require minimum \$10.00 order. No C.O.D.s accepted.

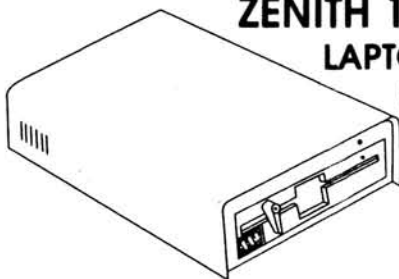
Questions regarding your subscription? Call Margaret Bacon at (616) 982-3463.



WELTEC

W 525 Subsystem

5 1/4" EXTERNAL DRIVE FOR ZENITH 181 & 183 LAPTOP COMPUTERS

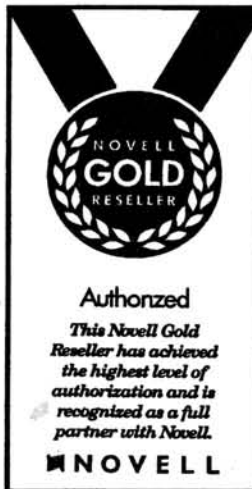


- EXTERNAL 5 1/4" DRIVE FOR ZENITH 181 & 183
- ACTS AS ADDITIONAL SYSTEM DRIVE
- TRANSPORT AND SHARE DATA WITH PCs
- POWER SUPPLY & CABLE INCLUDED
- EXTERNAL DRIVES FOR OTHER LAPTOPS AVAILABLE

WELTEC

17981 Sky Park Circle, Suite M
Irvine, CA 92714
(714) 250-1959
Reader Service #194

When Your Company Needs the Best
in Local Area Networking Services
...Go for the Gold!



When you want the best in networking design, service, and support - call First Capitol Computer. First Capitol is a Novell Gold Reseller, *the highest level of authorization* from the industry's #1 LAN manufacturer. Our experienced staff can provide assistance from initial design through on-site installation and training, as we have for more than five years - for companies from small law firms through Fortune 500 monsters.

Sure, we'll save you money on the components in the system, but more importantly, when we're done, the network will work like a dream - not like a nightmare!
Experience shows!



First Capitol Computer
#16 Algana Drive
St. Peters, MO 63376
Order Line: 1-800-TO-BUY-IT
Other Calls: 1-314-447-8697

Call or write for First Capitol Computer's unique new
Disk-Based Catalog, *FREE* for the asking!
...Or download from our 24-hour BBS system at 1-314-928-9228

Reader Service #191

REMOVABLE HARD DISK DATA STORAGE



FOR YOUR ZENITH
Z-150, Z-248, Z-286, Z-386
MICROCOMPUTER SYSTEMS

- RANDOM ACCESS
- 10MB OR 20MB HARD DISK CARTRIDGES
- MAXIMUM DATA SECURITY
- IMAGE BACK-UP
- 100% MEDIA INTERCHANGEABILITY
- SCSI OR ST506 INTERFACE
- NOVELL COMPATIBLE
- EXTENDED 24 MONTH WARRANTY

For product information see your local Zenith dealer
or call 1-805-964-0733.

In California call 1-800-654-8590.
FAX inquiries: 1-805-964-0734.

VISA & MASTERCARD accepted.

Also available on GSA contract GSOOK-88-AGS-5087.

DMA
TECHNOLOGIES

P.O. Box 1236
601 Pine Avenue
Goleta, California 93116

Reader Service #197

BUGGIN' HUG

Add a Hard Drive to the Z-138

Dear HUG:

I have a Zenith Z-138-42 computer and wrestled for a long time with a method of adding a hard drive to this system without sacrificing one of my floppy disk drives. The recent REMark article by Ray Isenson about installing a hard drive in an H/Z-148 led me to Bob Harris, Heath/Zenith Technical Consultant, who supplied me with the needed information on available power in the Z-138. Armed with this information, I was able to select a hard drive on a card, a Flashcard 30 meg drive, which I ordered by mail from CompuAdd, which met my power, capacity and cost requirements. I installed the drive in my system's one open slot in about five minutes, and it has performed admirably ever since. The Flashcard is comprised of a MiniScribe model 8438 fixed disk drive and a Western Digital model WD 1002-27X fixed disk controller. This hard card cost \$315.00, including shipping. Incidentally, I purchased the Z-130-1 daughter board, which supplies the open slot in the Z-138, from Zenith as a surplus part for only \$25.00. The total cost for adding a hard drive to the Z-138, not including telephone calls and the frustration of not having all the information I needed, was only \$340.00.

A second problem that took some detective work, and regular reading of REMark, to solve was the WordPerfect "clock problem". Version 4.2 of WordPerfect, and presumably, other versions as well, have this weird quirk of resetting the system clock of a Zenith computer to 2:00 am. I called WordPerfect and they sent a copy of a program called fixbios.com, which corrects the problem. I installed the file in the root directory of my hard drive, and added it to my autoexec.bat file, and it works fine. The file is very small, and it might fit on the main program disk of WordPerfect for those using floppy-based systems.

I have learned a lot from reading REMark, particularly in the past year. I hope this information will help some other Zenith user.

Sincerely,
Roger H. Colten RFD 1 Box 366
Miguelito Canyon Road
Lompoc, CA 93436

Any Knowledge of the 9000-48 Keyboard

Dear HUG:

I have been reading about the Enigma Model 9000-48 Keyboard (48 function keys). The descriptive literature, of course, paints a very glowing account of its capabilities — and they do seem impressive. It is also expensive. Do any of your staff, or others in the Heath/Zenith Users' Group, have any first-hand knowledge and experience with this item? I should like to know their opinions. Thank you.

Very truly yours,
Jack N. Rinker
6911 Andover Drive
Alexandria, VA 22307

Z-171 Users

Dear HUG and Z-171 Users:

Would you like to power your Z-171 with a 12-volt negative ground automotive electrical system?

You can with the power adapter from Heath/Zenith Model No. ZA-181-7 Part No. 800-2649 after these two modifications.

1. Drill the inside dia. of the coax plug to .093 Dia. x .437 Deep.
2. Disassemble the black plastic housing and desolder the two conductor cords at the circuit board, reverse those two conductors, solder back on the circuit board and reassemble the plastic housing.

Sincerely yours,
Jim Tichenor
1042 Willow Drive
St. Joseph, MI 49085

Reset Switch for PC Compatibles

Dear Mr. Buszkiewicz:

I want to tell you how pleased I am with the REMark magazine. Never an issue passes by without some useful article or tidbit about the H/Z machines.

I have graduated to a Z-386 from an H-151, and really am impressed with the Zenith. The ultra-expensive Compaq and the unimpressive IBM PS/2 just can't deliver the performance for the price. But what prompts me to write is this: Can you tell the few of us where to hook up the pushbutton for a reset switch? I have the article where you describe using the hard disk locking switch for the purpose, but my backplane board seems to be just backwards from the description. I was content with the ol' fashioned pushbutton on the C207 of the '151's CPU board. Is there a like part on the CPU of the '386 where I can attach the spring clip leads? I

have locked the keyboard up at least once a week — not all software seems compatible with DOS 3.30+.

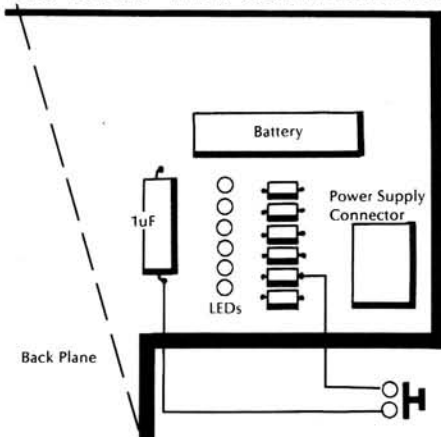
One of these days, I am going to rent a modem and try the HUG BBS. Is it really as simple as the female robot describes on the inner cover? 2400, N, 8, 1 dialing 1-616-982-3956 and have my HUG number ready, right?

Thank you for your time and consideration of this request. I am

Very truly yours,
David A. Secrist
1680 Wilderness Ridge Drive
Milford, OH 45150

For those of you who DIDN'T save your back issues of REMark, here's a reprint of how to install a reset switch in your H/Z-386 computer system. I originally presented the idea in the May 1986 issue of REMark, and then again in an editorial in the July 1988 issue. The method of "reset" applies to the H/Z-241, 248, and 386 computer systems. The following are excerpts from those two issues.

"How about an invisible hardware reset switch for the H/Z-386? I don't know why I didn't think of this sooner. I apologize for being so slow. Now answer truthfully; how often do you lock the keyboard on your '386? If you're like the majority of '386 owners, probably never! Why not make use of that unused switch, and turn it into a reset switch? Almost done as easily said! Eliminate the white wire from the terminal on the switch. Leave the black ground wire connected. Connect a single wire from the N/O (normally open) switch terminal (when the key is in the 'unlocked' position) over to the right terminal of resistor R108 on the backplane circuit board. This side of the resistor is marked: "DC-OK", and is next to the battery. The other side of the resistor is next to a green LED. When the key is used to switch momentarily from unlocked to locked and then back to unlocked, a hard reset will take place! This little 'trick' can be applied to any Zenith 'AT' series of computers. The "DC-OK" line is used in each case."



This layout may look slightly different on the H/Z-386.

CRTSAVER Revisited

Dear HUG:

At the risk of setting a record for revisiting CRTSAVER, I want to share some information with the other readers. I found Bob Brasfield's description of memory control blocks very useful in extending a bit of my knowledge of how DOS works. Since I read that article, I have seen two references — the documentation for Cris Dunford's PMAP and the


source code for MARK/RELEASE explaining how to use an undocumented DOS function to retrieve the segment of the first block in the chain. Function 52H returns a DWOPRD pointer to a location in low memory near the one of interest. Specifically,

```
MOV AH,52H ;ES:BX will point 1 word
INT 21H ; above the info we want
MOV ES,ES:-2[BX];Segment address of MCB
; to ES
```

This particular usage leaves me intrigued by an unanswered question. Is the one word offset an attempt to camouflage the purpose of the function or does ES:BX point to a table of information?

Sincerely,
Vern Reisenleiter
5216 Franconia Road
Alexandria, VA 22310





IBM XT in an H/Z-100

Scottie Board W/ZPC \$149.

OPTIONS AVAILABLE

- PC Compatible Serial Port W/Cable \$50
- 2nd Port W/Cable \$45.
- Clock/Calendar \$45.
- No Solder H/Z-100 MOD Kit \$5.

Requires H/Z-100 MS-DOS 768 of RAM, H/Z-100 Modification.

FACTORY NEW...THE REAL THING
Z-217-1 W/DOCS, HDWE, CABLES, INSTRS

- With 20 Meg ST-225 Drive \$599.
- With 40 Meg MS3650 Drive \$729.

****NEW****
Pair of AT Drives - External, Self Contained, Complete
Uses 1.2 Meg AT Floppy Drives • W/Power, Cables,
34 pin to 50 pin adapter, Instructions - \$299.
34 pin to 50 pin adapter alone - \$23.95

PC/XT/AT and WORKALIKES

LOGITECH	Scanman 4" x 10" Scanner -	\$199.
	C7 Serial Mouse -	\$ 79.
	HI RES BUS Mouse -	\$ 89.
	Complete PC Hand Scanner -	\$169.
	GRAVIS Joystick -	\$ 39.

HARD DRIVES - All Makes/Models at BEST PRICES
ADD ON CARDS, VIDEO, Controllers, Adapters
Tape Backup 40/60 MEG, 2 MEGS/minute.
Colorado Memory Systems DJ-10 \$299.
VCR Tape Backup "IMAGER" \$199.

BEST PRICES ON-
TURBO XT, 80286, 80386 Computers -

BUYING SERVICE - any item you wish us to purchase COST + 10%
GIVE US AN OPPORTUNITY TO BID ON YOUR NEEDS

Scottie Systems, Inc.
1609 S. Main St.
Milpitas, CA 95035
(408) 262-5021

IBM XT is a registered trademark of IBM Corp. • ZPC is a product of HEATH USER GROUP.

VISA & MC ACCEPTED, ALL PAYMENTS SUBJECT TO APPROVAL

Reader Service #121

VOICE MASTER KEY™

VOICE RECOGNITION SYSTEM

FOR PC/COMPATIBLES & TANDY 1000 SL/TL

A FULL FEATURED VOICE I/O SYSTEM

GIVE A NEW DIMENSION TO PERSONAL COMPUTING... The amazing **Voice Master Key System** adds recognition to just about any program or application. Voice command up to 256 keyboard macros from within CAD, desktop publishing, word processing, spread sheet, or game programs. Fully TSR and occupies less than 64K. Instant response time and high recognition accuracy. Voice recognition tool-box utilities are included. **A genuine productivity enhancer!**

SPEECH RECORDING SOFTWARE... Digitally record your own speech, sound, or music to put into your own software programs. Software provides sampling rate variations, graphics-based editing, and data compression utilities. Create software sound files or send voice memos through LANs. **A complete, superior speech and sound development tool.**

SOFTWARE CONVERSION CODES... The **Voice Master Key System** operates a growing list of third party talking software titles using synthesized phonetics (text - to - speech) or digitized PCM, ADPCM, and CVSDM encoded sound files. **Voice Master Key system does it all!**

EVERYTHING INCLUDED... **Voice Master Key System** consists of a plug-in card, durable lightweight microphone headset, software, and manual. Card fits any available slot. External ports consist of mic inputs and volume controlled output sockets. **High quality throughout, easy and fun to use.**

ONLY \$149.95 COMPLETE. TANDY VERSION ONLY \$89.95—SOFTWARE ONLY REQUIRES TANDY BRAND ELECTRET MICROPHONE

ORDER HOTLINE: (503) 342-1271 Monday-Friday, 8 AM to 5 PM Pacific Time

Visa/MasterCard, company checks, money orders, CODs (with prior approval) accepted. Personal checks subject to 3 week shipping delay. Specify computer type and disk format (3½" or 5¼") when ordering. Add \$5 shipping charge for delivery in USA and Canada. Foreign inquiries contact Covox for C & F quotes. **30 DAY MONEY BACK GUARANTEE IF NOT COMPLETELY SATISFIED. ONE YEAR WARRANTY ON HARDWARE.**

CALL OR WRITE FOR FREE PRODUCT CATALOG



COVOX INC.
675 Conger Street, Eugene, OR 97402
Telex 706017 (AV ALARM UD)
TEL: 503-342-1271 • FAX: 503-342-1283

Reader Service #196

Microsoft®
Windows \$25.00

Surplus Computer Parts

Seagate 4051
40 Meg. \$275.00

- Largest Surplus Electronics Dealer in Western Michigan
- In Business Over 40 Years
- Over 12,500 Square Foot Store and Warehouse

Examples of Zenith Salvage Products:

Z-100 Motherboards	From 20.00
Z-100 Power Supplies	25.00
Z-100 Winchester Controllers	50.00
Z-171 Modem Boards	100.00
Z-181 Modem Boards	10.00
Z-200 Power Supplies	From 25.00
PC Keyboards	15.00
5-1/4" Floppy Drives (360 or 1.2)	25.00
1/3 Ht. 5-1/4" Low Power Drives	25.00
CM1 Model 5412-12 Meg. Hard Drives	25.00
80287-6 Coprocessors	65.00
AT Backplane Boards	50.00
EPROMs and DRAM	Call for Pricing
Equipment Power Cords	2.00
Computer Fans — 12 volt and 120 volt	
Used Computers: 89's, 100's, PC's, AT's	

Many Other Zenith Salvage Parts Available
All Surplus/Salvage Sold AS IS — WHERE IS
No Catalog — Please Call for Quotes and Availability
We Ship U.P.S.-C.O.D. (ONLY) Sorry — No Foreign Shipments

Surplus Trading Corp.
THE HOUSE OF EVERYTHING "ALMOST"
2700 N. M-63 P.O. BOX 1082
BENTON HARBOR, MI 49022
CALL: LES — (616) 849-1800

Reader Service #153

Xplore Your Batch Files . . .

This article addresses two principal themes: (1) vagueness and omissions in the information that MS-DOS manuals provide for using batch file commands and some non-obvious strategies for batch files; and (2) The use of short assembly language programs as tools to find out things the manuals don't tell us about the computer, and about MS-DOS. It makes no pretense of being comprehensive as to either theme.

Many people view assembly language programming with considerable distaste, preferring a high level language because it lets them state what they want to do, rather than to laboriously describe all of the detailed steps of how to do it. My own view is that assembly language programming is fascinating for the degree of minute control that it gives over the operation of the machine. I must admit, however, that assembly language can get tedious for tasks where the executable code will be over 7 or 8 Kbytes.

In this article, I hope that the demonstrated usefulness of a picky little assembly language program will give the reader a (perhaps) new motivation for learning at least the bare rudiments of using assembly language. I am sure that many, if not a majority, of the readers of REMark have a consuming interest in learning more about their computer, and about its operating system. In an ideal world, one would only have to go to the index of the appropriate manual with a shrewdly chosen key word to find information on an area of interest. In this real world, however, not only are the manuals often incomplete and curiously indexed, sometimes what they tell us is just flat out wrong.

There is a marvelous way out of this problem though, and it is that if you pose the question in the right way, the computer itself must give you the answer. You can write short assembly language programs that will cause the computer and/or operating system to reveal some of their innermost secrets, printed right out on the screen.

A Problem with Mixed Switches and Parameters . . .

One issue that the operating system manuals fail to make fully clear is that of the similarities and differences of the parameters that you may enter after the name of a program, and the /switches that you may append to a command, or to one of its parameters, to alter its mode of

operation. A parameter is such as the file name, or its destination, when you use the COPY command, as in:

```
A>COPY thisfile B:
```

where thisfile and B: are parameters. Parameters must be separated from the command, and from each other, by delimiters which may be space (as in the



Robert Brasfield
10344 Dibble Avenue NW
Seattle, WA 98177

COPY example), tab, comma, semicolon, or equal sign. Some manuals will tell you that the plus sign (+) can also be used, but you may change your mind if you use the included RUN.BAT and SHOW.COM to check that out.

A switch may be such as those used with the FORMAT command, where,

```
A>FORMAT B: /S
```

will cause the FORMAT utility to include the system files on the formatted disk. Switches are often just / followed by a single character, but they may also be many characters in length. For example, the LINK command takes switches such as /DSALLOCATE and /LINENUMBERS.

With the sketchy descriptions of the uses, similarities and differences of parameters and command switches, many users may well come to think of them as more or less interchangeable, and applications programs may use them in ways that reinforce the appearance of interchangeability. Batch files that use replaceable parameters, can pass both /switches and parameters, from the original command line that invokes the batch file, through to applications programs named in the batch file, but the combined actions of the command processor and the batch processor may give strange twists to the result. This is an example of the sort of issue where a very short assembly language program can persuade the operating system to reveal much more than its manuals ever did.

Switches and parameters are very similar in that the only way that a program is going to learn about either one of them is from the fact that the operating system copies what appears on the command line, after the filename and before the Return key is pressed (*), to offset 081H in the program segment prefix. The material that is copied there includes both parameters and switches. This material is known as the "command tail", and its length in bytes is placed at offset 080H.

* There are exceptions to this if you use I/O redirection or piping symbols on the command line. SHOW.COM can be used to demonstrate the exceptions.

If you will use ASSEM.BAT to create the SHOW.COM file from the assembly language of Listing 1, you will have a tool with which you can recreate the exploratory exercises that are discussed later on in this harangue. ASSEM.BAT requires that you have SHOW.ASM, MASM.EXE, LINK.EXE AND EXE2BIN.EXE in the default drive, or locatable by reason of the PATH command.

SHOW.COM acquires the number at

. . . And Xplode Some Myths

offset 080H that gives the length of the command tail, converts it to printable decimal digits and places them in the XPAND string. It then prints out, starting at column 1 of the line, all of the characters that MS-DOS placed in the command tail area, followed by printing out the XPAND string with the command tail length. The two 9's (tabs) at the start of the XPAND string are to ensure a conspicuous visual separation of the number from the command tail. The command tail is printed out first so that the start of the line provides a visual reference for the start of the command tail, especially useful when the first character is a space.

The batch file RUN.BAT is set up so that it could pass up to three other parameters to an executable program that is named as replaceable parameter No. 1 when RUN.BAT is invoked. In the examples that follow, RUN.BAT and SHOW.COM were both available on a RAM disk, drive I.

If you enter `d>SHOW RAG TAG` on the command line (where `d>` is the drive letter and system prompt), SHOW.COM will print out:

```
RAG TAG           8
```

If you enter the switch character / after `d>SHOW`, instead of a space, as in `d>SHOW/RAG TAG`, SHOW.COM prints out:

```
/RAG TAG           8
```

and, if you enter a command `d>SHOW/RAG/TAG`, SHOW.COM prints:

```
/RAG/TAG           8
```

Ho-hum, you say? Well, hang on just a moment, things will be less routine very shortly. If you use RUN.BAT to pass the same command tail items, using the command `d>RUN SHOW RAG TAG`, SHOW.COM prints out:

```
;RAG;TAG;         9
```

For the command `d>RUN SHOW/RAG TAG`, SHOW.COM prints out:

```
/RAG;TAG;;        10
```

and for `d>RUN SHOW/RAG/TAG`, we will see:

```
/RAG/TAG;;;       11
```

Now, very obviously, the three combinations of parameters and switches yield quite different strings in the command tail area when passed through by a batch file than they do when placed there directly from the command line. This can have important consequences for a program that tries to use the number at 080H to decide how to interpret the command tail string.

So what is this batch file doing? Haven't we been conditioned to think that an input via a batch file will give the same results as would the input straight from the command line?

Let's describe first what happens with a command line that has only parameters, no /switches. When COMMAND.COM looks at the original command line, it uses the standard delimiters to locate the

boundaries between the parameters, saves the parameters and discards the delimiters. In copying to the command tail area, it recombines the parameters with whatever delimiters were used to separate the %1 %2 etc. of the batch file command, which also may use any of the standard delimiters.

In our example, %1 is SHOW, the name of the file to be executed, and COMMAND.COM, as instructed by the batch file, places in the command tail area:

```
delimiterRAGdelimiterTAGdelimiter%4
```

Since there was no fourth parameter, %4 is nothing, but the delimiter that we stipulated in the batch file as preceding %4 is put in place, just as the batch file directed. If you count up the characters, there are 9, and that is the number that will be put at offset 080H. The parameters that, when passed directly from the command line, resulted in 8 in the byte at 080H, now result in the number 9, because of the inclusion of a delimiter preceding the nonexistent %4.

With the form `RUN SHOW/RAG TAG`, the first replaceable parameter is SHOW/RAG, so there is no %3, as well as no %4 and we get two semicolons tacked on the end of the string, with a character count of 10. Similarly, with the form `RUN SHOW/RAG/TAG`, the first replaceable parameter is SHOW/RAG/TAG, three semicolons are tacked on for a character count of 11.

The implication of this is that, assuming that some of our programs may attempt to use the number at 080H in dealing with the command tail, then we should try to anticipate, when we write the batch file, how many parameters there will be. It is not sufficient to just put in %1 %2 %3 %4 etc. so that there are as many positions for substitution in the batch file as the maximum number of parameters that may ever be used. Some programs may ignore the value placed at offset 080H in dealing with possible parameters in the command tail area, but it would be a great nuisance to have to determine this for each program that one might invoke via batch files. Some other strategy is needed for coping with an indefinite number of parameters.

Another unfortunate result occurs when the command tail that is passed is a path/filename and the executing program does not rely on the number at 080H to locate the end of it, but goes down the string byte by byte, searching for the 0DH character to mark the end. If it replaces the 0DH with a null byte and then uses the path/filename as an Ascii string in a DOS function call, such as 3C, Create a File, an error return will occur if the delimiter preceding a nonexistent %3 or %4 was tacked on to the end. If you are writing a program that may be used in this set of circumstances, the remedy is to

both ignore the 080H number, and to not rely solely on the 0DH character, but to search down the command tail, placing the null byte at the first location of either 0DH or valid delimiter.

No doubt some reader will be tempted to solve this problem by not using delimiters in RUN.BAT, just %1%2%3%4. Here is one product of that strategy:

```
I>SHOWRAGTAG
```

```
Bad command or file name
```

And, if you still believe that + is a valid delimiter, try

```
d>RUN SHOW+RAG+TAG
```

```
which produces:
```

```
I>SHOW+RAG+TAG;;;
```

```
Bad command or file name
```

SHOW+RAG+TAG is treated as just one big ugly glob of a single parameter.

SHOW.COM can also illustrate one good reason to not use the = sign as a delimiter in batch files. Assume that you make a batch file WEIRD.BAT, similar to RUN.BAT, but using the = sign as the delimiter, and that you interchange the position of %2 and %3. The file would contain

```
%1=%3=%2=%4
```

and from the command WEIRD SHOW/RAG TAG you would get:

```
I:SHOW/RAG==TAG=
```

```
/RAG==TAG=           10
```

Since the == symbol has special meaning for batch files, who knows what this might lead to.

A Solution for Mixed Switches and Parameters . . .

RUN_MANY.BAT gives an example of how to deal with a variable or uncertain number of parameters that must be handled by a batch file. Commands `RUN_MANY SHOW RAG TAG`, `RUN_MANY SHOW/RAG TAG` and `RUN_MANY SHOW/RAG/TAG` will each produce the same output (except for the identity of the delimiters) from SHOW.COM as would have followed from commands `SHOW RAG TAG`, etc. Try it.

You will note that in RUN_MANY.BAT, where an IF command is used to determine if a particular parameter may not exist, it uses a string comparison `%2*==*`. In some manuals, in describing the uses of various commands, they illustrate a form of the string comparison `"%1"=="",` which tends to leave the impression that the use of quotation marks has some special place in the syntax of the batch commands. The reality is that the batch commands simply have no way to recognize a description of nothing. They can, however, recognize that the sequence of nothing and the character * is the same as the character *. They can recognize that the combination of quotation marks with a nothing substitution between them is the same as successive quotation marks.

In the command `IF %4*==* GOTO 3`, the string comparison, `%4*==*` could be replaced by any one of the following

forms:

```
"%4"==" " or "%4==" or "%4"=="  
or *'%'4==*
```

The rule is that if the parameter to be substituted for %4 may possibly be "nothing", you must set up the comparison so that there will always be at least one "real" character on each side of the equation. The quotation mark itself is just another useful "real" character, singly or in pairs. There are certain characters that you must not use. These are: < > | % and =.

It appears that the writers of manuals would rather have you just accept the form "", used in BASIC for describing null strings, than to confuse you with the facts.

The command "IF %4*==* GOTO 3" also contains some delimiters that you may not recognize as delimiters until you rewrite the command as "IF=%4*==* GOTO=3".

Try it! It works. Think of other things to try! You have nothing to fear but fear itself! (Or a reboot).

And Now Some Other Curiosities . . .

A capability of batch files that you may easily overlook, unless you have read your MS-DOS manual word for word, (and remember them all) is that the performance of batch files can be modified by information placed in the system environment by the SET command. For instance, if you use the SET command to put strings ROUTE=LONG and DESTINATION=HERE in the environment, you could use a comparison in a batch file to cause optional execution, controlled by those strings. The batch file might contain:

```
ECHO OFF  
IF %ROUTE%==LONG GOTO %DESTINATION%  
ECHO Route was %ROUTE%  
GOTO Last_Exit_On_Brooklyn  
:HERE  
ECHO Route was long  
:Last_Exit_On_Brooklyn
```

After running that with ROUTE=LONG, then enter a command:
SET ROUTE=SHORT
and run it again.

As you see, the information extracted from the environment can be utilized in a number of different ways. It appears that you can use the string from the environment as a parameter, or direct string substitution in many of the batch processing commands. Since a batch file could also contain SET commands, altering the content of the environment block, the possibilities are simply mind-boggling. Execute the command SET ROUTE= (with absolutely nothing, not even a space after the =), then try this batch file:

```
ECHO OFF  
IF %ROUTE%="" GOTO THERE  
If NOT %ROUTE%==TTTTTT GOTO THERE  
ECHO Route was too long and dreary.  
SET ROUTE=  
GOTO Last_Exit_On_Brooklyn (My friend o  
wns it)
```

```
:THERE  
SET ROUTE=T%ROUTE%  
ECHO Route is now %ROUTE%  
:Last_Exit_On_Brooklyn (Only the first  
8 bytes are significant)
```

Run this last batch file several times in succession, then just go on a wild orgy of experiments. One thing you could try is to see how many places the batch processor will let you get away with omitting the paired before-and-after % signs.

Manuals will also tell you that, although one batch file can call another there is no automatic return to the first when the second is complete, and will say that batch files can not be nested. Actually, batch files CAN be nested if certain conditions are satisfied:

1. The name of the second batch file, with the replaceable parameters that it will use, must be the string in the command, COMMAND/C string.
2. Command.com must exist in the currently active directory, or be locatable by reason of the environment PATH= string.

For a very simple example, set up condition (2) and run this:

```
ECHO OFF  
CLS  
ECHO "Before" message  
PAUSE  
COMMAND/C RUN_MANY SHOW RAG TAG  
ECHO "After" message
```

Then you can try NESTED.BAT, which

will allow you to create a brand new batch file, right in the middle of the execution of a batch file, have it execute, then continue with the original.

Much of what appears in this article is what you might call "undocumented". As a long time designer of electronic hardware, I have been conditioned to view it as close to a deadly sin to rely on undocumented properties of transistors, integrated circuits and the like, so it goes a little against the grain to advocate the use of undocumented features of software. However, much of the cost of placing those undocumented features in the software may well be for the purpose of laying the foundation for future upgrades, which we will be encouraged to buy, after having already paid for the undocumented foundations in their predecessors. So should we refuse to use those undocumented features that will be part of tomorrow's grand new improvements?

Consider also, that with the advent of OS-2, those grand new improvements may just die on the vine.

Further, having been a provider of source information to manual writers, I know that often undocumented just means that the writer didn't get the word.

It seems to me that the answer is "Use with CAUTION!"

```
RUN .BAT  
%1;%2;%3;%4
```

Most home finance programs have serious limitations.

Like the dodo bird, they're slow, awkward and not very smart. You'll waste eons of time, only to realize they're dead ends.

HFS-III, the Home Finance System, is fast, loaded with features and intuitively smart.

- Easily manage up to 100 checking, saving, CD, IRA and other accounts.
- Print checks on any form.
- Organize financial summaries and activity reports based on 100 expense and 15 deposit codes you define.
- Flag tax-deductible or medical expenses.

In short, control your finances to save money, save time and simplify taxes.

Find out why so many leading computer authorities and users are calling HFS-III the "superior species." See it today at participating Heath/Zenith Computers and Electronics Centers, or call toll free for more info.

Only \$49⁹⁵

Add \$3.50 handling & shipping.
Iowa residents add 4% sales tax.

Requires: DOS 2.0 or higher, IBM*
PC/ XT/AT or compatible or Zenith*
Z100 computer; 2 disk drives or hard
disk, 256K RAM.

Reader Service #137



Jay Gold Software, Inc.
P.O. Box 2024
Des Moines, IA 50310
(800) 541-0173



RUN_MANY.BAT

ECHO OFF

CLS

:The first 2 lines are just a convenience to avoid cluttering the screen with confusing and non-informative echoes.

:Note that in the following comparisons, the least number of parameters must be the first case checked out.

```
IF %2*==* GOTO 1
IF %3*==* GOTO 2
IF %4*==* GOTO 3
```

:The ; is used as a delimiter in the rest of this so that it will be obvious when printed out on the screen. With a space as the delimiter, its significance could be less clear.
:Non-referenced labels make handy comment lines, don't they?

```
%1;%2;%3;%4
GOTO EXIT
```

:3

```
%1;%2;%3
GOTO EXIT
```

:2

```
%1;%2
GOTO EXIT
```

:1

```
%1
:EXIT
```

ASSEM.BAT

```
MASM SHOW.ASM,SHOW.OBJ,NUL,NUL
LINK SHOW.OBJ,SHOW.EXE,NUL,NUL
EXEBIN SHOW.EXE,SHOW.COM
DEL SHOW.OBJ
DEL SHOW.EXE
```

NESTED.BAT

ECHO OFF

CLS

```
IF NOT EXIST JUNK.BAT GOTO TELL
DEL JUNK.BAT
```

:TELL

ECHO You may enter commands from the keyboard into a nested ECHO batch file which will be executed as the next process. ECHO provided that Command.com is in the current directory or ECHO can be discovered via the PATH= string in the environment. ECHO Your input must be terminated by CTRL Z and Return. ECHO Otherwise just press CTRL Z and Return to decline the ECHO opportunity.
COPY CON JUNK.BAT

```
IF NOT EXIST JUNK.BAT GOTO LASTEXIT
IF EXIST COMMAND.COM GOTO DO_IT
IF NOT EXIST %PATH%\COMMAND.COM GOTO WISEXIT
:DO_IT
COMMAND/C JUNK.BAT
:WISEXIT
DEL JUNK.BAT
:LASTEXIT
```

Listing 1

```
SHOW SEGMENT
ASSUME CS:SHOW,DS:SHOW,ES:SHOW,SS:SHOW
ORG 100H
BEGIN: JMP SHORT START
XPAND DB 9,9,0,0,0DH
START: XOR AX, AX
MOV SI, 080H
LODSB ;Get command tail length in AL
MOV DI, OFFSET XPAND + 4 ;Decimal goes into the 0's
CALL HX_DEC ;after DI is decremented
CALL DISPLAY ;SI points to 081H
MOV SI, OFFSET XPAND
CALL DISPLAY
MOV AX, 04C00H
INT 21H ;Just in case
INT 20H
DISPLAY PROC NEAR
MOV AH, 2 ;Write to screen function
DSP010: LODSB ;Termination character
CMP AL, 0DH
JE DSPXIT
MOV DL, AL
INT 21H
JMP DSP010
DSPXIT: RET
DISPLAY ENDP
HX_DEC PROC NEAR
```

; Procedure is good for no more than 2 decimal digits. If first digit is 0, it is changed to a space. AX, BX, CX, DX and DI may be altered.

```
MOV CX, 2
MOV BX, 0AH
```

```
HXD010: DEC DI
XOR DX, DX
DIV BX
OR DL, 00110000B
```

;Quotient to AX, remainder to DX
;DL always <= 9

POWERING

UP



William M. Adney

P.O. Box 531655

Grand Prairie, TX 75053-1655

Copyright © 1988 by William M. Adney. All rights reserved.

How to Select Application Programs

Choosing an application program — a word processor, spreadsheet, graphics or data base — is a difficult task for any computer user. The software market is literally flooded with products that provide one or more of these applications, and it is difficult even for an experienced user to select the right software to solve a specific problem. And if you are not generally familiar with the general capabilities of software in a given category, you may spend lots of money on software before you find the “right” one for you. This article will help you choose software that fits your needs, and if you are not quite sure what those needs are, you will find a couple of suggestions on how to define them.

Too many new computer users select an application program based on the recommendation of the sales staff in a computer store. In many cases, this recommendation may be based on the perception of the most “popular” software, and even the best software available for a given application may not be the best for you. Perhaps the best example as I write this is that Word Perfect is the best selling word processor on most lists and has been for some time. In most cases, however, I do not generally recommend Word Perfect for the simple reason that it is far more powerful than most users really need, even though it is an excellent word processor. The real question then becomes: What do you REALLY need?”

Examining Your Needs

There are two general approaches to selecting any application software. First, you can buy an ultra-sophisticated package that has all of the latest features and “grow into” using it proficiently. Some experts recommend that approach. I don’t because many of these sophisticated packages require a lot of time to learn some of the basic features, let alone the fancy ones. Many assume a basic knowledge of the terms related to an application that are usually quite puzzling to a new user. For example, all word processors worthy of the name should be able to perform the standard “block” functions: copy, move, and delete. If you don’t already know generally what those functions do, and how to do them, you will have to spend lots of time learning the terminology in addition to the software itself. For that reason, I suggest a somewhat simpler approach that requires a little explanation.

For most new computer users, learning how to use a software package, particularly the FIRST one, can be a traumatic experience. This is especially true when you have absolutely no experience with any microcomputer programs because you must first learn the general terminology (e.g., the “block” functions for a word processor or a “cell” in a spreadsheet). Then you must learn how to use that program’s commands to perform your

work. The combination of learning the general terminology, learning the program’s terminology, and learning how to use the commands takes a lot of time, and is generally frustrating. And most of the latest programs have so MANY features that it is nearly impossible for a new user to determine which ones will be needed on a regular basis and which will not. While it is nice to “have” hundreds of these features and capabilities, it is an unfortunate fact that the actual number of features results in an information overload. That leads to a lot of frustration that could be avoided. Therefore, I suggest a second approach.

Start with something simple and work your way up. I am reminded of a letter that appeared in “Buggin’ HUG” some time ago from a gentleman who could not figure out how to start the spelling checker in Microsoft Word. Although no details (such as what version) were mentioned in the published letter, I believe it was a user error either in the installation of the program or simply not looking up the word “spell” in the manual’s index to find the answer.

Regardless of what the problem was, his level of frustration with Microsoft Word was quite evident from the letter, and I suspect he was a victim of information overload resulting from the wide range of features in Word. As a result, he was not able to figure out how to perform

what should be a common feature in any word processor. I have seen that problem frequently in my computer consulting and teaching experience, and it is not at all unusual for frustration to lead to the inability to clearly think through a problem. This goes into a never-ending circle of frustration and causes many people to say: "I HATE COMPUTERS." Once this begins, it is extremely difficult to break the cycle. It is much easier to begin with a KISS and a SMILE.

With a KISS and a SMILE

I have spent over 20 years working with all kinds of computers, and I have always subscribed to the KISS and SMILE principles of data processing. KISS means: Keep It Simple, Stupid. SMILE means: Simplicity Makes It Lots Easier. And since a KISS is usually followed by a SMILE; well, what can I say? But enough of the philosophy for selecting a program, let's take a look at how you can do it on a practical level.

Keeping the selection of an application program simple is easy to say, but difficult to do. There is nothing worse than spending several hundred dollars on a sophisticated application program only to find out it is too difficult to use for even simple tasks. That particular point, in my experience, is what causes most of the frustration: you have spent all that money and the program is essentially useless to you. To avoid the frustration that often accompanies the attempts to learn a sophisticated and expensive program, there is one basic way to choose your first application program.

Choosing the First Application

The basic rule for choosing your first application is: GET THE CHEAPEST ONE YOU CAN FIND. For example, you can get "shareware" programs for nearly any type of application which cost little or nothing. The HUG library is another source of very inexpensive programs that you should not overlook, and I will mention some specific programs later in this article. Because of their cost, you will generally find that these programs do not have a lot of sophisticated features, but they will perform the basic function for which they were designed. This reduces the problem of information overload associated with the more sophisticated programs.

