The Official ZENITH/Heath Computer Users Magazine

REMark®

February 1989

A Software Fix
For Caps Lock and Ctrl Keys

See Page 7

Bug Zapping
Problem/Solution Column

See Page 63

Zenith
Leader in Laptops

See Page 47

More on Microsoft Word

See Page 37

"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!"

The Official ZENITH/Heath Computer Users Magazine

REMark®

Volume 10, Issue 2 • February 1989

On The Cover: *The Zenith SupersPort Laptop computer. See page 47 for an article on the fastest growing PCs on the market — the Laptop — by William J. Mitchell of the Detroit Free Press.*

Resources

Reader Service No.		***Page No.***
172	*American Micronics, Inc.*	*66*
104	*FBE Research Co., Inc.*	*45*
191	*First Capitol Computer*	*22*
192	*First Capitol Computer*	*46*
184	*First Capitol Computer*	*39*
105	*First Capitol Computer*	*62*
137	*Jay Gold Software*	*48*
117	*Payload Computer Services*	*16*
130	*Quikdata, Inc.*	*32*
188	*Serendipity Systems*	*48*
173	*Software Applic. of Wichita*	*56*
153	*Surplus Trading Co.*	*22*

*** PC Compatibles**

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

Features

Columns

Series

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..... Rupley's Avertising 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 voluems 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 enahnce 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.

HUG

H8 - H/Z-89/90

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
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 TRANSCEIVER	885-8016	HDOS	AMATEUR RADIO	20.00
MORSE CODE TRANSCEIVER	885-8031-[37]	CPM	AMATEUR RADIO	20.00
PAGE EDITOR	885-1079-[37]	HDOS	UTILITY	25.00
PROGRAMS FOR PRINTERS	885-1082	HDOS	UTILITY	20.00
REMARK VOL 1 ISSUES 1-13	885-4001	N/A	1978 TO DECEMBER 1980	20.00
RUNOFF	885-1025	HDOS	TEXT PROCESSOR	35.00
SCICALC	885-8027	HDOS	UTILITY	20.00
SMALL BUSINESS PACKAGE	885-1071-[37]	HDOS	BUSINESS	75.00
SMALL-C COMPILER	885-1134	HDOS	LANGUAGE	30.00
SOFT SECTOR SUPPORT PACKAGE	885-1127-[37]	HDOS	UTILITY	20.00
STUDENT'S STATISTICS PACKAGE	885-8021	HDOS	EDUCATION	20.00
SUBMIT (Z80 ONLY)	885-8006	HDOS	UTILITY	20.00
TERM & HTOC	885-1207-[37]	CPM	COMMUNICATION & UTILITY	20.00
TINY BASIC COMPILER	885-1132-[37]	HDOS	LANGUAGE	25.00
TINY PASCAL	885-1086-[37]	HDOS	LANGUAGE	20.00
UDUMP	885-8004	HDOS	UTILITY	35.00
UTILITIES	885-1212-[37]	CPM	UTILITY	20.00
UTILITIES BY PS	885-1126	HDOS	UTILITY	20.00
VARIETY PACKAGE	885-1135-[37]	HDOS	UTILITY & GAMES	20.00
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)

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
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

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
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	MSODS	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
SPELL5	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
Z100 WORDSTAR CONNECTION	885-3047-37	MSDOS	UTILITY	20.00
ZBASIC DUNGEONS & DRAGONS	885-3009-37	MSDOS	GAME	20.00
ZBASIC GRAPHIC GAMES	885-3004-37	MSDOS	GAMES	20.00
ZBASIC GAMES	885-3011-37	MSDOS	GAMES	20.00
ZPC II	885-3037-37	MSDOS	PC EMULATOR	60.00
ZPC UPGRADE DISK	885-3042-37	MSDOS	UTILITY	20.00

H/Z-100 and PC Compatibles

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
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	20.00
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

PC Compatibles

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
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
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 Veternary 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. Joe Katz says it was so easy to use, he didn't even need to look at the manual. "It's the only modem software that I use, and I'm in charge of both HUG bulletin boards!" 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.

HUG NEW PRODUCTS

TABLE C
Product Rating

10 – Very Good
9 – Good
8 – Average

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

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

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

ORDERING INFORMATION

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

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

NOTES

When ordering any version of MSDOS software, you must specify what type of media you want the software supplied on. If you want 5-1/4" floppies, add a "-37" to the 7-digit part number. If you want 3-1/2" micro-floppies, add a "-80" to the 7-digit part number.

All special update offers announced in REMark (i.e., ZPC II update) must be paid by check or money order, payable to the Heath Users' Group. **NO CREDIT CARDS ACCEPTED.**

HUG P/N 885-3047[-37] Z-100 WordStar Connection .... $20.00 (For PC WordStar Users, too!)

Note: This is a re-release of the Z-100 WordStar Connection disk first offered in 1988. It now supports release 5.0, as well as release 4.0 of WordStar Professional. If you have already purchased the previous release of this package, you can obtain this new version by sending your original Z-100 WordStar Connection disk and $7.00 to HUG.

The Z-100 WordStar Connection is a set of utilities that allows you to run the PC-compatible version of WordStar Professional 4.0 or 5.0 on a Z-100 (not PC) computer without any PC emulation required. All of the major features of WordStar 5.0 are supported, including Page Preview.

This version of the Z-100 WordStar Connection represents a different approach to the problem of running PC WordStar on a Z-100 computer. The old version applied a considerable amount of patches to WordStar, but this version uses a "parent" program to run WordStar as a child after only a few patches are made. With the new Z-100 WordStar Connection, you can make your WordStar work in almost exactly the same way it does on a PC, and all 40 of the PC function key assignments are supported.

The new Z-100 WordStar Connection adds two features to WordStar that make it even better than it comes out of the box.

The first added feature is character undelete. Characters deleted one at a time using the Backspace key or Delete are stored in a buffer, and they can be restored by typing a special key combination.

The second added feature is dynamic function key prompting. This feature shows you the usage of all 40 function key assignments using only a single screen line. Normally, WordStar shows 20 assignments using two screen lines.

A "parent" program for use on PC-compatibles is provided that adds character undelete and dynamic function key prompting to WordStar on those systems. It also gives you the option of configuring your keypad to work in the same H19 mode that was supported by the old Z-100 WordStar Connection, but without any patches to WordStar itself outside of the normal patch area.

Requirements: To use this package, you need a Z-100 series computer and the Z-100 version of MS-DOS version 2 or above. To run WordStar 4.0, you need at least 256k of memory, or 320k if you want to use Word Finder. To run WordStar 5.0, you need at least 384k of memory. WordStar comes configured for 512k of memory, so it is best to have at least that much installed. If you want to use the Page Preview feature of WordStar 5.0 with this package, your Z-100 must be fully populated with 768k of memory.

This package does not support TelMerge or PC Outline on a Z-100.