More importantly, you won't have a lot of money tied up in the software, and even if you don't like the program or its features, at least you will KNOW what you do and don't like. Because many applications generally function the same way and generally use the same terminology, such as spreadsheets, you will be able to learn the basic application concepts without a big investment. Since most of these programs can be found for \$40 or less, the fi-

nanacial impacts of a "throw-away" program will not lead to the frustration that a \$250 program will. But you may be surprised to learn that one of these cheap programs will do everything you need anyway, and it is not necessary to buy a more expensive program.

For example, let's say you are considering buying a spreadsheet program. You have never used one before, and you don't know what a cell is or how to construct a template. Don't run out and spend several hundred dollars for Lotus 1-2-3 despite the fact that you may have heard of it. Instead, get something cheap like HUG's Cheapcalc spreadsheet. For a modest \$20, you can learn something about spreadsheets, in general, and it won't cost you an arm and a leg to do it. You may also be able to find a copy of the PC-CALC shareware program that will cost you little or nothing.

If you are interested in simple text processing, you may want to try the HUG Editor to learn the basics. More advanced users may be interested in a more powerful editor, such as PC-WRITE that in its fully registered version still costs less than \$100 and has most features of word processors costing up to seven times as much. Don't spend hundreds of dollars on Microsoft Word, WordStar or Word Perfect to begin with — you may not like the software.

Perhaps you are interested in telecommunications with a friend or just looking at many of the computer bulletin boards that are available. Don't spend over a hundred dollars for CrossTalk; try HUG's MCP (Modem Communications Program) instead. For 40 bucks, you can use the software (it is easy to learn and works great, by the way), and you may find it's all you really need.

For data base work, I suggest trying PC-FILE. It is a simple, easy-to-learn program that has a lot of power. It will introduce the fundamental concepts of data base development and use, and again, it may be all you need. Most applications do not really require the power of a sophisticated program like dBase.

For one reason or another, you may not want to try these individual programs, and there is one other way to keep costs down and still have most of these capabilities.

The Integrated Approach

One of the other popular approaches to buying software is to buy an integrated package that usually includes at least a word processor, a spreadsheet, a data base, and a telecommunications module. Wait — don't run out and buy Symphony or Framework no matter what you have heard about them. They are very powerful integrated packages which have a lot of advanced features and capabilities with a list price to match: about \$695. You can

get the advantages of integrated software at far less cost.

One of the best values available is an integrated package called Ability. It includes the five basic applications that you might want or need: word processing, spreadsheet, data base, graphics, and telecommunications modules. Although its functions are not difficult to use, Ability er. At a list price of just under \$100, each of its basic applications costs about \$20 each which gives you considerable "bang for the buck." For about \$50 each, you will also find Ability Plus is a good value, too. It is, as you might expect, more powerful than its less expensive cousin, and it may be a better choice if you are using an EGA or VGA display.

Another excellent integrated package is First Choice. This software has received good reviews in a number of publications, and it seems to be especially easy to learn and use. It also includes a word processor, spreadsheet, database, graphics, and telecommunications modules. Although its functions are not difficult to use, Ability provides more power because you can easily "flip" back and forth between screens for the different modules. Ability also provides a "link" feature that works between multiple spreadsheets or between modules. For example, when the data on a spreadsheet changes, a word processing document will also change if they have been linked.

Perhaps the real advantage of these programs is that they are very cost effective. You get a lot of functions for the price, plus they are integrated which means that once you learn how to operate one program, other programs in the package work much the same way. Another advantage is that you can effectively "copy" information from one program's file to another. For example, you can use the spreadsheet to perform calculations and include that information in a document created by the word processor. Even though they do not have a lot of advanced features found in some other software, they do not cost as much either.

At this point, we have looked at some specific suggestions that can help you select your first application program in a specific category. But what happens when you need or want something more powerful than this?

Evaluating Software

There are any number of methods you can use to evaluate software, but each has its own limitations. One reasonable way to approach software evaluation is to look at various computer-related magazines and other publications. Many magazines, such as *PC Magazine*, feature comparative reviews of various kinds of software that can be quite helpful. While many of these review articles can be helpful in making a choice, it is important to

keep in mind that it is extremely difficult to objectively review software. All reviewers have a bias or prejudice for specific application software packages that may not be exactly obvious from reading a review. And even if you find a comparative list of features for various word processors, it may be less than helpful because the actual use of that feature may be clumsy or difficult. It may also be difficult to use a comparative list of features because you may not really need a full-featured program to begin with.

For example, my word processor of choice is WordStar, because it best fits my needs. Even though it does the things that I want to do, and it does them well, that does not mean that it is the best word processor for you. Still, when I look at any word processing software, I compare its features and capabilities with WordStar.

In order to place the subject of software evaluation in a more objective environment, I will use the word processing as a specific example, because more than 90% of all computer users buy or use word processing.

Before we get into specifics, let's develop a list of important features for a word processor and most other application software.

The Evaluation Checklist

The basic trick for setting up an objective evaluation for anything — software, computer hardware or an automobile — is to develop a list of the basic features that you want or need. For software evaluation, this list might include the categories shown in Figure 1.

Category	Value
Requirements	50
Speed	50
Features	150
Documentation	250
Support	
Dealer	50
Manuf.	150
Price	300

TOTAL	1000

Figure 1
Developing an Evaluation Checklist

In Figure 1, there are six general categories that we will use to evaluate software: Requirements, Speed, Features, Documentation, Support, and Price. The "value" for each category is a weighting factor that I have assigned based on each category's importance to me. For example, you can see that Price is the most important consideration because it has a maximum value of 300 points, more than any other category, out of a possible 1,000. The next most important category is the documentation (250 points), because I can't easily use the program if the

manual is poor or it may take an inordinate amount of time to figure out how to do something. I am also interested in the support (200 total points) in the event I have some kind of problem with the software. Notice that I have divided the Support category into two parts with emphasis on the manufacturer's support. Features come next at 150 points. And finally, I need to know about the Requirements to run the software (i.e., will it work on my system?), both hardware and DOS, and I want a program that performs at a reasonable speed.

All of the weighting factors add up to 1,000 points which makes it easy to see what the "score" is. I have also chosen a high total to help reduce any particular bias I may have for a program in a given category.

You can easily customize this kind of evaluation list any way you want. For example, you might want to check out the color capabilities of a program — Does it support EGA or VGA? Is a hard disk recommended or required? What about mouse support? Are there any limitations on file size? Can the program be customized for specific colors and for printers that are different from those usually supported by the program? Does the program directly support the printer you have without any changes? Does a word processor have an included spelling checker, thesaurus or other helpful utilities? These are just a few of the items that can be added to the basic list, and you can weight them as you prefer. But that is not all this kind of checklist can be used for.

With some changes, it can also be used for evaluating computer hardware. You can also use this technique for comparing just about anything, such as automobiles. Color, engine size, dealer service, warranty, and price are potential categories that you could use to compare different cars.

Where do you get the information necessary to complete an evaluation once you have developed it? You can find many comparative reviews of software and hardware in many microcomputer magazines. Many report on categories that are similar to Figure 1, and you may want to use that information in your own evaluation. There really is no limit to the number of things you can compare using this kind of technique. This approach to evaluating just about anything has been used in data processing for years, and I have used it for comparing a lot of mainframe hardware and software with one added twist.

When I need to justify the purchase of expensive mainframe computer hardware, I also include a number that I call the "bang for the buck" ratio. It is just the price of the product divided by the total score that the product received. To illustrate how this works, let's assume that we

did a full evaluation on Ability Plus (\$259.95), and it received a total of 810 points. A similar evaluation of First Choice (\$149.00) yielded a score of 680 points. To keep the numbers simple, I usually divide the final scores by 100 so Ability Plus has an 8.1 and First Choice is 6.8. Then, I can calculate that Ability Plus has a bang for the buck ratio of about 32.1 (259.95 divided by 8.1), and First Choice has a ratio of about 21.9 (149.00 divided by 6.8). Notice that the LOWER number indicates a better buy because it costs FEWER dollars per point. The basic assumption is that all evaluated products do approximately the same thing. Although it does not show up particularly well in this example, the bang for the buck ratio helps more when the products are closer to the same cost. There is one other suggestion that you should consider before buying any expensive software.

Check It Out

Even after going through a formal software evaluation, there is no substitute for trying it. Most computer stores have popular software available on their demonstration systems, and you can usually find someone who can show you the basic operation of a program. If possible, try to use a demonstration system that is similar to yours — if you have a floppy disk system, and the demonstration system has the software on a hard disk, you will not be able to get a good feel for how fast the program is on YOUR system.

This gives you the opportunity to see, touch, and get a feel for a program, and no matter how good your evaluation says it is, you still may not like it. You may find, for example, that the program is just too slow, but remember that speed may not have been heavily weighted in your evaluation. Don't try to perform any of the fancy things, because you probably won't know how to do them, and that can cause an immediate dislike of the program. Despite what your evaluation results are, you may find that a program is not as intuitive (i.e., easy to use) as you would like. Or you may not like the user interface or command structure at all. Be sure to keep in mind that you will not become an expert by "testing" the software for an hour or so.

Most software has some kind of tutorial or training program, and you should look at that during your testing process. For example, Microsoft Word has an excellent program-driven tutorial. Other programs may have various training "lessons" in the documentation, and they usually provide an excellent introduction to the software.

When you finally decide to shell out several hundred dollars for a new program, what should you do next?

When You Get It Home

When we get some new software,

most of us are anxious to get it up and running, but it is really important to take your time. Stifle the impulse to get it up and running in the next 10 minutes. Take your time. Don't power-on your computer just yet. After you get the package unwrapped, take a few minutes to look through everything. Sometimes pages for the manual are shrink wrapped, and you need to insert dividers for each chapter. As you do so, take a look at the chapter headings to become familiar with the general contents of the manual. Then spend some time looking through the Table of Contents to see what information is in which chapter. As you look through the documentation, make a note of where the installation instructions are.

In most software packages, the disks are usually in a sealed envelope of some kind, and many have the license agreement on the outside of the envelope. Take some time to read the license agreement and look through all of the documentation BEFORE you open the envelope. Once you open the sealed package containing the distribution disks, you generally cannot return the software.

Spend some time reading the installation instructions with a marker or highlighter in hand. It's still not time to power-on the computer yet. Mark the important notes. This can be especially critical because you may otherwise miss something that will affect the software's operation. For example, I was installing a new software package some time ago (I forgot which one), and I missed a statement that read something like: "the first printer selected from the menu is the primary printer." I installed my DTC StyleWriter (Diablo emulation) and my C. Itoh C-310 (Epson emulation) in alphabetical order, and it took me a while to figure out why the StyleWriter was always coming up as the default printer. I ended up having to reinstall the program because I missed that one critical statement. Since then, I have always double-checked the installation procedure and marked all critical information with a highlighter so I won't miss it during the process.

Now it's time to power-on the computer. Many software programs include some kind of a "read me" file on one of the disks, usually on disk number one. That file contains last-minute corrections and changes to the manual. Some software has a special "README" program that can also print the file. Or use the PRINT command to print it. Print the file and read it before you begin the installation. There may be some corrections or changes to the installation process that you have already read. I have found it helpful to make these corrections or changes to the printed manual before beginning the installation.

You are finally ready to begin the install process. Start the installation procedure,

and be sure to follow the instructions exactly. Take your time, and don't try to rush it. If you find something that you do not understand as you do the installation, go back and read the manual about it. If the software has 10 distribution disks or so, plan to spend about an hour for the installation. Be prepared to answer detailed questions about your system configuration: what port your printer is connected to, what kind of printer you have, what kind of video display you are using, and even how much memory (and what kind) you have.

As you do the installation, I recommend recording the serial number and the EXACT software version on the very first page of the manual. If there is more than one manual, I usually record the information in the one that contains the installation instructions so it is easy to find later. You may also want to keep the sales slip for the software in one of the manuals so that you know where the "proof of purchase" is if you need it. If you keep the sales slip in another file like I do, then you should also record the date of purchase on the first page of your manual, too. I also use a very soft point felt-tip pen (e.g., a Paper Mate Flair) to write that date on each of the distribution disks (that includes DOS disks, too). That can be important later if you get a disk that includes bug fixes or enhancements to your software, and it makes it easy to figure out which is the "latest" disk version. After the program is installed, it is usually best to power-off the computer unless you already know how to start and exit the program. More on that in a minute.

The installation procedure will take some time, and I usually plan to spend a total of 3-4 hours for the installation process. That includes looking through all the documentation, studying the installation procedure, and then doing it. In many cases, I will actually spend more time than that because I want to do special configurations. It is not unusual for me to spend an entire day or two doing the customization so that everything is exactly the way I want it. That is particularly true for the word processors I use most often. Now it's time for the real test: learning the new software.

Learning New Software

When you start learning any new software, the most important point is to READ THE MANUAL. Although you may think that is obvious, my experience is that many people seem to ignore the manual when they are learning new software. That is a bad mistake that can lead to considerable frustration and eventually result in a real dislike of your new and expensive software. To avoid that problem, I recommend an easier way that I have used to learn and teach a wide variety of

complex software packages. Here's how to do it.

Get your manuals, a pad of 8-1/2" x 11" paper, and a pencil; and go through the manuals page by page. The idea is to make notes about the software as you go through the manuals. That may sound like a lot of work, but it is incredibly effective in helping to learn the software. How many notes you make depends on your familiarity with the general application (e.g., word processing) and with that specific program. If you are learning a brand new application that you have never seen before, such as a spreadsheet, you may want to make a LOT of notes, especially on new or different terminology.

When I teach (or learn) new software, the very FIRST things I teach are: how to start the program from the DOS command prompt and how to exit the program (without saving data; i.e., ignore changes and exit back to DOS) back to the DOS prompt. There are few things more frustrating than starting a program and not knowing how to "stop" it. Then I talk about other ways to exit the program: save data and exit to the menu, save data and exit back to DOS, ignore changes and exit back to the menu, and any other exit commands that are appropriate. I also make the point that any data entered for any application is only recorded in memory; they are not saved to disk until an appropriate command is executed.

If you are familiar with much software, it is easy to see that virtually all software can be introduced this way. Learning how to correctly start and stop a program under all possible conditions is the first step. In order to keep all this information easily available, the very first thing you should do is make some notes about all these "start" and "stop" commands. What's next?

The next set of notes you make should be how to move the cursor around on the screen and in the file. The use of the arrow keys is usually obvious and depends on the application. Make notes how the keypad keys are used in their various combinations: alone, with the CTRL key, and with the ALT key, if applicable. What effect does the Num Lock and Scroll Lock have on the keypad functions, if any? What does the TAB key do?

Learning how to effectively and efficiently move the cursor around on the screen and in a file is essential. I have seen students using the PgUp and PgDn keys to get to the top and bottom of a file because they weren't taught that a word processor has simple commands for that. Knowing that the Home key moves you to the first cell in a spreadsheet is quite helpful. For a word processor, the Home key might move the cursor to the beginning of a line or to the top of the screen. And CTRL-Home may be used to move quickly to the top of the file (document).

Even if you don't immediately use or need all of the cursor movement commands, you will KNOW that they exist. And you can simply refer to your notes when you need it.

Once you know how to move around on the screen and in a file, it's time to look at the rest of the command keys, excluding the function keys. What do the INS and DEL keys do? How do you delete a character, word, line or cell from a file? What does the ESC key do? Are there any other special combinations for commands or other functions that you have overlooked, such as an "undo" or "cancel" command function?

At this point, you know how to start and stop the program, and how to move around on the screen and in the file. Since most of the rest of the command functions depend on what specific application is being used, I will use a word processor as an example application.

One of the major features of word processing is the capability to manipulate large blocks of text, and for that reason, I figure out the specific procedures and commands required for the block functions: copy, move, and delete. Virtually all word processors work in about the same way. First, you somehow mark a block of text (write down the procedure and commands for doing this), then you execute a command to copy, move or delete the block. Since many word processors also have an "undo" command that will recover a block deleted by mistake, I also make a note of that in this section of my list even though it may be duplicated somewhere else.

The next section includes all other special editing features that word processors have, such as search, search/replace, hanging indents, tab changes, forced page breaks, repagination, and anything else that I use frequently. At this point, I have all of a word processor's basic editing features included in my list. This information allows me to create a document and perform all editing tasks that I normally need to do. As you may have noticed, I have generally constructed my list in the order that I will need the information. Now I can get into various special features.

After a document is basically written, a spell checker is handy, and those commands are next. If a thesaurus is included with the word processor, that command is included here, too. Any other special commands, such as block math calculations and the on-screen calculator (such as in WordStar 4.0 and later), are also in this section.

All print-related features are part of this next section, including the commands/procedures for printing a document. All commands and procedures for implementing print enhancements (e.g., bolding, underlining, line spacing, font changes, etc.) are included here, too.

The last section includes other special features, such as how to create headers, footers, page numbering, and so on. Most word processors now provide the capability to generate a Table of Contents and Index, and this section includes information on how to mark those entries and the commands used to create a Table of Contents and Index.

The general organization of my "cheat sheet" is to go from the most often needed information to the least often needed information. For a word processor, I begin with information needed for writing and editing a document. Then, I find out how to add print enhancements and print it. And finally, I list any other special features that are not used too often.

If you take the time to create your own cheat sheet for software before you fire up your computer, I think you will be amazed at how much easier it is to learn any software. For me, this approach makes complex software much easier to learn because I have already become familiar with its terminology before I even begin to use the program. I have found this trick to be useful whether the software is a word processor, spreadsheet or compiler.

After learning the basic commands that a program uses, another way to become proficient in its use is to find a book on it. Regardless of how good the program or tutorial is, a book showing practical examples of how to do things is nearly indispensable. If you are learning Microsoft Word, for example, I particularly like Peter Rinearson's book "Word Processing Power with Microsoft Word" (Microsoft Press). There are also a number of good books available on WordStar and Word Perfect, too. In addition, you will find most of these books include a number of excellent and practical suggestions and examples for using most features. Using one of these books is an outstanding way to supplement the information provided in the software's tutorial.

If your software has a program-driven tutorial, use it. If your software has lessons in the manual, try them. Liberal use of the manual's Table of Contents and Index can be especially helpful during the learning process. Learning new software is not especially easy, and it does take time.

After you have developed your own cheat sheet and worked with the software a little, go to a bookstore and find a book on it. At that point, you will probably have come up with some general questions on how to do different things. With those questions in mind, browse through the various selections until you find a book that answers most of your current questions — then buy it. Chances are you will find that it will also answer your future questions too because its approach has already included your current ones. After

a while, you may also find that you need a second book because you have learned the basics and are ready for the more advanced features.

Unfortunately, I have noticed that many people get frustrated with new software after working with it for an hour or two. That is barely enough time to get it installed and running, let alone becoming proficient with it. A lot of today's software is so sophisticated and complex that it takes many hours of working with it to become proficient and expert in its use. Don't expect to become an expert with one of these packages, like Microsoft Word, in five or ten hours. It takes a lot of reading (the manual), time, and hands-on experience to understand how to use the power of this kind of software.

Powering Down

Selecting the "best" software is a difficult task, and many people spend hundreds of dollars on software before they find something that really fits their needs. Like many things, your choice of software is really a matter of personal preference. It's best to start with something cheap (so you can discard it later), especially when you don't know much about the general application. Then, you will know enough about an application's usual features to help make an intelligent choice of a more complex and powerful program. You can develop your own evaluation checklist, but always take some time to see how that application actually works before you buy it.

When you finally select an expensive program that you think will fit your needs, don't expect to become proficient with it overnight or in a week or two. Read the manual, and spend some time developing your own cheat sheet by going through the manual page by page. In addition to learning the basic software commands, this approach has the added advantage of helping you understand the organization of the documentation so you will know where to look for something that is not on your cheat sheet. If you follow some of the suggestions in this article, I think you will be pleased with your software because you also have seen one idea on how to learn and become proficient with it much faster. In today's fast-track world, many people expect that using a computer is just a matter of sitting down and pushing some keys. It isn't. A fast-track method of learning just hasn't been invented yet.

Next Time

This article has been an introduction on how to select and learn basic application programs. In the next article, we will look at some basic kinds of utilities that you will find helpful in organizing and using your computer.

If you have any questions about any-

Continued on Page 27

***** Z-100 SERIES SOFTWARE *****

PART NUMBER	DESCRIPTION	LIST PRICE	SALE PRICE
MS-463-1	Z-Basic (16 bit)	\$175.00	\$12.00
MS-463-7	Multiplan	\$195.00	\$12.00
MS-253-1	Basic-80 (8-bit)	\$175.00	\$12.00
CD-463-2	Condor File Manager	\$299.00	\$12.00
LT-Z100	All 4 Listed Above	\$819.00	\$40.00

***** IBM COMPATIBLE SOFTWARE *****

PART NUMBER	DESCRIPTION	LIST PRICE	SALE PRICE
MS-5063-30	Microsoft Windows	\$ 99.00	\$ 24.00
NU-413	Norton Utilities Adv.	\$150.00	\$ 99.00
WP-528	WORDPERFECT 5.0	\$495.00	\$269.00
BO-290	QUATTRO	\$239.00	\$179.00
CP-311	PC TOOLS DELUXE	79.00	\$ 68.00

***** ZENITH LAPTOP COMPUTERS *****

SUPERSPORT 184-1	2 3 1/2' Floppy Drives, 640K RAM	\$1587.00
SUPERSPORT 184-2	1 Floppy, 20 Meg Hard Disk, 640K RAM	\$2373.00
SUPERSPORT 286-20	12/6 MHz, 80286 CPU, 3 1/2' Floppy, 20 MEG Hard Disk	\$3292.00

***** VIDEO MONITORS *****

ZCM-1490	ZENITH Color Flat Screen VGA	\$718.00
MA2565	SAMSUNG Amber TTL 720x350	\$89.00
CW4644	SAMSUNG Color RGB 640x200	\$274.00
CM4531	SAMSUNG Color EGA 640x350	\$389.00
CN4551	SAMSUNG Multi-sync VGA 800x560	\$489.00
NC800	NEC Multi-sync II 800x560	\$639.00

***** ZENITH PC COMPUTER UPGRADES *****

SmartWatch from FBE Research Installs in ROM Socket on CPU Board in Zenith computer series Z-100/138/148/150/160. This clock/calendar contains a ten year battery and keeps your computer informed of both time and date at each boot-up. Instructions and software included. \$38.00

Z-150 Series Hard Disk Drive Kit Includes new generation High Speed (28 MS) Seagate Drive with Auto Park heads. Each kit is complete with controller card, cables, hardware and instructions to mount the Hard Disk under your two floppy drives in the Z-150 series computers. 32 MEG ST-138/150 Kit \$383.00

Z-148 Hard Disk Drive Kit Includes the Hard Disk Drive and controller in the kit above plus the Z-148 Expansion card described below. Each kit includes all cables, hardware and instructions required to replace one floppy drive with a high speed low power Hard Disk Drive. 32 MEG ST-138/Z-148 Kit \$459.00

ST-138/Z-148 Kit With SmartWatch \$489.00

Z-148 Expansion Card adds 2 IBM expansion slots \$79.00
with SMARTWATCH clock/calendar. \$109.00

INTERNAL MODEM Fully Hayes compatible (software included)
1200/300 baud \$94.00
2400/1200/300 baud \$159.00

EXTERNAL MODEM Fully Hayes compatible (software included)
1200/300 baud \$137.00
2400/1200/300 baud \$199.00

VCE 150 Video Eliminator for Z-150
Allows use of EGA or any video card. Required memory chip included. \$54.00

V-20 Chips High Speed NEC V-20-8 8088 replacement. These run at up to 8 MEG and are said to increase CPU speed 10-30% \$14.75

***** Z-100 SERIES COMPUTER UPGRADES *****

High Density 1.2 Meg Drives. External floppy drive set-up complete with drive, power supply, case and cable. Ready to connect to your 8" floppy controller Single Drive Unit 234.00 Dual Drive Unit \$351.00
Bare Drive and Cable for internal mount \$136.00

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

Gemini Emulator Board. Makes the Z-100 compatible with the IBM PC library of programs. \$432.00

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

UCI Easy87. Add an 8087 Numeric Coprocessor. \$69.00 for the board without an 8087 Chip. With 5 MEG 8087 \$188.00 or with 8 MEG 8087 installed \$234.00

ZMF100A by FBE Research. A modification package which allows 256K chips to be used on the old-style motherboard to reach 788K. Simple assembly with no soldering or trace cutting. Compatible with Easy PC and Gemini Emulator. . . \$60.00
Requires 27 256K RAM chips to complete the kit.

UCI Memory Upgrade Pal Chip Set For the Z-100's with the newer motherboard part number 181-4918 or greater. This chip set allows the installation of 256K RAM chips on the motherboard. With the addition of 27 256K RAM chips, a total memory of 788K is obtained. PAL Chip Set \$64.00

UCI Memory Upgrade Card We recommend this one highly. The board has sockets for up to 2 MEG of RAM. With no RAM installed \$288.00. Add \$35.00 for EasyDrive RAM Drive Software if desired. Either 64K or 256K RAM chips may be used to complete this kit.

UCI EASY-I/O S-100 board that provides IBM PC communications port compatibility with your EasyPC. Easy/I/O-1, One Serial Port \$81.00. Easy/I/O-2, Two Serial Ports, One Game Port, Clock-Calendar \$127.00

UCI EasyWin Winchester Drive Systems at reasonable prices. Complete Hard Disk Systems for mounting inside your Z-100. Systems complete with Seagate Drives, 21 MEG \$598.00, 31 MEG \$634. System without Drive \$317.00

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

***** FLOPPY DISK DRIVES *****

MITSUBISHI MF501	5.25" 48 TPI DS/DD 320K/360K	\$ 89.00
MITSUBISHI MF504	5.25" High Density 360K/1.2 MEG	\$106.00
MITSUBISHI M-353	3.5" in 5.25" frame 720K	\$98.00
MITSUBISHI M-355	3.5" in 5.25" frame 1.44 MEG	\$129.00
	M-355 Software Driver	\$ 19.00

M-355 runs on AT compatible or special controller only.

**** SEAGATE HARD DISK DRIVES *****

ST-125	21 MEG, 28 MS, Auto Park Heads With Controller & Cables	\$ 275.00 \$ 329.00
ST-138	31 MEG, 28 MS, Auto Park Heads With Controller & Cables	\$ 329.00 \$ 383.00
ST-238	31 MEG, 65 MS, RLL With RLL Controller & Cables	\$ 258.00 \$ 309.00
ST-251	42 MEG, 40 MS, Auto Park, Software With Controller & Cables	\$ 384.00 \$ 438.00
ST-251-1	42 MEG, 28 MS, Auto Park, Software With Controller & Cables	\$ 465.00 \$ 519.00
ST-4096	82 MEG, 28 MS, Auto Park, Software.	\$ 647.00

V-20 Chips High Speed NEC V-20-8 8088 replacement. These run at up to 8 MEG and are said to increase CPU speed 10-30%..... \$14.75

*** PAYLOAD CUSTOM ASSEMBLED COMPUTERS ***

We can assemble 8088 XT, 80286 AT or 80386 IBM compatible computers to your specifications. We use high speed turbo motherboards with Phoenix BIOS. Other features include clock/calendar, Keytronic 101 key keyboard, Mitsubishi floppy drives, Seagate hard disks, front panel mounted turbo and reset buttons, memory and I/O ports, all to your specification. Please write or call for price information. We can send you a work-up sheet showing items available and prices.

Example #1
IBM XT Compatible 8088 10 Meg CPU, two 360K floppy drives, 640K RAM, one serial port, one game port, two parallel ports, 101 keyboard, clock/calendar, 10 Meg CPU and amber TTL monitor \$831.00

Example #2
Same as above with the two floppy drives plus a Seagate ST-238 30 Meg RLL hard disk drive \$1126.00

Example #3
IBM AT Compatible 80286 12 Meg CPU, 2 Meg RAM, clock/calendar, one each serial, game and parallel ports, one 360K floppy, one 1.2 Meg floppy, 101 keyboard, Seagate ST-251 40 Meg hard disk, EGA color monitor \$2234.00

MS-DOS 3.3 Operating System - \$85.00

**PAYLOAD
COMPUTER SERVICES**



15718 SYLVAN LAKE
HOUSTON, TEXAS 77062
PHONE (713) 486-0687



Your satisfaction is guaranteed. All hardware carries a 90 day Payload warranty. VISA and MASTERCARD orders welcome with no surcharges. Add \$5.00 to all prepaid orders for handling and shipping in Continental USA, we pay the balance. Actual shipping costs for foreign, overseas and net billing orders to approved accounts. We accept purchase orders from schools, government and approved accounts. Mail or Phone your order for prompt service. Texas residents please add 8.0% state sales tax.

SOFTENING THE HARD DISK

In the last article, we looked at some of the inconveniences of DOS and started the process of setting up a hard disk in a way that minimizes them. Having completed formatting, partitioning and subdirectory creation, we now go on to rename the subdirectories we created.

The SUBST program does just that; but at the same time, it converts the subdirectory names to disk names, converting at a stroke the hierarchical subdirectory structure native to DOS into a flat list. The single-letter subdirectory names we created last time, from \g to \z, emerge as drives g: to z:, and save us typing CD \ every time we access one of them. And help stop some programs, like early versions of WordStar, from finding their overlays out of reach. And perhaps reduce the need to get expensive file, FAT and Directory recovery programs, and backup programs, and defragmentation programs. And generally, reduce the angularity of DOS. And maybe speed it up into the bargain.

These substitutions remain in force only until you power down the computer, like the assigned drive-letters of the partitions, so they must be made each time you want them functional. The commands may be placed either in AUTOEXEC.BAT or in another batch file, for occasional use, or used on the fly. As I was going to use them permanently and exclusively and for all the subdirectories, I made up a separate file, substdisk.bat in program Listing 2, and call it from AUTOEXEC.BAT.

The command takes the form:

```
f:subst 0: d:\o
```

- f: The name of the disk holding the program SUBST,
- O: The new (disk) name of the subdirectory,
- d: The name of the root directory, here the partition,
- \o The original name of the subdirectory, used in the MD command.

The batch file is a list of exactly similar lines, with the o's replaced by the other letters you are using, from \g to \z, and the d: replaced by the root directory in question, from c: to f: in the case of four partitions. You also replace the f: with the name of the disk where you have the program SUBST. The subdirectories \k, \n and \z end up, for instance, as the drives k:, n: and z:.

Program Listing 2: Substdsk.bat
Substituting drive letters for subdirectories

```
c:subst g: c:\g  
c:subst k: c:\k
```

```
c:subst t: c:\t  
c:subst x: c:\x  
c:subst m: d:\m  
c:subst p: d:\p  
c:subst q: d:\q  
c:subst r: d:\r  
c:subst v: d:\v  
c:subst w: d:\w  
c:subst y: d:\y  
c:subst h: e:\h  
c:subst n: e:\n  
c:subst o: e:\o  
c:subst z: e:\z  
c:subst i: f:\i  
c:subst j: f:\j  
c:subst l: f:\l  
c:subst s: f:\s  
c:subst u: f:\u
```

PART II

JOHN A. NEGUS

07150 VALLON PONT D'ARC
FRANCE

There are two minor complications to watch out for, the total number of drive letters you want to use and the order in which the 'drives' are created.

If you want more than five drives, you have to warn DOS by means of the line:

```
lastdrive = z
```

in your CONFIG.SYS file, replacing the z with the letter relevant to your setup. This opens up for use all letters through to z as available to DOS. I use z, keeping several subdirectories spare, unoccupied. That is easier than coming back later and modifying things. It also gives convenient temporary storage for defragmenting files.

DOS keeps tabs on the number of drive letters in use and will, if asked, automatically grab and allocate the next available drive letter without the user having to specify it. The Epson driver quoted above uses this characteristic; DRIVER.SYS, useful for non-standard floppies, does too. You have to watch out for this when composing config.sys and when planning your subdirectories, remembering that drive letters can be 'consumed' in both CONFIG.SYS and AUTOEXEC.BAT. Place the command lines for these devices that consume the next available drive-letter in the order that gives you the letters you want in the places you want.

Programs like ramdisks that require you to name the drive letter to use pres-

ent fewer problems. If you call a ramdisk program in your AUTOEXEC.BAT file, specify the letter to use after taking account of the contents of CONFIG.SYS, which is executed first, and according to any substitutions you initiate in AUTOEXEC.BAT.

I hope one day to add a 1.44 Megabyte floppy, which will require the use of driver.sys, so I have left a vacant drive letter in the right place in the chain, (in config.sys) so that DOS will use that and spare me the trouble of reconfiguring the whole system. You could, for instance, choose to keep the name d: for a floppy by placing the disk partition command line after the driver.sys command that sets up d:, so the partitions would become e: to g:. If you reverse the order, the floppy will appear as g:, the first name free after the four partitions. Equally, if the configuration switches on the main board of a PC are set for three floppies, DOS will automatically call your hard drive D:, not C:.

Most of what is written above supposes that C: is the hard drive, simply because that is the configuration the majority of users are likely to adopt. If you find inconsistencies in the lettering in these articles, you can ignore them, and the three previous paragraphs, if yours is a simple one- or two-floppy system with one hard drive: your hard disk will be C: and the three additional partitions will be D:, E: and F:. If your config.sys or autoexec.bat files already have lines in them creating partitions or floppy drives or ramdisks, or if your hard disk is the sort that you cannot boot from, your final configuration depends on where you add any new lines, before or after those. Try altering the order of the lines to get the configuration you want; you may have to alter the letter a ramdisk is known under, if you find one in autoexec.bat.

Once you have put the substitutions into force, you can view the current state of the game by typing SUBST alone, without parameters; the program announces its findings with lines like:

```
S: => G:\S
```

=> there has the meaning 'takes the place of'. To cancel an existing substitution, type SUBST plus the drive name plus /D. SUBST z: /d, for instance, deletes the substitution on z:, leaving z: in its pristine state, in my case, as the subdirectory \z.

While most programs have no difficulty at all handling substituted drives through z:, unfortunately, some of the DOS programs do. Substituted drives are unsuitable game for the programs Assign

(the new one), Backup, Diskcomp, Diskcopy, Chkdsk, Fdisk, Format, Join, Label and Restore. The fly in the ointment comes from the same tube — DOS's own auxiliaries. Nevertheless, I have not been able to test more than a very limited number of applications programs and you should regard each program you use with suspicion, until you have tested it to your satisfaction.

The substitutions do not prevent accessing subdirectories by the normal means, so programs set up to do so should work as before.

If you wish to use any of the 'difficult' programs, it is quite easy to undo the substitutions temporarily, see program Listing 3, unsubst.bat, and then reinstate them afterwards, by running program Listing 2, substdsk.bat, again. In fact, it is not a bad idea to create all the batch files you will need at the same time, using the replacement capabilities of your word processor to write three batch files all at the same time — one for initial subdirectory creation, one for setting up the substitutions and one for undoing the substitutions — it saves a lot of writing.

Program Listing 3: Unsubst.bat
Cancelling the substitutions, restoring the subdirectory names

```
c:subst g: /d
c:subst k: /d
c:subst t: /d
c:subst x: /d
c:subst m: /d
c:subst p: /d
c:subst q: /d
c:subst r: /d
c:subst v: /d
c:subst w: /d
c:subst y: /d
c:subst h: /d
c:subst n: /d
c:subst o: /d
c:subst z: /d
c:subst i: /d
c:subst j: /d
c:subst l: /d
c:subst s: /d
c:subst u: /d
```

It is not necessary to put in the leading c:, or its equivalent, if you always run it from the default disk, but it does no harm and it allows you to run the program from anywhere, when another disk is the default, or to get it executed by a DOS front-end program, for instance.

Having created the subdirectories and substituted drive letters for them, you can "fill 'em up", copying your files from your floppies to their new destinations. Again, some care now will save time later. The ideal is to put all the files which will not change size, like programs, into place first, and this for all the 'disks'. Remember that the different 'disks' sharing the same partition share exactly the same part of the disk surface. The ideal is to place the files that will change size 'above', or after,

those that won't; this in order to reduce the number of files you have to defragment, or copy out in order to achieve defragmentation, for you do not have to disturb the bottom layer of unchanged program files. So, first copy all your programs and other unchanging files into their different destinations before copying any of the data files and others that will change size.

If you add a constant-size file to a disk 'on top of' data files that are changing size, you are effectively placing an obstacle in the way of defragmentation. One or two obstacles of this sort are obviously of no importance, but if the number is large, simply copying data files out and back will not be enough.

To defragment your files that change size, if you haven't got a special program, you need first to copy them out to another location, be it floppy or hard, then delete them on the original surface and then copy them back — the equivalent of a move, not just a copy. If you want to know where the 'pad' of fixed length files ends and the later additions start, the command DIR, or DIR /W, will give you the information, for it lists files effectively in the order in which they were placed on the disk. This order is not to be relied upon, though, where you have deleted a file and created, or copied in, a new one later: the new one will take the first directory slot that is available, be it virgin or 'occupied' by the no longer wanted details of an erased file, and DIR's display will be no guide to the file's placement on the disk surface.

How do you know what sort of state your disk, or any file, is in — when, in short, defragmentation is necessary? The ideal would be a way of generating reports on both the state of file fragmentation and the space occupied by each subdirectory, this last for checking if there is enough space on the target disk for the copied material. Preferably quickly, and written either to a file or the printer.

No problem. The shortness of the disk names in effect allows you to specify them all in the command line to a batch file and 'shift' through them one at a time, carrying out any operation you want. In this way, you can have reports on the state of all your files and all your subdirectories in the matter of a minute.

Program Listings 4 and 5 show a pair of files that does this, allowing a choice of operation in the bargain. You specify a DIR of all subdirectories, for instance, by typing the command:

```
Repeat dir
```

The batch program REPEAT.BAT picks up the command, reads the parameter 'dir' and calls REP.BAT with the requisite command line parameters: first, the operation; second, all the drive-names to be operated upon. REP then cycles through the operation, plugging in each

target disk name in turn.

The only complication is the renegade programs, like chkdsk, that can't handle substituted drive names; this necessitates a return to the bad old days, specifying both root drive and subdirectory name, and incidentally complicates the programming so much that it seemed more trouble than it was worth; I hard-coded the chkdsk routine in REP.BAT. Repeat, therefore, sends the CHKDSK routine no drive parameters, only the operation specifier, ck, in this case. The chkdsk code, unsubst and substdsk apart, is an image of the action that the other sections carry out, but in their case, they do it by following not a simple list of commands, but the program loop.

A hard disk contains so many files that I chose to put the information output by these programs into disk files rather than on to the screen. You can replace the file name following the >> sign with PRN, for output to the printer, or any valid DOS device name. If you do so, you may want to alter >> to >, for the former means "append to the end of", while the latter is just "send to". Alternatively, you can remove both the > sign (or signs) and the file name and have the report on screen.

Not being a great fan of the DOS BACKUP program, I try to keep all subdirectories no larger than a single floppy, 720k at present, perhaps 1.44M in the future. This explains the choice of XCOPY (you can use COPY in its place) and allows easier handling of individual files. For I back up important files immediately after changing them, unimportant ones, and fixed-size files, at intervals. It is so easy to do that it seems a mistake to rely on infrequent, easily forgotten and long and boring whole disk BACKUPS. The effect of the /m parameter is to mark the file as backed up; if you do not rewrite the file, xcopy will know it has not been modified since the last time and will not waste time copying it again.

The xco routine is designed to use two floppy drives, instead of one, for speed. If you only have one floppy, delete the lines in the xco section of rep.bat starting with the first 'shift' and down to, but not including, the second 'shift', and alter the B:, if necessary. Equally, you can omit a source drive, if say it is empty or if it is too big to fit on a floppy or if it contains nothing but unchanged files, as I've done in the xco part of Repeat.bat.

As I try to keep the contents of each subdirectory down to one floppy in size, I need a means of knowing if there is any dead wood taking up space on my floppies. The line:

```
for %%f in (*.*) do if not exist %1:%%f
dir %%f >> b:delta.dir
```

reads each file name in turn on the default floppy disk, checks to see if it exists on the current hard subdrive, and if

not, writes its name to the file delta.dir.

As the programs are usually running on a Ramdisk, I take care to let the batch file copy the output files to terra firma, before forgetting. Using delta.dir, it is then easy to move or delete the unwanted files from the floppies, freeing up space.

The SD option in the program listing is a public domain program you will find on the HUG Bulletin Board that gives a four-across directory display, with file sizes in k, the total space, both occupied and remaining, and the disk name. The file the routine produces is an excellent disk catalog, in conjunction with a word processor, as well. This is a version of SuperDirectory that I modified to handle hard disks correctly, recognizable by its date, 4/19/88 and its size — 1676 bytes.

Using the models presented, you can easily add any operation that you want. For each new operation it is enough to add four elements — a command parse line (if xx=yy goto zz) in each of the two files and an execution section in each of the two files — the part lying between, and including, the relevant label (recognizable for starting with a colon) and the 'goto end' line.

The colons at the start of the early lines of Repeat.bat are a trick allowing comments in a file that will not display, ever. Colons are the reserved symbols required at the beginning of a label; the most DOS does with them is read the contiguous word, and then only if it is searching for a label. To be able to read the 'colonized' comments you need to either TYPE them or view them with some sort of program. If you have not already got one, I strongly recommend some sort of DOS front-end, or shell, program. I use Vfiler 2.8, Vf150, in fact, a public domain program no more than 12k in size which allows viewing, executing, copying, renaming, deleting files (singly or the ones you mark for batch operations), plus easy changes of subdirectory and other functions. Irreplaceable. It, and its associated .doc file, is available on the HUG PBBS, in a version modified to remove one or two incompatibilities with DOS 3. It, or a similar program, makes it easy and quick to read the comments in batch files. The competition demands 90k and \$90 for the same functionality. A word of thanks to Mike Nice and the others responsible.

Vf150 has got a peculiarity that is worrying when you first meet it, but of no importance once you know it. When it is waiting for your command, with the list of files on the screen notice the position of the cursor — roughly mid-screen and three lines up from the bottom. If ever you give it a directory to read, on first loading it up that is empty, the program does not clear the screen or display anything (like an error message), but simply places its cursor at the usual position, waiting for your input. To correct the

Program Listing 4: Repeat.bat

The first of the Repeat pair, and the one which selects and specifies Rep.bat's task and parameters

```
echo
: A file to repeat a specified operation on the series of disks from c: to z: .
: the possible operations are:
: - chkdsk all (sub)directories & report in the file ckh.out
: - dir all subdirectories
: - sd all filenames into the file Hdfiles
: - xcopy all files to a: and b:, resetting the archive bits, and
: list any superfluous files on the floppies to the file delta.dir.
: The default disk should contain this file plus Rep.bat and any
: program required. The output files do not have to exist already on
: the default disk; if they do, the new material may be added to
: the end of them.
:
: Call as
: Repeat op
: where op is any one of ck, dir, sd, xco.
:
echo on
if %1==ck goto ck
if %1==dir goto dir
if %1==sd goto sd
if %1==xco goto xco
echo on
echo Problems with the parameters
goto end

:ck
rep ck

:sd
rep sd c d e f g h i j k l m n o p q r s t u v w x y z

:dir
rep dir c d e f g h i j k l m n o p q r s t u v w x y z

:xco
echo Xcopy of all drives except F:
rep xco c d e f g h i j k l m n o p q r s t u v w x y z

:end of listing
```

problem, enter either the drive letter you want the directory of, or, if that does not work, L, for login, followed by the drive letter you want the directory of. From there on everything will return to normal.

If you want the comments in the batch files to appear during execution, alter any colon followed by a space to REM and make sure echo is on. I turned echo on for debugging purposes, off when the programs worked all right; that way you have an uncluttered screen during execution. Echo is a variable that keeps its last set value, even while no batch files are running, so the way you leave it at the end of one file affects how it will behave at the start of the next. You can, of course, delete the entire 'echo' line, but programs always seem to need modifying at some time or other . . .

Colons are also a handy means of modifying batch files quickly, converting a command into a label, so preventing its execution. If you have a file zap program, you can quickly overwrite the first letter of say a Pause line with a colon and it will no longer be executed. It saves loading a word processor or deleting the line. The

SD portion of rep.bat gives an example.

There are so many programs involved with these two batch files that I have made up a third that does nothing but set the scene, copying all the needed files on to the same disk and setting it as default. The same file, whose make-up is left to the reader's imagination, could if you want type reminders of the options and the syntax. A painless way to get it to do so is to include them as REM comments at the end of the file.

Among the required programs, which might need copying into place, is COMMAND.COM, needed for loading and executing one batch file on top of another, here the SUBST batch files on top of REP. Calling a second batch file from a first without the COMMAND /C command will leave you back in DOS afterwards instead of returning to and continuing the execution of the first file. (DOS 3.3 has a 'call' command which avoids the need for COMMAND /C). One advantage of DOS 3.2, however, over 2.11 is that you can COMSPEC a ramdisk, again relieving your hard disk of unnecessary wear and tear, so there is no need for it to be present on

Program Listing 5: Rep.bat

The second of the Repeat pair, the one which does the work.

```
echo on
rem This is the subsidiary file called by Repeat to do its bidding.
rem It, Repeat.bat and any program needed must be on the default disk.
rem See Repeat.bat for instructions for use.
echo on
if %1==ck goto ck
if %1==dir goto dir
if %1==sd goto sd
if %1==xco goto xco
echo on
echo Problems with the parameters
goto end

:ck
Echo Checking the four root directories on the hard disk.
chkdsk c:.* /f > ckh.out
chkdsk d:.* /f >> ckh.out
chkdsk e:.* /f >> ckh.out
chkdsk f:.* /f >> ckh.out

Echo Checking the sub-directories on the hard disk.
command /c unsubst
chkdsk c:\g\.* /f >> ckh.out
chkdsk c:\k\.* /f >> ckh.out
chkdsk c:\t\.* /f >> ckh.out
chkdsk c:\x\.* /f >> ckh.out
chkdsk d:\m\.* /f >> ckh.out
chkdsk d:\p\.* /f >> ckh.out
chkdsk d:\q\.* /f >> ckh.out
chkdsk d:\r\.* /f >> ckh.out
chkdsk d:\v\.* /f >> ckh.out
chkdsk d:\w\.* /f >> ckh.out
chkdsk d:\y\.* /f >> ckh.out
chkdsk e:\h\.* /f >> ckh.out
chkdsk e:\n\.* /f >> ckh.out
chkdsk e:\o\.* /f >> ckh.out
chkdsk e:\z\.* /f >> ckh.out
chkdsk f:\i\.* /f >> ckh.out
chkdsk f:\j\.* /f >> ckh.out
chkdsk f:\l\.* /f >> ckh.out
chkdsk f:\s\.* /f >> ckh.out
chkdsk f:\u\.* /f >> ckh.out
copy ckh.out y:
command /c substdsk
goto end

:dir
shift
if "" == "%1" goto end
pause When you are ready for the next dir,
dir %1:/w
goto dir
goto end

:sd
shift
if "" == "%1" goto end
:aue When you are ready for the next dir,
sd %1: >> hfiles
goto sd
copy hfiles y:
goto end

:xco
shift
if "" == "%1" goto cop
A:
pause Xcopying %1: to A:. When the disk is ready in A:
xcopy %1:.* A: /m
for %%f in (*.*) do if not exist %1:%%f dir %%f >> g:delta.dir
shift
if "" == "%1" goto cop
B:
pause Xcopying %1: to B:. When the disk is ready in B:
xcopy %1:.* B: /m
for %%f in (*.*) do if not exist %1:%%f dir %%f >> g:delta.dir
goto xco
```

```
:cop
copy g:delta.dir y:
goto end

:end
echo on
:end of listing
```

the default disk. The line:

```
SET COMSPEC=Z:\COMMAND.COM
```

in autoexec.bat tells DOS to load COMMAND.COM, whenever needed, from disk Z, even if Z is a ramdisk. Intelligent. And quick.

There's many a slip 'twixt the cup and the lip. Most readers will have to customize at least one of these programs before they will work satisfactorily, due to the inevitable differences between systems. None of the readers of REMark ever introduce bugs into their programs, I know, but should something not work as intended, there are several things to check before painting an author's ears red.

First, check your partitions — that they are in place, that their names are correctly entered in all the programs, wherever you find C: to F:. Next, check the subdirectories — that they are created correctly, that the substitutions take place without problems, that the lines of disk names in repeat.bat are correct, that source disks are correctly named, as in substdsk. And finally, that any target disks for output files written by Rep.bat are correct. The most likely alteration necessary is that you have a different list of drives available: if this is the case, be careful to make the changes in each of the five files. One of the most likely errors is to leave DOS searching for a program which is not on the default disk. Another is to have saved your batch file in normal word processing format, instead of in ASCII.

When you are ready to try the programs out, or have made any change necessary, start by running the Repeat Dir command, which is the most tolerant of errors, making sure that echo is on. You will, amongst other things, see how the IF "" == "%1" works. If changes prove necessary, again make them in all the relevant files. Once you have gotten the programs running correctly, you have powerful addition to your operating system.

The answer to the puzzle posed earlier, the number of the first FAT slot belonging to the third of the files — but you won't need me to tell you — is 33.

With the help of DOS and a few batch files you can: tame your hard disk; make it easily understandable by novices; save yourself a great deal of typing, perhaps some money, certainly some disk space; render the housekeeping chores easier and have reports on every nook and cranny of your hard disk — from A to Z — and cock a quiet snook at your DOSo-disko-sado-macho the while. He would have to write, or buy, an extremely complex program in order to be able to do what you can now do. *

Printing With Hewlett-Packard's DeskJet Using WordPerfect Version 5.0 A "Demonstrative Review"

William N. Campbell, M.D.
1063 Green Glen Drive
Boothwyn, PA 19061

This sentence is printed with a font of Times Roman 10 point proportional print and is a test to demonstrate how it appears. **This text is bolded because the font was changed to Times Roman 10 point bold.** *Now, the font was changed to 10 point italic and this is printed in italic mode at this point. The font has been changed again to Times Roman, Bold, Italic, and 14 point. Therefore this is printed in BOLD, and ITALIC and 14 POINT. Note that line spacing of text, and keeping different size text between the same left and right margins are automatically adjusted whenever there is a change in font point size!* The font has been changed again and this is printed in 10 point Times Roman and is the same as the first printing done in this paragraph. The only thing I have done, other than entering text, is to insert a new "font code" before each font change by typing a Ctrl-F8 (which is the "Font" function key), and then select the desired font from a list of fonts that are displayed. I typed from the beginning of a paragraph to the end of that paragraph before I pressed the Enter (Return) key - WordPerfect "word wrapped" automatically at the end of each line to the beginning of the next line and this produced what you see. Note that I purchased and am using an accessory Times Roman Proportional Spacing Cartridge in this

printer which is a Hewlett-Packard DeskJet. Incidentally folks, these font changes are made in software in computer memory - initially they are all within integrated circuits within the easily snapped-in cartridges. Hewlett-Packard offers numerous cartridges, and you can easily insert any one or two at the same time. WordPerfect 5.0 supports IBM and true compatibles such as my H386.

(Note that this paragraph is the only paragraph in this review that is printed in fixed pitch. Therefore, I have turned hyphenation on for this section only.) If you use your printer mainly for writing letters, and memos, consider using the DeskJet's internal Courier fixed pitch fonts as shown in this paragraph. They include 5, 10, 16.67 or 20 characters/inch (pitch) in either 6 or 12 point size; in draft or letter quality mode; plus some optional embellishments offered by WordPerfect 5 or other word processor. For example, if you desired to print a spreadsheet you might opt for this 16.67 condensed pitch in 12 points.

You just tap a couple of keys on your keyboard to select different fonts as you desire them, whenever you want to jot down a memo or write that masterpiece! *True "Letter Quality Typing" and/or "Desktop Publishing" have just become quite affordable, and easy to accomplish!* Here are other font size and

appearance demonstrations. This sentence is printed with a font of Times Roman 12 point proportional print and is a test to demonstrate how it appears. **This text is bolded because the font was changed to Times Roman 12 point bold.** *Now, the font was changed to 12 point italic and this is printed in italic mode at this point. The font has been changed again to Times Roman, Bold, Italic, and 8 point. Therefore this is printed in BOLD, and ITALIC and 8 POINT. Note that line spacing and margins are automatically adjusted whenever there is a change in font point size!* The font has been changed again and this is printed in 6 point. The only thing I have done throughout this text is to insert a new "font code" before each font change by typing a Ctrl-F8 which is the "Font" function key, and then select the desired font from a list of fonts that are present. Note that I purchased and am using an accessory Times Roman Proportional Cartridge in this printer which is a Hewlett-Packard DeskJet. Hewlett-Packard offers numerous other font cartridges for the DeskJet.

THIS IS PRINTED IN 14 POINT BOLD STYLE. The fonts available in this cartridge are 4, 5, 6, 7, 8, 10, 12, and 14 points in size; in addition, each point size is available in "normal", bold, italic, or bold and italic appearance. **ALSO, WORDPERFECT 5.0 ADDS SOME OF ITS OWN STYLES!**

(The following paragraphs are printed using a Helvetica 10 point font which is available in a separate cartridge with identical point sizes and attributes as described above in the preceding paragraph.)

The DeskJet has gotten outstanding reviews. It is a joy to use. It is **exceptionally quiet**, and will fit right beside your computer since it has an extremely small footprint (17" wide, 15" deep and 8" tall, and it weighs under 15 pounds.) It has both serial and parallel connections and is very easy to set up. It works well with all the types of paper I have tried, including letterhead bond, ordinary "house bond" (cheap white paper that you can buy from paper distributors - usually at the lowest price), and what HP recommends, good grade "copier" paper. It takes either 8 1/2 by 11 or 8 1/2 by 14 paper and also prints addresses on standard #10 business envelopes. (This is an ingenious mechanism as the paper does not need to be removed to print an envelope.) The paper is loaded from the front (up to 100 sheets at a time) and also ejected into a separate tray in the front of the printer. Nothing could be easier. It is a clever, and quick paper handling system. The printer does not handle continuous forms, incidentally. I have found it pleasant to be able to forget about removing the perforated carrier strips of continuous forms.

The DeskJet uses **no ribbon**, and the ink is contained in easily installed cartridges. The ink jet nozzle is built into each cartridge, so that when you install a new ink cartridge you automatically are installing a new nozzle. I have had absolutely no problems with the ink distribution system and I have never had a paper jam since I've been using the printer! It is actually faster and easier to install a new ink cartridge than to change a ribbon on most other printers. Also, I have yet to get an ink smudge on my fingers when changing cartridges. Although Hewlett-Packard conservatively rates a cartridge to print 500 to 800 sheets of 8 1/2 X 11 inch paper, my experience so far is that one cartridge prints in excess of 2 reams of paper (over 1000 sheets). Each ink cartridge costs about \$18.00, quantity one.

An accessory font cartridge takes less than 10 seconds to install. They install

on the top front of the printer, just in back of the very accessible keypad.

The DeskJet's speed is rated by the manufacturer at 120 cps in Letter Quality Mode, and 240 cps in Draft Quality Mode.

The owner's manual is thorough, well thought out, has an excellent index and includes a floppy disk containing numerous printer drivers. I have not needed to access this disk. **Do see a DeskJet in operation!**

WordPerfect 5.0 is loaded with **new features including graphics** and I believe it will set a new standard in word processing software. It is far superior to its predecessor, Version 4.2, which is currently the world's best selling word processor. **WordPerfect also has the world's best software support! (I wonder if there could be a connection?)**

I have been using the DeskJet for about 2 months now and received my update to version 5 of WordPerfect about the latter part of May. Documentation, setup, installation and Reference Manual all rate excellent. A hard disk drive is almost a necessity. This review has been extremely easy to write. For example, it took less than 5 minutes perusing the Reference Manual to learn how WordPerfect 5.0 handles columns. The combination of the DeskJet and WordPerfect 5.0 certainly have exceeded my wildest expectations. Both are highly recommended! The last time I was this enthusiastic about anything was when I assembled my first H-8!

The list price of the DeskJet is under \$1000.00; list price of described cartridges is \$125.00 each. The list price of WordPerfect, Version 5.0 is around \$400.00 (update from version 4.2 is only \$60.00.) However, all these products mentioned here are widely discounted, quite heavily in some instances. With "group buys" and from right firms (look in back pages of some computer magazines) you may well be able to acquire the DeskJet, one or two cartridges, and WordPerfect Version 5 for between \$900.00 and \$1200.00 (or even less)! Remember, laser printers, without software or extra cartridges.

cost **much** more than all these items combined! Check with your local Heathkit/Zenith dealers for group and "bundled" prices as well. Most carry both the DeskJet and WordPerfect, version 5, and will be happy to demonstrate both for you. In some ways I prefer the DeskJet's printout to that from a Laser Printer! What is your opinion? How does the printout you have seen here compare with that of a 9 pin or a 24 pin dot matrix printer?

It has been difficult to exactly match the left and right, and the top and bottom margins of REMark's layout. As near as I can figure out, the body type of REMark must be about 9 points. At least the printed material in the body text of an actual issue of REMark seems to be 9 points. Of course it is quite possible that it was printed in 10 points and then photographically slightly reduced in preparation for offset printing. In any event, you should be reading actual 10 point printing unless it has been photographically altered.

Also, everything (except page numbers) on the 2 pages of this review was produced with WordPerfect version 5.0 and the material was printed on my DeskJet. This includes the graphic reproduction below, which is one of thirty samples of "clip art" included with WordPerfect 5.0. Additional illustrations can be ordered at additional cost directly from the manufacturer who supplied WordPerfect Corporation with the 30 samples.

Last, and **in 4 points** (can't you visualize that H-P engineer saying "**4 points? - WHAT IF?**"):

DeskJet is a Trademark of Hewlett-Packard
WordPerfect is a Trademark of WordPerfect Corporation



ENABLE

A Tutorial

Part 16

Telecommunication

In the preceding articles, I have discussed four of the modules in ENABLE, the word processor, spreadsheet, graphics, and data base. ENABLE is a full featured integrated package that also includes a telecommunication module. While most people will not use this capability, it is available and it does work. This article will focus on this module.

Most people who use personal computers have their favorite communication program. Many of these programs have many bells and whistles and permit you to have auto logon, along with the usual download and upload of files. Some have the capability to do these functions unattended. They have different transfer protocols and can have autodial and auto logon to other computer systems. Most of these programs do not have a word processor or the other integrated capabilities like ENABLE. Using one of these programs requires you to download the file and then drop out of the program to run or to look at the file.

ENABLE will permit you do the above functions, along with having the rest of ENABLE's integrated capability available at the touch of a key. Although the module does not have all of the bells and whistles of a stand alone telecommunication package, it is usable and fairly easy to use. I have sent large files across the U.S. with ENABLE at both ends using XMODEM protocol without problem. At home, ENABLE is the only telecommunication package I have on the H-386 which my son uses. He works many bulletin boards around the country without a problem. I am aware of one major company in the U.S. that uses ENABLE's telecommunication module to link to a mainframe to download parts of data bases. The individual can then work these records off-line and then upload them after completion of the task.

The Software Group has written the manual for the telecommunication mod-

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

ule for people with limited experience using a modem. They go into discussions of modems, comm ports on the computer, and even down to the RS-232 cable requirement. The blue 158 page book provides information on how to use the module. The Federal Sales division of the Software Group has placed a bulletin board on the air for Federal users. This system has files and a place for questions and answers on ENABLE.

As part of ENABLE's telecommunication package, it is possible to establish up to 256 unique logon sequences or SETUPS. These setups can include all of the information that is necessary to connect with a remote computer from your computer. Along with the usual capabilities of a setup program, ENABLE will also display connect time and charges, if so desired.

ENABLE's setup will prompt for nine bits of information that will make establishing a connection with another computer painless, and make it possible to do at night. The setup procedure will prompt for the telephone system, including the number to dial. It will ask for information on the modem to include speed and other communication parameters, including the communication ports. You can insert your user ID and password if you desire. If you want connect time and approximate charges displayed on the bottom of the screen, this can be inserted into the setup. If you wish to use a terminal emulation, this is also inserted into the Setup. Once completed, these setups are available with a keystroke upon entering ENABLE. All of the possible setups are stored in the file TPSETUP.\$TP.

ENABLE's telecommunication mod-

ule is entered in the same manner as all of the other modules. From the Main Menu select (U)se System and (T)elcom. You are then prompt as to the mode you wish to enter, (C)ommunicate or (S)etup. Communicate is the method used to do the actual communication. If you do not wish to develop a Setup routine or just want to try connecting to something on a one-time basis, select (Q)uick Connect. This selection will permit you to select the baud rate first. ENABLE supports speeds from 110 to 19,200 baud. In the last article, the discussion was about menus. ENABLE uses this feature to assist you in making decisions on selections. In the bottom half of the screen, a set of helpful information is provided for each choice. Note that this changes with each selection in Figures 2 and 3 in this article. Next, you must select one of four options that ENABLE provides covering parity, word size, and stop bit. If one of the three settings does not match your requirement, you can select option four and insert the setting for each of these items. The number three option, no parity, 8-bit word, and one stop bit are normal and is the one that the Software Group Bulletin Board uses. Next, comes the type duplex you need to use, half or full. The last selection is the Comm port that the modem is connected to.

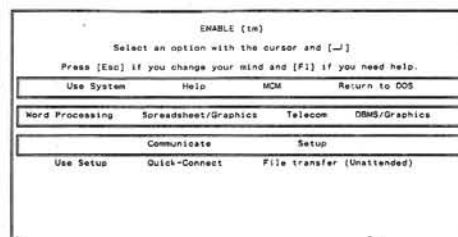


Figure 1
ENABLE's Main Menu

After you have established the information necessary to communicate, ENABLE will then link the Telecommunication

Quick Connection Form

Baud rate: 1=110 2=300 3=600 4=1200 5=2400 6=4800 7=9600 8=19200

Select one of the options described below: 1 2 3 4

Select type of duplex: Half Full

Which of your computer's COM-PORTS are you using? COM1 COM2 COM3

The baud rate is the speed at which data is transmitted between your computer and the one at the other end. Select the baud rate you prefer. Be sure both the network and the information service you intend to use support the baud rate you select. NOTE: 19200 is not valid for many computers. Press [END] to bypass the above prompts or [ESC] to exit.

Figure 2

Quick Connection Form Page One, Option One

Quick Connection Form

Baud rate: 1=110 2=300 3=600 4=1200 5=2400 6=4800 7=9600 8=19200

Select one of the options described below: [1] 2 3 4

Select type of duplex: Half Full

Which of your computer's COM-PORTS are you using? COM1 COM2 COM3

1 --> parity even, word size = 7 bits, # stop bits = 1.
2 --> parity odd, word size = 7 bits, # stop bits = 1.
3 --> parity none, word size = 8 bits, # stop bits = 1.
4 --> You will be prompted for parity, word size and stop bits.

Figure 3

Quick Connection Form Page One, Option Two

module to the Word Processor. This permits you to capture on-screen text in a word processing document and use it immediately. In the next screen, this link is established and you are prompted for a file name. If you do not insert a file name, ENABLE will establish a default name, "TPnname.WPF." ENABLE will then display a screen telling you what it is doing. It first initializes the Comm port on the computer and then the modem.

TELEPROCESSING SYSTEM

Establishing Interface With Word Processing

You may capture all or portions of the data displayed on your screen during your telecommunications session. This feature allows you to review data which has scrolled off the screen. Also, you may edit this data using all of the Word Processing functions. At any time during the session (or at the end of the session) you may save this data on disk as a Word Processing document. If you wish to immediately begin capturing data, enter a file name in the area below. Otherwise just press ENTER.

Enter file name (or just press ENTER): []

If you are using a modem, please make sure the modem has been turned on.

Figure 4

Telecomm Word Processing Linking Screen

You or Enable will take these actions:

Enable → Initializing communications port
Initializing modem

Establishing a connection for setup quick

Figure 5

ENABLE's Initializing Screen

ENABLE then informs you that these steps have been completed and you must now continue. Pressing a <RETURN> will present you with a blank screen. You must now press F10 to display the Top Line menu and make a selection from here. To make your call, press (T)elcom and then (C)all. ENABLE will then provide you with an area to insert the telephone

number, including the requirement for a (P)ulse or (T)one. You can insert commas for delays, just like you would use for a Hayes compatible modem. Once you have the number input, press the <RETURN> key and the call will be made. You will have interactive control of the system using this feature. Once you have logged on, all procedures are the same as any other Comm package.

Break Capture Disconnect Files MCM Other Print [F10] WP
Enter telephone number: []
Use commas for pauses, T for touch tone, P for pulse. (Ex: TR,555-1212)

WARNING: If there is a call currently in progress it will be disconnected.
Press ESC to return
Press ENTER to proceed

*1 TP0Name.WPF X FB Cap

Figure 6

Telecom Top Line Menu, Telephone Call

The Capture mode is the default when you enter the above mode. All information displayed on the screen can be captured on a disk file. If you do not want to capture data, this can be turned off by pressing F10 to display the Top Line menu and then pressing (C)apture, which is a toggle. If you wish to capture files from the other computer that are ".COM", ".EXE", ".ARC", or other binary files, you can press F10 and then (F)iles and (R)eceive, then select a transfer protocol. ENABLE supports six methods of file transfer, no protocol, ENABLE's protocol, Xmodem, or three variations of Kermit. Selecting (N)one is not recommended for file transfer as no error checking is completed which may result in transmission errors. Most bulletin boards support the Xmodem protocol, so this is the method that would be selected. If you were connected to another ENABLE user and wanted to transfer ENABLE files, then select the ENABLE protocol, which is an "enhanced version of Xmodem." Kermit protocol is another type of error checking transfer that is used on mini or mainframe computers. Once you have the protocol selected and named the file to be received, the transfer will be accomplished. The usual block count will be made during these transfers. If you wish to send files to another user, select (T)ransmit, instead of receive, and then insert the name of the file you wish to send. ENABLE will return to the basic screen once the file has been received and saved.

[Receive] Transmit Unattended Auto timeout: 1=Receive 2=Transmit
Select protocol: None [ENABLE] Xmodem Kermit 1=Kerm Logout 2=Kerm/Finsh
Modified Xmodem with additional error checking. ENABLE <to> ENABLE only.

Warning! Your modem must be set to full duplex.
When you use a Setup and specify that your modem is an auto-dial it will be properly set by Enable. If you have set the duplex option on the modem, please make sure it is in full duplex mode.

*1 TP0Name.WPF X FB Cap

Figure 7

ENABLE Protocol Option

[Receive] Transmit Unattended Auto timeout: 1=Receive 2=Transmit
Select protocol: None [ENABLE] Xmodem Kermit 1=Kerm Logout 2=Kerm/Finsh
Standard Xmodem. (The other system must also use Xmodem.)

Warning! Your modem must be set to full duplex.
When you use a Setup and specify that your modem is an auto-dial it will be properly set by Enable. If you have set the duplex option on the modem, please make sure it is in full duplex mode.

*1 TP0Name.WPF X FB Cap

Figure 8

Xmodem Protocol Option

After completing your work on the computer, you press F10 to display the Top Line menu and select (D)isconnect. ENABLE will then prompt you if you are sure you want to disconnect. The next prompt will be if you wish to capture any of the screen data. These are not the files that you have downloaded as they were saved at the end of the transmission. If you respond with a (Y)es, ENABLE will save all data that has been sent across the screen.

At any time during the course of the session, you can drop into the word processing mode. All data that was presented on the screen will be displayed at that time. You can do whatever is required with the data. To move from the telecommunication module to the word processing module, just press PgUp. Shift/F9 will return you to the Telecomm mode.

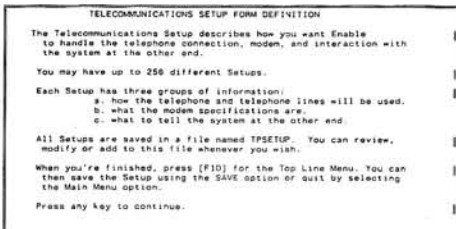
ENABLE's Telecomm Top Line menu also has a "Break" key. This will send a break signal to the other computer to indicate a desire to stop what it is doing.

While in the Air Force, the office I was in captured vast amounts of data from a mainframe computer. We would use the Z-100 and the Air Force Comm package "COMPAC". The data would be captured as ASCII and then be brought into WordStar for additional formatting. WordStar would insert a hard return at the end of every line which made the word processing painful. ENABLE does not have this problem, and text captured can be readily formatted. The only thing that is required is to put in the hard returns at the end of the paragraphs.

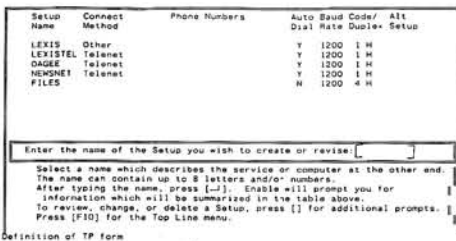
If you wish to print data that is coming across the screen, press shift/F2. This keystroke is a toggle so that printing can be stopped by press shift/F2 again. Another method is to press F10 to display the Top Line menu and press (P)rint. Using the Top Line menu, you can select the port that the printer is connected to. ENABLE will not place the text coming across the screen into a print queue so you may lose data if it is coming faster than your printer can handle, unless the sending computer supports the Xon/Xoff protocol.

As indicated earlier in this article, ENABLE supports up to 256 setups. The development of a setup will permit you to log on to a remote computer with a minimum of keystrokes. To develop this

setup, select (U)se System, (T)elcom, and (S)etup from the main menu. ENABLE will open with a general statement on the setup procedure. The screen that is displayed is very similar to the data base definition screen. You can revise or create a new setup from this screen. ENABLE has five sample files on the system when you get it. They are LEXIS, LEXISTEL, OAGEE, NEWSNET, and FILES. "Files" is a set of default settings for communicating with other generic computers.



**Figure 9
Telecomm Setup Opening Screen**

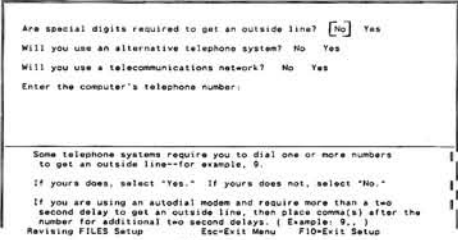


**Figure 10
Setup Create/Revise Screen**

The first set of information requested by ENABLE is about the telephone system. You are prompted if special digits for outside lines are required and then you must input them. If this is selected, ENABLE will wait approximately two seconds for the outside line dial tone. The next question is about the telephone system that will be used. ENABLE will support the extra digits required to access the "other" telephone systems, if used. If you respond with a (Y)es to this prompt, ENABLE will request the telephone number and the access code. Commas can be used to insert wait states in the system, one comma equals two seconds. If the other telephone system requires tone signals and you have rotary phone and an autodial modem, you can insert a "t" at the end of the number and ENABLE and the modem will take care of the requirement.

The next prompt from ENABLE requests information on the telecommunication network being used. ENABLE is set to recognize "Telenet", "Tymnet", and "Uninet". If you use one of these networks, ENABLE is programmed to perform specific commands related to their use. You are then prompted for the telephone number of the network, including the area code, if necessary. The next question is the terminal identifier required by the network. Again, ENABLE is programmed with this information for the above listed net-

works. The last prompt on this screen is the connection address. The Software Group says that this is the special address on the system for the type of information that you are looking for. Our company has an address on one of the electronic mail services (MCI Mail) and this is what would be inserted here.



**Figure 11
Telecom Setup Screen One**

The next screen prompts you for information on the modem. You have to specify if it has an autodial capability and the type dial requirement. Next comes the baud rate. You next have to select one of the three predefined setup options or insert your own unique setup as to parity, word length, and stop bits. ENABLE will permit you to select Even, Odd, or No parity, word sizes from 5 to 8, and stop bits of 1, 1.5, or 2. Next, you are prompted for the duplex type. After duplex comes, the inter-character delay and then Xon/Xoff. The inter-character delay is inserted if the other computer or modems can not keep up with the transfer. The specified delay inserts a millisecond delay between each character. Normally, this is set to 0. Xon/Xoff is the signal sent between computers telling them to stop sending information because one does not process information as fast as it is being sent. An example would be the printer buffer being filled up and without the Xon/Xoff capability some data may be lost. The next prompt is turn-around character sent by the other computer. This character is what the computer sends when it is ready for the next input. This could be a "?" or ">". This turn-around character can be a control character or a series of characters and is similar to the Xon/Xoff signal. ENABLE will support this if it is required and supported. The control characters or special characters are inserted in the same manner as the macros. To enter a ctrl-C you would insert {^C} in the block. Escape would be entered {ESC}. Next, you are prompted for the password or first response. This is usually the userid code that you type in after the other computer acknowledges that you are connected. If you need to press <RETURN> after this response, insert a "~" (tilde) after the word. ENABLE will not display this response on the screen if you type in {echo off} before the word. The last response on this screen is the second identifying word for the system if required. This is normally the password. The

password on systems I have used is case sensitive, so make sure that you enter the information exactly as required. Again, you can insert an {echo off} to prevent the code from being displayed on the screen. ENABLE permits a code up to 130 characters in length. Again, a tilde can be entered if a <RETURN> is required.



**Figure 12
Setup Screen Two**

ENABLE now moves on to the next screen of information. Here, you can tell ENABLE to display the approximate cost of the connection. ENABLE can use only one price to determine this cost, so if you use a system like Compuserve with different rates for 300 baud and 1200 baud, you must have two different setups, each with different costs and baud rates. The cost is updated on the minute. It does not include special charges for using extra cost options of the system. The next prompt asks if you would like to see the time of the connection. This time is in minutes and if selected, is updated every minute along with the cost, again if that option was selected. The next selection is the communication port being used on the computer, normally COMM1. At times, a setup may fail because of a busy telephone line on the other computer or the telephone system. ENABLE permits you to establish the number of times to try the setup before stopping. If the setup fails, the next prompt permits you to try another setup. This is where you would enter the name of the secondary setup. The last prompt on this screen is the terminal emulation requirement. ENABLE can emulate the VT-52 (native H-89 and Z-100 modes), VT-100, an ATT4410 or nothing. ENABLE will not emulate the graphics or double-width or double-height letters available on the VT series terminals. Pressing the NUM LOCK key on the PC will convert the numeric keypad into the VT series number pad. The ENABLE telecommunication manual shows the PC keys that can be used during the emulation, i.e., the F1 - F4 are the VT PF1 - PF4 keys. Depending on the emulation, additional questions can be asked.

The next screen asks if you wish to change any of the four listed options. You can change the mapping tables for the computer including the keyboard, the wait time until the other computer, how linefeeds, tabs and character are handled, or to disable some automatic parameters.

Inter-character transmission delay: [6] 1 2 3 4 5
 Does the system at the other end support Xon/Xoff? No Yes
 Enter the password or first response:
 Enter the second response:

Enable may send data faster than some computer systems or "smart" modems can process the transmitted data. For these devices, you may cause Enable to pause (delay) between transmission of each character. The option number you select will be multiplied by 1/8 to calculate the inter-character delay.

SELECT "0" UNLESS YOU ARE SURE THIS DELAY IS REQUIRED.
 Revising FILES Setup Esc=Exit Menu F10=Exit Setup

Figure 13
Setup Screen Three

The keyboard remapping is like macro procedures or the data base dot command procedures. It is developed from the Main Menu, (T)ools, (T)elcom, (K)eyboard Mapping, and (C)reate. Using this capability, you can remap a key so that it does several steps. You can remap it so that a key, when pressed, will send a string like "go pcs-48". This command will, in Compuserve, send you to the Heath area. After completing the setup, you press F10 and (S)ave.

Do you wish to see the approximate cost on the Status Line? [No] Yes
 Do you wish to see the approximate time of the connection? No Yes
 Which of your computer's COM-PORTs are you using? COM1 COM2 COM3
 Enter number of times to retry this Setup:
 Enter name of Setup to try if this one fails:
 Terminal emulation? No 1=VT100 2=VT52 3=ATT4410

If you wish, Enable will display the approximate cost of the connection on the Status Line while you are telecommunicating.
 The cost is determined by multiplying the connect-time by the cost per hour. It is only an estimate and not an accurate prediction of your total expenses.
 The cost is recomputed every minute.

Revising FILES Setup Esc=Exit Menu F10=Exit Setup

Figure 14
Setup Screen Four

Do you wish to select any of the options listed below? [No] Yes
 Wait time (in seconds) for other system to answer:
 Enter receive filter option: 1 2 3 Use Filter/mapping table
 Specify how received linefeeds will be handled: 1 2 3
 Specify how received tabs will be handled: 1 2 3 4

Use transmit or receive mapping tables or keyboard mapping table.
 Change default wait time for other computer to answer (1 minute).
 Change default display options for characters, linefeeds and tabs.
 Disable automatic adjustment of communication parameters for 3modem and Enable file transfers.
 Revising FILES Setup Esc=Exit Menu F10=Exit Setup

Figure 15
Setup Screen Five

Will you use a transmit filter/mapping table? [No] Yes
 Specify when linefeeds will be transmitted: 1 2 3
 Will you use a keyboard mapping table? No Yes
 Do you wish to change the default described below? No Yes

Select YES if you require a filter/mapping table for transmitted characters.
 This table must be created using the Enable MCM TOOLS option which appears on the Main Menu.
 This feature allows you to specify which characters to be transmitted will be ignored (not transmitted) or replaced.
 All needed instructions are provided within the TOOLS option.