Program Author: Patrick Swayne, HUG Software Engineer

The Z-100 WordStar Connection disk contains these files:

README	.DOC	WSPC	.DOC
WSZ100	.DOC	WSPC	.COM
WSZ100	.COM	PC	.PAT
WINSTALL	.COM	WSZ100	.ASM
WSCHANGE	.COM	WINSTALL	.ASM
Z100	.PAT	WSCHANGE	.ASM
COLORS	.PAT	WSLIST	.ASM
KEYS	.PAT	FIXWSL	.ASM
WSLIST	.COM	PRCHANGE	.ASM
FIXWSL	.COM	PDFEDIT	.ASM
PRCHANGE	.COM	LSRFONTS	.ASM
PDFEDIT	.COM	PF	.ASM
LSRFONTS	.COM	PFINST	.ASM
PF	.COM	READ	.ASM
PFINST	.COM	WSPC	.ASM
READ	.COM		

Continued on Page 61

BUGGIN' HUG

Items of Interest

Dear HUG:

A couple of items that I thought might be of interest:

Item 1: In the September 1988 issue of *REMark*, you published my plea for information concerning the upgraded (?) memory address decoding chips for the H/Z-100-PCs that are being offered for sale. The response from other "HUGGIES" was really surprising. Based on the new input, I bought one of them (ZP148 Memory Decoder PAL from FBE Research Company), installed it in my H/Z-148 and am writing this as a report to any others in the Group who might have the same question.

When I press "Ctrl-Alt-Ins", my computer now tells me that I have Memory Size: 704K and when I "force-feed" the memory to capacity there is every indication that the added 64K is being used. Just what I'd wanted. The modification will work so long as there is not an attempt to use it on a computer with an EGA board; they're not compatible! Otherwise, it's a great, inexpensive way to get more working memory.

Installation is a snap. The instructions are quite adequate. It isn't necessary to remove the floppy drives or even to temporarily disconnect any cables.

Item 2: Reference the article on installing a hard disk drive in the H/Z-148 (November 1988 *REMark*): The ZA-148 expansion accessory card is no longer offered by Heath/Zenith. Meanwhile, Premier Technologies, the marketer of the other board appears to be out of business. Fortunately, the PTZ-148, manufactured by Databyte Technology is available through others who advertise in *REMark*. Additionally, FBE Research is offering an alternative card that, allegedly, will accept the hard disk controller card in either slot. This, by permitting use of a large controller card, will give you more flexibility in choice of controllers.

Sincerely,
Ray Isenson
4168 Glenview Drive
Santa Maria, CA 93455

Letter to Ray Isenson

Dear Ray:

I read with great interest your article, "A Hard Disk Drive for your H/Z-148" in the November issue of "REMark", the Heath/Zenith magazine. In your article, you indicated that you could not locate Premier Technologies, the manufacturing company that produced the PTZ-148X accessory expansion card.

Originally, Databyte Technology, Inc. manufactured the card in our California factory and Premier Technologies marketed it. In October, 1987, Premier Technologies got out of the computer business so Databyte took over all responsibilities for the expansion card, thereby insuring uninterrupted product availability and support to our customers. Additionally, we have expanded our offering to two different models: one with a clock, which is the one most people use when adding a hard disk; and a second, lower priced version without the clock that is specifically intended for networks.

In late 1987, Zenith stopped deliveries of their expansion card and began referring customers to us. We market primarily through dealers and most Heath/Zenith stores stock our products.

I hope this information assists you in your pursuit of expanding the capabilities of the H/Z-148. If you need any additional information, please contact our sales office at (714) 968-6099.

Regards,
John Hayashi
V.P. Sales and Marketing
Databyte Technology, Inc.
400 South Date Avenue
Alhambra, CA 91803

Move the Clock

Dear HUG:

I would like to express my appreciation once again for the continued contributions you each have made to our community and special thanks to Mr. Swayne for his efforts in keeping the Z-100 alive.

I have just received HEPCAT, Both Sides and HUGMCP programs which I purchased during the discount days. Thanks. I was happy to see the new version of SCRNCLK on the HEPCAT disk as I had experienced some problems with format and did not realize it was caused by the SCRNCLK program.

I could not get HEPCAT to work with Lotus 123, there may be a way, but I am not smart enough to figure it out. However, I did make a small change in the SCRNCLK.ZSM program to move the clock display over to the left by 10 spaces so that the 123 prompts would be visible. I am sure that feat would be easy for any programmer to accomplish. However, for those of us who know just enough to be dangerous when it comes to programming, I thought I would pass this along to anyone who uses 123 and likes the time displayed.

Only one line of the code need be changed: Under PTIME: 20 lines down in the code find MOV BP,70 and change the 70 to a 60. Reassemble into a .COM program.

Sincerely,
Emory Howell
Rt 11 Box 494A
Tyler, TX 75709

Need Help in Alaska

Dear HUG:

Just purchased an 80 track, double-sided, full-height, 5-1/4" drive. Would like to buy another. Part of the label has been cut off — however, the part number of the drive is SA 460. There is an 'S' on the label — could this mean that it was made by Siemen?

So:

1. Does anyone have such a drive for sale.
2. Would anyone have the address of the Siemen's Company?

Also, I've heard of an EXTENDED ASCII character set. Would like very much to get my hands on a copy of that. It is said to be designed for use by scientists, technological sorts, and the like.

Here, then:

1. Could someone send a photocopy of this character set to me?
2. Would anyone have the address of the folks that established this as a standard.

Thanks a lot!

Cordially,
Mark Hunt, RN
LT, US Public Health Service
USPHS Alaskan Native Hospital
Barrow, AK 99723

Beginners Appreciate Being Recognized

Dear HUG:

Just want to say a quick thank you for running William Adney's series, POWERING UP. Though I have been using a computer for a little while already, I found that most of the articles in HUG were somewhat beyond my comprehension. I recognized much of the information, but had no idea as to what to do with it. I was about to let my subscription lapse when Mr. Adney's series started. I decided to give it another try. While many things he discusses I am already somewhat familiar with, there are many things that are new. Even the things that are a little repetitious to me have been helpful seeing them presented in a clear, coherent manner.

Just wanted to let you know that there were some of us beginners out there and we appreciate being recognized. Thanks to you and to Mr. Adney.

PS. I even venture into the other articles and am finding that I understand more each time.

Sincerely,
Steve Waxler
8200 Indian Hill Road
Cincinnati, OH 45243

The Z-181 and the Logitech C-7

Dear HUG:

I have been a Z-181 user since January of 1987 and have thoroughly enjoyed using this machine, my first computer. For the most part, I use WordPerfect 4.2 and WordPerfect Executive (a very useful package for laptop owners), but I also use other well-known software, including an earlier version of Autocad. There are certainly drawbacks to running this sophisticated CADD software on a monochrome 640k machine with no hard disk, but I've been able to learn quite a bit, and the CGA graphics and backlit LCD screen of the '81 allow use of most of Autocad's features.