Revising FILES Setup Esc=Exit Menu F10=Exit Setup

Figure 16
Setup Screen Six

Another of ENABLE's telecommunication capabilities is Unattended File Transfer. This permits you to send or receive files at night when phone rates are the lowest. ENABLE does this by using a script that you develop. The script is created from the main menu. Select (M)CM, (T)ools, (T)elcom, (U)nattended Transfer Form, and (C)reate. You are prompted for a name and can enter an extension. Because you are in the word processor, EN-

ABLE will enter an extension of ".WPF" if you do not insert one. Again, the structure of this script file is the same as the dot commands in the data base. ENABLE provides a lot of on-screen help during the development of the script. By pressing F10 and then selecting (I)nstruction, you can receive detailed instructions on several related topics. Some of these screens are shown in the figures in this article. You will specify the setup to be used and the time that the call is to be made, along with the files to be sent. This script can be set to not only operate at a specified time, but also day of the week. The script commands can not be used to call the quick connect method of ENABLE, the setup command is the only thing permitted. This capability includes gosub, goto, return, and if statements to make it a powerful mini programming language.

Enable Development System
 Tools Menu, Version 1.0
 Select an option with the cursor and [ENTER]
 Press [Esc] if you change your mind.
 Menu: [Telecom] Quit
 Transmit/Receive Map [Unattended Transfer Form] Keyboard Mapping
 [Create] Revise

Figure 17
Telecom Develop Menu

The Telecommunications Unattended File Transfer Form is described on screen as well as in your documentation.
 Complete instructions plus examples can be obtained by pressing F10 and selecting the INSTRUCTION option.
 Press ENTER to continue.

Figure 18
Information Screen One

```
.TELECOMMUNICATIONS DIVISION. VER=1
;The above line is required.
.TIME 03:00
;This line tells the script to run at 3 am.
.SETUP HEATH
;This line tell the script to use the heath setup.
.IF 1, "NO CARRIER, .GOTO QUIT
;This line tell the script that if there is no carrier to goto the quit
;statement.
.SEND "~"
.WAIT4 10, "Enter Your FIRST Name:"
.SEND "George-"
;The above two lines tells the program to wait for 10 seconds for the prompt
;'First Name', after which it will send 'George'.
.WAIT4 "Enter Your LAST Name Or <cr> For FIRST Name Prompt:"
.SEND "Elwood-"
.PASSW "XXXXXXXXXX"
;These lines finish the logon sequence including the password.
.WAIT4 "Function Or <H>elp: "
.SEND "U"
.TX X,HUG16.ASC
;These lines move to the upload area and send the file HUG16.ASC
.SEND "G"
.LABEL QUIT
.LABEL DONE
.EXIT
.END
```

Telecommunications Transfer Form
 The Word Processing document used to develop your Transfer Form has two parts: the header and the Script
 A sample format is shown below.
 .TELECOMMUNICATION DIVISION. VER=1 <----- header
 .TIME 17:00 <----- script
 .SETUP FILES
 .SEND "ACCOUNT 30"
 .WAIT4 60,"HELLO"
 .PASSW "PASSWORD"
 .END
 Press any key to view the next page TPTABLE Cap L00001C00

Figure 19
Information Screen Two

Telecommunications Transfer Form (Script)
 1. The remainder of the Transfer Form is made up of script commands from the following list. (One command per line starting in column one)
 2. Currently supported commands:
 [] indicate optional parameters.
 .TIME MM.MM
 .SETUP setuptime
 .SEND "string"
 .PASSW "password"
 .LOCAL "string"
 .WAIT4 timeout,"string" [,label]
 .LABEL label
 .GOTO label
 .GOSUB label
 .RETURN

Press any key to view the next page TPTABLE Cap L00001C00
 Telecommunications Transfer Form (Script)
 .TX mode,filename (mode can be X,E,N,K)
 .RX mode,filename
 .IF ##,"string",scriptcommand
 .CLEARIF [##][,##]
 .RUN scriptname [,label]
 .SUBFILE scriptname [,label]
 .BEEP
 .DAY dayofweek [,label]
 .KEY bracedcodes
 .DELAY seconds
 .EXIT
 Press for Top Line menu, to view previous page TPTABLE Cap L00001C00

Figure 20
Information Screen Three

Telecommunications Transfer Form (EXAMPLE)
 This is an example of a transfer form. Your Word Processing document should look like this before you press F10 and select the CREATE FORM option.
 .TELECOMMUNICATION DIVISION. VER=1
 .TIME 23:00
 .SETUP Dwjones
 .SEND "ACT 1234"
 .TX X,FILE1
 .SEND "BYE"
 .END
 Press any key for the Top Line Menu
 B:HUG.WPF TPTABLE Cap L00001C00

Figure 21
ENABLE Sample Script

The sample script file will provide a small idea of how the script file works.

This script would send a file to another computer at 3 am. Once the script file is completed and saved, it can be used at any time. To run the script, you must load it from the Main Menu. Select (U)se System, (T)elcom, (C)ommunicate, and (F)ile transfer (Unattended) and then the name of the script. The computer must be left on and the file will run at the specified time. ENABLE will also create a log that can be reviewed after the action has been completed to show what has run and what did not work.

This short discussion has focused on the ENABLE telecommunication module. This is the fifth module in ENABLE and can be useful. It has a lot of capability if you want to make the effort to use all of it. The next article will return to the word processing module again. *

Continued from Page 15
 thing in this column, be sure to include a self-addressed, stamped envelope (business size preferred) if you would like a personal reply to your question, suggestion or comment.

Products Discussed

HUG Software
 HUG Editor (885-3012) \$20.00
 HUGMCP (885-3033) 40.00

```
Continued from Page 10
MOV   BYTE PTR [DI], DL
LOOP  HXD010

CMP   DL, 030H
JNZ   HXD020

MOV   BYTE PTR [DI], 020H

HXD020: RET
HX_DEC ENDP
SHOW  ENDS
END    BEGIN
```

CheapCalc (885-6004) 20.00
 Heath/Zenith Computer Centers
 Heath/Zenith Users' Group
 P.O. Box 217
 Benton Harbor, MI 49022-0217
 (616) 982-3463
 Ability Plus \$295.95
 Ability 99.95
 Heath/Zenith Computer Centers
 Heath Company
 Hilltop Road
 St. Joseph, MI 49085
 First Choice \$149.00
 Software Publishing Inc.
 1901 Landings Drive
 Mountain View, CA 94039
 (415) 962-8910

PC-File+ \$69.95
 PC-Calc 59.95
 Buttonware Inc.
 P.O. Box 5786
 Bellevue, WA 98006
 (800) J-BUTTON
 PC-Write
 Disks only \$15.00
 Manual only 35.00
 Registered version 89.00
 Quicksoft Inc.
 219 First N., #224
 Seattle, WA 98109
 (800) 888-8088 *

BUG ZAPPING

BUG: *My three-year-old spilled prune juice on my only copy of my MS-DOS manual. How can I obtain a new copy of this book?*

ZAPPED: If your manual is torn or otherwise irregular when you receive it, we'll replace it free of charge within 90 days of purchase, provided you return the original to us with your proof of purchase. The defective item and the sales receipt should be sent to:
 Heath Service Receiving
 Hilltop Road
 St. Joseph, MI 49085

If your manual becomes torn or otherwise defaced through use, you can return it together with your credit card number, check, or money order to the above address, and we'll replace it for a charge. The cost of the manual varies by product, and you should call Parts Replacement (616) 982-3571 for the exact cost — they will need the part number of the manual (595-xxxx-xx).

The only way you can get extra software manuals is to buy extra software packages in their entirety. "Software" — the programs on the disks and all documentation (quick reference cards, key templates, manuals, etc) — is one single entity identified by

name and model number. It is the exclusive property of the manufacturer, even though it may be licensed to you, the end user. Parts of the software (the manual, for instance) may be exchanged for replacement but cannot be sold separately.

BUG: *I have a program called Sidekick, and some data is not visible on the screen of the Z-180 computer. Why?*

ZAPPED: The video controller in the Z-180 computer emulates a color graphics adapter. If the program you're using checks the video, the program will display color. However, the Z-180 computer only produces a gray scale, not a color scale, so you will be unable to see certain data that would otherwise appear in color on your screen.

To correct this problem, use the FN-F8 and FN-F9 key combinations that allow you to toggle the LCD display forward (FN-F8) or backward (FN-F9) through eight different gray-scale palettes. Optionally, you can use the PALETTE program found on MS-DOS 3.2 BIOS 3.30 and above. This program allows you to select a palette by number. *

QUIKDATA - THE ONLY NAME TO REMEMBER!

For all your H/Z 8-bit and PC/XT/AT needs

ACCELERATE YOUR PC!

From PC Technologies, the fastest and most proven way to really speed up your Heath/Zenith PC/XT computer, giving it -AT compatible speeds! Turn your turtle into a -286 rabbit with a 12Mhz 80286 accelerator board.

There are other ways to speed up your H/Z PC/XT computer, but you spend too much and get too little. The **286 EXPRESS** is the effective solution, making your H/Z150/160/150/159 series of computers, or any general PC/XT computer in many cases faster than a standard IBM AT type computer! Simple half-card plug-in installation with keyboard software switchable speed control.

Complete with 16-bit -AT -286 processor, CACHE memory for dramatic speed increase, 12Mhz 80286 processor, and software. **You won't believe your stop watch!**

\$379 SPECIAL (\$645 list)

ACCELERATE YOUR 241/248

With the Aox Z-MASTER **20Mhz 80386**, your Z241/248 computer will be faster than the Zenith Z386 and more compatible. The Aox 20MHz Z-MASTER 386 is a one-board CPU card that replaces the Zenith 80286 CPU board of the Z241 and Z248. With support for up to 8MB of onboard 32-bit memory and 80387 math co-processor on one single card. The Z-MASTER 386 comes standard with an efficient 8KB cache providing 0 wait state access to the onboard 32-bit memory and optimum access to existing 16-bit Zenith and conventional memory. The Phoenix ROM BIOS Plus is also standard providing a higher level of software and hardware compatibility. All existing 16-bit memory is accessed efficiently via the onboard cache controller. Installation is simple with only one board to replace, approximate installation time is only 10 minutes. See H-SCOOP #107, or request a copy for a report on this board, or inquire for more information.

AOX386 - SPECIAL - \$1295 0K RAM
APX386-1 - \$1695 with 1MB RAM

MEMORY UPGRADES

Note: All memory upgrades come without memory chips. Call for current chip pricing.

Z150MP - \$19 Will allow you to upgrade your H/Z150/160 to up to 704K on the main memory board, using up to 18 256K DRAM chips.

MEGARAM - \$43 Upgrades your H/Z150/160 series with up to 704K of main memory, and about 512K for RAMDRIVE memory. Includes documentation, software RAMDRIVE disk, PAL and jumper wire. For the full 1.2 megs total memory, 45 256K DRAM chips are required.

ZMF100 - \$53 Will allow you to upgrade your H/Z110/120 (old motherboards; with p/n less than 181-4918) to 768K system RAM. Requires 27 256K DRAM chips.

Z100MP - \$76 Allows you to upgrade your H/Z110/120 (new motherboards with p/n 181-4918 or greater) to 768K system RAM. Requires 27 256K DRAM chips.

PRINTERS

We have a full variety of Panasonic printers; narrow and wide carriage dot matrix, 24-pin dot matrix and daisywheel. We also have cables, buffers, ribbons, tractors when not included, etc:

KXP-10911	80 column 160 cps parallel std	\$225
KXP-10921	80 column 240 cps parallel std	\$359
KXP-1595	132 col 240cps ser/par std	\$489
KXP-1524	132 col 240 cps 24pin par std	\$579
KXP-1124	80 col 192 cps 24pin parallel	\$369
KXP-3151	132 col daisy 22cps par std	\$539

8-BIT

We carry a full line of hardware and software products for the H8/H89/90 computers. This includes diskettes, printers, H89 working boards and parts, some H8 parts, etc. Call or write to obtain a catalog of all our 8-bit goodies.

H37 SOFT SECTOR CONTROLLER	\$179
H8 SOFT SECTOR/WINNIE CONTROLLER PACKAGE	\$275
TM100-2R Tandon DS ST 48 TPI refurb disk drive	\$79
H89 4MHZ - Double your H/Z89/90 CPU speed	\$43
Z47 REMEX DRIVES slaves and masters - special	\$95

WINCHESTER UPGRADE KITS

PCW20 - \$295 Complete winchester setup for a H/Z150, 148, 158, 159, 160, PC etc. Includes 21 meg formatted half-height Segate ST-225 65ms drive, WD WX1 winnie controller board, cable set, documentation.

PCW40 - \$449 42 meg with 37ms Segate ST-251.

PCW80 - \$695 80 meg with Segate ST-4096 full size drive.

EWIN20 - \$555 Complete 21 meg winchester package for the Z100. Includes S100 host adapter card, WD winchester controller, Segate ST-225 winchester drive, cables, software and documentation. We also have this in 42 megs (\$669), and 80 megs.

We also have winchester systems for the H8 and H/Z89/90.

H-SCOOP

Of course, don't forget about the best source of Heath/Zenith related information you can obtain - our monthly newsletter, **H-SCOOP!** Just \$24 for a 12-issue year (\$35 foreign), it will help you get the most from your computer investment. Get sound technical advice, helpful hints, find out what the problems are, fixes, reports, reviews, information from other subscribers, classifieds and much more.

BARE WINCHESTER DRIVES

ST-225 - \$229 65ms half height bare winchester with 21 megabyte storage capacity.

ST-251 - \$395 37ms autopark half height bare winchester with 42 megabyte storage capacity. Good choice for -AT compatibles.

ST-251-1 - \$429 Same as ST-251 but 28ms speed.

ST-4096 - \$629 28ms autopark full height bare winchester with 80 megabyte storage capacity. Best choice for -AT compatibles.

Call or write in to place your order, inquire about any products, or request a free no obligation catalog. VISA and Master Card accepted, pick up 2% S&H. We also ship UPS COD and accept purchase orders to rated firms (add 5% to all items for POs). All orders under \$100 add \$4 S&H. Phone hours: 9AM-4:30PM Mon-Thu, 9AM-3PM Friday. Visit our bulletin board: (414) 452-4345

QUIKDATA, INC.

POB 1242
SHEBOYGAN, WI 53082-1242
(414) 452-4172



Z-100 Survival Kit #3

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

Writing Text to the Z-100 Screen

One requirement of almost every program is the ability to display text on the computer screen. If you write some of your own programs, there may be several different methods available to you for outputting text to the screen. We're going to look at some of those methods in this installment of Z-100 Survival Kit.

Let Me Count the Ways . . .

First off, let's take a look at all the possible ways that can be used to get text on the screen . . .

1. High-level routines
2. DOS function calls
3. BIOS calls
4. Monitor ROM calls
5. Writing directly to video memory

High-level routines can be thought of as the resident print commands available in most programming languages. For instance, the PRINT and PRINT USING commands in BASIC. Another example would be the printf() or puts() function in the 'C' language. If you have done batch file programming, the ECHO command may also be thought of as a high-level text output routine. These are commands (or function calls) which make life easy for those who are programming with a high-level language.

DOS function calls offer a quick and easy way of outputting text for the assembly language programmer. Some high-level languages (notably BASIC and FORTRAN) make it very awkward to use DOS function calls in a program, but quite a

few (like 'C' and PASCAL) include library functions specially used for DOS Functions.

BIOS calls might be used when speed is critical to an application program. Calling the BIOS for text output is noticeably faster than most built-in language print functions. But in exchange for the increase in speed, you must give up the portability (the ability to use the program on more than one type of computer) that normally comes with using high-level routines or DOS function calls.

Calls to the MTR-100 monitor ROM will give much the same results as BIOS calls, but one more layer of overhead will be stripped away.

Writing directly to the video memory is the most versatile way to write text to the screen. It is also the most complicated and least portable way, and should therefore be reserved for those applications where custom fonts or fast display are important. Unlike the PC compatibles, the Z-100 has no dedicated text mode where ASCII text can be written directly to video memory. Thus, in order to display text on the Z-100 screen, you must actually transfer the font design to the screen one pixel or byte at a time.

Now that we have the preliminary descriptions out of the way, let's take a more detailed look at each of these alternatives.

High-Level Routines

Most programmers (or those who dabble in programming) will never use, or

need to use, anything other than the built-in capabilities of their language interpreter or compiler for text display. As long as the PRINT statement will do what you need to do, there is probably no need to add extra complication to your life. I'm not going to give any examples of using these high-level routines because that should be documented thoroughly in your programming language manual.

What I am going to talk about, with reference to high-level text routines, is under what circumstances you might need to use an alternative text outputting scheme. The obvious case, of course, is where more speed is required. You can greatly speed up text display by using BIOS calls, or writing directly to video memory. Using DOS function calls probably won't do much to help text display speed, since the high-level routines usually make use of DOS calls anyway.

Another case where the high-level routines won't do the job is when you need text to be displayed in positions other than the normal rows and columns on the screen. Most pixel graphics and CAD programs, for example, will allow you to place text anywhere on the screen, on a pixel-by-pixel basis. In order to do this, they must write directly to the video memory. If you want this same type of capability in your own program, you also will have to write directly to video memory.

Along the same lines, if you want to use custom text, which is not a standard size (i.e., 8 x 9 pixels), you must use direct video memory access. Text charac-

ters which are smaller or larger than the normal font, as well as italic text, are examples of the possibilities.

Another situation which may require use of a DOS or BIOS routine is when you don't want your program to be interrupted when a user types Control-C at the keyboard. Many high-level languages do not allow you to disable this feature during screen output, or keyboard input.

One last note about high-level text output routines. Be sure to get familiar with whatever resources are offered with your particular language interpreter or compiler. It is possible (but not likely) that the text output routine already uses the BIOS for output, thus making it unnecessary to look further for a speed upgrade. Most languages offer more than one alternative for displaying text on the screen — check them out. If you are working with a compiled language, your compiled program will usually be much smaller if you stay away from the more versatile print functions (like `printf()` in the 'C' language, or `PRINT USING` in BASIC). For instance, in the 'C' language, using the `printf()` function may cause the compiler to drag in all of the floating point functions, increasing the program size by several thousand bytes.

DOS Function Calls

The MS-DOS operating system, in addition to taking care of disk I/O responsibilities, has an entire library of useful functions which may be called from a user program. These functions aren't described in the Users' Manual that accompanies MS-DOS. To find out about them, you have to either buy the Programmer's Utility Pack (PUP), or pick up a book about MS-DOS at the bookstore. If you're a programmer type, you really should invest in the PUP while they are still available from Heath/Zenith. You'll wonder how you ever lived without it! Or if you're just curious, most bookstores these days have dozens of books describing the inside workings of DOS. Don't worry about finding one that describes the Z-100 (You'll be looking for a long time). The MS-DOS function calls for any machine that runs DOS are the same.

There are three DOS function calls which deal specifically with output to the computer screen. They are numbers 2, 6, and 9. Here is a brief description of each...

Function 2 Display Character

This function is used to display a single character on the screen at the current cursor position. Load register DL with the character, and call interrupt 21H with register AH set to 2. Like this . . .

```
MOV DL, 'P' ; prepare to display a 'P'
MOV AH, 2   ; with function call 2
INT 21H    ; output to screen
```

This is the most bare-bones of all the DOS screen output functions. It will break

to the Control-C routine if Control-C is typed at the console during the time the function has control.

Function 6 Direct Console I/O

This function is similar to number 2, except that it also allows you to input characters from the keyboard (which we aren't concerned with here). Load register DL with any printable character, load AH with 6, and call interrupt 21H.

```
MOV DL, 'P' ; we'll display a 'P' again
MOV AH, 6   ; this time using function
INT 21H    ; output to screen
```

One important thing to note is that DOS function 6 does not check for Control-C at the keyboard. If it is important that your program not be interrupted, this is the screen output function to use.

Function 9 Display String

This function allows you to output an entire string of characters to the screen at one time. To use it, load register pair DS:DX with the segment and offset of the start of the text string. Load AH with 9, and call interrupt 21H. Note that the text string must end with a dollar sign ('\$') character, which is considered the string terminator. Example . . .

```
MOV DX, OFFSET TEXT ; point DX to text
MOV AH, 9           ; use function call 9
INT 21H            ; output to screen

...                ; other instructions
```

```
TEXT DB 'Hello there!$' ; the text string
```

This example assumes a couple of things. First, that the register DS has previously been set to the segment containing the label TEXT. It also assumes (as indicated by the ...) that some instructions (like a RET, or the end of the program) occur between the function call and the TEXT label, since obviously, the DB instruction cannot be interpreted as code instructions.

You might ask, "what if I want a dollar sign to be printed?". Answer . . . you'll have to use function call 2 or 6.

How to Use DOS Function Calls in Your Programs

The next question which seems obvious is how to use these function calls in your programs. Obviously, the functions are easy to handle with assembly language, as the examples above show. Using the calls with other languages may not be as easy. Some languages may have provisions for calling DOS functions (like most 'C' and Pascal compilers), while others may make their use very difficult (like BASIC). Even among different implementations of the same language (like Microsoft 'C' and Turbo 'C') the methods for using DOS functions may differ.

Using Microsoft 'C' as an example, here is how we would use DOS function number 9 to print a string . . .

```
char text[14] = "Hello there!$";
```

```
main() {
    bdos(9, text, 0);
}
```

The `bdos()` function will load register DX with a pointer to the start of the string text, load register AL with a zero (this is a dummy argument, since we don't need register AL), and then execute DOS function number 9.

Most 'C' and Pascal compilers come with a function similar to the Microsoft

`bdos()` function, but the syntax will vary.

Most of you who program in 'C' will be asking "wouldn't it be a lot easier to use the `puts()` function?, like this . . .

```
main() {
    puts("Hello there!");
}
```

Well, yes. That's the way I would normally output a string to the screen. It's hard to imagine writing an entire 'C' program using DOS function 9 for screen output. But if your program is short and simple (you don't get much shorter or simpler than "Hello there!"), you might be interested to note the following statistics; our example above, using the `bdos()` function, compiles into an .EXE program which

is 2357 bytes long. The example using the `puts()` function compiles into a program of 5245 bytes. And if you wanted to take it another step, and use the `printf()` function, the result is 7233 bytes.

Okay, now I notice that all the assembly language buffs are laughing, because writing this simple example in assembly language . . .

```
CODE SEGMENT
ASSUME CS:CODE, DS:CODE
ORG 100H
; start at 100H for .COM program
START: MOV DX, OFFSET TEXT
        MOV AH, 9
        INT 21H
        INT 20H
; this function ends the program
TEXT DB 'Hello there!$'
CODE ENDS
END START
```

. . . will result in an executable program that takes only 22 bytes!

Now that I have shown some examples of 'C' and assembly routines to make use of DOS function calls, I'll acknowledge that most of you probably use BASIC as your primary computer language. BASIC is one of those rather stiff languages (oh, I see the letters coming, now!) that doesn't have a great deal of flexibility for communicating with DOS or the hardware directly. But where there is a will,

there is a way. Witness the following . . .

Use DEBUG's mini-assembler to enter the following program fragment (don't include the comments). You can do this by running DEBUG, and then giving the 'A' command.

```
MOV BP, SP           ; get the stack frame
MOV SI, [BP+4]       ; get address of string descriptor
MOV DX, [SI+1]       ; get pointer to start of text string
MOV AH, 9            ; use function 9
INT 21H              ; display the string
RETF 2               ; return to calling program
```

Now if you unassemble what you have entered (using the 'U' DEBUG command), you can see the hexadecimal numbers that make up this program. They turn out to be . . .

```
89, E5, 8B, 76, 04, 8B, 54, 01, B4, 09,
CD, 21, CA, 02, 00
```

This sequence of bytes actually represents the small program subroutine given in our example above. You can call this assembly language subroutine from a BASIC program like this . . .

```
50 TEXT$="Hello there!"
90 FOR I=1 TO 8:READ PROC%(I):NEXT
95 DATA &HE589,&H768B,&HB404,&H0154
96 DATA &H09B4,&H21CD,&H02CA,&H0000
100 DEF SEG
110 DOS9F=VARPTR(PROC%(0))
120 CALL DOS9F(TEXT$)
130 PRINT
```

Note that the hex integers used in the DATA statements to make the program have each byte pair reversed. This is because BASIC stores an integer with the least significant byte first in memory, then the most significant byte. Also, if your assembly language routine doesn't use an even number of bytes, don't worry — just stick a zero on the end.

If this kind of thing interests you, you might want to refer to the Appendices of your BASIC manual for a more thorough description of how to use the CALL statement. Microsoft suggests writing the original program with an assembler, and then loading it with the BLOAD command, but for short assembly routines, I like my way better.

For the most part, it is ridiculous to go through all of this trouble to display a string on the screen using a DOS function call from a high-level language. But there are times when the knowledge of how to do so may come in handy.

BIOS and Monitor ROM Calls

Up until this point, all of the techniques we have discussed for placing text on the screen have been useable for any MS-DOS based computer. So if you have a PC clone taking up space on the desk next to your Z-100, what I have said would apply to it, too. But from here on out, we're talking Z-100 only. Making BIOS calls, monitor ROM calls, and writing directly to video memory, are very machine specific actions. Any use of these methods on a PC compatible is definitely guaranteed to cause an instant crash.

The Z-100 BIOS (Basic Input/Output System) contains a user interface which may be used to communicate with the disk drives, console, printer, or auxiliary device. Using the BIOS interface is something that is fairly simple once you get the

hang of it, but a proper tutorial is not really possible in the space we have left here. We'll do justice to using the BIOS interface in a future installment of Z-100 Survival Kit, but for now I'll just give you enough information to output characters to the screen (since that's the main subject we're trying to cover).

The BIOS function we're interested in is affectionately known as BIOS_CONFUNC. That's what they call it in the Pro-

```
' our text string
' read the assembly program
' the program (decimal integers)
'
' use BASIC's data segment
' get the start of the program
' call with string as argument
' do a CR/LF
```

grammer's Utility Pack, at any rate. You call the BIOS functions by making a long call to a jump table located at segment 40H. The address of the BIOS_CONFUNC vector is 40:0051 (hexadecimal). You can use the BIOS_CONFUNC routine to read or write a character to the console, check console status, etc. To write a character to the console (another word for the computer screen), you simply load register AL with the character, load AH with a zero, and call 40:0051. Like this . . .

```
BIOS SEGMENT AT 40H
ORG 51H
CONFUNC LABEL FAR
BIOS ENDS
```

```
CODE SEGMENT
MOV AL, 'P' ; display a 'P'
MOV AH, 0 ; code for CHR_WRITE function
CALL CONFUNC ; do it to it
... ; rest of program
CODE ENDS
```

The purpose of the first portion of this code is to tell the assembler where the BIOS_CONFUNC vector is located. If you will be using DEBUG to assemble the program, you can use the 40:0051 address in the call statement . . .

```
MOV AL, 50 ; this is a
'P' to DEBUG
MOV AH, 0 ;
CALL 40:0051 ; call BIOS_CONFUNC
```

The BIOS_CONFUNC character output routine ends up calling the MTR-100 monitor ROM to get its work done. It uses an entry point into the MTR-100 named MTR_SCRT which is at address FE01:0019. If you prefer, your program can call

CLASSIFIED ADS

LIKE NEW Orchid Designer VGA \$250.00. R. Speidel, Box 95E, Rt. 1, Emmaus, PA 18049.

Z-100. PRINTER, MODEM \$\$\$\$ Extras. Call for list. \$900.00. (808) 878-6096.



Want New And Interesting Software?
Check Out HUG Software

Are you reading
a borrowed copy of REMark?
Subscribe now!

the monitor ROM directly by using this code . . .

```
MTR SEGMENT AT FE01H
ORG 19H
SCRT LABEL FAR
MTR ENDS

CODE SEGMENT
MOV AL, 'P' ; display a 'P'
CALL SCRT ; using the MTR-100 ROM routine
... ; rest of program
CODE ENDS
```

The advantage of going straight to the MTR-100 routine is to tweak the last bit of speed out of the system. To give you an idea of the speed difference, I tried displaying 50,000 characters using a looping assembly language routine. The results . . . using the BIOS_CONFUNC routine took 23 seconds. And using the MTR-100 SCRT routine, the time was 17 seconds. Just for laughs, I also tried using MS-DOS function call 2 — it took 58 seconds.

In defense of the BIOS interface, I must say that for most programs it is more convenient to use the BIOS because of the other console functions available. Another advantage to using the BIOS interface was to avoid problems if a new

Prime H/Z Enhancements!

Clock Uses no Slot

FBE SmartWatch: On-line date/time. Installs under BIOS/Monitor ROM. Ten year battery. Software included. Works with all Heath/Zenith MSDOS computers. For PC's \$35; Z-100 \$36.50. Module \$27.50

H/Z-148 Expansions

ZEX-148: Adds 1-1/2 card slots. \$79.95. ZEX-148 + SmartWatch \$109.95
XP-148: PAL chip expands existing 640K memory to 704K. \$19.95

H/Z-150 Stuff (Not for '157, '158 or '159)

VCE-150: Eliminate video card. Install EGA/VGA card. All plug in. Includes VEM-150, RM-150. Requires SRAM chip. VCE-150 \$39.95, SRAM Chip \$15
VEM-150: Card combines existing two BIOS ROM's into one socket. \$34.95
RM-150: Decoder PROM used in removing video card. With detailed instructions. \$9.95

ZP640 PLUS: Expand to 640K/704K by adding 2 banks of 256K RAM chips (not included). ZP640 PLUS \$19.95 (first one); \$9.50 thereafter.

LIM 150: 640K RAM plus 512K of simulated Lotus/Intel/Microsoft EMS v3.2 expanded memory. Installs on H/Z-150/160 memory card. No soldering. Requires forty-five 256K RAM chips (not included). LIM150 \$39.95
Mega RAM-150: Get 640K/704K main memory plus 512K RAM disk on H/Z-150/160 memory card. No soldering. Without RAM chips \$39.95

COM3: Change existing COM2 port address. Internal MODEM at COM2. Don't lose serial port. COM3 \$29.95

Maximize Your Z-100

ZMF100A: Put 256K RAM chips on "old" motherboard (p/n 181-4917 or less). Expand to 768K. No soldering. Without RAM chips. \$65.00
ZRAM-205: Put 256K RAM chips on Z-205 board. Get 256K memory plus 768K RAM disk. Contact us for data sheet before ordering. Without RAM chips. \$49.00

Z-171 Memory Expansion

MegaRAM-171: Put 256K RAM chips on memory card. Get 640K memory plus 384K RAM disk. \$59.95

H/Z-89 Corner

H89PIP: Two port parallel printer interface card. With software. H89PIP\$50.00; Cable \$24.00

SPOOLDISK 89 and SLOT 4: Cards still available. Contact us for information.

Order by mail, phone or see a Heath/Zenith Dealer. UPS/APO/FPO shipping included. VISA or MC. WA residents add 8.1% tax. Hours: M-F 9-5 PST. We return all calls left on answering machine!

FBE

FBE Research Co., Inc.
P.O. Box 68234, Seattle, WA 98168
206-246-9815

Reader Service #104

version of the MTR-100 ROM was released with different entry points. But since the Z-100 is obsolete machinery, I doubt if Zenith will be providing any more MTR-100 versions, so that is no longer a factor — all of the MTR-100 versions I know of have the MTR__SCRT entry at FE01:0019.

Another consideration when deciding whether to use the BIOS interface or the MTR-100 ROM routine, is that the BIOS causes a user interrupt to be generated whenever a character is output to the screen. Memory resident utilities that depend on this interrupt to gain control will not work correctly if you bypass the BIOS by going straight to the MTR-100.

Using BIOS Output with BASIC

Here is a routine that you might actually find useful in your BASIC programs. Enter the following program using DEBUG's mini assembler . . .

```
100:  MOV BP, SP           ; get stack frame
      MOV SI, [BP+4]     ; find start of string descriptor
      MOV CL, [SI]       ; get length of text string
      CMP CL, 0          ; if null string,
      JZ 11F             ; return immediately
      MOV CH, 0          ; CX is now character count
      MOV SI, [SI+1]     ; SI points to start of string
111:  MOV AL, [SI]        ; get a character
      MOV AH, 0          ; code for CHR_WRITE function
      PUSH SI            ; save string pointer
      CALL 0040:0051     ; call BIOS_CONFUNC
      POP SI             ;
      INC SI              ; point to next character
      LOOP 111           ; loop until done
11D:  RETF 2             ; return to BASIC
```

Now if you unassemble this program with DEBUG, you will get the following hex bytes . . .

```
89 E5 8B 76 04 8A 0C 80 F9 00 74 13 B5
00 8B 74 01
8A 04 B4 00 56 9A 51 00 40 00 5E 46 E2
F2 CA 02 00
```

Here's an example of how to use this assembly routine in a BASIC program . . .

```
40 DIM PROG$(17)
50 TEXT$="Hello there!"+CHR$(10)+CHR$(13)
90 FOR I=1 TO 17:READ PROG%(I):NEXT
95 DATA &HE589,&H768B,&H8A04,&H800C
96 DATA &H00F9,&H1374,&H00B5,&H748B
97 DATA &H8A01,&HB404,&H5600,&H519A
98 DATA &H4000,&H5E00,&HE246,&HCAF2,&H0002
100 DEF SEG
110 BIOS=VARPTR(PROG$(0))
115 FOR I=1 TO 1000
120 CALL BIOS(TEXT$)
125 NEXT
```

You'll notice that this assembly routine lets you print any BASIC string using the Z-100 BIOS interface. This will be about twice as fast as using the BASIC PRINT statement. The program sample above will print "Hello there!" on the screen 100 times.

Handling Special Characters

All of the methods of displaying text that we have discussed up to this point will handle non-printable ASCII characters in a special way. For instance, they will all recognize a carriage return or line feed

code. They should also recognize special escape sequences that may be used to clear the screen, position the cursor, etc.

If you are using a high-level language, the character output routine you use may filter out some of these special characters. Therefore, you may need to experiment with the results (or consult the manual) to determine exactly how a particular high-level routine affects the character stream. For example, some routines may automatically insert a carriage return character whenever a line feed is sent to the screen. And some routines may filter out escape characters, making it impossible to send Z-100 escape sequences to the console.

To Be Continued . . .

In the next installment of Z-100 Survival Kit, I'll wrap up our discussion about displaying text on the screen, by describing how to write directly to video memo-

ry. I'll be including code samples for a character output routine, and we'll also talk about how to use large or fancy text in your programs. *





ZORK

for the Z-100

Dan Gnagey
 21 Park Street
 Caribou, ME 04736

My first introduction to text adventure games came while a college student working at the university computing center. As a student programmer, I had access to (almost) unlimited computer funds on the IBM-370 mainframe the university used for its computing tasks. Someone had loaded Willie Crowther's Original Adventure game on the system and between projects, or when boredom struck, I would spend hours playing the game at a Decwriter terminal. Several years ago when I bought my Z-100, I again spent hours typing in "Plugh" and "Xyzzy" while playing HUG's version of Adventure. I read several reviews about a game called "Zork" and decided that this game might just be for me. I called several companies that advertised Infocom games in REMark and Sextant magazines and found to my dismay that "according to them" the games would not work on the Z-100. This bothered me as I could not understand why a text adventure game would not work on my "generic" MS-DOS machine. After some digging around and quite a bit of asking, I finally found all the necessary information to make most of the games work on a Z-100 without ZPC or another IBM emulator.

ANSICON.DVD and CTTY

One of the first requirements I found was the need to have ANSI.SYS on the

system boot disk. This necessitates the use of MS-DOS 2.0 or higher on the Z-100 which makes the ANSICON.DVD device driver available. ANSICON.DVD is the Z-100 equivalent of IBM's ANSI.SYS device driver. It drives the keyboard and monitor much as the CONSOLE device driver does and, in addition, translates ANSI escape codes, that cause special console functions, into their Z-100 equivalents. It can be found on MS-DOS 2.0 disk number two under the directory \DEV. The device driver can be installed on the Z-100 by adding the following statement to your CONFIG.SYS file:

```
DEVICE = ANSICON.DVD
```

When you boot your system, the device driver will be loaded into memory and ready to go. This is the only way the device driver can be loaded. Now, a way must be found to get the input directed to the console to route through the ANSI device driver. One way to do this is to simply redirect the input to the ANSI driver at the console through the use of a command line, such as:

```
TYPE filename.ext > ANSI
```

This will also work with a COM file such as those that compose Infocom's text adventures. Redirection in this manner can be accomplished by typing (or adding to your AUTOEXEC.BAT file) a line such as:

```
WISHBRIN > ANSI
```

Another (and I feel more elegant) method of redirecting the console input is through the use of the MS-DOS CTTY command. CTTY is a command that al-

lows one to change the device from which one issues commands. Although casual reading of the MS-DOS User's Guide may lead one to believe that CTTY will only accept the device names AUX, CON, COM1, and COM2, careful scrutiny reveals that the command will accept the name of any (character-oriented) device. Thus, all console input can be routed through the ANSI device driver by typing or including the following statement in your AUTOEXEC.BAT file:

```
CTTY ANSI
```

Control can be returned to the CONSOLE device river by typing:

```
CTTY CON
```

Although the ANSICON device driver included with MS-DOS 2.0 and 3.0 is adequate to use with Infocom's text adventures, there is at least one other ANSI device driver that has some advantages over the "bare bones" Zenith version. The device driver I prefer is called ZANSI.DVD Version 1.03 and is available from Joe Keenan's Black Cat 100 Bulletin Board. His user-supported (shareware) version supports the color-setting sequences used on PC-compatibles, the extended ASCII character set, and all sequences as defined for either the Z-19/89, Z-100, or Z-100/PC series as much as is possible on the Z-100. The only real difference, as far as the Infocom games go, between the two device drivers is that with ZANSI the game's status line is shown in reverse video while ANSICON apparently does not support that function.

Infocom's Text Adventures

I hesitate to classify the Infocom programs as games, lest some of you underestimate their sophistication. Infocom refers to them as "interactive fiction" and I certainly will not argue with that description. Playing one of these games is very similar to reading a well-written novel, with the exception that your actions have a definite effect on the outcome of the story. Zork is certainly the most famous of Infocom's products and lies in the same vein as Willie Crowther's original Colossal Cave Adventure. It takes you to the ruins of the Great Underground Empire, where you travel in search of the incomparable Treasures of Zork. Infocom's products have expanded far beyond those humble beginnings, and their stories now include fantasies, thrillers, science fiction, and mysteries; all with a good measure of humor thrown in. They are guaranteed to keep your attention for hours while you explore new worlds, solve problems, and enjoy the superb descriptions of places and objects.