This letter is really intended to share my experience with the purchase of a mouse for use with this computer. After reading about the various mouse brands and features in PC Magazine and Info World, I unwittingly purchased a Logitech C-7 Mouse bundled with their Paint software. By all reports this mouse offered the best features and has a good price. Unfortunately, I had not been warned of one very important fact: the Z-181's serial port will not support the Mouse due to inadequate power supply!

I was informed of this fact upon requesting 3.5 inch software from Logitech. After haggling with the customer support and sales people, I was informed that it was my responsibility as a potential purchaser to research the product more fully than I had. There was no mention of this problem in any trade literature that I had seen, and Logitech admits that this is in no way a common problem. I feel that I am a cautious consumer and that some mention of this incompatibility should have been made in one of several articles regarding this product. The seller, of course, offers absolutely no warranty of hardware compatibility.

At this point, I have not approached Zenith with this problem; I'm really not expecting very much at this point. I am open to suggestions outside of tearing down the computer. As the situation stands now, I have a mint condition Mouse and software waiting for a computer to use it with.

Sincerely,
Jon Linton
3709 7th Avenue #4
San Diego, CA 92103

Bug Report

Dear HUG:

I recently received a bug report for my REMIND program that you published in the February '88 issue of REMark. The problem shows up in some cases where the reporting interval begins in one year and ends in the next. In such cases, the date conversion to integer format that is done in the test_dat() function sometimes fails to preserve the chronological order of the dates. the conversion assumes that a 'year' contains 12 months of 31 days each, which results in a 372 day 'year'. However, the year multiplier in the conversion is 365. The fix is to change 365 to 372 in the computations for scode, ecode and tcode in the test_dat() function.

Readers may also be interested in learning that the program has been enhanced to allow the user to condition a reminder report on one or several specific days of the week during the reporting interval. This allows reminders to be displayed on every Friday and Saturday during the interval, for example. I am willing to provide copies of the current version of the program source code and an MS-DOS executable file on disk for $10. I can also write most soft-sector CP/M disk formats, but since there is no standard for reading a system clock under CP/M, I do not see any practical way to provide executable code to CP/M users.

Lawrence W. Conklin
105 Riverglen Road
Liverpool, NY 13090

Glitches

Dear Jim:

It's always a nice, positive event to get an advance copy of REMark. Not least, of course, because I know when that happens I'll find my name in the Table of Contents. I am certainly happy that you published "A Light Tap . . . Without a Sledgehammer" (November 1988). Unfortunately, not quite all of the source code for the program described in the article found its way into the magazine. And, in the middle column on page 55, one character of a pointer name described in the article text is missing; two program statements appear as:

```
++f;        . . . and should be:      ++pf;
++pu;                                 ++pu;
```

Thank you for publishing my article.

Sincerely,
Don Keller
1330 Eden Valley Road
Port Angeles, WA 98362

Elwood's Article in October 1988

Dear HUG:

I was delighted with the October issue of REMark. My Z-100 continues to be a valuable tool in my work developing laser-based instruments for atmospheric measurements, and articles helping maintain and enhance its usefulness keep me subscribing.

In response to Mr. Elwood's article in that issue, I have a Miniscribe SCSI hard disk and CDR host adapter in my low-profile Z-100, along with *both* half-height floppies. The hard disk mounts easily on top of the cover plate of a floppy with a hole in the front plate to check the drive access light. ZPC with a Scottie board makes my machine adequately compatible with our IBM environment here.

Sincerely,
Ronald L. Schwiesow
Research Engineer
National Ctr for Atmospheric Research
Research Aviation Facility
P.O. Box 3000
Boulder, CO 80307-3000

Response to Gary Evens' Letter of 9-88

Dear HUG:

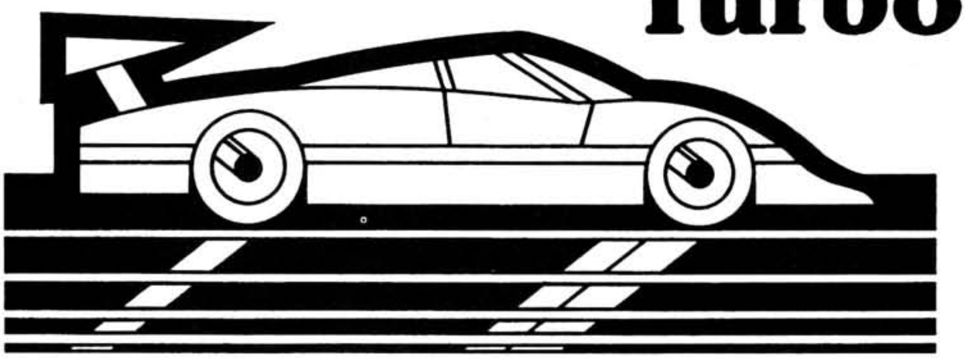
I am writing in response to Mr. Gary R. Evens' letter entitled "WordPerfect V5 Not for the Z-100" that appeared in the September 1988 issue of REMark (Volume 11, Issue 9). Mr. Evens wrote that he has had problems with WordPerfect version 5.0 using a Gemini PC emulator board. I have been using WordPerfect version 5.0 (and before that version 4.2, IBM version) on the Z-100 with ZPC version II. Patching of either version is not necessary. The program works without difficulty.

To use WordPerfect 5.0 on the Z-100, a small virtual drive is helpful. I create one that is 50k ("device=vdisk.sys 50" in config.sys). Use this drive for all the files that WordPerfect creates during operation by entering the switch "d-i" as explained below. These files are erased when WordPerfect is exited. This virtual drive is also a good place to keep the PC.COM and Z100.COM files that allow changing between PC and Z-100 mode under ZPC. Also, "PATH=B:" in the AUTOEXEC.BAT file will allow WordPerfect to find the thesaurus or speller files when needed.

To begin, switch to PC mode number 7. Insert WordPerfect disk 1 in drive A: and enter "WP/d-i:" at the system prompt. This starts WordPerfect. Insert WordPerfect disk 2 when prompted. WordPerfect disk 2 must ALWAYS remain in drive A. Put a data disk in drive B to save text on. WordPerfect is now up and

Continued on Page 64

Turbo Pascal



Matt Elwood
1670 N. Laddie Ct
Beavercreek, OH 45432

Part 4: The CRT Unit

Welcome back again. This is the fourth article in the series that deals with programming in Turbo Pascal. In the last three articles, we've talked about the basic system unit and the DOS unit. In this article, we'll talk about another large unit, the CRT unit. This unit includes procedures and functions that help you take control of the PC's screen, including sound, windows, colors, and other interesting commands. Let's get to it now.