Wishbringer

For those of you that have not experienced interactive fiction I will present a short review of Wishbringer, an Infocom program written for newcomers to this wonderful distraction. Infocom's attention to detail is evident as soon as the shrink wrap is removed from the package. In addition to the Wishbringer disk and instructions, the package contains a mysterious sealed envelope (with a real postage stamp), a postal map of the village, a white plastic stone, and a booklet titled "The Legend of Wishbringer". These items are useful in helping set the stage for the game and are typical of the extras inserted into all Infocom products. The story was written by Brian Moriarty with the newcomer to interactive fiction in mind. It starts out fairly easy and the challenge increases as you progress through the story. You are the central character (as in all Infocom stories), a postal clerk in the small village of Festeron. As the story begins, you are tasked with delivering a small envelope to a magic shop, where you discover that an old woman's cat has been kidnapped. Getting to the shop is an adventure in itself, but the story only really begins when you deliver the envelope. When you leave the shop you find that your small town has been transformed into a horrifying world full of trolls, vultures, and assorted other nasty creatures. So I won't ruin the story for anyone, I will stop here by saying that Wishbringer is a wonderful story for anyone 9 to 90. It serves as a good introduction to Infocom's world of interactive fiction. Should you think that your children will be wasting time playing the game, I can assure you that their minds will be hard at work trying to solve the many puzzles in the

story.

From a technical standpoint, the Infocom games are sophisticated and use very powerful sentence parsers to examine the plain English sentences you use to perform actions in the story. The programs examine each sentence for nouns and verbs, then base their actions on them. Some sentences that would be understood by Wishbringer include:

>WALK NORTH
>DROP THE ENVELOPE ONTO THE COUNTER
>WALK INTO THE POLICE STATION
>TAKE THE BOX. OPEN IT. PUT IT ON THE TABLE
>ASK MISS VOSS ABOUT THE VIOLET NOTE

The program's ability to understand your sentences allows you to spend time playing the game, not trying to figure out what combination of words are needed to perform a task. Along with the excellent sentence parser, several other features enhance Infocom's games. You can make a transcript of the story on your printer by using the SCRIPT command and then stop it with the UNSCRIPT command. You can save your position in the game at any time through the use of the SAVE command. This puts a "snapshot" of your place in the story on disk so you can return to that exact spot at a later time or if you should get yourself "killed" during play. The RESTORE command lets you resume the story at the point you used the SAVE command. You can save a number of positions by using different file names for them (the default is WISHBRIN.SAV). In addition to the excellent quality of the programs, Infocom has a technical problems helpline, publishes a newsletter, "THE STATUS LINE", that you receive free upon registering your program, and has hint books available for those who just cannot solve that one especially difficult problem. Wishbringer is one of a number of Infocom's games that are available by direct mail at the ridiculously low price of \$14.95. Others available at this great price include:

Ballyhoo	Mystery
Cutthroats	Adventure
Deadline	Mystery
Hitchhiker's Guide to the Galaxy	Science Fiction
Infidel	Adventure
Planetfall	Science Fiction
Seastalker	Adventure
Starcross	Science Fiction
Suspect	Mystery
Suspended	Science Fiction
Witness	Mystery
Zork I	Fantasy
Zork II	Fantasy
Zork III	Fantasy

I hope this article saves those of you that would like to start reading interactive fiction on the Z-100 some of the confusion I had while trying to get Infocom's products to work on my computer. For those of you that would like more infor-

mation on ANSICON.DVD and CTTY refer to your MS-DOS User's Manual and the Programmer's Utility Pack. ZANSI.DVD comes with a good documentation file that covers just about everything you would want to know about ANSI device drivers and the author promises support through his bulletin board to registered users. For those of you with PC compatibles, Infocom's interactive fiction is easily installed on the PC and provides the same great fun. I hope those of you that have not read interactive fiction give it a try. Infocom has a story for every taste.

Products List

ZANSI.DVD \$15.00
Black Cat Software
c/o Joe Keenan
5430F Lynx Lane, Suite 306
Columbia, MD 21044
BBS (301) 598-8248

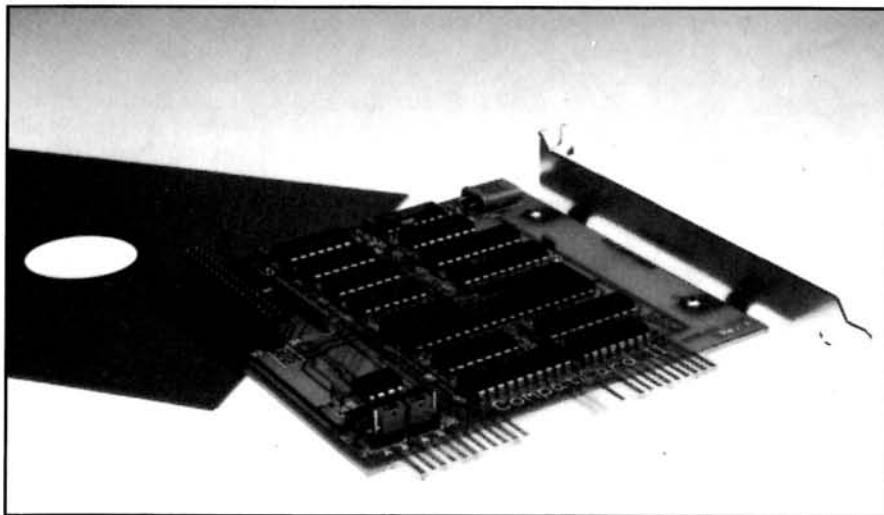
MS-DOS Version 3 \$150.00
Model Number OS-63-30
MS-DOS Version 3
Programmer's Package \$225.00
Model Number CB-3163-30
Heath/Zenith Computers & Electronics
2987 Lakeshore Drive
St. Joseph, MI 49085

HUG MS-DOS Adventure \$10.00
P/N 885-3016-37
Heath/Zenith Users' Group
P.O. Box 217
Benton Harbor, MI 49022-0217

Infocom Games for MS-DOS
Infocom
P.O. Box 478
Cresskill, NJ 07626
(800) 262-6868 Orders
(617) 576-3190 Technical Problems *

**EXPLORE
NEW WORLDS
WITH
HUG
GAME
SOFTWARE**

A 1.44 MB Mini-Floppy



Mark R. VanSickle
1115 N. Altadena
Royal Oak, MI 48067

For Your Heath/Zenith PC-Compatible

Background

Ever since I put a 20 Meg Winchester in my Z-158, I've been enduring the periodic ritual torture of backing up the hard drive to forty or fifty 360K floppies. On the newer AT-class 80286 machines at work, I've been able to store less-often-used large programs or just great gobs of data on those nifty little 3.5" 1.44 Megabyte disks, keeping space on the hard drives free for things that really need to live there. Backups at work are almost fun now, and there are the additional benefits of shirt-pocket disk size and better data survivability with the hardshell case. Okay, the 1.44 Meg 3.5's are still a bit pricey. Their cost-per-360K, at the best price I've found, is still nearly three times that of a 40-cent bulk 360K floppy. But despite the current cost of the disks, all of the advantages noted above made being able to use the 3.5" High Density (as opposed to the 720K variety) disks with my Z-158 a very attractive idea and would extend the useful life of my "tiny" 20 Meg hard drive besides.

Any of the Heath/Zenith 8088-based PC-compatibles can run the smaller 720K 3.5-inchers by just upgrading to Zenith MS-DOS 3.2. To use the 1.44 Meg drives, an aftermarket Disk Controller must be added to the system. I have recently successfully installed such a controller and

3.5" outboard drive on my Z-158, with the assistance/guidance of Ray Massa of Studio Computers. My initial inquiries to him stemmed from ads seen in the back of Infoworld for XT-compatible 1.44 Meg drive support systems from two vendors: Sysgen, who sell the Bridge family of products; and Micro Solutions, makers of the CompatiCard series of controllers.

Sysgen's ad was a bit fancier, I had cut out their ad first, and the fact that they had an (800) number was not lost on this cheap Dutchman either. So, based on these swell criteria, I called Sysgen first. The woman answering the WATS line call told me that there would be a two-day wait on a callback from a tech person. This proved fortuitous, as my 1980's "once I've thunk of it, I want it yesterday" mindset dictated that I immediately call Micro Solutions in Illinois, toll call notwithstanding.

"Will It Work With . . ."

Speaking with Dave, a Micro Solutions technical support person, I explained my concern about the 158's backplane-only nature and the fact that the newer controller apparently would have to replace or coexist with the original floppy disk section of the 158's two main system boards. Dave said that he had dealt with a Z-158 situation involving

one of their other boards earlier that same day (my paranoia antennae vibrated at this — it sounded too pat), and that a 158 wanted supplemental controllers to live in the third I/O port. Though unsure whether any re-jumpering of the 158 board would be necessary, he offered to check. I had already made some inquiries about this, thinking that the new disk controller would replace the old one. Since the nature of the actual installation is such that both controllers are functional, it turned out that no 158 jumper changes are necessary.

Dave pointed out that I would need a version of DOS that supports the high-density minifloppies (which I already knew) and that all necessary driver software would be supplied. When asked about specific drive brands, he said that their cards worked with all makes tested, except that some Sony units require a cable adapter. The drive Ray Massa and I had discussed was a Toshiba FDD4603, so this wasn't a problem.

Taking the Plunge

The specific nature of Dave's answers lent them the ring of truth (as opposed to the thud of snow job), so I ordered the \$125.00 CompatiCard II through Studio Computers. Buying the board through Ray gave me something of a safety net in that

the past couple of upgrades to my system have been done in his store on a try-before-you-buy basis. If we couldn't get this setup to work, he would return the board and I wouldn't be stuck holding the \$125.00 mail-order bag; but judging from the quality of the rest of their operation, Micro Solutions may in fact have a return policy — I didn't ask.

After I'd already ordered the Micro Solutions card, I did get a callback from Sysgen; however, their technical guy was unable to give me any info specific to a 158 or to H/Z PC's, in general. The gist of what he said was "call us if it works". I was pretty sure at this point that I had managed to meander in the right direction, at least from a purchasing confidence standpoint. Sysgen's Bridge board is cheaper at \$95.00, and it may well work fine for those with less fear of the unknown. They also offer a drive for around \$225.00. At that price, I would hope that a case and supply are included.

At the time my order was placed (Nov. '88), two boards were advertised by Micro Solutions: the \$175.00 full-blown CompatiCard, which comes with a variety of edge, header, and DB-type connectors and supports up to four additional drives; and the \$125.00 CompatiCard II that I bought, which has one 34-pin header and will run two drives. Since then, they've been advertising the Megamate, a drive/cable/card/software package for \$349.00, but I expect that the separate cards are still available.

Installation

When the CompatiCard arrived, I took my 158 system box (hidden implication here is that the Power Cord had been removed!), plus an external drive housing with power supply which I already had, over to Studio Computers. After a speed-read of the excellent manual, I set the board's two I/O port jumpers, per the Address Jumper Settings Table, for Third Controller (Port 360h-367h) configuration — as Dave had instructed. The half-size card was plugged into an available slot, and the null bracket for the neighboring empty slot was removed to make an expedient, non-pinching exit for the ribbon cable (grinding or filing off some metal from the CompatiCard's bracket edge would be slicker). The manual fully describes the correct cable orientation with respect to connector pin 1 at both drive and controller ends.

The only noteworthy condition found during the mounting of the drive in the external housing was that we had to omit one screw of the four that secure the housing cover to its frame. It was found that the screw was just long enough to protrude through the housing frame and push against the drive chassis, torquing the drive enough to keep the disk from ejecting smoothly. This may be some-

thing to keep an eye out for, whether you mount your drive internally (which should be about as easy as this external setup) or "outdoors".

We then turned to the Toshiba drive's jumper block and introduced one glitch into what would have been a smooth installation. The Toshiba drive manual's FDD4603 jumpering diagram said to locate the A/B jumper at A for an XT or compatible, and this is correct. Where we goofed was in moving the DS1/DS2 jumper from its factory-set DS2 position to DS1. I later found it mentioned in the CompatiCard manual that the address selector should be set for the second drive (so much for speed-reading). The drive comes with DS2 set to conform to the twisted-cable AT "standard" initiated by IBM so that an out-of-the-box drive will be software-selected. Heaven forbid that their computer techs should ever have to move a jumper — let's introduce confusion into drive cabling instead (and leave it to IBM to create something twisted). We moved to DS1 in the mistaken assumption that the only drive on a new node should be the lower drive-select number. The mistake was detected when the drive light stayed off on our first "G:<Return>" test. It wasn't hard to figure out the problem, since all other aspects of the installation were so simple and clearly defined. So leave the jumper at DS2 already and only ensure that the A/B jumper is at A. Micro Solutions has neatly handled the DS2 and Twisted/Straight cable situation in the way you set up the device driver.

Software Setup

After the computer was reassembled and powered up, we booted from a system 5.25" floppy. The modification to CONFIG.SYS described below was made to this disk, and we then used it to test the 3.5" software and drive. This seemed a more prudent method than involving the winchester in what was still an experiment.

The software driver for the CompatiCard is CCDRIVER.SYS. After it and the 3 other files on your working copy of the Micro Solutions disk have been copied to your system disk, the physical drive address that the driver will use is easily determined from the first of two Drive Installation Tables in the CompatiCard manual. Which of the 16 possible address numbers (0-15) is correct is derived from: 1) how the two card jumpers were set, 2) which of the four card connectors (four are possible with the full CompatiCard; the "II" is simpler) is being used, and 3) whether the ribbon cable used is the familiar straight type or the IBM type, which has lines 10 and 16 transposed/twisted. In my installation, this address # was 9.

The second Drive Installation Table assigns a number from 0 to 7 to each of

the eight types of drives which can be used. For my 3.5 inch high capacity 720K/1.4M drive, this # was 7.

These numbers are passed to the device driver in the form of switches after a DEVICE= statement in your CONFIG.SYS file. The format is:

```
DEVICE=CCDRIVER.SYS /x,y,z
```

where x is the drive address #, y is the drive type #, and z is the step rate, which defaults to 6 mS and can usually be omitted, since 6 mS is typical of almost all drives. My CONFIG.SYS file ended up being

```
BUFFERS=15
```

```
FILES=20
```

```
DEVICE=CCDRIVER.SYS /9,7
```

Brief Recap of the Main Steps

1. Remove power cord, other external cables, case.
2. Set the jumpers on the new card.
3. Check the drive jumpers.
4. Install the card, cable, and drive.
5. Re-install case, monitor, keyboard, power, etc.
6. Copy driver files & set CONFIG.SYS driver line.
7. Reboot the computer from the modified floppy.
8. Put a disk in the new drive; type G:<Return>.

... And the Drive Light Comes On

The drive has worked flawlessly for nearly a month now. Formatting and general disk I/O times appear competitive with those seen on HP Vectra 286 drives at work. The only test of the system still remaining is to perform a hard drive backup using FastBack Plus. I've already gotten exact instructions from another knowledgeable Micro Solutions tech person on how to configure FastBack. What's lacking at this point is the software itself (it's Dec. 28 and the tiny madwomen living here had to have their toys too — LOTS of them). Until I get the program, I have Micro Solutions' assurance that it will work — and I have every reason to believe them.

Other Utilities and Documentation

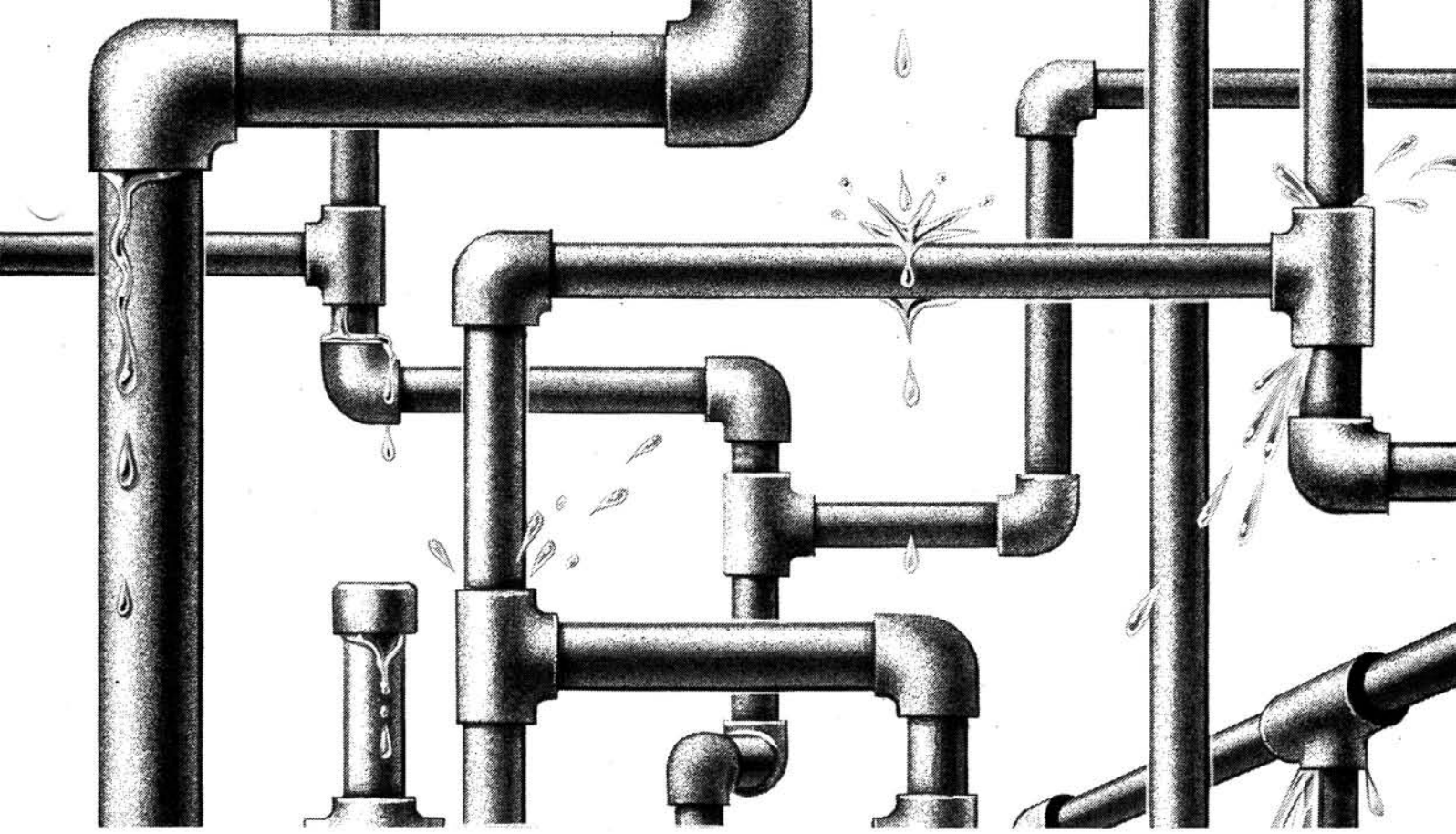
CCDRIVES.COM — When your computer is rebooted with the new CONFIG.SYS and CompatiCard files aboard, the CCDRIVES utility, which runs at each bootup or on demand, announces the drive letter and particulars for all drives connected to the CompatiCard. On my system, the status message looks like this when CCDRIVES is summoned from the C:> prompt:

```
CCDRIVES Version X.XX
```

The following drives are available:
Drive G: — 3.5 inch High Capacity.

If you're running multiple drives, you can rearrange the drive letters assigned to different drive types by changing the order in which the switches for each drive appear in the CONFIG.SYS file.

Continued on Page 46



Getting Started With . . .

The multi-tasking that OS/2 provides will probably be financially out of the question for many micro users. Yet, no one can deny the benefits of running more than one application, of being able to transfer data between your spreadsheet, data base, and word processor. I mean, who wouldn't want to print a spreadsheet chart in a document, or easily use data base records for form letters or labels?

The alternative to multi-tasking for these functions is an "integrated" package — a single piece of software that combines the four basic applications (word processing, spreadsheet, data base, and graphing), and perhaps adds in a communications package for extra measure.

This approach is certainly not new. Lotus 1-2-3 started it all, Symphony tried to make the package complete, and other integrated programs have shown up over the years. All of them, however, are always faced with the same comparisons — how do the individual parts of the package compare with their stand-alone competitors. Does the spreadsheet stand up against Lotus? Does the data base function measure up to dBASE? Is the word processor as complete as WordPerfect or

Word?

Perhaps these questions are justified, perhaps not. Integrated packages play a rather unique role in the software industry. Unique in the sense that they provide functions, and powers, to a class of users who might really not care about these comparisons.

Microsoft Works, available for both the IBM and MAC worlds, is one of these new breed of integrated packages. It includes a word processor, spreadsheet, data base, and communications application, as well as a spelling checker and keyboard macro program. Like others in its class, Works is designed for users who want the basic functions of applications programs in an easy to use and inexpensive package, who want to quickly switch back and forth between applications without the expense of OS/2 and megamemory, and who want to share their data between programs. To gain these decided benefits, they are willing to forgo some of the power-features found in stand-alone programs, while really not giving up any of the practical functions. Keeping the meat and potatoes — the functions that a majority of users need — while giving up the caviar, if you forgive the analogy.

For example, Works has found a solid niche among MAC users and in education, two areas where the typical user is more concerned with ease of use and learning

than anything else. According to Microsoft, over 50,000 schools have adopted Works as an educational tool, teaching basic applications and applying it to courses ranging from day care to carpentry. At the same time, Works can provide the business user with the same set of tools, a handy package to use in the executive laptop or desktop.

Let's take a look at some of the advantages of an integrated package, then jump into each of its major applications.

First, whenever possible, the function keys, command structure, and pull down menus are the same across all of the applications. For instance, you press Shift-F1 for the tutorial, or F7 to repeat a search no matter which part of Works you are using. There is no need to learn one set of commands for the word processor, a completely different set for the spreadsheet. So once you're comfortable with one program, you know the basics of them all.

If you are already experienced on multiple applications, you might not see the benefit in this. But new users find the different command structures of separate applications confusing and a stumbling block to productivity.

Second, there is no need to struggle

Alan Neibauer

**1138 Hendrix Street
Philadelphia, PA 19116**

Works

with sharing data and files between applications. You don't need a utility program to use the data base file with the spreadsheet, or special graphics programs to combine a chart with a memo or report. From within the word processor, for instance, you can print labels from data base records, or use the records for form letters. Works will even display a list of data base files and their field names for your reference. Using similar techniques, you can add to your document spreadsheets, data bases, and charts that you created with other applications. In fact, since Works includes a communications module, you could then have it dial the phone and transmit your file to the office for printing, or pick up your electronic mail, all without exiting the program or even closing the word processing application. While you cannot see more than one application on the screen, as with Windows, you can easily switch back and forth between up to 8 different windows, moving and copying text.

That's all pretty good for a package with a list price at \$149 that can be used on systems with floppy, as well as hard disks. How much do you have to forego for this versatility? Let's take a look.

Installing Works

An easy-to-use Setup program loads the necessary printer information onto your working floppy or hard disk. You designate both a text and graphics printer (they can be the same) for printing documents and charts created with the spreadsheet module. If you have a two drive system, you can then use almost all of Works without ever changing a disk. You can create documents, spreadsheets, data bases, and communicate via modem from the program disk in drive A, while storing your documents and other files in drive B. You will have to change disks to use the help files and the integrated spelling checker. But except for that, all of the applications are available to you.

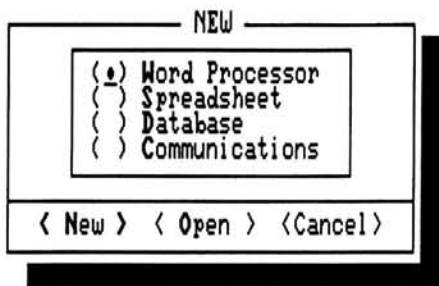
Let's assume that you've followed the setup program and now have a working copy of Works. To get you started, we'll complete a small document, complete with a spreadsheet, and print form letters to persons in a data base. That's a lot for one Getting Started article, but it will show you in a nutshell the versatility of Works and get you started with three of its applications. This way, you'll see how the different parts interact more than the details of the applications themselves.

The Spreadsheet

To start Works, insert the installed program disk in drive A or log onto the Works directory of your hard disk. Type WORKS, then press Return. In a moment, you'll see the initial menu screen showing four major applications (Figure 1).

Press the down arrow key to select

File Window



Creates new Works file.

Figure 1

the Spreadsheet option, then press Return. The spreadsheet application appears as seen in Figure 2. The top row lists the commands available. This will be similar in all of the applications. The line beneath that is called the formula bar. It will display the contents of the active spreadsheet cell, identified at the bottom left of the screen, the active cell reference. Notice that cell A1 is now active. The last line on the screen is called the message line. Here you'll see prompts or messages from Works.

If you've ever used any spreadsheet before, you'll have no trouble here.

Let's begin by entering the labels, or the names of each of the rows and columns.

1. Press the right arrow key to reach cell B1. Make sure B1 appears in the bottom left.
2. Type First, then press Return. "First" appears both in the spreadsheet and in the formula bar.
3. Press the right arrow key to reach C1, then type Second.
4. Press the right arrow key instead of Return. The next cell, D1, becomes active and "Second" appears in cell C1.
5. In a similar manner, enter Third in cell D1, and Fourth in cell E1.
6. Press the Home key to activate cell A1. Home moves the cursor to the first column of the row.
7. Press the down arrow key to reach A2.
8. Type Sales, then press the down arrow key.
9. Now complete the spreadsheet so it appears like this:

	First	Second	Third	Fourth
Sal				
Expenses				
Rent				
Salary				
Overhead				
Total				
Net Profit				

As you can see, the spreadsheet shows sales and expenses for the four

quarters of the year. It contains "labels", or words which describe the contents of the rows and columns. Now it's time to enter the number.

10. Use the arrow keys to reach cell B2, type 10000, then press Return to add the figure to the cell. If you type 10,000, Works will automatically drop the comma.
11. Now enter the following numbers into the spreadsheet at the cells given:

Cell	Enter
B5	500
B6	3500
B7	400

We want to have the total of the expenses in cell B8 and the net profit, the difference between sales and expenses, in B10. Rather than manually calculate the figures and type them in, we'll enter formulas to do the work for us.

12. Place the cursor in cell B8.
13. Press =, the equal sign. The message line changes to Edit formula — telling you that Works is ready to receive a formula rather than a label.
15. Type SUM(B5:B7), then press Return. While the formula bar shows that the cell contains SUM(B5:B7), the cell itself shows 4400, the sum of the items in cells B5 through B7. In fact, it's important that you understand the difference between the real contents of the cell and what is simply displayed there in the spreadsheet.

The formula in the formula bar is what is really in that cell of the spreadsheet. Works computes, or evaluates, the formula, and displays the results on the screen. You'll see why this distinction is important in a moment.

Sum is a built-in function of Works. It calculates the total of the range of cells shown (the first and last cells of the range are separated by a colon). The cell contains the formula, but displays the result.

16. Place the cursor in cell B10.
17. Enter the formula for the net profit.
 - a. Press =.
 - b. Type B2-B8, then press Return. The difference between the two cells, which represents the profit made for that quarter, is shown in the cell.

To see the power of spreadsheets, let's change some of the numbers in the

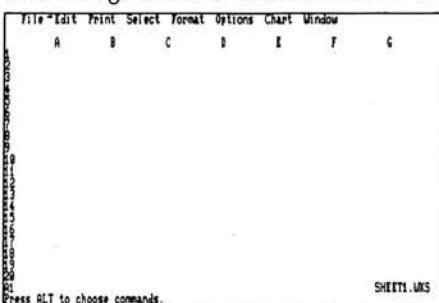


Figure 2

other cells. If you change any of the cells referenced by the two formulas, the amounts will be automatically recalculated and displayed.

18. Place the cursor in cell B5, type 1000, then press Return. Not only did that cell change, but so did the total expenses in cell B9 and the net profit in cell B10.

It's now time to complete the spreadsheet for the last three quarters. But since we want to enter formulas in the other columns for the total and net, let's copy those we've already entered.

Start by copying the formula in cell B8 to cells C8, D8, and E8.

19. Select cell B8.

20. Press and hold down the shift key, then press the right arrow three times. Leave go of them both.

Notice that with the shift key pressed down as you move the cursor, the cells all become active, or selected. (This same technique is used to select text in the word processing application as well.)

21. Press Alt to activate the command bar, then E for the Edit options. The option we want to select is Fill Right. This means to use the first cell to fill the selected cells to the right.

22. Press R. Now zeroes appear in each of the other cells, not the contents of B8. When cells are filled, or copied, in this manner, the formula is copied and automatically adjusted to its new location. Because there are no numbers in the referenced cells, they calculate to a zero value.

23. Press the right arrow key. Notice that the cell contains SUM(C5:C7). Works adjusted the formula to include the new columns. This type of cell reference is called "relative".

If you had entered the formula as SUM(\$C\$5:\$C\$7), however, Works would have copied that exact formula to the other cells, not adjust it across columns. This type of reference is called absolute.

24. In the same way as you did in step 23, copy the formula in cell B10.

a. Select cell B10.

b. Press and hold the shift key.

c. Press the right arrow three times.

d. Press Alt, E, then R for fill right.

Instead of typing in all of the numbers in the remaining cells, let's use the fill right command again.

25. Select cell B2.

26. Press and hold down the shift key while you move the cursor to cell E7.

27. Press Alt, E, then R. Finally, change the sales figures for the last three quarters. Change cell C2 to 12000, cell D2 to 13000, and cell E2 to 8000. Leaving the other figures alone for now, look at the impact of changing sales when expenses remained constant.

	First	Second	Third	Fourth
Sales	10000	12000	13000	8000
Expenses				
Rent	1000	1000	1000	1000
Salary	2000	2000	2000	2000
Overhead	400	400	400	400
Total	4900	4900	4900	4900
Net Profit	5100	7100	8100	3100

Figure 3

28. Use the arrow keys to activate cell A1.

29. Press Ctrl-; (Press and hold down the Ctrl key, then press the semicolon.) Your system date appears in the formula bar.

30. Press Return to insert the date in cell A1. (Figure 3)

Now save and print the spreadsheet.

31. Press Alt, F, then S, for save. Since this is a new spreadsheet, the Save As box appears with the default file name Sheet1.WKS at the prompt. Press Return to save the spreadsheet.

32. Press Alt, then P twice. The print menu appears listing a number of options.

33. Press Return to print a copy of your spreadsheet. You'll notice that the column headings do not appear to line up properly over the numbers. We'll take care of this in a little while.

There's a great deal more that you can do with the spreadsheet application beyond simple budgets like this one. Cells can be formatted and protected from changing, and dates can be converted to and from Julian. You can use a number of financial and mathematical functions, include If commands for conditionally copying cells, and plot charts and graphs. While Works is easy to learn, and may not include all of the features of a power-house like Lotus, it is nonetheless a powerful spreadsheet in its own right.

The Data Base

Without exiting, or closing, the spreadsheet application, let's now create a small data base of executives to receive the letter we want to type that will include the spreadsheet. This will show you how easy it is to move between applications without having to save and exit each one.

1. Press Alt, F, then N, for new. The menu of major applications appears just as it did when you first started Works.

2. Press the down arrow twice to reach the data base option, then press Return. The forms design screen appears. You can design a data base in two ways — either in rows and columns, like a spreadsheet, or in a form. Let's create the data base in the form mode, then see how it looks as a spreadsheet.

3. Type Last Name:, then press Return. The colon tells Works that this is a field name.

4. Press the down arrow key to reach the next line on the screen.

5. Type First Name:, then press the down arrow key, just as you could after entering data into a spreadsheet cell. After you press the arrow key (or Return) in each case, the field name and a line appear in the form itself, again just like cell entries in a spreadsheet.

6. Now in the same manner, enter the next field names. Remember to type each name, followed by a colon. Then press the down arrow key (or Return and then the arrow) to move to the next line.

Address:

City:

State:

Zip:

Department:

In each case a line ten characters long appeared after the field name (Figure 4). That might be long enough for the state and first name, but not for other fields. So let's format the fields for the type and size of data we plan to enter. However, since you're already familiar with the Work's spreadsheet, we'll finish the data base in the list mode.

Figure 4

7. Press F10 to leave the form design mode and display the data entry form for the data base you just created. While you could add data here, let's display the data base in the list mode.

8. Press Alt, O, for options, then V for view list. The data base is now displayed very much like a spreadsheet, except each column is headed by the field name, not a letter. But before entering the data, let's widen several of the columns to make room for our information.

9. Press Home to reach the first field (or cell) in the first row.

10. Press Alt, t (for format), then w (width). Type 15, then press Return. The last name column is now 15 characters wide.

11. In the same manner, widen the First Name column to 15, the address column to 20, and the city column to 15. This widens the column only in the list mode — you'd have to repeat the

same procedure in the form mode to display the full width of the entries.

It's now time to enter data in the data base. Just as you did in the spreadsheet, use the arrow keys to activate the field (the equivalent to the spreadsheet cell), then type the entry.

12. Enter the following data into the first two rows.

Chesin
Adam
124 Lock Road
Philadelphia
PA
19116
MIS

Williams
Paul
62 Walnut Street
Camden
NJ
01888
Finance

Your screen should look like Figure 5.



Figure 5

Just like the Works spreadsheet, there's a great deal that you can do with the data base. You can design reports and queries, format fields and individual entries; search, sort, and edit data base records. Fields can be formatted as dollar, fixed decimal, percent, logical (true and false), and time or date. You can even create calculated fields. For instance, the field entry `=Sales*1.06` will create a field that's six percent greater than whatever is in the Sales field.

But for now, let's save the data base and move on to our form letter.

13. Press Alt, F, then S and Return to store the data base using the default name DATA1.WDB.

The Word Processor

Without really exiting the spreadsheet or the data base, we'll now create a form letter to send to those on the mailing list.

1. Press Alt, F, then N, for new. The menu of major applications appears just as it did when you first started Works.
2. Press Return to accept the selected option, Word Processor. The word processing window now appears, as in Figure 6.

As with the other applications, the word processor is a full and robust



Figure 6

program. But our aim here is to get you started and to see how easy it is to work with more than one application at a time, sharing information between them.

So start out by typing the form letter. Use the arrow keys, backspace, and delete to correct any errors you make.

3. Press Ctrl-C (center) to center the cursor on the screen.
4. Type your inside address and the date. Press Return an extra time after the date.
5. Press Ctrl-L (left) to return the cursor to the left margin.

We would like each of the persons in the data base to receive the letter, so we want their first and last names on this line.

7. Press Alt, then E for the edit pull down menu.
8. Press F for insert field. The insert field menu appears showing the name of the active data base, DATA1.WDB. (Figure 7)

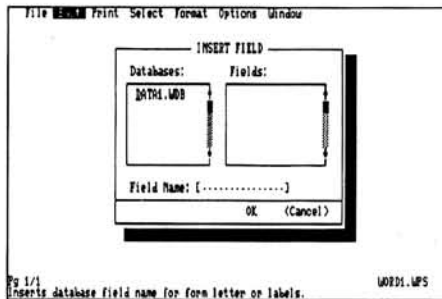


Figure 7

9. Press the right arrow key to select that data base and display its fields in the right-hand box.
10. Press Tab to place the cursor in the box, then the down arrow key twice to select the first name field.
11. Press Return. The line now looks like this
First Name
12. Press the spacebar, then select the last name field.
 - a. Press Alt E F to display the insert field menu.
 - b. Press Tab to reach the field list.
 - c. Press the down arrow to select the last name field.
 - d. Press Return. The line now appears
First Name Last Name

13. Complete the next section of the letter by inserting their address, city, state, and zip code. It will appear like this:

First Name Last Name
Address
City, State Zip

14. Now type the salutation using their first name:

Dear First Name:

15. Type the first words of the paragraph: Below is the

16. Press Ctrl-B to turn on the boldface format.

17. Type projected.

18. Now press Ctrl-spacebar. This turns off the boldfaced mode.

19. Complete the paragraph:

Below is the projected budget for the next fiscal year. Please review it at your earliest convenience.

20. Press Return twice after the paragraph.

You now want to include the data from the spreadsheet in the document. So let's return to the spreadsheet window and select the rows to be copied.

21. Press Alt W (windows). Notice that the three currently open documents are listed at the bottom of the window in the order they were opened:

- 1 SHEET1.WKS
- 2 DATA1.WDB
- 3 WORD1.WPS

22. Press 1 to display the spreadsheet.

Remember that the column headings do not line up properly? That's because labels start at the left side of the column, numbers at the right. Let's format the labels correctly now.

23. Activate cell B1.

24. Press down the shift key and press the right arrow three times to select, or highlight, the four labels.

25. Press Alt, t (for format), then S, for style. The style menu appears as in Figure 8.

26. Press the down arrow key three times to select right alignment, then press Return. The labels now appear directly over the numbers in the column.

27. Save the spreadsheet before going on — press Alt, F, S, Return.

Okay, now lets copy the spreadsheet into the document.

28. Place the cursor in cell A1.

29. Hold down the shift key, then move the cursor to cell E10, selecting the entire spreadsheet.

30. Press Alt, E, then C to copy the data. Now return to the document.

31. Press Alt W 3 to return to the third application opened, the word processor.

32. Press Return. The text of the spreadsheet is inserted into the document.

33. Complete the letter.

Sincerely Yours,
Alvin A. Aardvark

Continued on Page 46

Do You Really Need a More Powerful Computer? Options, Upgrades, and Alternatives

William M. Adney
P.O. Box 531655

Grand Prairie, TX 75053-1655

Copyright © 1988 by William M. Adney. All rights reserved.

Given the rapid advances in computer technology, all of us consider buying a new, faster, and more powerful computer from time to time. How seriously we consider that is usually dependent on whether or not we can afford it, but there are other far more important considerations that will also determine how satisfied you are with a new computer.

For discussion's sake, I will choose the old, reliable Z-151 computer. And we'll take a look at some of the pros and cons of upgrading to a newer system, such as the 12 MHz 80286-based HS-2526 or the 16 MHz 80386-based HS-386. Although these newer models are shown with the Heathkit numbers, this also applies to all of the Zenith computers as well. For those of you not familiar with the Heath and Zenith model numbers, Heath computer models are usually preceded with an "H" indicating they are a Heathkit that you build. Zenith computer models are fully assembled and are usually preceded with a "Z." For the most part, I usually refer to the same model computer with its "Z" part number, even though it may be a kit. Although I have personally built all of my computers from Heathkits (except for the Z-171 which was not available as a kit), I still refer to them as the Z-100, Z-248 (originally a Z-241), or Z-386. Even though I do not have a Z-151, I built one for a friend so that I would know something about its internals.