To include the CRT unit in your programs, do exactly like I told you in the last article. To incorporate a unit into the program, use the "uses" keyword followed by the unit name. For example, to include "crt", use:

```
uses crt;
```

Our first subject will be some variables that set special flags. They check for certain things when set, or give you the status of something. Our first variable is "CheckBreak", which is a boolean variable, which if set to True, enables checking for "Ctrl-Break", which when pressed will quit the program. If false, "Ctrl-Break", when pressed will do nothing.

The second variable is "CheckSnow", which enables or disables snow checking. This shouldn't be a program on Zeniths, and should be set to False.

The third variable is "DirectVideo", which enables or disables direct video memory access when writing to the screen. "DirectVideo", when true, cause "Write" and "Writeln" to write directly to video memory. When false, "Write" and "WriteLn" use BIOS calls. This is advanced, and should be set to True.

The fourth variable is "LastMode". Whenever a video mode is changed (we'll cover this later in the article) like Co80 to Bw40, the old video mode is stored in this mode and can be recalled or reset.

There are a few more codes that I haven't covered here because one isn't really important and two are encoded and you need to know how to decode them.

Now we'll get into the procedures defined in the CRT unit. With the CRT unit, you can get faster output by defining a text file called CRT. To do this you can use "AssignCRT". An example of fast screen output is this:

```
program FastCRT;
uses crt;
var fil:text;
a : integer;
begin
AssignCRT(fil);
a:=0;
Rewrite(fil);
while a<20 do begin
WriteLn(fil,'This is fast!');
a:=a+1;
end;
Close(fil);
end.
```

This program first opens the CRT fast text file using "AssignCrt", then "Rewrite"s it to actually open the CRT file. Then using a "while" loop, the program writes the message twenty times to the screen, and finally closes the file. Another good feature of this is you can ask if the user wants to output to the screen or printer and open the file variable as the printer, and then the "WriteLn" would be compatible for both. This is just like the "PRINT #1" in BASIC.

Our next procedure clears the characters from the cursor to the end of the current line. This procedure, "ClrEol" takes no parameters and doesn't move the cursor.

We're now to a very important procedure, "ClrScr". This procedure clears the screen and "homes" the cursor.

Our fourth procedure, "Delay", delays for number of milliseconds contained in the parentheses after the word. An example using "Delay" is:

```
program DelayUse;
uses crt;
var x:integer;
begin
Write ('Enter a value in milliseconds
       for me to wait (1-65530) ');
ReadLn(x);
Delay(x);
end.
```

This program takes an integer input for the amount of milliseconds to wait and passes it on to the "Delay" procedure. "Delay" is used in a procedure I'll cover a little bit later, "Sound".

The next pair of procedures both delete and insert a line. "DelLine" deletes the line the cursor is on (the current line), and moves all the lines below that up. "InsLine" inserts an empty line at the cursor position, and moves all of the text below that one line down. The bottom line is lost, though.

An important procedure that is needed to make "DelLine" and "InsLine" work is the "GotoXY" procedure. This moves the cursor to the X and Y position you specify. The next "Write" or "WriteLn" will start here. An example using "GotoXY" is shown below:

```
program GotoPos;
uses crt;
var X,Y : integer;
begin
WriteLn('Input the X value for the text :');
ReadLn(X);
WriteLn('Input the Y value for the text :');
ReadLn(Y);
ClrScr;
GotoXY(X,Y);
Write(X,',',Y);
end.
```

This short program asks the user for the coordinates on the screen, X having to be in the range one to seventy-nine and Y having to be in the range one to twenty-four for the normal screen. In the special 43 and 50 line modes covered later, the Y can be one to fourty-three or fifty depending on the display mode.

A feature saved from the old Turbo 2 and 3 is the "HighVideo", "LowVideo", and "NormVideo" procedures. In this version, "HighVideo" turns the intensity on, making brighter colors, "LowVideo" turns the intensity off, making darker colors, and finally "NormVideo" resets the intensity bit to what it originally was. In version 3, "HighVideo" and "NormVideo" was the same.

Sound is another procedure included in the CRT unit. To make sound, you must first start the sound with the Sound procedure. There is one parameter to the sound procedure, which is the frequency in Hertz. Then a delay can be used to hold out the sound, and then NoSound shuts the speaker off. Here's an example:

```
program Snd;
uses crt;
var x:integer;
begin
WriteLn
 ('Enter the frequency of the tone : ');
ReadLn(x);
Sound(x);
Delay(1000);
NoSound;
end.
```

The program above will ask you for the frequency of the tone. A good value for this would be in the range of 50 to 10,000. Then the tone would be held out for 1 second.

Now we get to colors. You can set both the foreground colors, using "TextColor", and the background color, using "TextBackground". The colors may be zero through 31, and the background colors may be zero through seven. To change the color of the entire screen, set "TextBackground" to the color you desire and clear the screen. This will destroy the data on the screen, though.

The crt unit contains a nice windowing capability, which allows you to set a window smaller than the normal screen and be able to clear the window, change the background and foreground color, and write to without disturbing the outside screen. Of course, the window is destructive, writing over the part of the screen it occupies. To restore the window to the entire screen, just use "Window(1,1,80,25)" or "Window" with the coordinates of the entire screen. Here's a program demonstrating the use of windows:

```
program WindwUse;
uses crt;
var x,y:integer;
begin
for y:=1 to 23 do begin
for x:=1 to 79 do begin
GotoXY(x,y);Write('#');
end;
end;
Delay(2000);
Window(28,3,40,10);
ClrScr;
WriteLn(' This is inside a window!');
Delay(2000);
Window(1,1,80,25);
ClrScr;
WriteLn
('Window restored to regular boundarys');
end.
```

In the above example, the two "for" loops at the beginning fill the screen with a background of number signs. After a short delay of 2 seconds implemented by the "Delay" command, a window is opened up in the middle of the screen and cleared. You should see a screen full of number signs except for a box in the middle of the screen. At the top left-hand of the box, text will be written. Then after another delay, the windows are restored to the full screen boundarys.

Our last procedure, "TextMode", changes the CRT text mode of the screen. The default mode is "CO80", which is 80 columns by 25 rows in color with a color adaptor, or "Mono", which is 80 columns by 25 rows with the monochrome adaptor. Changing this is simple. To change the text mode, use "TextMode(mode)", with the modes below:

MODE	ADAPTOR	DESCRIPTION
Bw40	Color	Black and white 40x25
Bw80	Color	Black and white 80x25
Mono	Mono	Black and white 80x25
Co40	Color	Color 40x25
Co80	Color	Color 80x25
Font8x8	EGA/VGA	80x43 with EGA, 80x50 with VGA

That is it for the procedures, and now we move on to the four functions included in the crt unit. Our first two functions have to do with the keyboard and trapping keystrokes that are typed. The first of the two, "KeyPressed", checks if a key was pressed. This is useful when used with loops, like shown in the example program below:

```
program CntlKey;
uses crt;
var x:integer;
begin
while not KeyPressed do begin
x:=x+3;
WriteLn(x);
end;
end.
```

The above program will count by threes until any key is pressed (except Ctrl, Alt, Shift, NumLock, Caps Lock, and Scroll Lock) and then quits. This shows how the "KeyPressed" function, which outputs a boolean (true or false) value, can be used in a loop.