Although this discussion will center on the question of getting a newer computer to replace the '151, it applies to any of the Heath or Zenith 8088-based compatibles, such as the Z-138 or Z-148, as well as the entire Z-150 series of computers. Some of the 8088-based computers, such as the Z-148 and Z-158, also have

the dual-speed capability which allows you to switch computer speeds between 4.77 MHz and 8.0 MHz. Although multi-speed computers are quite common today, I suspect that most of you who have these dual-speed computers probably switch them to the higher speed and leave them there unless you have some of the older software (especially games) which uses a speed-sensitive copy protection scheme that requires the 4.77 MHz speed. Which brings up the first question: Is speed really that important?

Myths About Computer Speed

There are so many mistaken ideas about the importance of the speed of a computer that it would take far more than a single article to explode all of these wrong notions. For example, I remember reading an article some time ago (I think it was in BYTE) which compared a 16 MHz Z-386 computer with a similar 20 MHz 80386 Compaq computer. The test results were interesting because they showed that the Z-386 PROCESSING speed was just a hair less than the Compaq's. In other words, these tests were specifically designed to see how fast a computer could really PROCESS data as a unit, rather than use the simple-minded approach that a faster clock speed on the Compaq meant that it was a faster computer.

One of the major reasons the Z-386 is so fast is that it uses zero-wait-state memory, and this particular Compaq model used memory with one wait state. Although nearly all of the computers available today use the zero-wait-state technology, that was not previously true. The IBM AT (both models), for example, used wait states in memory, some of the memory boards that worked fine in an AT would

not work in a similar Zenith computer (e.g., Z-248) because of the zero-wait-states. Since all Zenith computers I can think of have used zero-wait-state technology from the beginning, it really becomes a moot point for us. And lest the uninitiated think they don't need to know this kind of computer gobbledy-gook, I think you will find these are many of the same people who make poor computer-related decisions and end up being unhappy with their computer, whatever the brand. If you are curious about some other aspects of adding memory to your computer, you will find a more detailed discussion of it (including wait states) in the Powering Up article on memory that was originally published in March 1989 REMark or in the book available from HUG.

But is clock speed really that important to most users? The answer is, as usual, that it depends. In the Z-151 (or the Z-100 for that matter), you will find that some kind of a "turbo" kit which increases clock speed to something around 8 MHz can make a difference you can see on the screen. But if you really want to keep an older computer, like the Z-151, I recommend a hard drive as the best single and least expensive way to improve overall system performance. In fact, most of the older 8088-based systems will demonstrate much better performance with a hard drive to the point that you may not really need a newer computer.

Although the actual performance difference between two computers with different clock speeds can be measured fairly accurately, the other factor that really affects performance is the type of microprocessor unit (MPU) that a system uses: an 8088, an 80286 or an 80386. All other

things being equal (e.g., memory wait states and MPU), it is obvious that a "faster" computer will do things faster than a "slower" one. Since 80386-based computers are the current latest and greatest systems, let's consider clock speeds on them to keep at least one parameter the same.

Even though the Z-386 is a fine computer, you can find 80386 computers with clock speeds up to 25 MHz, and if you keep up with the industry, you know that 33 MHz systems are not too far away. Part of the availability of these faster systems depends on how soon Intel can provide these faster MPUs in production quantities for hardware manufacturers, and I predict that you will see new Zenith computers with clock speeds at least in the 25 MHz range this year. One would have to be unconscious or dead (I like to think I am neither!) to think that Zenith will not keep up with the technology and the competition. And when Intel finally makes the 80486 MPU available, I am also confident that Zenith will design a computer for that, too.

In short, I would not upgrade to a newer computer based simply on clock speed. Although the performance of the 25 MHz 80386 systems may be impressive, most users don't need it if they already have a system with that same MPU, such as a Z-386. Unless you are doing some real MPU-intensive work, such as graphics or lots of calculations, the faster speed may only give you a visible performance difference of a few milliseconds, if you can see it. Most people can't. I suppose that it goes without saying that it is ridiculous to buy one of these high-powered computers without buying a hard disk, too. If you use floppy disk technology with one of these systems, in mainframe systems we would say that you are "I/O bound" because you are limited by the input and output speed of the floppy disk.

Clock Speed and Applications

Most home and business computer users run applications, such as word processing, that really don't require a fast MPU. Although both my '248 and '386 systems have hard drives, I have not really noticed any real visible difference in how fast I can run Microsoft Word or WordStar. I can see some minor differences in how fast a spreadsheet (e.g., SuperCalc or Quattro) recalculates, and how fast a graphics program runs (e.g., GEM Draw or Generic CADD), but that difference is too small to make me want to upgrade from a '248 to a '386 for only that reason. Depending on exactly what I am doing, the exact difference is so small that it would be difficult for me to justify buying a '386 on that basis alone.

It did not surprise me to see the considerable difference in processing speed

between the two systems for programming applications. Some programs I have written in GW-BASIC, for example, run MUCH faster in the interpreted mode on the '386 than the '248. Program compiles also run much faster, even though there is little apparent difference in their execution speed on either system. For one long C program that I wrote using the Ecosoft compiler, there is a difference of several minutes in the compilation time between the Z-100 and the Z-386 which is also not surprising. If you do much programming, you will probably find that a computer with a newer and faster MPU can make a significant difference in your productivity.

For a business that requires a large data base application (e.g., dBASE), a faster system can also reduce the processing time assuming that both systems have a hard drive. I noted the hard drive specifically here because I think it is foolish (and sometimes impossible) to run a major data base application on a floppy disk system.

Most of the discussion to this point has considered many of the common applications that microcomputers are used for. There is at least one other consideration you should be aware of.

New Operating Systems and Applications

If you expect to use a new operating system, such as OS/2, then you will need a computer that has at least an 80286 MPU in order to run it. And if you expect to use OS/2 to its full potential, you will also have to get new application software, too.

Some newer applications, such as Windows/386, even require at least an 80386 MPU. And some of the very newest programs, such as Samna Corporation's Ami word processor, have very specific hardware requirements. Ami requires an 80286 MPU and a Hercules, EGA or VGA video card. Ami is a Windows-based application that requires about 2.1 MB on my hard disk with the runtime version of Windows that came with the package. And because Windows is notoriously slow, Ami is considerably slower on my '248 than it is on my '386. If you contemplate using any kind of Windows, including OS/2's Presentation Manager, I strongly recommend getting at least an 80386 system. The reason I am using Ami as an example here is because you will begin to see more and more software that is quite hardware specific. And even though Ami's hardware requirements are clearly listed so they can be easily seen on the package, you must KNOW what kind of hardware your system has so that you can correctly determine whether or not you can use Ami on your system. And if you know that your Z-151, Z-158, or Z-148 has an 8088 MPU, it is easy to recognize the fact that Ami

will NOT run on your system.

Lest you have any doubt, the days are nearly gone when you could buy just about any PC compatible program with a high degree of confidence that it would run on a computer like the Z-151. Some of the current applications, like desktop publishing, simply won't run on a standard '151 because at least an EGA video card is required. Some current applications require at least 512 KB of conventional memory, and a few require 640 K. Memory requirements of some applications may even prevent your using some of your favorite TSR programs because there simply is not enough system memory available for DOS, that application, and any memory-resident program. And some new applications, even require that you have at least an 80286 MPU in your system just to run the program.

As these new applications become available, you will find that my discussions of them will also include their hardware-specific requirements. I have not normally mentioned hardware before because most programs I have described in my recent columns will run on any Zenith PC compatible computer up to and including the Z-386. Unless I find some kind of real quirk in the software that does something strange on one of my systems, I will only mention the basic manufacturer's published system requirements, such as an 80286, EGA, hard disk, extended memory or whatever. Although I will still mention Heath and Zenith computer model numbers, I will assume that you know enough about your computer hardware and applications to correctly determine whether or not a program I discuss will run on your specific system.

Perhaps the most important reason for only listing the manufacturer's "generic" requirements is that it is not possible to make the statement that a certain program will run just fine on a Z-248 computer, unless it is absolutely identical in all respects to mine. You might, for example, have one of the original CGA video cards (the Z-409) installed that is not capable of EGA. My '248 has a Video 7 Vega EGA card which I am using with the original NEC MultiSync monitor. As I write this, my Z-386 has a Z-449 video card with a Zenith FTM monitor.

Although many of my articles will continue to discuss some of the unique Zenith-specific hardware (e.g., the FTM monitor) and software (e.g., Zenith MS-DOS and OS/2) for these systems, I will not attempt to identify all of the specific model numbers for general software that I discuss. That would obviously be a hopeless task.

The real point of this current discussion is to illustrate how important a detailed knowledge of your system is. It is also important to know what kinds of things are being developed and what you

will need to use them. It is this perspective that can help you decide whether you really need a newer and faster computer or whether you can upgrade your current one to do what you want.

New Versus Upgrade

If money is not a problem for you, then you will probably buy the latest and greatest computer available. Unfortunately, most of us have more modest means, and we can't go out and simply spend a few thousand dollars because we want or think we "need" a new system. But before we get too involved with some of the options and alternatives for upgrading an older system, there is one other important point to consider.

If you really need a computer with a more powerful MPU, don't fool around and try to upgrade a Z-151 with a board that provides 80286 support — you will find it more cost effective in the long run to simply get a newer computer. One of the more pointed examples of this is that the original Microsoft 80286 upgrade board, called the Mach 20, reportedly would not run the OS/2 operating system. I do not like or recommend this kind of upgrade simply because it usually causes all kinds of strange problems that are impossible for most users to fix. And regardless of the quality and manufacturer, it is absolutely impossible for any third-party manufacturer to guarantee total hardware and software compatibility for an upgrade that uses hardware and features not originally included in the basic design specification for that system, such as the Z-151. For example, I do not recommend speed-up kits or video card eliminator kits for the Z-151 because I have received letters from many of you who have had software compatibility problems with one or both of them.

My response to most of these letters has been to explain how to identify where the problem is, but unfortunately, there is little that can be done to "fix" it. In general, most users find there is only one program which does not work with their specific upgrade(s), so the answer is to suggest not using the problem program. Although that kind of apparent glib answer irritates most people and does not solve the problem, it is actually the only answer because of some incompatibility in the add-on or software. Even worse is the fact that many people are only now finding some of these problems because new kinds of programming "tricks" and compilers are being used.

For those of you who have been HUG members for a few years, you may remember my July 1986 REMark column in which I said: "If you are planning to do a lot of heavy-duty work with PC software, there is one clear choice . . ." Don't upgrade a Z-100. This was the third of three articles in which I discussed three ways to

add PC compatibility to the Z-100: the Gemini Board, the EasyPC, and HUG's ZPC software. In this article, I also said: "The point is that, if you really want to get into the PC compatible software, you really should have a PC compatible." And I think that general recommendation can be slightly modified to include the question of upgrading to a newer system.

If you need an 80286 or 80386 system to run specific software, buy it. Don't fool around with trying to upgrade bits and pieces of an older system, like the Z-151, because you will inevitably encounter some kind of compatibility problem unless you try one of the alternatives mentioned later in this article. Although it may initially appear to be more cost effective to attempt to "salvage" the investment by adding a relatively inexpensive upgrade, you may eventually find that some of the newest software will not run on one of these upgraded systems because of some obscure incompatibility. In short, I think it is wiser to go ahead and bite the bullet if you really need some of the advanced features and speed of the newer systems. Now let's move on to a discussion of other options and alternatives that you have for upgrading your current computer.

Cost-Effective Upgrade Options

Regardless of how old your computer is, I think you will find the best and least expensive way to improve system performance is to add a hard disk, especially if you have a lot of programs and many data files. You might be surprised at how much faster a 4.77 MHz Z-151 computer will appear to be with a hard disk. When you consider that a standard DS/DD 5.25" floppy disk spins at 300 RPM, and a standard hard disk spins at 3,600 RPM, it is easy to see how adding a hard disk can improve system performance. Of course, it will not make your system 12 times as fast, but it will at least double the apparent speed because programs can be loaded faster, and files can be read and written faster.

Aside from a hard disk, the best thing I ever did was buy a color monitor for my system. I started out with a CGA compatible C. Itoh CM-1000 CRT that I added to my Z-100, and I later used that on my '248. I still have the old Zenith ZVM-122 amber monochrome CRT that I keep attached to my Z-100 because I don't use it much now. Display clarity and resolution is important to me because I do so much writing, and I decided to upgrade my '248 to EGA. I bought a Video 7 VEGA Deluxe EGA card and the original NEC MultiSync monitor, and have been extremely pleased with the results. For my '386, I bought a ZCM-1490 FTM monitor which has an even better display. My only real complaint about my '386 system, as it stands today, is that it is noisy. While that

probably would not be very noticeable in a normal office environment, it is quite obvious in my study. When I power up the system, the two fans in the '386 make a little noise, and the hard disk winds up to operating speed adding to it. And of course, the cooling fan in the FTM fires up, too. Three fans and a hard disk really make more noise in my usually quiet study than I was used to with my '248, and it took me a while to get accustomed to it. I guess that is one of the hazards of progress, and the Z-386 with the FTM monitor is definitely worth it.

For the Z-151, there really aren't too many other upgrades that I feel comfortable in recommending, primarily because I have received too many letters from HUG members about various software incompatibility problems. This includes speed-up kits, video card eliminators, and 80286 CPU add-on cards. Because of the original design of the Z-151, I do not recommend most of the other add-ons that you may find, with the exception of expanded memory that you may find helpful with software that can use it. Although most of these add-ons are quality units made by reputable manufacturers, adding one or more of them will probably result in your finding some kind of compatibility problem eventually with hardware, software, or both.

Upgrades for a Z-150 Series Computer

Although I have used the Z-151 as an example because it is the oldest PC compatible made by Zenith, the discussion applies to all of the 8088-based systems made by Zenith and Heath, including the Z-138, '148, '151/2/7/8/9, and '161. If you decide to add one of these upgrades, just keep in mind that you may be sacrificing either current or future compatibility with one or more hardware or software products. But there are some other alternatives.

Depending on exactly which computer model you have, you still may be able to upgrade it to a newer and faster configuration at a reasonable price. Moreover, these alternatives provide a COMPLETE conversion, rather than just some kind of a kludge to improve performance, and I believe you are far less likely to find compatibility problems with this approach. And although I have not personally tested any of these options, I have included them to make this discussion complete so you will be able to check them out and judge for yourself.

If you have a Z-151/2/8/9 or Z-161, First Capitol Computer has two upgrades that you may be interested in. The first one — FCC-150/286-UP — will upgrade your current system to an 80286 AT compatible system. The second option — FCC-150/386-UP — will upgrade your system to an 80386 configuration. If you are not familiar with First Capitol Computer,

they are one of the best and most knowledgeable vendors of Heath/Zenith computers and products, and I have no hesitation in recommending these products based on First Capitol Computer's excellent reputation.

Upgrades for a Z-200 Series Computer

If you have a Z-200 system (i.e., a '241 or '248), you have three choices if you want to upgrade to an 80386 system: the HUG-386 or HUG-386-C (includes a dual floppy/hard disk controller) upgrade, First Capitol Computer's FCC-248/386-UP upgrade or AMI's ZX-386 upgrade.

The HUG upgrade provides all of the boards necessary to upgrade your Z-200 system to a basic Z-386 configuration which includes 1 MB of memory. The basic kit (HUG-386) includes a new 32-bit backplane board, a new 80386 CPU board, and a new I/O board. If you are converting a Z-241, you MUST order the HUG-386-C kit that includes a dual floppy/hard disk controller because the old '241 controller is not compatible with the Z-386 hardware. If you are converting a Z-248 that already has a dual floppy/hard disk controller, you can simply order the basic HUG-386 upgrade since that controller is compatible with the Z-386. And since some '248s were sold with a floppy-only controller, you MAY want to order the HUG upgrade that includes the controller (HUG-386-C) so that you can easily add a hard disk to the system.

I think the HUG upgrade has two major advantages. First, the upgrade includes all Zenith boards which means that your system is still a Heath/Zenith computer. And of course, the second advantage is that the upgrade is completely compatible with all software and hardware that works on the Z-386. A possible disadvantage of this approach is that you must disassemble your current system and install the upgrade yourself, but that is only a matter of using a screwdriver, exchanging some boards, and connecting all the plugs. In order to preserve your sanity, be absolutely, positively, and completely sure that you backup a hard disk (if any) BEFORE you begin the conversion, just in case. If you also change the disk controller, you are almost certain to need a hard disk backup since changing controllers frequently means that the new controller cannot read the hard disk because something in the low-level format (e.g., the interleave byte) has changed. Besides, when you do change hard disk controllers, it is always good practice to rerun a low-level format program (e.g., PREP) to ensure reliability with the new controller. In that case, you MUST have a hard disk backup because virtually all low-level format programs effectively destroy everything on a hard disk.

The second upgrade alternative for the Z-200 series is First Capitol Comput-

er's FCC-248/386-UP upgrade that changes a '241 or '248 computer to an 80386 system. This is a particularly good choice if you don't want to disassemble your system and change the boards — First Capitol will do it for you and test the system. Of course, this approach requires that you send your system unit to First Capitol, and a disadvantage is that you will be without your system for a short time. If you decide to use this approach, be sure to call First Capitol Computer to check it out and make arrangements for the upgrade.

The third alternative is to add the AMI ZX-386 upgrade to your Z-200 system that was discussed in Pat Swayne's article "A Review of the AMI ZX-386 Upgrade" that appeared in the January 1989 issue of REMark (page 23). Since Pat did a lot of detailed testing and discusses the results in this article, I recommend you read it carefully before making any decisions.

The advantage of changing out all of the "old" hardware in a Z-150 or Z-200 series computer gives you the advantage of having a high degree of confidence in the new hardware and its compatibility with your current software. If you are fortunate enough to be able to afford a completely new system, you still can have that same, or increased, level of confidence. Still, you must keep in mind that, whatever approach you choose, you are working with what amounts to a completely new system. Because of the new hardware, it will probably not work exactly like the "old" one (e.g., a Z-150 or Z-200), and there are some changes and considerations that you must know about.

Upgrade Considerations and Operation

Whether you are working with an upgraded or completely new system, be absolutely sure that you test all your software thoroughly before you begin any kind of important work. More importantly, do not make software changes of any kind (including AUTOEXEC.BAT and drivers for CONFIG.SYS files) until you are satisfied that you know how to operate the new system. If you are only upgrading one hardware level to a newer system (e.g., 80286 to 80386), you may not find too many changes, but if you are skipping several levels (i.e., 8088 to 80386), the differences may be quite striking.

To illustrate how significant these differences can be, let's consider the difference between a Z-150 and a Z-386. The Z-150's 8088 can address a maximum of one megabyte (1,024 KB) of memory, and the Z-386's 80386 can address up to four gigabytes (1024 MB) of memory. Let's work out how much difference that really is in bytes, although it should be clear that the 80386 can address 4,096 times as much memory as an 8088.

A kilobyte (KB) is 1,024 bytes; and a

megabyte (MB) is 1,024 KB or 1,048,576 bytes. A gigabyte (GB) is 1,024 MB or 1,048,576 KB or 1,073,741,824 bytes. Note that the order of magnitude of each of these is 1,024 (not 1,000) times the previous value. In other words, a gigabyte is 1,024 times as large as a megabyte, and since the 80386 can address up to four gigabytes, that is 4,096 times as much memory as an 8088. In pure numbers of bytes, the 8088 can address 1,048,576 bytes, and the 80386 can address 4,294,967,296 bytes or 4,096 times as much. In words, we can say that the 8088 can address a million bytes, but an 80386 can address 4 BILLION bytes which is probably just a little larger than you will ever need, at least for main memory. This is, by the way, only one of the more important technical differences in these chip. There is one other interesting point about these two chips, and that is the way they can "talk to" memory.

Because of the memory address limitations in the 8088 chip, DOS runs in what we can call "real" memory which includes up to 640 KB of conventional or system memory. Without getting into all of the technical garbage, suffice it to say that the 80386 chip essentially can operate in two memory modes: the REAL mode (for DOS and its applications) and the PROTECTED mode (for OS/2 and its applications). The point is that the 80386 talks to and manages memory differently than the 8088, and there are bound to be some specific differences in what software will work on an 80386. Although this difference may have little or no impact in running DOS and its applications on an 80386, it will definitely make a difference as to what runs under OS/2 and what does not. For now, however, let's stick with simply running DOS applications on an 80386, specifically a Z-386.

Starting a New System

When you bring up a new system or upgraded one for the first time, be sure that you check out EVERYTHING before you begin any production work. If you have the Zenith Disk Diagnostics for the system, run them to check out the hardware. Take a look at your AUTOEXEC.BAT and CONFIG.SYS files that you probably copied from a previous system. For the AUTOEXEC.BAT file, go through it and precede each TSR program with a REM command to disable that program. For the CONFIG.SYS file, you will need to delete all of the DEVICE= commands unless one is absolutely required to even use the system, such as a special driver for a video card. In most cases, the beginning CONFIG.SYS file should contain nothing more than FILES= and BUFFERS= commands. Above all, do not implement any TSR or device driver that uses any of the advanced features of your new system — it may cause unexpected problems that

will be difficult to troubleshoot and pinpoint the cause. Take the time to make sure that the previous AUTOEXEC.BAT and CONFIG.SYS files work from your previous system before trying anything new. You may find that something, like a device driver, will not work the same way as it did in your old system. And the bigger jump you make in terms of the MPU (e.g., 8088 to 80386), the more possibilities you are likely to find.

For example, most 80386 upgrades, including the ones previously mentioned, include a standard 1 MB memory which is treated a little differently than memory in previous systems. For the Z-386 system, you want to be sure that you read the Zenith 80386-Based Desktop Computer Owner's Manual to understand the set up, verify, and use this memory. For example, you need to check out the switch SW401 on the first memory card to verify that all switches are in the "ON" position, as shown on pages 4-28 and 4-29 of the Owner's Manual. In fact, you should carefully read all of Chapter 4 of the Owner's Manual and double-check all of the switch settings on all boards, especially if you are having any kind of problem.

I have found it to be extremely helpful to make notes of all the recommended switch and jumper settings as I read, and I make special notes as to why I set each switch in a certain position. On switch 401, for example, positions 0 and 1 are set to ON to indicate the first or base memory card in the system. Position 2 is set to ON to enable expanded memory (EMS), and position 3 is set to ON to set the base system memory to 640 K. Positions 4 through 7 are set to ON because this is the first memory board in the system that begins on the "0 megabyte boundary."

After my list is complete, I then go through each board to make absolutely certain that each switch is in the correct position specified in my notes. In this case, I have found that it is not enough just to eyeball each switch — I actually push each switch to make sure it is making proper contact. If a board gets knocked around in shipping, a switch may LOOK like it is making contact when it really is not. If I have any doubt, I move the switch to make sure it "feels" like it is making proper contact. After I check each board for correct jumper and pin settings, I also press down on all socketed chips to make sure all pins are firmly seated in the socket and making good contact. The final step is to eyeball all of the pins for all of the socketed chips to make sure that no pin is bent underneath a chip — even mechanical chip inserters can make mistakes, too. You might be amazed at how many parity errors and other strange diagnostic problems can be fixed by a simple chip massage! I have fixed perhaps half a dozen computers with parity errors in the

last few years simply by removing each board and giving it a chip massage. Be firm, but gentle with your computer. You may find that a massage will make strange problems go away as quickly as they appeared. Or maybe a board simply needs a little help to get its electronic circulation going again . . .

Although I presented the idea of chip massage in a somewhat whimsical vein (I like puns, too!), I recommend it in cases where you find strange or intermittent problems that seem to have no apparent cause. In particular, intermittent problems can be caused by a very slight vibration, such as a floppy disk drive, so that the pins on a socketed chip sometimes make proper contact and sometimes they don't.

To return to the Z-386 now, you will also find that page 4-26 of the Owner's Manual generally shows how the standard 1 MB memory is used by this computer. The manual notes that 128 KB are reserved for both the System ROM and Optional ROM code, and up to 640 KB is allocated as System RAM. Although the manual clearly states that the "balance can be used as EMS RAM", I am uncomfortable with that because this memory space is reserved in a standard PC compatible for the various kinds of video memory and other things. I have not fooled around with that yet until I find out more about it. When I do, I will write about it, too. Until then, I suggest you be very cautious about using this memory for EMS (with the Zenith EMM.SYS driver), especially if you are upgrading from another system.

As you can see, I am cautious about recommending or trying things when I do not have the sufficient knowledge to predict what might happen. In all likelihood, everything would work fine, but I like to have an idea of what SHOULD happen before I try it.

Powering Down

In the next article, we will take a look at the new Zenith MS-DOS 3.3+, with some information that you MUST know about. In particular, we will examine why older versions of the Mace and Norton Utilities do not work with this MS-DOS version. Compatibility is not the issue here because these programs (and others) also do not work with current IBM PC-DOS releases either.

For help in solving specific computer problems, be sure to include the exact model number of your system (from the back of the unit), the ROM version you are using (use CTRL-ALT-INS to find it), the DOS version you are using (including both version and BIOS numbers from the VER command), and a list of ALL hardware add-ons (including brand and model number) installed in your computer. The list of hardware add-ons should specifical-

ly include memory capacity (either added to an existing board or on any add-on board), all other internal add-on boards (e.g., modems, bus mouse or video cards), the brand and model of the CRT monitor you have, and the brand and model of the printer, with the type of interface (i.e., serial or printer) you are using. Also be sure to include a listing of the contents of the AUTOEXEC.BAT and CONFIG.SYS files, unless you have thoroughly checked them out for potential problems (e.g., TSR conflicts). If the problem involves any application software, be sure to include the name and version number of the program you are running when the problem appears.

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

Products Mentioned

HUG Z-241/248 to Z-386 Upgrades
 HUG-386 (no controller) \$2999.00
 HUG-386-C (w/dual controller) 3149.00
Hug members receive a \$1200 discount off these prices. -ed.

Heath/Zenith Computer Centers
 Heath/Zenith Users' Group
 P.O. Box 217
 Benton Harbor, MI 49022-0217
 (616) 982-3463 (HUG Software only)

HS-386 80386 Computer Kit	\$3349.00
1 MB Memory Board for Z-386 (Z-505)	799.00
4 MB Memory Board for Z-386 (Z-515)	2999.00
Cache Memory Board for Z-386 (Z-525)	599.00
HS-2526 12 MHz 80286 Computer Kit	2199.00
FTM Monitor (ZCM-1490)	999.00

Heath/Zenith Computer Centers
 Heath Company Parts Department
 Hilltop Road
 St. Joseph, MI 49085
 (800) 253-7057
 (Heath Catalog orders only)

Z-150 Series Computer Upgrades	
To Z-286 80286 system (FCC-150/286-UP)	\$1295.00
To Z-386 80386 system (FCC-150/386-UP)	2495.00
Z-241/248 Series Computer Upgrade	
To Z-386 80386 system (FCC-248/386-UP)	1995.00

First Capitol Computer
 #16 Algona Drive
 St. Peters, MO 63376
 (314) 447-8697 Technical Information
 (800) 862-8948 (TO-BUY-IT) Orders

Z-248 Upgrade to 80386 configuration ZX-386 (20 MHz) with 1 MB	\$1945.00
--	-----------

ZX-386 (20 MHz) with 2 MB \$2495.00
 ZX-386 (20 MHz) with 4 MB \$3595.00
 ZX-386 (20 MHz) with 8 MB \$5795.00
 SMS Data Products Group, Inc.
 1501 Farm Credit Drive
 McLean, VA 22102
 (703) 827-0640 *

Continued from Page 36

CCFORMAT.COM — This utility must be used to format the card's drives. An option switch determines which format will be used. If the option argument is omitted, the disk will be formatted to the highest capacity supported. In my case, **CCFORMAT G:/1.4** or just **CCFORMAT G:** will format a 1.44 Meg disk. To format a 720K disk, **CCFORMAT G:/720** is entered.

INT13.DOC — This text file is for programmers needing to access sectors and the like. It provides the Assembly-level BIOS-type Interrupt 13 handling routines used by the card.

Almost the entire latter half of the **CompatiCard** manual is devoted to specifications, including detailed sections on Cabling and Adapters, Special Drive Types, I/O Ports, Control Registers, Troubleshooting, and other technical data. Micro Solutions has crammed a heck of a lot of clearly-stated information into a nicely-printed 34-page booklet.

Studio Computers Package (And Kudos)

First, permit me a minute to applaud ZDS dealer Ray Massa. My first business with Studio Computers (then called Keyboard Studio — didja get a lot of calls for Wurlitzers, Ray?) was when I mail-ordered his CP/M Adventure Maker software for my new H-89 in 1982. Since moving to the Detroit Metro area, I've bought an MPI printer, the 20 Meg Winchester kit, a VGA board and multiscanning monitor, and this 3.5" subsystem from him. He's supported the '84 printer with upgrades, software, and the occasional "bring it in and we'll fix it" visit for this long, mostly for free — and after MPI has gone under or otherwise disappeared. Ray treats his customers like businesses used to. I'm sure few, if any, of those customers ever left ticked off about an unresolved problem. In a climate where many FranchiseLand computer sales people are not-too-distant kin to used car salesmen, Studio Computers is an oasis.

Since the 3.5" installation was a success, Studio Computers will offer a package consisting of the Micro Solutions **CompatiCard II** and software, Toshiba **FDD4603 3.5" 1.44 Meg drive**, and a ribbon cable for \$315.00. He also carries the external drive housings. Contact Ray Massa at (313) 645-5365 for details.

Sources Mentioned

3.5" Drive System Pkg. \$315.00
 Studio Computers, Inc.
 999 South Adams

Birmingham, MI 48009
 (313) 645-5365

CompatiCard II \$125.00
CompatiCard 175.00
Megamate 349.00
 Micro Solutions, Inc.
 132 West Lincoln Highway
 DeKalb, IL 60115
 (815) 756-3411

Omni-Bridge Controller \$ 95.00
 Sysgen, Inc.
 556 Gibraltar Drive
 Milpitas, CA 95035
 (800) 821-2151

FastBack Plus ver. 2.0A \$179.00
 Fifth Generation Systems, Inc. (list)
 11200 Industriplex Boulevard Around
 Baton Rouge, LA 70809 \$100.00
 (800) 873-4384 mail-order *

Continued from Page 40

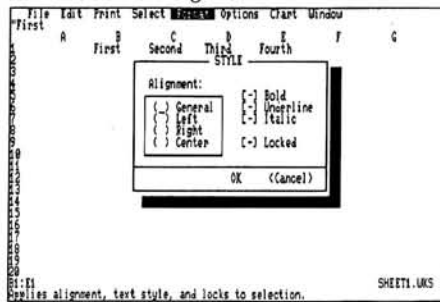


Figure 8

You're now ready to print the form letter. The data base file is already open, so you can create the form letters right now. But let's just see how easy it is again to move from application to application.

34. Press Alt W 2 to display the data base window. Confirm that all of the data is correct.
35. Press Alt W 3 to display the word processor.
36. Press Alt P M (for Print Merge). The print merge menu appears with the name of the active data base already selected.
37. Press Return to use that data base. The Print menu appears.
38. Press Return to print the form letters.

Using the Menu Bar

Before leaving the word processor, let's just see the options on the menu bar.

1. Press Alt to select the menu bar.
2. Press Return to pull down the file menu.
3. Press the right arrow key to pull down and display the other menus.

You'll see options to move and copy text; format characters, paragraphs, and pages; use the spelling checker; and enter headers and footers — all of the usual features you expect from a word processing program.

Now let's save the letter, then exit Works. The menu bar should still be highlighted.

4. Press the arrow key to pull down the file menu.
5. Press S, then Return to accept the default document name, **WORD1.WPS**.
6. Press Alt, F, then X to exit Works.

Works offers an alternative to the costly move to OS/2 or Windows. While the applications only runs one at a time (not true multi-tasking), you can easily switch between multiple applications by using the window pull-down menu. The flexibility that Works provides actually gives you many of the benefits of power-user systems without the cost or complications. *



ACCOUNTING & TAX

Not sure if you need the expensive 'Chinese Flower 1-2-3', or 'Spanish Numeral Four' spreadsheet programs? Then find out for only \$20! "CheapCalc" will do double precision addition, subtraction, multiplication, division, power, SUM, and roots (using fractional powers). CheapCalc has many other functions too numerous to mention (just like the expensive spreads!) CheapCalc is available for all Heath/Zenith computers and operating systems. For more information, check out page 58 of the Software Catalog Update #1, or call HUG and order your copy today.

Why didn't somebody think of this sooner! Books and magazines are printed on both sides of the page, why not listings from your computer's printer? With HUG's "Both Sides" printer utility, now you can send your printer's output to both sides of the page; properly positioned, and page numbered! Cut your paper usage, and binder space usage in half! Order HUG's "Both Sides" printer utility today. **Both Sides** is available for MSDOS and is **HUG P/N 885-3048**.

TK Solver Plus

Part 1

Edwin G. Wiggins
13 Clare Drive
E. Northport, NY 11731

Introduction

TK Solver Plus is an equation solver software package sold by Universal Technical Systems. In order to use it, you type in the equations that you want to solve. The software makes a list of the variable names you use. If the value of a variable is known, you fill it in next to its name. When you have the same number of equations as unknowns, you press the F9 key and the computer solves for the unknowns. If trial and error is required, TK Solver Plus takes care of that, too. The equations may contain trigonometric, exponential, hyperbolic, and logarithmic functions, as well as the more common algebraic functions.

If you have a series of input values for a particular variable, you can create a list containing this series, and TK Solver Plus will solve the equations for each value on the list. It will generate lists of output values as it goes along. From these lists of input and output values, you can generate professional looking tables and graphs.

Engineers and scientists have discovered that they can do lots of technical calculations with spreadsheets. TK Solver Plus can do much, much more than a spreadsheet, and it requires about the same amount of time and effort to learn. Beginning with this first article, I will show you some of the things TK Solver Plus can do, and I will show you how to do them.

Basic TK Solver Plus

When you load TK Solver Plus you get a title screen for a few seconds. Then, the first working screen, shown in Figure 1, appears. The upper half of the screen is the Variable Sheet, and the lower half is the Rule Sheet. Equations are entered on the Rule Sheet in fairly simple syntax. When you complete an equation and hit the ENTER key, TK Solver Plus scans the equation and makes a list of variable names on the Variable Sheet. The up and down arrow keys move the cursor up and down the Rule Sheet.

```
===== VARIABLE SHEET =====  
St Input---- Name--- Output--- Unit----- Comment-----
```

```
===== RULE SHEET =====  
S Rule-----
```

Figure 1
First Working Screen

After you have typed all the equations needed, you move the cursor to the Variable Sheet with the semicolon key. On the Variable Sheet, you type input values for some variables. Comments and units can be added in the appropriate columns. The arrow keys move the cursor up and down a column or from one column to another.

A specific example will illustrate the use of the software. Suppose we want to know the best height and radius for a tin can. "Best" is defined as minimum surface area for the required internal volume. The equations for surface area and volume of a cylinder are:

$$A = 2(\pi * R^2) + 2 * \pi * R * H$$
$$V = \pi * R^2 * H$$

In these equations, R is the radius of the cylinder and H is its height. The first term on the right side of the area equation, represents the areas of the top and bottom; the second term represents the area of the curved side.

Figure 2 shows the Rule and Variable sheets after these equations have been typed. Note that in TK Solver Plus, pi is entered as pi(). The cursor has been moved to the Variable Sheet by hitting the semicolon key. Comments have been added to explain each variable name, and a value of 1 has been assigned to the volume. Since there are four variables and

two equations, we must fix the value of one more variable. That will leave us two equations in two unknowns. Let's fix the value of H as 1 and have TK Solver Plus solve for the required radius and surface area.

The solved Variable Sheet is shown in Figure 3. For our choice of V and H, the radius must be .56418958, and the surface area is 5.449077. Note that no units have been specified in the Unit column. Specification of units is optional, but recommended. Once units have been specified, unit conversions can be defined. I'll tell you all about this feature next.

Since we're looking for the smallest possible area for a volume of 1, we now proceed to enter different values for H and have TK Solver Plus calculate the radius and the area each time. By trial and error we find that the smallest area occurs when the height is 1.08. Figure 4 shows this solution.

Using Units

Units can be added to our Variable Sheet by moving the cursor to the Unit column and typing the appropriate unit. Suppose we want to do our calculations in inches. We move the cursor to the Unit column in the row for the variable A and type IN^2, which means inches squared. Similarly, we move the cursor to the R row

```

===== VARIABLE SHEET =====
St Input---- Name--- Output--- Unit----- Comment-----
          A              CAN SURFACE AREA
          R              CAN RADIUS
          H              CAN HEIGHT
          V              CAN VOLUME

```

```

===== RULE SHEET =====
S Rule-----
* A=2*(pi()*R^2)+2*pi()*R*H
* V=pi()*R^(2)*H

```

Figure 2
Problem Set Up

```

===== VARIABLE SHEET =====
St Input---- Name--- Output--- Unit----- Comment-----
          A              5.5449077    CAN SURFACE AREA
          R              .56418958    CAN RADIUS
  1      H              CAN HEIGHT
  1      V              CAN VOLUME

```

```

===== RULE SHEET =====
S Rule-----
A=2*(pi()*R^2)+2*pi()*R*H
V=pi()*R^(2)*H

```

Figure 3
Problem Solved

```

===== VARIABLE SHEET =====
St Input---- Name--- Output--- Unit----- Comment-----
          A              5.535828    CAN SURFACE AREA
          R              .54289168    CAN RADIUS
  1.08    H              CAN HEIGHT
  1      V              CAN VOLUME

```

```

===== RULE SHEET =====
S Rule-----
A=2*(pi()*R^2)+2*pi()*R*H
V=pi()*R^(2)*H

```

Figure 4
Minimum Surface Area

and type IN, to the H row and type IN, and to the V row and type IN³.

So far, we have only been using the Rule and Variable sheets which appear on the screen when TK Solver Plus is loaded. There are many other sheets that can be called to the screen. Right now, we want the Unit Sheet, which is where unit conversions are defined. Press =U to bring the Unit Sheet to the screen in place of the Variable Sheet. The Rule Sheet will remain on the lower half of the screen.

In the From column we type GAL. After moving the cursor to the To column, we type IN³. There are 231 IN³ in a gallon, so we type 231 in the Multiply By column. In the next row of the Unit Sheet, we type the conversion from FT to IN, and in the third row, we type the conversion from FT² to IN². The completed Unit Sheet is shown in Figure 5. You might expect TK Solver Plus to know that there are 144 IN² in an FT² if you tell it that there are 12 IN in an FT. Unfortunately, it

doesn't. It will, however, make a conversion in either direction (FT to IN and IN to FT). It will also chain together two conversions. If you give the conversion from FT to IN and FT to YD, the program will convert from IN to YD on command.

Let's return to the Variable Sheet by typing =V. Now we move the cursor to the IN³ entry and overwrite GAL. When we hit ENTER, the value of V changes from 1 (IN³) to .004329 (GAL). If we change the Unit column entry for H from IN to FT, the value changes from 1.08 (IN) to .09 (FT). Figure 6 shows the modified Variable Sheet.

Using Lists

You're probably thinking that there must be a better way to find the best choice of R and H, and there is. Instead of typing one value of H after another and solving each case, we can create a list of H values and have the program calculate and store the corresponding values of R and A.

First, we move the cursor to the Status column at the far left of the Variable Sheet. We type L (for list) in the rows for variables A, R, and H. This tells the program that there are lists associated with these variables. H will have an input list, while A and R will have output lists. We must put an entry in the input column for each variable that has an input list. Any numerical entry will do, since the actual value is not used in any calculations. The value 1.08 is already in the input column for H. We'll just leave it there.

Next, we need to create the input list for H. We move the cursor to the H in the name column of the Variable Sheet and press the > key twice. This brings us to an empty List Subsheet for the variable H. Here we type the values of H that we want to use in the Value column. Figure 7 shows a filled in List Subsheet. There are also features that allow automatic filling of the List Subsheet, but we'll come back to those later.

We move back to the Variable Sheet either by pressing the < key twice or by typing =V. Instead of pressing F9, we now press F10 to initiate a list solve. The program calculates R and A for each value of H on the list. These R and A values are automatically stored in lists as the calculation proceeds.

Making Tables

The three lists (H, R, and A) can be turned into professional looking tables or graphs. Typing =T brings up the Table Sheet in place of the Variable Sheet on the top half of the screen. We type Table1 in the Name column and Best Tin Can in the Title column. This Table Sheet is a directory of any and all tables associated with this problem. Pressing the > key takes us into the subsheet for the table we just named. The top portion of the Ta-

ble Subsheet controls the format of the table. Since the default settings are acceptable, we move immediately to the lower section where we specify table contents. We can expand the subsheet to fill the entire screen simply by pressing the F6 key. We can bring the Rule Sheet, or any other sheet, back to the lower portion of the screen at any time by typing /W followed by the first letter of the sheet's name (R for the Rule Sheet).

In the first row under List, we type H. We move the cursor down the column with the cursor arrow key and type R in the second row. In the same manner, we type A in the third row. The other columns here are optional. For the moment we will ignore them. Figure 8 shows the filled in Table Subsheet. Pressing F8 produces the table shown in Figure 9.

Making Graphs

Making graphs is similar to making tables and just as easy. Typing =P brings the Plot sheet to the screen. This is a directory of all graphs associated with this problem. We enter the graph name of our choice in the Name column, the graph type (line, bar, or pie) in the Plot Type column, and the graph title in the Title column. TK Solver Plus automatically determines the type of video card installed in the computer and fills in the Mode column with the appropriate entry.

Figure 10 shows the filled in Plot Sheet. We press the > key to dive into the subsheet for the plot we just named. The upper portion of the Plot Subsheet controls display format, and we can accept the default values. Notice that the title we typed on the Plot Sheet appears in the Title row of the subsheet. Since we want a plot of area versus radius, we type RADIUS after X-Axis Label: and AREA after Y-Axis Label:. The x-axis values are in the list named R, so we type R after X-Axis List:. Similarly, we type A under the heading Y-Axis at the bottom of the sheet. Figure 11 shows the filled in subsheet. Finally, we press F7 and the graph shown in Figure 12 appears on the screen. A printed copy of the graph can be obtained by pressing the P key while the graph is on the screen.

Trial and Error

If we specify the volume and the height, TK Solver Plus uses the equation for V to determine R, since it is the only remaining unknown in that equation. Note that the unknown being solved for need not appear alone to the left of the equals sign. Next, the program calculates the area from the area equation. No trial and error is involved in these calculations.

Suppose we fix the values of area and volume and ask TK Solver Plus to calculate the corresponding radius and height of the can. Since R and H both appear in each of our equations, a direct solution is

```

===== UNIT SHEET =====
From----- To----- Multiply By--- Add Offset---
GAL         IN^3         231
FT          IN          12
FT^2       IN^2         144
  
```

Figure 5
Unit Sheet

```

===== VARIABLE SHEET =====
St Input---- Name---- Output--- Unit----- Comment-----
              A         5.535828  IN^2         CAN SURFACE AREA
              R         .54289168 IN          CAN RADIUS
.09           H              FT          CAN HEIGHT
.004329      V              GAL          CAN VOLUME
  
```

Figure 6
Modified Variable Sheet

```

===== LIST: H =====
Element-- Value-----
1         .8
2         .9
3         1
4         1.1
5         1.2
6         1.3
7         1.4
8         1.5
  
```

Figure 7
List Subsheet

```

===== TABLE: table1 =====
Screen or Printer:      Screen
Title:                  Best Tin Can
Vertical or Horizontal: Vertical
Row Separator:
Column Separator:
First Element:          1
Last Element:
List----- Numeric Format-- Width-- Heading-----
H              10
R              10
A              10
  
```

Figure 8
Table Subsheet

Best Tin Can		
H	R	A
.066666667	.630783131	5.67066184
.075	.594708039	5.58521695
.083333333	.564189584	5.5449077
.091666667	.537933661	5.53611238
.1	.515032269	5.54991849
.108333333	.494826999	5.58027818
.116666667	.476827227	5.62296279
.125	.460658866	5.67494086

Figure 9
TK Solver Plus Table

not possible. In this case, TK Solver Plus can solve by trial and error.

Figure 13 shows the familiar Variable sheet. Values of 5 IN³ and 20 IN² have already been specified for V and A, respectively. The letter G appears in the status

column in the row for the variable H. This indicates that trial and error is required, and a guess value is entered in the input column. The guess value 5 has been entered. It is not necessary to make an accurate guess, but the closer the guess, the fewer iterations are required to find the real answer.

As before, we press the F9 key to initiate the solution. TK Solver Plus writes a message at the top of the screen during the solution process to inform us of its progress. It will iterate until a solution is found or until 10 iterations have been performed without finding a solution. If a solution is found, the value will appear in the Output column. Otherwise, the current guess will appear in the Input column. In this case, 10 more iterations will be done if F9 is pressed again.

```

===== PLOT SHEET =====
Name----- Plot Type-- Display Mode-- Title-----
plot1      Line chart  CGA high-res  FIGURE 12 BEST TIN
plot2      Line chart  CGA high-res  Raduis vs. Height

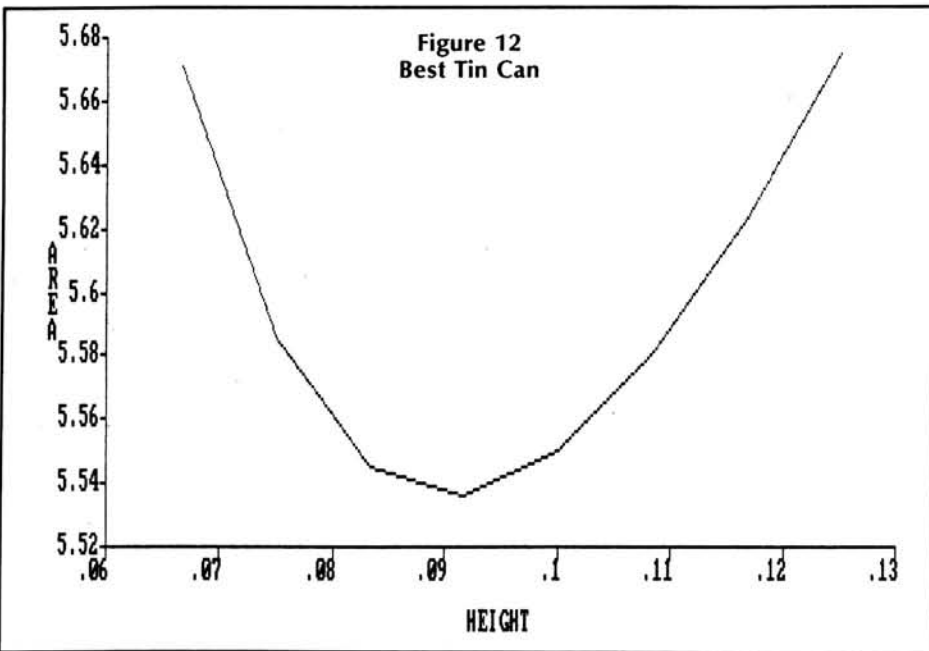
                Figure 10
                Plot Sheet

```

```

===== LINE CHART: plot1 =====
Display Scale:      Yes
Display Zero Axes:  None
Display Grid:      No
Logarithmic Axes:  None
Title:             FIGURE 12      BEST TIN CAN
X-Axis Label:      HEIGHT
Y-Axis Label:      AREA
X-Axis Minimum, Maximum:
Y-Axis Minimum, Maximum:
X-Axis List:       H
Y-Axis--- First-- Last-- Style----- Character--
A           1           *
                Figure 11
                Plot Subsheet

```



```

===== VARIABLE SHEET =====
St Input--- Name--- Output--- Unit--- Comment-----
 20         A          IN^2    CAN SURFACE AREA
           R          IN      CAN RADIUS
G 5         H          IN      CAN HEIGHT
 5         V          IN^3    CAN VOLUME

                Figure 13
                Ready for Trial & Error

```

```

===== VARIABLE SHEET =====
St Input--- Name--- Output--- Unit--- Comment-----
 20         A          IN^2    CAN SURFACE AREA
           R          .55318016 IN    CAN RADIUS
           H          5.201001 IN     CAN HEIGHT
 5         V          IN^3    CAN VOLUME

                Figure 14
                Trial & Error Solution

```

The solved variable sheet is shown in Figure 14. For this problem, there are actually two solutions. That is, there are two different pairs of values (R and H) that satisfy the equations. Depending on the values of A and V, one of the pairs may contain a negative value. This is mathematically correct, but physically impossible. For other choices of A and V, there are two valid solutions. Where there are two solutions, TK Solver Plus tends to find the one closer to the guess value.

Sometimes when there are two solutions, TK Solver Plus tends to oscillate. First, it moves toward one solution, then toward the other. The result is neither solution is reached. This is not a bug in TK Solver Plus. Rather, it is a common problem in numerical analysis, the mathematical procedure that TK Solver Plus uses. When this behavior shows up, the problem can usually be corrected by trying a different starting point.

Numerical Format

You may have noticed that TK Solver Plus often displays a lot of digits after the decimal point. The TK Solver Plus Table, Figure 9, is a good example. A tin can could never be manufactured to the accuracy of the numbers in this table. The table would also look better if there were fewer digits after the decimal. Notice in Figure 8 that one of the columns on the Table Subsheet is named Numeric Format. We ignored this column when we set up the table.

Let's go back and fix up the numeric format in this table. First, we call the Numeric Format Sheet to the screen by typing =N. We can have many numeric formats associated with one problem. The Numeric Format sheet is simply a directory of these. We simply give a name for our numeric format (call it DECIMAL) on this sheet and then dive into the associated subsheet. This is like what we did for tables and graphs.

Figure 15 is a filled out Numeric Format Subsheet. This subsheet allows us to control many aspects of numerical display. At the moment, we are concerned only with the number of decimal places. We move to the "Numeric Notation:" row and change "Either Scientific or Decimal" to "Decimal" by simply typing D. This prevents TK Solver Plus from using scientific (power of 10) notation. Then, we move to the "Decimal Places:" row and type 3. This indicates that we want 3 digits after the decimal.

We return to the Table Subsheet by typing =T and then the > key. The screen now looks just like Figure 8. We move to the Numeric Format column and type DECIMAL (the name of our numeric format) for each of the 3 variables. Since DECIMAL indicates that we want 3 digits after the decimal, our new table looks like Figure 16.

```

===== NUMERIC FORMAT: DECIMAL =====
Comment:
Numeric Notation:           Decimal
Significant Digits:        15
Decimal Places:            3
Padding:                   None
Decimal Point Symbol:      .
Digit Grouping Symbol:
Zero Representation:       0
+/- Notation:              - Only
Prefix:
Suffix:
Justification:             Left
Left Margin Width:        0
Right Margin Width:       0

```

Figure 15
Numeric Format Subsheet



**EXPLORE
NEW WORLDS
WITH
HUG
GAME
SOFTWARE**

Best Tin Can

H	R	A
.067	.631	5.671
.075	.595	5.585
.083	.564	5.545
.092	.538	5.536
.1	.515	5.55
.108	.495	5.58
.117	.477	5.623
.125	.461	5.675

Figure 16
Revised Table

Conclusion

We've seen the basic features of TK Solver Plus in this first article. There are many interesting advanced features, and these will be the subject of future articles. We'll look at user defined functions, automatic list filling and much more. *

FORM FILL-R & FORM EDIT-R

Fill in pre-printed forms, or create your own forms and print on blank paper. The printed form can be a copy of the screen layout (including lines and boxes), or create a separate data-entry screen -- this allows you to position data fields for easy entry, and put text on the screen which does not appear on the printed form.

The FORM EDIT-R creates form definitions. It handles:

- Placement of data entry fields on the data-entry screen and the printed form
- Attributes of each data field (type -- Text, Integer, Real, Dollars, etc. -- right justify, uppercase only, required for printing, etc.)
- Form characteristics (page length, perforation-skip, left and top margins, etc.)

Once you create the form definition, use the FORM FILL-R to fill in forms and print them, or save them to disk files for later recall and/or correction. FORM FILL-R can accept input from, or create output as, plain text files -- letting you interface with many other programs. A number of example form definitions are included.

FORM FILL-R and FORM EDIT-R run on any PC or compatible computer, and also on the Z-100. **\$49.95** Postpaid.

EGAD

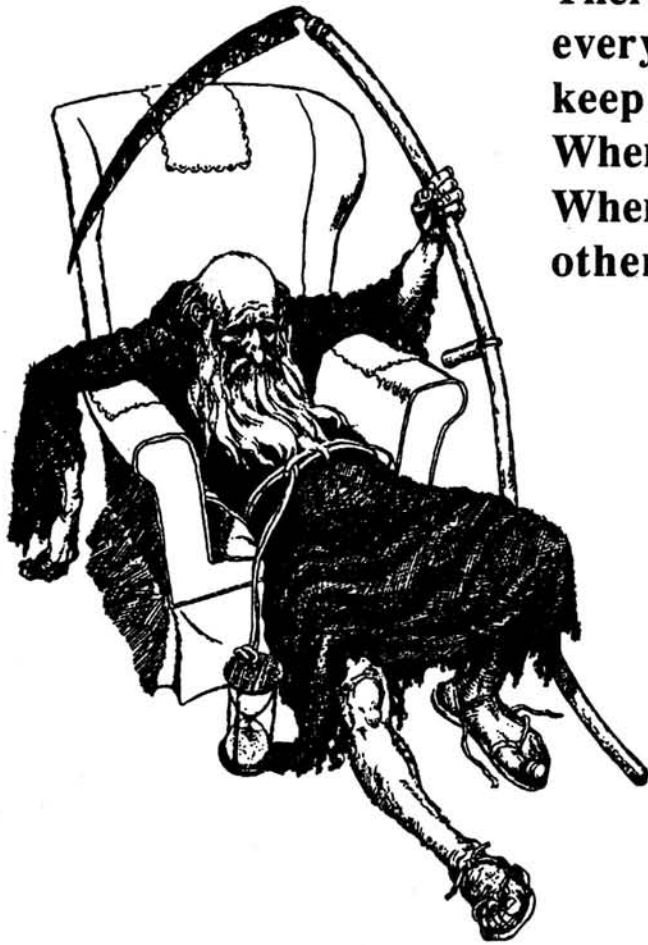
Graphics and Text Screen Print Package for the VGA, EGA, and CGA displays.

- Print any part of the screen ('crop box' lets you use the cursor keys to select any rectangular area)
- Enlarge for emphasis (1 to 4 times in graphics modes).
- Color graphics print in color (on color printers) or in black and two shades of gray (for black-only printers). Color text also prints in color, on color printers.
- Set program lets you determine which screen colors map to which printer colors (or gray and black tones).

In this list of supported printers, the number of possible printer colors is shown (Up to 16 printer colors may be chosen for each screen graphic mode). **Epson** and compatibles (Black and two Grays); **Star NX-1000 Rainbow** (30 colors including black and two grays); **Xerox 4020** (Over 64 colors); **Dataproducts 8020** (Black and 6 colors); **NEC-8023/C.Itoh 8510** (Black and two Grays). EGAD, **\$25.00** Postpaid.

**Lindley Systems 4257 Berwick Place,
Woodbridge, VA 22192 (703) 590-8890**

Call or Write for Catalog and Product Information.
Reader Service #136



There comes a time in the life of every computer when it just can't keep up with the grandkids...
When its memory comes up short...
When it begins to be passed over for other, more youthful CPU's...

So, What's the Catch?

The catch is that since these upgrades involve components from your old computer, you'll lose use of that machine for about a week (counting shipping time in both directions). However, we will schedule your system upgrade before you send it to us to minimize the lost time, and make every effort to turn the machine around within 48 hours of receipt.

Still Sitting in that Easy Chair?

Don't rest for long! Prices and availability are subject to change without notice! Better to lock your upgrade in now, while there's still time! *Prices based upon models available at the time this ad was placed.* Call for current prices, availability and full details.

Let First Capitol Computer rejuvenate your old computer.

No, there's no way to reverse the hands of time, but First Capitol Computer CAN give your computer a new mind and body! Our upgrades aren't mere circuit boards, modifications, or kits! We take your original disk drives and move them into a new Zenith Data Systems chassis. The result is an almost entirely new machine indistinguishable from a unit right off the production line! No compatibility problems and all work is performed by factory authorized service technicians.

The Cost? A Small Portion of What a New Computer Would Cost You!

That's right! Since First Capitol uses parts from your old machine in the new cabinet, you only pay for what you need - and you receive a built-in trade-in credit for your old machine! We'll even provide trade-in credit for any third-party products which won't work in faster machines towards replacements - and install them for you at no extra charge!

Upgrades Currently Available:

FCC-150/286-UP: Z-151/2/8/9 or Z-161 to Z-286 80286 AT compatible (16 bit). \$1295.

FCC-150/248-UP: Z-151/2/8/9 or Z-161 to Z-248/12 12MHZ 80286 AT compatible. \$1895.

FCC-150/386-UP: Z-151/2/8/9 or Z-161 to Z-386 16MHz 80386 computer. \$2495.

FCC-248/386-UP: Z-241/248 to Z-386 16MHz model (32 bit processor). \$2095.

FCC-150/CL-AT: Z151/2/8/9 or Z-161 to First Capitol 12MHz CL-AT (16 bit) Clone. \$995.

FCC-150/CL38TW: Z151/2/8/9 or Z-161 to First Capitol 20MHz CL-386TW (32bit) Tower. \$1995.

FCC-248/CL38TW: Z-241/248 to First Capitol 20MHz CL-386TW (32 bit) Tower. \$1695.

Call for Prices on Z-386/25 Upgrades!

These Also Work on Kit Models!

Add 2% shipping and handling (plus 6.725% sales tax, if a Missouri resident).

 **First
Capitol
Computer**

#16 Algana
St. Peters, MO 63376
Orders: 1-800-TO-BUY-IT
Technical: 1-314-447-8697

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

Customize Your Programming Language

Introduction

Customize your programming language? It sounds silly. What is a programming language if not a collection of rules to be followed. How can you or I fiddle with the linguistic dictates of FORTRAN or BASIC, Pascal or C? Of course, we can't. What we can do is take advantage of our language's flexibility to create our own personal language augmentations — visual signposts which can improve comprehension of any code we may write.

It's likely you are already customizing your programs, although you may not put a name to the process. If you're using tricks like consistently inserting blank lines in given situations, mixing upper- and lowercase letters to send yourself messages, or highlighting important comments, you're customizing. A customization technique may be as simple as using statement indentation to highlight logical program structures. It may be as complex as developing hierarchies of comment types for different applications.

I have never known an experienced programmer who didn't have their bag of tricks full of personalized language extensions. In this article, I will illustrate some of the customization techniques I have encountered during my quarter century of programming. You can select those which fit your needs, or invent your own. Be a little cautious, however. Some of my examples may not be compatible with one another, and the use of one extension may preclude the use of another.

My examples will come from Fortran. Fortran is a very serviceable, widely used language which is rich in customization opportunities. Many of the basic ideas will be adaptable to other languages.

Opportunity Knocks

Fortran is a compiled language originally developed to program scientific and engineering calculations. Being an old language, it was designed to conform to the constraints of the once ubiquitous 80 column punch card. Although the language has evolved, the 80 columns have remained. They are assigned as follows: col-

umns 1-5 contain the statement number, if one is present; column 6 is used to indicate if the "card" is a continuation of the preceding statement; columns 7-72 are for the actual statement; and columns 73-80 are available for whatever use the programmer can find for them.

If the letter C appears in the first column of a card, the statement is a comment. Comments appear in program lists,

but they are ignored by the compiler.

The most recent and widely used version of Fortran is ANSI Fortran-77. This version allows us substantial creative license and provides many customization possibilities.

- Fortran-77 allows us to
- Intersperse blank lines at will to create white space.
- Use spaces (blanks) at will to create white space, except within constants representing text.
- Use upper- and lowercase letters interchangeably, except within constants representing text.
- Use any printable character within comment lines.
- Use statement numbers between 1 and 99999 in any order.
- Use columns 73-80 for any desired information.

In addition, many Fortran compilers allow the non-standard underscore () in variable names, as well as variable names considerably longer than the six characters permitted by the standard. Each of these opportunities is discussed below.

Blank Lines

Blank lines create white space, which in turn creates visual buffers between groups of statements. I habitually use blank lines to set off different types of declarative and other non-executable statements at the beginning of a program unit. These statements (e.g., DIMENSION, INTEGER, REAL*8, DATA, etc.) are often consulted during program development

and debugging. Blank lines make it easy to locate the desired statements quickly.

An excellent use of blank lines is to call attention to important comments, such as those that announce major sections of code within a program unit. The visual effect can be compounded by beginning and ending the comment with lines of eye-catching characters not normally used in Fortran. For example,

```
C  ! ! ! ! !
C  ! Begin third stage integration calculations !
C  ! ! ! ! !
```

Blank lines are also handy when used immediately before and after a group of related statements beginning with one or more comment lines. A simple example is given below.

```
C READ THE POLAR COORDINATES
    READ (1,50) R,THETA
    50 FORMAT(2F8.3)

C CALCULATE THE CORRESPONDING CARTESIAN
    COORDINATES
    X=R*COS(THETA)
    Y=R*SIN(THETA)

C PRINT X AND Y
    WRITE (3,100) X,Y
    100 FORMAT(' X=' ,F8.3 , 'Y=' ,F8.3)
```

These last two examples illustrate an idea mentioned in the introduction, developing a hierarchy of comment types. In the cases presented here, the applicable range of the comments is pretty obvious. In the first case, the comment highlighted with exclamation points will be in effect until another such highlighted comment is encountered. In the second case, each comment applies only to the statements preceding the next blank line. Unfortunately, program logic may force you to have many comments in effect simultaneously. Very quickly your hierarchy of comments can become uncomfortably complex. I have yet to find a completely satisfactory system. Perhaps you can do better.

Blank Spaces

Blank spaces can be used almost anywhere within statements to improve pro-

gram readability. They should be used liberally, but consistently. Frequently, they are employed to set off the individual elements of a statement. Compare the appearance of the following statements, identical except for the use of blanks.

```
IF (I1.LT.I2)X(I)=I*TIME
IF (I1.LT.I2) X(I)=I*TIME
IF (I1 .LT. I2) X(I) = I * TIME
```

They differ in readability. I personally prefer the spacing of the second. The first is clearly run together; the third seems a little airy. Experience has taught me, however, that many programmers like the third. That's what customization is all about.

The creative use of blanks can also make many non-executable statements easier to read. Compare the two DATA statements below for ease of comprehension. In the second case, we have continued the statement to additional "cards" and aligned the variable names.

```
DATA ALPHA/21.0/,BETA/14.66/,GAMMA/0.0013/,DELTA/1,2,3/
```

```
DATA ALPHA /21.0 ,
& BETA /14.66 ,
& GAMMA /0.0013/,
& DELTA /1,2,3/
```

Another frequent use of blanks is to indent statements selectively to highlight certain logical structures. Consider as an example the IF-THEN, ELSEIF, ENDIF structured programming construct. Again, the utility of this practice can be judged by a simple comparison.

```
IF (X.LT.Y) THEN
Z=X
ELSEIF (X.GT.Y) THEN
Z=Y
ELSE
Z=X*Y
ENDIF
```

```
IF (X.LT.Y) THEN
Z=X
ELSEIF (X.GT.Y) THEN
Z=Y
ELSE
Z=X*Y
ENDIF
```

The improvement is obvious. The only likely controversy would concern the number of columns to indent. Another prime candidate for indentation is the range of a DO loop. A DO loop is the basic repetitive logical structure of Fortran. I have provided nested DO loops to illustrate this case.

```
DO 10 I=1,15
DO 10 J=1,10
X(I,J)=Y(I)*Z(J)
DO 10 K=6,20
ARR(I,J,K)=X(I,J)
10 CONTINUE
```

```
DO 10 I=1,5
DO 10 J=1,10
X(I,J)=Y(I)*Z(J)
DO 10 K=6,20
ARR(I,J,K)=X(I,J)
10 CONTINUE
```

Enough said about blanks.

Underscore

One of the most annoying problems a programmer faces is assigning adequately descriptive names to his or her variables. The ANSI Fortran standard does nothing to help the situation, allowing a maximum of six alphanumeric characters in symbolic names. To indicate a case counter, for example, you're stuck using CASCNT or NCASE or something equally fuzzy. Fortunately, many implementations of Fortran allow longer names and/or use of the underscore () as a legitimate character in names. Both these extensions are helpful, giving you the opportunity to try CASE_COUNT, case_ct, and many other more descriptive names. If your compiler has these extensions, use them.

A couple words of caution about long symbolic names are in order, however. First, the compilers that allow them al-

most always recognize only a handful of characters at the beginning as being significant. MicroSoft Fortran, for example, considers only the first six as significant, although it will let you use up to 660(!). The trap is that you might try naming three counters COUNTER1, COUNTER2, and COUNTER3, not realizing that the compiler, using only the first six characters, will not distinguish among them. Second, if you one day transport the code to a compiler which adheres to the ANSI standard, you will have an entertaining conversion job facing you.

Upper- and Lowercase Letters

Unless explicitly specified otherwise, variables names in Fortran beginning with the letters I through M are integers. Any other initial letter indicates a real variable. In selecting variable names, this can be a nuisance, for your choice of variable name may not match this convention. A simple remedy is to use "i" and "x" as **prefixes** to troublesome uppercase variable names. The integer counter you want then becomes iCOUNT, and the real integration interval you need is xINT or even x_INT.

We can take this idea a step further. Let's suppose that you want to be able to tell by a variable's name if it is a constant, or perhaps if it is an input to the program. This information can be appended to variable names with lowercase **suffixes**. The suffix "k" in ACCk could indicate a constant. An "i", as in BETAi, could denote an input quantity. But why stop with a single letter suffix? How about Eik or E_ik to mark an input constant?

Another possibility is to use lowercase letters for program variables and uppercase letters for function and subroutine names. The statement

```
z = SQRT(x**2 + y**2)
```

is a simple example.

My favorite use of upper and lower cases is illustrated below. I put all my Fortran statements in upper case and all my comments essentially in lower case. This practice makes it very easy to distinguish the two, especially when combined with blank lines. Compare the following with the earlier version which was entirely in uppercase letters.

```
C Read the polar coordinates
READ (1,50) R,THETA
50 FORMAT(2F8.3)
```

```
C Calculate the corresponding cartesian
coordinates
X=R*COS(THETA)
Y=R*SIN(THETA)
```

```
C Print X and Y
WRITE (3,100) X,Y
100 FORMAT(' X=',F8.3,' Y=',F8.3)
```

Statement Numbers

In Fortran, when a statement is referenced by another statement, it requires a unique statement number between 1 and 99999. In broad terms, the referencing statement may be either a DO statement which begins looping logic, a GOTO statement to transfer program control, or a READ or WRITE statement for transferring data into or out of the code in a specific format (as described in a numbered FORMAT statement). Statement numbers are assigned by the programmer, and they need not be in any specific order.

This freedom of choice permits a programmer to reserve groups of statement numbers with a certain characteristic to denote common statement characteristics. For example, in past times, I would collect all my format statements at the end of each program unit (yet another small freedom Fortran allows). I would denote input formats by numbers from 1001-1999 and output formats by numbers from 2001-2999. This served several purposes. First, almost no thought was needed when numbering formats. I would just take the next higher unused number. Second, I knew immediately which formats were used for input and which for output. Lastly, I always knew where to find the format statements. This was especially helpful if some formats had multiple references. I eventually carried this even further and used the 3000s for format statements added for debug output. This made them easy to delete when they were no longer needed. (Since writing PRONEAT, a code to renumber and otherwise tidy my Fortran codes, I no longer follow this practice.)

Another common signpost is the use of a statement number like 9999 to signify the end of a program unit. Many programmers will also adopt numbers like 1000, 2000, etc. to mark the start of major program segments.

Columns 73-80

Originally, columns 73-80 were used in Fortran to specify card sequence within a program deck. Small decks were usually held together with rubber bands, while larger decks of more than a few hundred cards were kept in card trays. When the program was run, a computer operator would move the cards from the tray to the card reader and then return them to the tray after they had been read. There was a substantial chance the deck would be dropped. When it did happen (and eventually it always did), the sequence numbers made the reassembly task easier. Today, the sequencing of card images in a text editor serves no purpose. This frees these columns for other uses.

Several obvious uses immediately come to mind. A simple notation in these columns could indicate temporary statements which are to be removed after the program is debugged. Alternatively, they might mark recent additions to the logic, thereby drawing attention to themselves during debugging. Yet a third use would be for identifying statements as being part of a particular set of changes. This might be done by dates or an alphanumeric identifier.

Opportunities in Other Languages

Well and good, you say. Now how about those of us who don't program in

	Blank Lines	Spaces	Comments	U/L Case	Statement #'s/Lbls	Under-score
Assembly	Yes	Limited	Yes	Yes	Labels	Yes
Basic	No	No	Yes	No	Numbers	No
C	Yes	Yes	Yes	Yes*	Labels	Yes
Fortran	Yes	Yes	Yes	Yes	Numbers	Yes
Pascal	Yes	Yes	Yes	Yes	Numbers	Yes
ZBasic	No	Yes	Yes	Yes**	#'s/Lbls	Yes

* Distinguishes upper and lower case

** Can be set to distinguish or not distinguish case

Table 1

Fortran? What's available for us? You can get an idea from Table 1, which compares a variety of "generic" languages feature-by-feature with Fortran. I refer to them as generic because, with the exception of ZBASIC, which is produced only by Zedcor, a particular vendor's language implementation may or may not have the features indicated. (ZBASIC is a very useful hybrid of BASIC and structured programming whose editor will automatically indent all appropriate logical constructs.)

As you can see, with the exception of

BASIC, all these languages offer a number of options to the programmer seeking to customize his codes. The rest is up to you. Take the concept of customization and run with it. You'll see the benefits when you debug your code today; you'll see them six months from now when you pick it up to make modifications.

Good luck!



Zenith's New Z-386/25 is the Fastest PC Compatible on the Market Today, BUT...



First Capitol Computer will MEET or BEAT any competitive price anywhere that isn't below our cost, and we'll still build the system to suit you!

...What do you do if you don't want to buy your system the way the factory wants to sell it to you? Why, you turn to First Capitol Computer, of course, as over 45,000 others have for more than 8 years now!

First Capitol has been building customized systems the way our customers wanted them ever since the Z's were first released - and we don't charge extra for this service! In fact, our prices are some of the lowest in the country! Unlike other discount houses, we also retain engineering and technical personnel on staff to evaluate new product compatibility, and keep you out of trouble when you purchase upgrades for the new computer dynamo.

So, when you decide it's time to upgrade, call First Capitol for help. We'll show you the least expensive way to configure your system, and build it to suit your exact needs - including those of your wallet!

Call for the latest Z-386/25 pricing and upgrades

**First
Capitol
Computer**

#16 Algona Drive
St. Peters, MO 63376
Orders: 1-800-TO-BUY-IT
Other Calls: 1-314-447-8697

Spring Specials from Heath

Kit projects...

Heathkit Flood Alarm detects unwanted water Avert damage from leaking water pipes or basement wall seepage. Three hours to build. Needs 9V battery. 13 1/4" H x 3 3/8" W x 4 7/8" L. Kit GD-1701 (2 lbs.) ... **\$12.95**



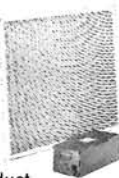
Program your doorbell with a favorite tune Arrange wire leads on one-octave keyboard inside doorbell to program any tune. Includes songbook with 50+ college, seasonal, Christmas and special occasion songs. Back door button activates a part-tune. Volume, tone, speed and delay adjustable. Two-evening kit. Takes standard 16 VAC, 50/60 Hz bell transformer. Kit TD-1089 (3 lbs.) ... **\$32.95**



Build your own Heathkit draft detector Easily detect heat loss areas. Turn it on, adjust for silence, then move sensor in areas where heating or cooling losses occur. Temperature changes set off beeping alarm and flashing LED. Used in temperatures from 59-95F. Takes a 9-volt battery (not included). Kit NE-2112 (2 lbs.) ... **\$14.95**



Heathkit Furnace Air Cleaner. Works with heating or A/C system to reduce the effects of pollen, dust and cigarette smoke. Power supply mounts onto a cold air duct, and turns on collecting cell when blower is running. 1" collecting cells below directly replace original furnace filter. 120VAC also works with GD-2196 2" filters. Kit GD-3196 (9 lbs.) ... **\$79.95**
Accessory filters 4 sizes available. each ... **\$74.95**



Heathkit Portable Air Cleaner. Helps solve tough air-cleaning problems, removes dust, smoke, other pollutants from 6,000 cubic feet (25'x30'x8' room). One-evening kit includes power supply, 3-speed fan control and assembled filter. 120 VAC, 60 Hz. Dimensions 26 1/2" H x 17 1/4" W x 13 1/2" D. Kit GDS-1297 (59 lbs.) ... **\$199.00**
Accessory: Replacement Charcoal Filter Assd. GDA-1297-2 (2 lbs.) ... **\$9.95**



Charging system tester Diagnose faulty components in your car's charging system with accuracy and ease. A two-wire setup and three quick tests show you if the battery has sufficient charge to reliably start the engine, if the battery is being charged by the alternator, if the voltage regulator is faulty (causing the battery to become overcharged), or if the alternator stator windings and rectifier diodes are functioning properly. No external power or battery required. Dimensions are 5 1/2" H x 2 3/4" W x 3 1/4" D. Kit CI-2065 (2 lbs.) ... **\$19.95**



Noise generator kit To properly check out any stereo or sound system with a spectrum analyzer, you need a noise source with a nearly constant energy output. This one-evening Pink/White Noise Generator will do just that...and at a fraction of the cost of comparable noise generators. Kit AD-1309 (2 lbs.) ... **\$24.95**



For the electronic hobbyist...

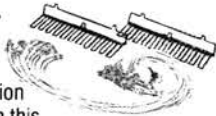
Pocket-size digital meter for home and shop Compact digital meter includes 3 1/2 digit, LCD display for readings up to 1999, audible continuity checker and diode test. Overload protected on all ranges, new transistor hFE test. Assd. SM-2310 (1 lb.) Now only **\$24.95**



Connector adapter kit Make 108 different coaxial connector adapters with gold-plated pins and Teflon insulation. Kit contains male and female, N, BNC, UHF, SMA TNC and mini UHF connectors. Special in-between fittings allow you to assemble your own adapters in seconds. Assd. HCA-3000 (2 lbs.) ... **\$69.95**



Coax adapter cables Solve frustrating coax connection problems with this convenient adapter cable assortment. Twenty cables in all help you make almost any connection. Two handy



storage racks keep your cables neatly organized. 50 ohm impedance. Assd. TPI-5000 (5 lbs.) . Now **\$49.95**

Unbeatable Scanner Prices...

Uniden 10-Channel/10-Band Handheld Scanner. Save \$31. Superportable, programmable, direct channel access, channel lockout, keyboard lock switch prevents accidental reprogramming. 55 dB @ +25kHz selectivity; audio output 300 milliwatts maximum. Receive 29 to 512 MHz with a 15-channel-per-second scan speed. Requires 5, AA (not included) or rechargeable NiCd batteries. Assd. BC-55X-LT (1 lb.) ... **\$108.88**

100-Channel/11-Band Uniden Handheld scanner. Reduced \$41. 100 programmable channels, 10 scanning banks, 11 bands of coverage. Automatic and manual search, weather search, 10 priority channels, squelch, lockout and delay. Selectivity is -55 dB @ +25kHz; audio output is 450mW maximum. Snap-on battery pack gives full-powered portable scanning. Assd. BC-100X-LT (3 lbs.) ... **\$188.88**

Uniden Tabletop Scanner. 11 bands of coverage, 5 or 15 channels-per-second scanning speed. Features automatic memory search, backup, priority, programmable lockout, scan speed control and automatic squelch setting. Selectivity -45 dB @ +25 kHz, audio output one watt at 10% THD. Takes 117 VAC, 60 Hz from an external wall outlet. Originally priced at \$169.95. Assd. BC-175X-L (3 lbs.) ... **\$148.88**

100 Channel/11-Band Uniden Mobile scanner. Auto search aircraft, police, marine and emergency bands. Fully programmable. Fits neatly under dash, flip-down stand and telescopic antenna included. Audio output is 1.5 watts at 10% THD; require 13.8 VDC (vehicle battery or AC adapter). Save \$61+. Assd. BC-580X-LT (5 lbs.) ... **\$198.88**

Deluxe Uniden Mobile Scanner. Save \$81+. 100-channel/12-band mobile scanner. Covers 800 MHz band for full 12-channel coverage. Weather search, priority, squelch, lockout, delay, auto and manual band search. Mount under dash, or AC adapter (included), flip-down stand and telescopic antenna. Takes 13.8 VDC. Assd. BC-760X-LT (5 lbs.) ... **\$248.88**

Make life easier...

Full-featured remote control center for your TV, VCR and CATV Assd. GDZ-143 (1 lb.) ... **\$29.95**
**Zenith cannot guarantee that the PCC will operate every model TV or VCR.