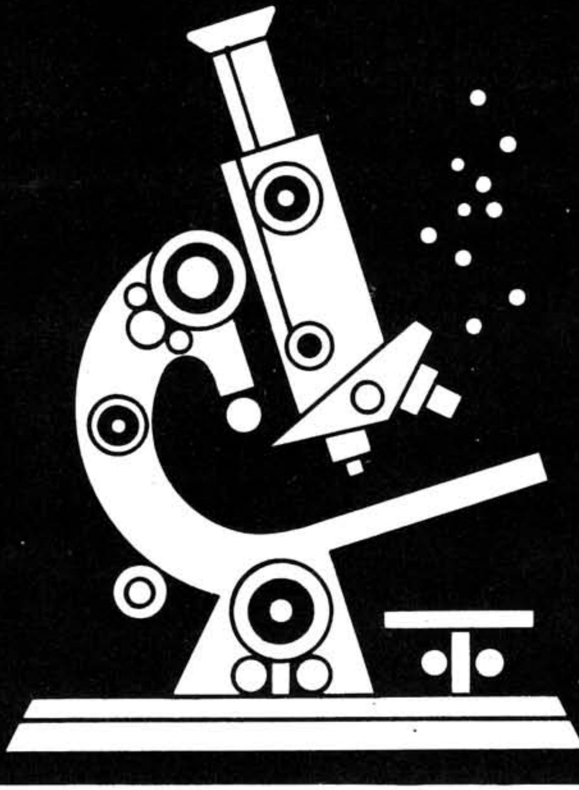
After a key is pressed (except when using "Read" and "ReadLn"), it is kept inside a buffer until it can be read with "ReadKey". "Readkey" outputs one value when a normal key is pressed, and two values (the first of which is a 0, useful for designating function keys from normal keys) are returned. Here's a small example program using "ReadKey":

```
program RdKy;
uses crt;
var x,fk : char;
begin
ReadLn('Press (A,B,C) or Shift and a
                         function key');
while not keypressed do x:=x+1;
x:=Readkey; if x=#0 then fk:=Readkey;
case x of
#65: WriteLn('You pressed the letter A');
#66: WriteLn('You pressed the letter B');
#67: WriteLn('You pressed the letter C');
#0 : case fk of
84..93: WriteLn('You pressed shift and
                              a F-Key');
end;
end;
WriteLn('Done');
end.
```

The above program uses both "Keypressed" and "Readkey". The program first looped around (while not KeyPressed) until you pressed a key. Then the program read in the first value, x, which is the standard ASCII code for the letter and number keys. In the ASCII chart, 65 would be equal to the letter A, 66 would be B, and so on. The next value is only generated if a special key (or combination of keys) is pressed. In this value, numbers 84 through 93 would be equal to the key combination Shift and a function key. The case statement matches the values generated by "Readkey" to the specific values.

The final two functions, WhereX and WhereY, simply contain the cursor X and Y position. WhereX and WhereY output a byte type variable.

That's it for the fourth article about Turbo. Next article we'll cover the most interesting part of Turbo 4.0, the graphics. Turbo 4.0 has an extremely advanced graphics system and I'll cover it all. Until next time, happy programming! ✱



A Chaos Microscope

Hugh Kenner
103 Edgevale Road
Baltimore, MD 21210

Editor's Note: "Read this twice!"

Chaos is in. And expositions of Chaos Theory — James Gleick's *Chaos* (1987), p. 74, or Peitgen and Richter's *The Beauty of Fractals* (1986), p. 25 — show you a graph of values that commence orderly, then go crazy. Here's a Turbo Pascal program to create that graph and let you explore details of it. Amid the craziness you'll find patches of order that resemble (but never replicate) the shape of the whole. Playing with the program will give you a feel for one domain Chaos Theory explores.

First to get a sense of the governing equation, key in this little program.

```
PROGRAM Verhulst;
VAR
   Pop, Rate : REAL;
   i : INTEGER;
BEGIN
   WRITE ('Rate <1.0 .. 4.0> ? ');
   READLN (Rate);
   Pop := 2/3;
   FOR i := 1 TO 300 DO
   BEGIN
      Pop := Rate * Pop * (1 - Pop);
      WRITE (i:12, '.', Pop:10:6);
      IF i MOD 2 = 0 THEN WRITELN;
      IF i MOD 46 = 0 THEN REPEAT UNTIL KeyPressed
   END
END.
```

"Verhulst" asks you for a variable called Rate, in the range 1 .. 4. Next it initializes a second variable, Pop, to 2/3. Then it cycles 300 times through an equation, each time getting a Pop which is first printed, then sucked through the equation again to produce the next Pop. That's the business part of the program; the rest is convenience. The last line stops it when the screen is full while you look for a pattern in the results. When you're ready, a press on any key will trigger the next screenful.

Compile, run, and for your first input try 2.7. Alternate outputs will wobble to and fro:

```
1. 0.600000      2. 0.648000
3. 0.615859      4. 0.638757
5. 0.623016      6. 0.634141 ...
```

But even this early you can see the odd-numbered ones getting bigger, the even-numbered ones smaller. They're inching toward a merger? Sure enough, after the 37th iteration all results are identical: the 6-place value of "Pop" has converged to 0.629630.

Try an input of 2.9, and convergence will be slower: 113 iterations to settle on 0.655172. Try 2.95, and we converge to 0.661017, though still more slowly: 185 iterations.

Be aware, by the way, that in peering through a six-digit window we may think we see true convergence before it's there. By displaying all the digits Turbo can allow us, we'd see churning go on at the right-hand end of the number long after the first six digits are stable. (For 10 digits, 2.9 takes 185 iterations to converge instead of 113). Still, trend-spotting is the way of the human mind, and in equations of this class a trend was long ago spotted. Not so many years ago it was axiomatic that *all non-linear equations, when iterated over and over, eventually settle down to a stable output.*

This is a good time to explain why I called our little program "Verhulst". P.F. Verhulst wrote its central equation in 1845, with population dynamics on his mind. Imagine fish in a pond, reproducing wildly at a fixed annual rate, the population zooming upward as offspring join the game. (You see why our variables are called "Rate" and "Pop".) With infinite food in an infinite pond for an infinite lifespan, each generation's Pop would simply Rate * Pop. onward to infinity, but the world is an imperfect place, where the food supply runs out and fish are mortal. Hence Verhulst's last multiplier, (1 - Pop), which after some generations of overshoot and undershoot makes "Pop" level off at a stable value.

That works between limits. The lower limit is 1. With a growth rate of less than 1, the population dwindles toward extinction. Try 0.9 and watch the fish die out. And 4.0 is the equation's upper limit. Try 4.1 and watch it blow up.

Between those limits, where the population levels off depends *solely on the rate of growth*, not on the number of fish we stocked that tank with. Our program starts with an arbitrary Pop of 2/3 (as it were, a pond 2/3 full), but in the long run the starting value affects nothing save the speed of convergence.

So it looks as though, within a set of finite constraints, a population growing at whatever rate will always stabilize. Right? Wrong.

Run "Verhulst" one more time with an input of 3.1. And what is this? The alternate values are not converging, they are drawing apart. But watch the odd-numbered ones; *those* are converging. And so are the even numbered ones. Sure enough, by the 79th iteration, we have two stabilities, 0.764567 and 0.558014. That means: one year (assuming fish breed annually) our fish pond will be three-quarters full, the next year half-full; and so on, back and forth forever.

Two-value stability obtains between growth rates just over 3 and just under 3.449. (Inputs close to those limits can take *very* long to settle.) Now try 3.49, and, lo, *four* stabilities; by the 25th iteration we have

0.872392　　0.388521　　0.829128　　0.49446

... BOOM, bust, BOOM, bust, the alternate booms larger and smaller, the alternate busts smaller and larger.

Now try 3.553, and quite soon the output will be cycling among *eight* stable values. So you've spotted a trend, periodic doubling? Well, try a 3.8 input and rub your eyes. For we've crossed a threshold, into a domain where numbers simply jump all over the place without a pattern anyone can discern. That would be true no matter how long we went on with the iteration. Welcome to Chaos.

And now it's time to get an overview, which our long listing, "Chaos.Pas", provides. It is written in Generic MS-DOS Turbo Pascal for the Z-100, and to put its graphics point-by-point on the screen it uses the Assembly listings which Randy DeBey publishes in the December '86 *REMark*. By assembling those as per DeBey's instructions, you obtain three compact files which you rename CLS.INC, PS.INC, and POINT.INC. The job of CLS is not just to clear the screen but to enable the 25th line and prepare all three planes of video RAM for writing to. After that PS will turn on a dot of any color at any location, and POINT will report what color any dot presently has. Color is nice, by the way, but the program runs fine in monochrome. (For a PC, you'd omit the three lines with the DeBey EXTERNAL files, and — depending on your graphics board — perhaps jigger the colors.)

With those three .INC files on your working disk, key in the program and compile it. (In the course of filling a screen, CHAOS.PAS makes 175,950 floating-point calculations, so if your machine has an 8087 math chip, Turbo-87 will speed things up by a factor of three.) Now run it, and, for your first try, answer each request for an input with RETURN to use the default values. The pattern you will see in a few minutes should resemble Figure 1.

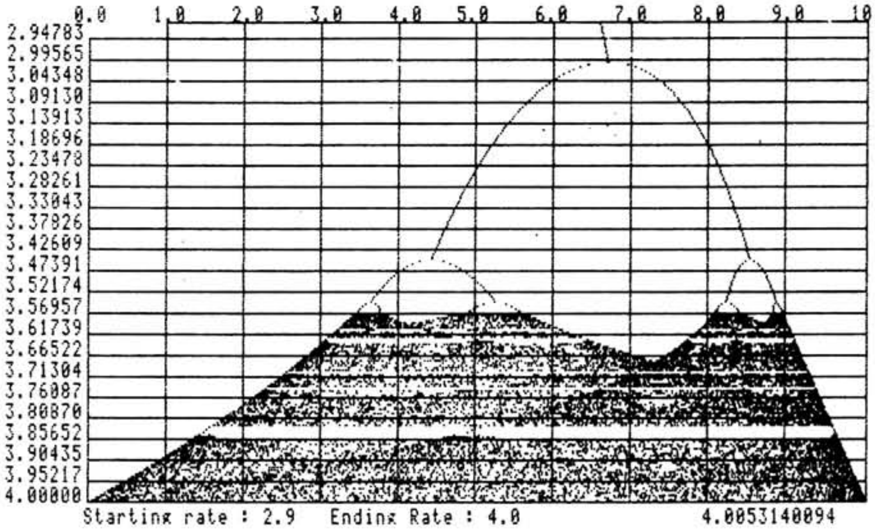


Figure 1

The numbers across the top of the graph are populations, on a scale of 0.0 to 10.0. ("Verhulst" returned 0 to 1.0, but here we've scaled up to get more convenient numbers.) Down the left margin is a range of growth rates, from 2.9 to 4.0. Your experiments with "Verhulst" will help you understand what is happening. At the top, growth rates under 3.0 converge to one stable population; on each scanning line we are turning on a dot 250 times, but it is always the same dot in the same place. After 3.0, we see oscillation between two values. These move further and further apart; then round about 3.45 a second split gives use *four* stable states. (The exact point at which this happens, by the way, is 1 plus the square root of 6, or about 3.449.) Lower down an 8-state cycle appears. Next, doubling (16, 32, 64 ...) steps up, so sensitive to the rightmost digits of Rate values that our screen can no longer represent it. And at about 3.57, chaos suddenly takes over.

It's an odd kind of chaos, looking random through it isn't; every dot on the graph is rigidly determined by the Verhulst equation, so that running the program again will generate the same pattern dot for dot. In effect, the equation is a pseudo-random-number generator that uses the current value of "Rate" for its seed. The dots along any line on the screen plot the pseudo-random numbers it generates during 250 iterations. If we go into the CHAOS.PAS source code and change the constant "Generations" from 250 to something higher, we'll get a denser plot.

But the region of chaos is crossed by one very conspicuous black band, and we may want a closer look at that. Easy. Run the program again, with these inputs: 3.826, 3.86, 1, 10. What you'll see is that black band, but stretched vertically so it fills the screen. Lo and behold, it contains three near-duplicates of the big design — main stem, "doubling" scenario, return to Chaos. The one at the right is so skinny its resemblance to the Big Picture is hardly apparent. Let's zoom in on it: 3.84, 3.85, 9.56, 9.63. There it is, doubling its way to Chaos just like its grandfather. Its subsets, in turn, resemble *it.* Try a closer look at its left-hand branch: use 3.8465, 3.854, 9.55, 9.6.

In these highly detailed blowups we confront a major theme of Chaos Theory, *self-similarity*. Anyone who has played with Fractal patterns is familiar with the way their tiny details tend to resemble the whole from which they're extracted. Likewise, the Verhulst Graph's domain of chaos, the zone from 3.57 down to 4, is punctuated with ever-smaller images of the main graph. Note that the one we've just generated repeats that tell-tale wide dark band at the very bottom. We can try zooming in on it; suitable input values are 3.8535, 3.8545, 9.54, 9.64. Run that. Looks familiar, doesn't it?

The wisps and veils on the bottom half of this emphasize what you may have suspected already, that as pseudo-random-number generators go the Verhulst equation is marked ill-behaved. See how the outputs cluster here, thin out there, exquisitely sensitive to the input seed! If it's true that iterative non-linear equations like this one model natural processes, then nature abounds in faulty pseudo-random-number generators. The well-behaved ones we use in computer programs, scattering output with reliable uniformity, are unnatural in the extreme. So admire the work that went into making them that way.