Recharge your NiCd batteries Assd. BP-1234 (4 lbs.) ... **\$17.95**

Amber screen fish finder Assd. MI-2020 (6 lbs.) ... **\$244.95**
Accessories: Transom Mount Transducer Assd. MIA-2020-1 (4 lbs.) ... **\$44.95**
Through-Hull Transducer Assd. MIA-2020-2 (6 lbs.) ... **\$99.95**



Full color fish finder Assd. MI-2040 (10 lbs.) ... **\$399.00**



Portable weather computer travels in your shirt pocket Save over \$15 on this compact weather instrument that goes with you hiking, biking, boating and camping. Gives time, date and current temperature, plus keeps a record of highest and lowest temperatures. Also acts as a stopwatch. Automatic power-down feature extends battery life. Requires three AAA batteries (not included). Assd. BW-100 (1 lb.)...Now only **\$24.95**

Heathkit®
to order CALL TOLL FREE
1-800-253-0570
Use order code 216-063



for credit card orders, 24 hours a day.
Some items are closeouts. All items are available in limited quantities. Prices for some items were previously reduced.

For your free Heathkit catalog call 1-800-44-HEATH.

A Windows Small Memory Model Template

William S. Hall
3665 Benton Street, #66
Santa Clara, CA 95051

Introduction

Writing even the simplest program in Microsoft Windows can be really intimidating. To reduce the pain (pun inevitable), we present you with a template which can help you to understand the basic Windows programming model and which you can use as a starting point for more interesting applications.

The program we present here is very simple. All it does is to create a window on the screen. When the window is minimized, a frame is drawn around the edge of the icon window and some identifying text is displayed. In addition, the program responds to the standard system commands. It does nothing else. But, in order to write something even this simple, the programmer must understand how to register a window class, create and show a window, borrow previous instance data, load resources, and finally, to read and act upon messages from Windows.

This template has been used many times to write other simple applications. For example, it is not difficult to create a digital clock by adding the necessary support code [1]. The template has been invaluable to us in writing Windows device driver testing programs, and we have even used it to develop a similar template for Presentation Manager [2]. Although the Application Programming Interfaces (API) in Presentation Manager and Windows are different, the programming models are very similar, and it was relatively straightforward to make the conversion.

You will probably find this article fairly difficult on a first reading. A lot of information has to be covered in a small space to describe what is essentially the minimum Windows program. But, there are authors who have done a very nice job of

starting the reader from an even more basic level [3], [4]. These writers have a bit more space to lead you more gently through the Windows maze. One, Charles Petzold, is an especially good expositor, and you will gain much by reading his book.

At any rate, if you have the Windows Software Development Kit, by all means use the Programmer's Reference manual in conjunction with reading this article. Look up each call to Windows as they are discussed and examine any related data structures. It will help you to gain the insight you need in order to understand and generalize the information we present.

This program is written for the Windows 2.0 and Windows/386 environments. Windows 1.0 is already obsolete, and if you are a registered owner of Windows, you should take advantage of the \$25.00 2.0 upgrade offer from Microsoft.

If you actually type in the source code, you will find it a greater learning experience than just reading it. However, I will send you a disk with the template for \$10.00. Perhaps HUG will even make it available on the bulletin board for free.

Hardware and Software

You will need to obtain a C compiler which is 'Windows aware'. Certain Windows functions require a special prologue and epilogue, and the compiler must be able to furnish this code. Microsoft's C compiler, version 4.0 or greater is one possibility. You must also have a copy of the Microsoft Windows Software Development Kit (SDK). Here, you will find the necessary libraries, headers, and utilities you need in order to write programs in Windows. Although each of the SDK and the compiler lists for about \$500.00, they

are frequently discounted to around \$270 each. If you already own earlier versions of either the compiler or SDK, by all means upgrade them.

If you have a machine with a hard disk and a display capable of running Windows, then you have the basic hardware needed to write Windows programs. But, you may not be well prepared to debug them. Because Windows owns all the machine's resources, you need an auxiliary terminal connected to one of the communications ports. Then you can run SYMDEB (supplied with the SDK) and direct all debugging input and output to the terminal. You can even debug at the source code level if you set the right switches during compilation and linking. However, you cannot see automatic variables unless you understand how the stack is arranged. Nevertheless, despite these and other limitations, SYMDEB can be a useful tool for tracking down problems in Windows.

You can use temporary 'printf' statements for debugging purposes if you run Windows with output redirected to an appropriate device, such as an auxiliary terminal, printer, or file. Standard output will then appear there, and this can be very helpful in analyzing what a Windows call is actually doing. However, you must not leave any permanent printf's in your program.

As an alternative to the auxiliary terminal, you may want to run dual monitors using an EGA, CGA, or VGA as the primary video and a monochrome or a Hercules board as the secondary display. If you combine this arrangement with the public domain program OX.SYS, you have a very useful arrangement. By redirecting Windows to AUX, you can see standard out-

put displayed on the monochrome screen. SYMDEB can be redirected there as well, and you do not lose the use of a communications port.

Memory Models for Windows Programs

Because Windows is a multitasking environment, several programs may be in CPU memory at the same time. Lacking a real paging system, Windows takes advantage of the segmentation feature of the Intel processor. Most Windows programs are made up of several segments, each located in its own area of the global heap. These segments can be moved, shared, and even discarded. Although the effects of memory management can generally be ignored by the programmer, there are times when he or she must be aware of the consequences. For example, a long pointer into a segment which can be moved will not be valid indefinitely. And, if the program's code is all in one module, then Windows cannot make efficient use of the limited memory available. Thus, it is really necessary to plan a program's layout in an optimal way.

One of the simplest schemes for taking advantage of Windows memory management is to divide a program into two segments, one of which contains initialization and the other run-time code. If the initialization segment is so marked, it will be discarded when it is no longer needed, and this space will be made available for other programs. Since at least two code segments are required, the so-called medium memory model is implied. In fact, this is the best model for writing Windows programs having only one data segment simply because it makes the mechanics of program segmentation very easy.

However, it is also possible to have more than one code segment in the small memory model, as we have done here. But, there are certain cautions which must be observed. For example, calls between different segments must be labeled FAR. You can call a library function only from the `__TEXT` segment (the default code segment) since all such functions are NEAR. `WinMain`, the entry point for Windows, must be in `__TEXT`. Finally, unless you say something specific about the segments in the program's definition file, they will be grouped together by the linker anyway. We will explain this in greater detail below, but first let us look at how a Windows program is organized.

Windows Programming Philosophy

Many of the simple programs that we write start from the beginning and run to completion. Other types of programs spend most of their time waiting for input from the keyboard. When the data is received, the program processes it and returns to wait for more input. A Windows program most closely resembles the second model. Most of the time the program

Figure 1

```
# Smltpl - Windows small memory model template.
# Written by Bill Hall, Olivetti ATC.
#

# no debug compile
cp=cl -d -c -AS -Gw -Ox -Zp

# Microsoft C 5.x compiler with PM libraries
LIBPM=slibw slibcr libh/NOD/NOE

# Dependencies
smltpl.res : smltpl.rc smltpl.h
            rc -r smltpl.rc

smltpl.obj : smltpl.c smltpl.h
            $(cp) smltpl.c

smltplnt.obj : smltplnt.c smltpl.h
            $(cp) -NT _INIT smltplnt.c

smltpl.exe : smltpl smltpl.def smltpl.res smltpl.obj smltplnt.obj
            link4 smltpl smltplnt,smltpl/align:16,smltpl/m,$(LIBPM),smltpl.def
            rc smltpl.res
            mapsym smltpl
```

is waiting to receive a message from Windows. This message may contain the current mouse coordinates, information as to which key was just pressed, a request to redraw the client window, or other information.

In fact, about one hundred types of messages can be sent to a Windows program. Only some of these are ever processed by the application; most are simply passed back to Windows requesting the default action. A program gets its messages by calling Windows, and while it is waiting for one to be placed in its queue, other applications can run. Thus, the act of reading messages not only supplies the program with a set of actions to be performed, but it also allows multitasking to occur.

Hence, a typical Windows application must be built quite differently from what is perhaps your concept of a program, and it takes some effort to get used to this sort of programming philosophy. In addition, even if you are very comfortable in C or Pascal, the code can appear very strange. New ideas plus a very rich programming interface make the learning curve for Windows initially quite steep. You will most likely find that even our simple template, which does practically nothing, requires a lot of effort to understand.

The Make file

A Windows program has so many pieces that it really must be managed by some kind of 'make' program. Such a tool is supplied with the Microsoft C compiler. Its syntax is similar to the UNIX version and so will be familiar to many of you. We have tried to keep our make file simple, but we have taken advantage of the macro feature in order to help you accommodate a variety of compiler versions and

libraries. You can use these macros to tailor the make file to your own environment.

So, beginning at the top of Figure 1, you see the 'cp' macro for compiling non-debugging versions of the C source files. Although you should review the exact meanings of the switches used with the compiler, here is a summary of the ones we use:

```
-d show each pass of the compiler
-c make an object file only
-AS compile for the small memory model
-Gw compile for Windows (PASCAL keyword)
-Zp pack structures
-Ox maximum optimization
```

You cannot omit `-Gw` nor `-Zp`.

Next follows a macro describing the libraries needed at link time. You will probably have to change it depending on the version of your compiler. We are using the Microsoft 5.1 compiler, and we did NOT build composite libraries when the compiler was installed. If you have built composites, please develop your own library macro; there are too many variations for us to guess at your configuration. At the very least, you must include the special window library `SLIBW` and some version of the small model C runtime library `SLIBC`.

If you are planning to use floating point numbers, please read how to do this in the Windows SDK. You must follow special procedures when dealing with the emulator library.

After the macro definitions is the command needed to make the resources for the program. Resources are exactly those objects which interact with the user. Examples are dialog boxes, fonts, strings, icons, bitmaps, and menus. A source resource `.RC` file describes these objects in text format using the appropriate syntax. The resource compiler `RC`

then converts this description into a binary .RES file. After the executable .EXE file is made, RC is used to attach the .RES file to the .EXE. SMLTPL has only some strings, which is not typical of a Windows program. However, we wanted to keep it simple.

Next are the commands to make the .OBJ files from the .C sources. You should note the named segment specified by the -NT switch used in making smltplnt.obj. This information is used by the linker in combination with the definitions file to separate the code into two segments.

Finally, the Windows executable file is put together using several commands. Note the use of a special linker, LINK4. Windows programs use the same .EXE format as OS2. Some of the new features which are provided are dynamic linking (linking to libraries at run time rather than at link time), and memory management (some aspects of which we have already described).

The 'align' command is not necessary; it just specifies the separation of segments in the disk file. The default is 512, so the disk file is a bit smaller if 16 is used. Next is the library macro, and finally, we see a new element which we examine below, the definitions file.

After the executable is made, the resources must be attached. Again, the resource compiler is run, this time without the -r switch. Finally, a symbol table file which can be read by SYMDEB is created by the MAPSYM utility.

The Definitions File

Figure 2 shows a new element, the definitions file for SMLTPL. In addition to the name (which must agree with the executable's file name) and a description string, the .DEF file contains important information about stack and heap sizes, segments and how they should be handled by Windows, a so-called stub program, names of functions in the program which are called by Windows (Exports), and those functions which are imported from other programs (Imports).

In this program, we use the minimum size recommended for the program's heap and stack. Such a large size for the stack is necessary, since many screen operations, such as bit block transfer (bitblt), use the stack extensively.

The stub is simply a program which is placed at the beginning of a Windows executable and which runs if the program is inadvertently activated from MS-DOS. The stub used here simply announces to the user that 'This program requires Microsoft Windows.' and exits. However, one could conceivably have two versions of the same program, one for Windows and one for DOS with the latter as the stub for the former.

The CODE and DATA statements declare general attributes of these segments.

Figure 2

```
; Definitions file for the Windows small model template
;
; Written by Bill Hall, Olivetti ATC
;

; Program name
NAME Smltpl

; Description string
DESCRIPTION 'Windows small memory model template'

; Program to run if run from MS-DOS
STUB 'Winstub.exe'

; Heap and stack sizes
HEAPSIZE 1024
STACKSIZE 4096

; Attributes of code and data segments
CODE MOVEABLE
DATA MOVEABLE MULTIPLE

; Attributes of particular segments
SEGMENTS
    _INIT PRELOAD MOVEABLE DISCARDABLE

; Functions called by Windows
EXPORTS
    MainWndProc @1
```

Since several instances of the same program can be running, there must be as many copies of the data as there are instances. Code, however, can be shared, and this is the default. We allow both code and data to be moveable.

Next follows specific information about the __INIT segment. If you want a multiply-segmented program in the small memory model, such declarations must be made. Otherwise, all segments are simply placed into __TEXT at load time and you lose the flexibility of separate, discardable segments.

Any function in your program which is called by Windows must be exported. In this program, only the main client window function falls into this category. In programs using callback functions and dialog boxes, these handlers must also be exported.

Similarly, your program may want to access functions in that class of executables known as dynamic link libraries. Since your program makes contact with

such libraries only at run-time, you must import any such functions by a corresponding statement in the .DEF file to resolve externals at link time. Learning to use libraries is a bit of a black art; in this program, no imports are needed.

The Resource File

A Windows resource file contains objects which are used to provide the user interface. They are precisely those items which must change if a program is, say, rewritten in another language. In fact, Microsoft currently distributes Windows in English, French, Spanish, Italian, German, Swedish, and Dutch. So, it is sensible to keep resources in their own segment and with their own attributes. In addition, the resources can be changed without recompiling and linking the remaining source. It is only necessary to use the RC compiler to remove the old resources and attach the new.

Figure 3 shows the template's resources, which consists only of strings.

Figure 3

```
/*
 * Resource file for the Windows small memory model template
 *
 * Written by Bill Hall, Olivetti ATC
 */

#include 'smltpl.h

STRINGTABLE
BEGIN
    IDS_APPNAME "Smltpl" /* application name */
    IDS_ICON "STPL" /* icon string */
    IDS_TITLE "Windows Small Memory Model Template" /* title bar */
END
```

They are identified by the manifests IDS_XXX, which are themselves defined in the header file, `smi1pl.h`. You will see later how the strings are used in the program.

The Header File

Figure 4 lists the contents of the header file, which is shared among the C sources and the resource file. At the top, we use a little trick to insure that any global variables are declared only in the module where the preprocessor directive `#define EXTERN` appears. In any other module where `'EXTERN'` is not defined, it is automatically converted to `'extern'`, which is a C key word.

Next follows the definitions for the string constant manifests and which are needed in both the C sources and the resources. Following are the three global variables used in the program. The first is a handle to the main window. As Windows is partially object oriented, many items, including instances, tasks, menus, and dialog boxes, are identified by handles, which are simply unsigned, 16-bit quantities.

The two string variables are shared by several routines. One contains the application name and the other the string which is displayed by the program when the program is iconic. Note the naming conventions used. The prefix often tells you the general nature of the object. Thus, `'h'` is handle, `'hWnd'`, indicates a handle to a window, `'sz'` means an ASCII (null-terminated ASCII) string, etc.

Similar techniques are used in naming functions. For example, `'CreateWindow'`, `'RegisterClass'`, and so on. Here, the format is action + object. In Presentation Manager, this naming convention is even more highly developed.

The header file concludes with declarations of two functions needed by both C modules. Note the unusual keyword `PASCAL` on `MainWndProc`, and the naming of `InitProgram` as `FAR`. Even though we are using the small memory model, this function is in a different segment and so can be reached only by an intersegment call. The word `PASCAL` requires a bit more explanation. Normally in C, when a function is called, the variables are pushed on the stack right to left and the caller cleans up the stack upon return. This is fine for a language which can allow a variable number of arguments to a subroutine, but it means that the same `'clean up'` code is generated in every function which calls that subroutine. In the `PASCAL` calling convention, a fixed number of arguments are pushed left to right and the subroutine itself adjusts the stack. All Windows functions are `PASCAL`; in addition, any function in your program which is called by Windows must also use the `PASCAL` convention. As you will see, `MainWndProc` is called by Windows when it has a message, and this explains the declara-

Figure 4

```

/*
 * Header file for the small memory model template
 *
 * Written by Bill Hall, Olivetti ATC
 *
 */

/* This trick insures that globals are declared in only one place */
#if !defined(EXTERN)
#define EXTERN extern
#endif

/* Resource string constants */
#define IDS_APPNAME 100
#define IDS_ICON 101
#define IDS_TITLE 102

/* Global variables */
EXTERN HWND hWndMain; /* main window handle */
EXTERN char szAppName[10]; /* name of the application */
EXTERN char szIcon[5]; /* icon string */

/* Declarations which need to be known in all C source files */
LONG FAR PASCAL MainWndProc(HWND, unsigned, WORD, LONG);
BCOL FAR InitProgram(HANDLE, HANDLE, LPSTR, int);

```

tion.

The Main Code Module

Finally, we can now take a look at some of the code (Figure 5). At the top, we note the inclusion of `windows.h`, a file supplied on the Windows SDK. Here is defined everything needed by a Windows program, such as data structures, type definitions, macros, constants, and all Windows function declarations. You can find many undocumented features of Windows here. And, although it is very large, it is possible to substantially reduce compile time by inhibiting certain definitions. The technique is described at the beginning of the file.

Next, we see the word `EXTERN` defined as a null string followed by the header file. So, the global variables are declared in this module.

`WinMain` is the entry point for a Windows program. `WinMain` receives four parameters. Two are instance handles; the first is to the current instance and the second is to a previous invocation of the program. The instance handles are used in this program primarily during initialization, which is covered in the discussion of the next module. There is also a long pointer to the command line, but this string is not broken into its fields as is `argv` in C. Finally, a parameter, `cmdShow`, indicates how the window is to be initially displayed — iconic, normal, or maximized (zoomed).

`WinMain` is really quite simple in form. After calling `InitProgram`, the program enters into its message-reading loop. This loop remains active until `GetMessage` returns zero. `WinMain` then returns causing the program to exit. Whereas a typical non-Windows program gets input by querying the operating system, a windowing system waits for information to arrive in the form of messages. These are re-

trieved until the application's message queue is empty. Control then passes to another program, and the application remains suspended until a message is again available.

If a message is retrieved, it is translated (an action which converts `WM_KEYDOWN` and `WM_KEYUP` messages to `WM_CHAR` messages and which is analogous to obtaining an ASCII value from a keyboard scan code), and then dispatched to the application's window procedure. Windows then calls the window procedure with the message, which carries out the appropriate task and returns.

Note that if an application does not read messages, no other process can run. Although multitasking, Windows is not preemptive. In addition, the `GetMessage` call does not return until a message is available, and it may appear that no Windows program can carry on lengthy operations in its Windows procedure. In fact, there are methods to maintain tasking using `PeekMessage`. In a future article, we will try to give some examples of this kind of message handling.

Over a hundred different kinds of messages can be sent to a program. Mouse movement, mouse button clicks, keyboard action, windows movement and sizing, requests for painting, etc., all generate messages. Each contains data in the word parameter (`wParam`) and long parameter (`lParam`), whose meanings are unique to the message. Most messages are never specifically acted upon by a program and are simply returned for default processing. In our program, we take special action on only two.

The `WM_DESTROY` message is the signal that the application has received a request to exit. Usually a program performs any final cleanup and calls `Post-`

QuitMessage. A WM_QUIT message will then be received, GetMessage will return zero in the message loop, and the program will exit.

A WM_PAINT message is placed in the message queue whenever the window client area must be drawn for some reason. Windows may send the message because it has information that some other application, dialog box, or menu has destroyed part of the client region. Frequently, the application sends itself a WM_PAINT message as a way to update the screen.

The first action the program takes in response to a WM_PAINT message is to retrieve a paint structure using the BeginPaint function. This is passed to our paint routine where all updating is performed. The EndPaint call terminates a paint message. Even if an application has no local paint function, these two function calls must be executed whenever a paint message is received.

Included in the paint structure passed to MainWndPaint is a handle to a display context, which must be obtained from Windows before any output can be performed. A display context is basically a data structure describing the objects to be used when drawing on the output device. The device itself can be a printer, screen, or even a memory bitmap. A display context has a number of default objects, such as a black pen, white brush, system font, text mapping mode, etc. If the application requests something different, such as a color brush or a different font, these must be selected into the display context.

In this program, we carry out a special operation in MainWndPaint only if the window is iconic. If so, the size of the client rectangle is retrieved and the Rectangle function is used to draw a border with the default pen and fill the interior with the default brush. Then, the text contained in the szIcon string is drawn in the top third of the icon area.

If you have read this far, you should try extending the action of the paint function by displaying something in the client area when the window is not iconic. For example, you may want to try printing a message or drawing some geometric figure. As a slightly greater challenge, see if you can figure out how to do this with a different pen or brush.

The Initialization Module

This is the most complicated part of the program. In the previous module, we discussed the actions required to keep the program running. Here, we perform all the steps needed to get it into operation.

Below the include statements and the local function declarations at the top of Figure 6, you see the first function called when SMLTPL is invoked, InitProgram. The parameters passed are the same as those given to WinMain. If this is

Figure 5

```

/*
 * Resident segment for the Windows small memory model template
 *
 * Written by Bill Hall, Olivetti ATC
 */

#include <windows.h>

/* all global variables are declared in this module */
#define EXTERN
#include 'smltpl.h'

/* local function declaration */
static void NEAR MainWndPaint(HWND hWnd, HDC hDC);

/* Entry point for program */
int PASCAL WinMain(hInstance, hPrevInstance, lpszCmdLine, cmdShow)
HANDLE hInstance, hPrevInstance;
LPSTR lpszCmdLine;
int cmdShow;
{
    MSG msg;

    /* If initialization is not successful then exit */
    if (!InitProgram(hInstance, hPrevInstance, lpszCmdLine, cmdShow))
        return FALSE;

    /* Retrieve messages from Windows */
    while (GetMessage((LPMSG)&msg, NULL, 0, 0)) {
        TranslateMessage((LPMSG)&msg);
        DispatchMessage((LPMSG)&msg);
    }
    return msg.wParam; /* exit program */
}

/* All messages are processed here */
long FAR PASCAL MainWndProc(hWnd, message, wParam, lParam)
HWND hWnd;
unsigned message;
WORD wParam;
LONG lParam;
{
    PAINTSTRUCT ps;

    switch(message) {
        case WM_DESTROY:
            PostQuitMessage(0);
            break;

        case WM_PAINT:
            BeginPaint(hWnd, (LPPAINTSTRUCT)&ps);
            MainWndPaint(hWnd, ps.hdc);
            EndPaint(hWnd, (LPPAINTSTRUCT)&ps);
            break;

        default:
            return ((long)DefWindowProc(hWnd, message, wParam, lParam));
            break;
    }
    return(0L);
}

/*
Caveat programmer: This function is written in the new format.
If your compiler complains, write it as
static void MainWndPaint(hWnd, hDC)
HWND hWnd;
HDC hDC;
{
    ...
}
*/

```

```

static void MainWndPaint(HWND hWnd, HDC hDC)
{
    /* draw the icon */
    if (IsIconic(hWnd)) {
        RECT rIcon;
        GetClientRect(hWnd, (LPRECT)&rIcon);
        Rectangle(hDC, 0,0,rIcon.right, rIcon.bottom);
        TextOut(hDC,2,rIcon.bottom/3,(LPSTR)szIcon,strlen(szIcon));
    }
}

```

Figure 6

```

/*
Initialization segment for the Windows small memory model template.
This code can be discarded after it is used.
*/

#include <windows.h>
#include "smltpl.h"

/* local function declarations */
static BOOL NEAR RegisterWindowClass(HANDLE);
static void NEAR GetPrevInstanceData(HANDLE);
static BOOL NEAR MakeAndShowMainWnd(HANDLE, HANDLE, int);

/* This routine is FAR since it is called from another segment */
BOOL FAR InitProgram(hInstance,hPrevInstance, lpszCmdLine, cmdShow)
HANDLE hInstance, hPrevInstance;
LPSTR lpszCmdLine;
int cmdShow;
{
    /* if this is the first instance of the program ... */
    if (!hPrevInstance) {
        /* register the window */
        if (!RegisterWindowClass(hInstance))
            return FALSE;
        /* get the icon string */
        LoadString(hInstance, IDS_ICON, (LPSTR)szIcon, sizeof(szIcon));
    }

    /* A previous instance already exists so get global data from there */
    else
        GetPrevInstanceData(hPrevInstance);

    /* Create and show the window */
    if (!MakeAndShowMainWnd(hInstance,hPrevInstance, cmdShow))
        return FALSE;

    return TRUE;
}

/* Every window must belong to a class. We register ours here */
static BOOL NEAR RegisterWindowClass(hInstance)
HANDLE hInstance;
{
    PWNDCLASS pWndClass;
    HANDLE hTemp;

    /* Load the name string from resources */
    LoadString(hInstance, IDS_APPNAME, (LPSTR)szAppName, sizeof(szAppName));

    /* allocate space for the WNDCLASS structure and lock it down */
    hTemp = LocalAlloc(LPTR, sizeof(WNDCLASS));
    pWndClass = (PWNDCLASS)LocalLock(hTemp);

    /* fill the structure */
    pWndClass->hCursor = LoadCursor(NULL, IDC_ARROW); /* standard cursor */
    pWndClass->hIcon = NULL; /* no icon */
    pWndClass->lpszMenuName = NULL; /* no class menu */
    pWndClass->lpszClassName = (LPSTR)szAppName; /* our class name */
    pWndClass->hbrBackground = GetStockObject(WHITE_BRUSH); /*white background*/
    pWndClass->hInstance = hInstance; /* instance handle */
    pWndClass->style = CS_VREDRAW | CS_HREDRAW; /* standard redraw values */
    pWndClass->lpfnWndProc = MainWndProc; /* pointer to our window proc */
}

```

the first invocation of our program, then hPrevInstance will be NULL. In this case, we load the string to be displayed when the window is iconic, and register the class of the main window. If this is a later instance, then we simply borrow these strings from the previous instance.

Next, we create and display a window of the class which we have just registered. If any of these actions fail, FALSE is returned to the caller and the program exits. Now, let us study more closely each routine called by the initialization function.

A window class is basically a data structure containing a collection of attributes common to all windows of that class. Included are the mouse cursor style, menu, class name, background brush, class style, and, most important, the window procedure. To register a class, it is only necessary to fill in a WNDCLASS structure and pass it to Windows. Since Windows copies it, we can destroy the structure after RegisterClass returns successfully.

So, we begin by allocating space in the local heap to hold the WNDCLASS structure. Allocations in the heap can be moved, compacted, or locked in position, and Windows manages this area in somewhat the same way as global memory. So, the first step is to obtain a handle to a memory block with LocalAlloc. Then, when it is to be used, it is locked in order to retrieve its pointer. Later, when the memory is to be deallocated, it is unlocked, then freed.

When we use the attribute LPTR in the LocalAlloc call, we insure that the memory retrieved is set to zero and is fixed. Although not documented, the handle returned in this case is the actual pointer, and a call to LocalLock is not really necessary. But, we go through the formalities to illustrate proper techniques for working with the heap.

Now we fill in the WNDCLASS structure for the program. We use a standard arrow cursor. Since we have no bit-mapped icon nor menu, we enter a NULL for each. As the class name should be unique, we use the name of our program. Because we want Windows to maintain the background, we specify a brush (which in this case is white). (If an application wants to paint its own background, then a NULL is entered in the WNDCLASS background brush field, but the application must then be prepared to create its own brush and to respond to the WM_ERASEBKGD message from Windows.) Next, we supply the instance handle, and we specify in the style field that we want the window to be redrawn if it is resized either vertically or horizontally. Finally, we name a long pointer to MainWndProc, which Windows will call when it has a message for the application.

There are two other fields in the


```

/* register the class. if fail, abort */
if (!RegisterClass((LPWNDCLASS)pWndClass))
    return FALSE;

/* free the memory used */
LocalUnlock(hTemp);
LocalFree(hTemp);

/* show success */
return TRUE;
}

/*
If this not the first instance, we can retrieve static data from a
previous invocation of the program
*/
static void NEAR GetPrevInstanceData(hInstance)
HANDLE hInstance;
{
    GetInstanceData(hInstance, (PSTR)szAppName, sizeof(szAppName));
    GetInstanceData(hInstance, (PSTR)szIcon, sizeof(szIcon));
}

/*
Create the window, making sure that its position and size are suitable
for the display.
*/
static BOOL NEAR MakeAndShowMainWnd(hInstance, hPrevInstance, cmdShow)
HANDLE hInstance;
HANDLE hPrevInstance;
int cmdShow;
{
    char szTitle[50];

    LoadString(hInstance, IDS_TITLE, (LPSTR)szTitle, sizeof(szTitle));

    /* create the window, letting it fall where Windows wants */
    hWndMain = CreateWindow((LPSTR)szAppName,
        (LPSTR)szTitle,
        WS_OVERLAPPEDWINDOW,
        CW_USEDEFAULT, 0,
        CW_USEDEFAULT, 0,
        (HWND)NULL,
        (HMENU)NULL,
        (HANDLE)hInstance,
        (LPSTR)NULL);

    /* if we fail, give up */
    if (hWndMain == NULL)
        return FALSE;

    /* finally, display the window and show success */
    ShowWindow(hWndMain, cmdShow);
    UpdateWindow(hWndMain);

    return TRUE;
}

```

structure which we have left at zero, namely the class- and window-extra data. By asking for space here, the programmer can store useful information which is common to the class or unique to each window in the class.

If the registration is successful, we then unlock and free the memory used and return TRUE.

In the registration procedure, we obtained the class name by reading it from the resource segment. If this is a second or later instance of the program, it is not necessary to register the class again. Nevertheless, we still need the application name, as well as the icon string. So, we load this data from a previous instance of

the program.

Finally, we create and show the main window. The first parameter in the CreateWindow call is the class name, which is, of course, the same as we have just registered.

We complete the initialization by creating and displaying a window of the class we just registered. We begin by reading the title bar string from the resource file. Then, CreateWindow is called with several parameters. The first, the class name, is the same as we have just registered. The title bar string is what was just read from resources. The style WS_OVERLAPPEDWINDOW is the default for a top level window, and hence, our win-

dow will have minimize and maximize buttons, a system menu button, will be movable, and will have a thick sizing frame. The position and size of the Window will be determined by Windows, since the pair CW_USEDEFAULT, 0 is specified. However, the programmer can override the default values by specifying the left and top position and the width and height.

The instance handle passed back to Windows is, of course, our current one. Finally, since our window has no parent, no menu, and no extra data, we place NULL handles in these positions. CreateWindow then returns a handle to our Window if successful. If we fail, we simply return FALSE and the program exits. Otherwise, we show the window using the cmdShow parameter we received when the program was started. The final call, UpdateWindow, causes Windows to send us a WM_PAINT message. This will insure that our initial screen is up to date. In particular, if the user started our program with the shift key held down, then the program would begin in an iconic state. But, since UpdateWindow is called, the icon will appear correctly drawn on the screen.

Using the Template as a Basis For Other Programs

We have described the minimum steps that must be taken by most Windows programs. You can use the template to write more interesting programs by adding menus, dialog boxes, and by responding to additional messages in the window procedure. To begin the process, the first step is to convert the template. Suppose you are going to write an application called MYPROG. First, create a directory and copy into it the template files. Then, give the command 'ren smtplt*.myprog*.*'. Next, edit each source file, changing any occurrence of the string 'smtplt' to 'myprog'. You may also want to change some of the comments, the DESCRIPTION string in the .DEF file, and the strings in the .RC file. When done, try to make the file and test it. If everything works, you are now ready to let your imagination go to work and to write a program which actually does something.

References

- [1] M. J. Babcock and W. S. Hall, "A Digital Clock for Microsoft Windows", Programmer's Journal, 6.4, July/August 1988, pp. 40-59.
- [2] W. S. Hall, "A Presentation Manager Template", Programmer's Journal, To appear September/October 1988.
- [3] D. Durant, G. Carlson, and P. Yao, "Programmer's Guide to Windows", Sibex, San Francisco, 1987.
- [4] C. Petzold, "Programming Windows", Microsoft Press, Redmond, WA, 1988.





Jim Buszkiewicz
HUG Managing Editor

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

Z-386/25 Product Announcement

Zenith Data Systems is pleased to introduce the Z-386/25. The Z-386/25 is based on the powerful Intel 25 MHz 80386 microprocessor providing today's ultimate processing capabilities within the Industry Standard Architecture.

Zenith Data Systems' innovative system design pushes the 25 MHz 80386 processor to its fullest potential, offering the power and compatibility demanded by high-end personal computer applications and environments.

Z-386/25 Feature Highlights

- 25 MHz 80386 microprocessor
- Support for 80387 and Weitek co-processor
- 2 Mb standard memory

4MB EMS Memory Card for the SuperSport 286

American Cryptronics has introduced an EMS Memory Card with 4 Megabytes of RAM. The card plugs into the existing memory card connector in the computer just like the 1 Megabyte EMS Memory Card. 4 Megabytes will give the SuperSport 286 the capabilities of other powerful laptops and allow it to run OS/2 applications.

Retail Price — \$2895. Prices are FOB, Costa Mesa, CA, and are subject to change without notice.

Ordering information: Call (714) 540-1174 or FAX (714) 540-1023. Order P/N 7PC-17AC.

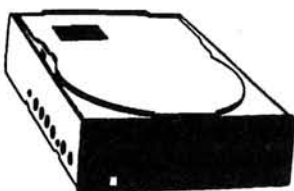
The EMS Card carries a one year warranty and comes with a thirty day unconditional return policy.

Also available: SuperSport 286 — 1 Megabyte EMS Memory. Z-181 — CMOS

RAM Disk.

- 64 K cache memory standard
- Three open 32-bit SuperSet expansion slots
- Two open 16/8 bit expansion slots (on videoless model)
- One 3.5" 1.4 Mb floppy drive
- ESDI hard drives
 - 70 Mb (18 ms)
 - 150 Mb (16 ms)
 - 320 Mb (16 ms)
- I/O interfaces standard
 - Two RS-232C 9-pin AT compatible serial ports
 - One parallel port
- Zenith enhanced 101-key keyboard
- MS-DOS 3.3 PLUS
- MS-WINDOWS/386 (with hard drive models)
- One year carry-in warranty

Winchester Drive Sale!



Complete Hard Disk Upgrade Packages With Dual Warranty!

All models include drive, controller, cabling, instructions, and 90-day replacement warranty in addition to manufacturers' warranties.

PC/XT Upgrades

WIN-XT-25	20Meg, 65ms, MFM, 1/2ht	\$289
WIN-XT-38	30Meg, 65ms, RLL, 1/2ht	\$319
WIN-XT-51	40Meg, 28ms, MFM, 1/2ht	\$495
WIN-XT-96	80Meg, 18ms, MFM, Full ht	\$749

AT/386 Upgrades

WIN-AT-25	20Meg, 64ms, MFM, 1/2ht	\$395
WIN-AT-38	30Meg, 65ms, RLL, 1/2ht	\$449
WIN-AT-51	40Meg, 28ms, MFM, 1/2ht	\$519
WIN-AT-96	80Meg, 18ms, MFM, Full ht	\$795



#16 Algana Drive
St. Peters, MO 63376
Orders: 1-800-TO-BUY-IT
Other Calls: 1-314-447-8697

GUARANTEED COMPATIBLE with Heath/Zenith PC compatibles!

Reader Service #193

Attention Zenith Z-248™ Users!

Convert your 286 to a 386 with our one Board Solution!

The Aox Z-MASTER 386

CALL US FOR OUR GSA SCHEDULE NUMBER!


800-822-0386



Strategic Alliance, Inc.
477 Washington Street (617) 762-7485
Norwood, MA 02062

Since 1979 Aox has been a leading designer of accelerator, coprocessor and compatibility cards providing superior technological solutions.

Reader Service #195



“How Can You Take Advantage of Me”

“... If you don't call? I have everything you could possibly want! My software selection continues to grow, and remains my most popular feature. I'm fast, but, if you don't have the time to download, these software disks can now be purchased for a small copying charge! My message base has also become quite popular. Through it, HUGgies are exchanging more information than ever before. Finally, there's my legendary Bargain Centre. It alone, will make you come back for more! Did you know how inexpensive I am? Why pay \$14 per hour connect time to someone else when your phone company charges less than \$12 per hour (less on weekends) from anywhere within the continental U.S.!

So, go ahead and take advantage of me.

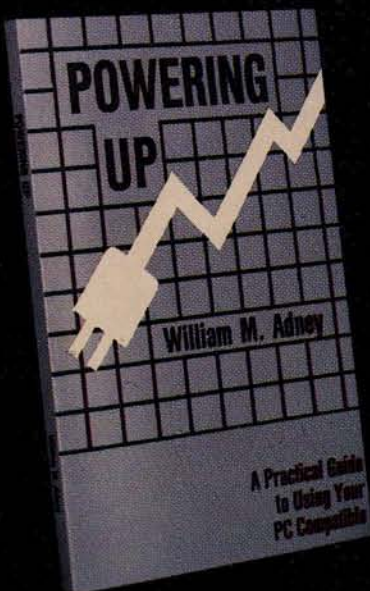
Just set your modem for 300, 1200, or 2400 baud (8N1) and call (616) 982-3956.

You needn't type anything, I'll know you're there!”

MOC

POWERING UP

First, there was "POWERING UP", the series. Now, the long awaited "POWERING UP" book is finally here! Written in a non-technical manner by William M. Adney, this book was prepared especially with the beginning PC-Compatible computer user in mind. Even if you've just purchased your first system, or perhaps have had one for awhile, this book is for you. You can advance to "Power User" status in just a few short readings. The information within is presented in a clear, concise, easy to find manner, making this book attractive as a quick reference guide, to all levels of PC-Compatible users! The following is a list of subject material covered by the individual chapters:



- Setting Up Your Computer System
- Powering Up Your System
- Using File Names and Subdirectories
- The DOS Command Line
- Important DOS Commands You Must Know
- Connecting Peripherals to Your Computer
- Using Batch Files and Config.sys
- Using Input/Output (I/O) Redirection
- Understanding Video Hardware
- Adding More Memory to Your Computer
- How to Select Application Programs
- How to Select Utility Programs
- Selecting and Setting Up a Hard Disk
- Other Useful DOS Commands
- Maintaining Your Computer System

"POWERING UP" is HUG part number 885-4604, and is available for \$12.00 exclusively from the Heath/Zenith Users' Group, and Heath/Zenith Computers and Electronics Centers. Order your copy today by calling HUG at: (616) 982-3463. HUG members can order "POWERING UP" directly from the HUGPBBS Bargain Centre by calling (616) 982-3956 (modem).



P.O. Box 217
Benton Harbor, MI 49022-0217

BULK RATE
U.S. Postage
PAID
Heath Users' Group

POSTMASTER: If undeliverable,
please do not return.

\$2.50
P/N 885-2111