Do not jump to the conclusion that all black bands contain *three* cascades. The main graph has narrower bands, and they are different. Try 3.737, 3.745, 2, 10. Five cascades this time; to enlarge the one on the left, try 3.737, 3.745, 2.24, 2.3. You'll note that the regions of chaos, from which this pattern emerges and to which it returns, are starting to look sparse; most of the dots our 250 iterations generate are being thrown away to the right of the display.

And there's the *big* black band again, just above 3.7443. For a closer look at that, try 3.744104, 3.744107, 2.242, 2.243. Sure enough, three cascades. But if the left-hand one makes you curious? Alas, there you'll bump into a design limit on the present version of this program. While (whithin the scope of Turbo Pascal's REAL numbers) we can slice the growth rates as thin as we wish, the population range, which sets our left and right boundaries, is restricted to three-place values. 2.242 is accepted, but not 2.2421. To see why, observe the top line of the present graph; there's not room to label a 5-digit subdivision. If you want to attack the source code and extend the range, perhaps suppressing inner labels, feel free. Be prepared for some debugging. Procedure VerBars and its parent Procedure DrawGrid are the program's most finicky parts, mainly because Turbo's ways with REAL numbers force so many dodges. I'll be glad to hear from someone who can simplify.

Finally, some comments on the program listing. The constant "Offset" (line 21) shifts the displayed graph rightward to make room for those 7-character labels. Function "Formatted" (lines 53-69) strips trailing zeroes from a REAL number; that lets lines, such as 210, label the graph neatly; it also lets the conditional at lines 105 and 118 reject entries that exceed three places.

Functions FracPart and Clean (Lines 159, 167), along with other details in Procedure VerBars (174-214), deal with bugs that plagued me till I discovered the pitfalls that inhere in Turbo's representation of REAL number. In particular, a number like 1.4 will be rounded up to 1.4000000000 for display, though what the computer is working with may be 1.3999999999. Then the FRAC function yields not 4 and a string of zeroes, but 3.99999999999-01, which can drive VerBars crazy. FracPart is my homemade improvement on FRAC; it simply converts the number to a short string, then eats off leading digits to the decimal point, then reconverts what's left to a number. That way, FracPart (12.0) is *sure* to be zero, even when the computer really has 11.9999999999 on its mind. Likewise, Function Clean ensures that right-most digits contain no garbage that can make comparisons fail. 3.0000000001 is *not* equal to 3.0000000000.

Pressing any key (line 275) will terminate the program in mid-run. The picture stays on the screen until a second keypress (line 283) performs the definitive exit to Procedure Cleanup, which clears the 25th line, restores the cursor and reverts to a white-on-black screen.

Lines 258-9 cycle the Verhulst equation 600 times, to shake it down before display values start getting extracted. There are critical points where 600 isn't enough; in *The Beauty of Fractals* Peitgen and Richter mention their routine 5,000-cycle shakedown, O.K. on a mainframe but too slow for us. To inspect one problem area, input 3, 3.00004, 6.666, 6.667. Toward the bottom left corner you'll see dots smearing as they oscillate among adjacent values. And those values are all of them wrong, as you'll learn if you change the constant "Shakedown" to 5000 and (after an excruciating wait) see how differently the curve diverges. Not to worry. For most purposes Shakedown = 600 performs fine. It's only near a few forking-points that the equation gets jittery, and the forking-point just below 3.0 is much the worst of all. I contrived the example to take you very near it, as it were near the lip of Niagra; not a place, in your daily affairs, where you're likely to be.

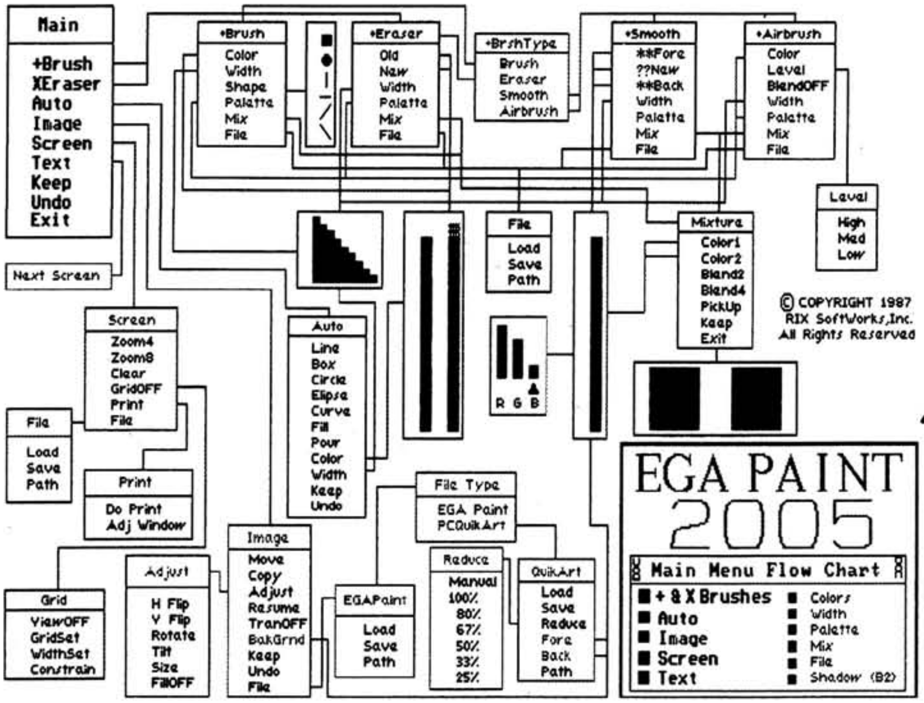
Listing of file "CHAOS.PAS"

```
 1  PROGRAM Chaos;
 2  (* Algorithm from Scientific American, July '87, p. 108;
 3     graphs Verhulst's Equation: growth rate vs. population.
 4     Assembly language "External" modules by Randy DeBey.
 5     Last rev. 2/28/88.  Program asks for beginning and
 6     ending growth rates, left and right indices.  Indices
 7     admit fractional parts to 3 decimal places, to permit
 8     zooming in on small portions of the graph.            *)
 9
10  {$C-}
11
12  TYPE
13     Str20 = STRING[20];
14
15  CONST
16     CharHeight  =   9;    Black    = 0;     Esc    = #27;
17     ScreenDepth = 216;    Blue     = 1;     Hues   = 'm';
18     ScreenWidth = 553;    Magenta  = 3;     Cursor =   5;
19     Generations = 250;    Green    = 4;     Off    = 'x';
20     ShakeDown   = 600;    Cyan     = 5;     On     = 'y';
21     Offset      =  60;    White    = 7;
22
23  VAR
24     i, XVal, YVal  : INTEGER;
25     LeftX, RightX, StartRate, EndRate, Rate, Increment : REAL;
26
27  PROCEDURE Cls; EXTERNAL 'Cls.Inc';
28
29  PROCEDURE Plot (x, y, Color : INTEGER); EXTERNAL 'Ps.Inc';
30
31  FUNCTION Point (x, y, : INTEGER): INTEGER; EXTERNAL 'Point.Inc';
32
33  PROCEDURE SetColor (TextColor : INTEGER);
34  BEGIN
35     WRITE (Esc, Hues, TextColor, Black)
36  END;
37
38  PROCEDURE SetCursor (State : CHAR);
39  BEGIN
40     WRITE (Esc, State, Cursor)
41  END;
42
43  PROCEDURE HorLine (x1, y, x2, Color : INTEGER);
44  BEGIN
45  For i := x1 TO x2 DO Plot (i, y, Color)
46  END; {Procedure HorLine}
47
48  PROCEDURE VerLine (x, y1, y2, Color : INTEGER);
49  BEGIN
50     FOR i := y1 TO y2 DO Plot (x, i, Color)
```

```
END; {Procedure VerLine}

FUNCTION Formatted (x : REAL) : Str20;
VAR
   i : INTEGER;
   OutString : Str20;
BEGIN
   OutString := '';
   STR (x:10:10, OutString);
   i := LENGTH (OutString);
   WHILE OutString [i] = '0' DO
   BEGIN
      DELETE (OutString, i, 1);
      i := PRED (i)
   END;
   IF POS ('.', OutString) = LENGTH (OutString)
     THEN OutString := OutString + '0';
   Formatted := OutString
END; {Function Formatted}

PROCEDURE GetInputs;
VAR
   OK : BOOLEAN
BEGIN
   Cls;                        {Enables access to Screen RAM}
   YVal      := 9;             {Top of graph}
   StartRate := 2.9;           {Default starting rate}
   EndRate   := 4.0;           {Default end rate}
   LeftX     := 0.0;           {Default left limit}
   RightX    := 10.0;          {Default right limit}
   REPEAT
      GoToXY (1, 1);
      WRITE ('Starting growth rate? <RETURN = 2.9>  ');
      CLREOL;
      {$I-} READLN (StartRate) {$I+};
      OK := (IOResult = 0) AND (StartRate >= 1.0) AND (StartRate < 4.0)
   UNTIL OK;
   GoToXY (39, 1);
   WRITE (Formatted (StartRate));
   REPEAT
    GoToXY (1, 3);
      WRITE ('Ending growth rate?   <RETURN = 4.0>  ');
      CLREOL;
      {$I-} READLN (EndRate) {$I+};
      OK := (IOResult = 0) AND (EndRate > StartRate) AND (EndRate <= 4.0)
   UNTIL OK;
   GoToXY (39, 3);
   WRITE (Formatted (EndRate));
   REPEAT
      GoToXY (1, 5);
      WRITE ('Lower population limit?  <RETURN = 0.0>   ');
      CLREOL;
      {$I-} READLN (LeftX) {$I+};
      OK := (IOResult = 0)
        AND (POS ('.', Formatted (LeftX)) + 3
                   >= LENGTH (Formatted (LeftX)))
          AND (LeftX >= 0.0)
            AND (LeftX < 10.0)
    UNTIL OK;
   GoToXY (43, 5);
   WRITE (Formatted (LeftX));
   REPEAT
      GoToXY (1, 7);
      WRITE ('Upper population limit?  <RETURN = 10.0>  ');
      CLREOL;
      {$I-} READLN (RightX) {$I+};
      OK := (IOResult = 0)
        AND (POS ('.', Formatted (RightX)) + 3
                   >= LENGTH (Formatted(RightX)))
          AND (RightX > LeftX)
            AND (RightX <= 10.0)
   UNTIL OK;
   Increment := (EndRate - StartRate) / (ScreenDepth - CharHeight);
   Cls;
   SetCursor (Off);
   SetColor (Cyan);
   GoToXY (8, 25);
   WRITE ('Starting rate : ', Formatted (StartRate));
   WRITE ('':3, 'Ending Rate : ', Formatted (EndRate))
END; {Procedure GetInputs}

PROCEDURE DrawGrid;
VAR
   X, Y, j, k, Mark, CellWidth, LeftCellWidth : INTEGER;
   XDiff, CellCount, Index, IndInc, Try, Fraction : REAL;
   Temp : Str20;

   PROCEDURE EndBars;
   BEGIN
      X := Offset;
      GoToXY ((X DIV 8), 1);
        WRITE (Formatted (LeftX));
        VerLine (X, PRED (2 * CharHeight), Screendepth - 2, Blue);
      X := ScreenWidth + Offset;
      GoToXY ((X DIV 8), 1);
        WRITE (Formatted (RightX));
        VerLine (X, SUCC (CharHeight), ScreenDepth - 2, Blue)
   END; {Procedure EndBars}

   PROCEDURE DrawLeftCellBound;
   BEGIN
      X := Offset + LeftCellWidth;
      Mark := X;
      GoToXY ((X DIV 8), 1);
      WRITE (Formatted (Index));
      VerLine (X, SUCC (CharHeight), ScreenDepth - 2, Blue)
   END; {Procedure DrawLeftCellBound}

   FUNCTION FracPart (Number: REAL) : REAL;
   BEGIN                        {Evades FRAC garbage}
      STR (Number:0:4, Temp);
      WHILE Temp[1] <> '.' DO DELETE (Temp, 1, 1);
      VAL Temp, Number, K);
      FracPart := Number
   END; {Procedure FracPart}

   FUNCTION Clean (Number : REAL) : REAL;
   BEGIN                        {Kills REAL garbage}
      STR (Number:0:4, Temp);
      VAL (Temp, Number, k);
      Clean := Number
   END; {Function Clean}
```

```
  PROCEDURE VerBars;
  BEGIN
     SetColor (Magenta);
     EndBars;
     IF (ROUND (Cellcount) = 1) AND (FracPart (CellCount) = 0.0)
        THEN CellCount := 10.0;
     CellWidth := ROUND (ScreenWidth / CellCount);
     IF XDiff >= 0.1
       THEN LeftCellWidth := ROUND (CellWidth
                             * (1.0 - (FracPart (LeftX) * 10)));
     IF XDiff > 1
       THEN LeftCellWidth := Round (CellWidth
                             * (1.0 - FracPart (LeftX)))
     ELSE LeftCellWidth := CellWidth;
     j : 10000;
     WHILE j >= 1 DO
     BEGIN
        Try := Clean (1/j);
        IF (XDiff) > (Try) THEN
           Index := LeftX - (FracPart (LeftX *j) * Try) + Try;
        j := j DIV 10
     END;
     IF LeftCellWidth <;ScreenWidth THEN DrawLeftCellBound;
     WHILE (Index > 0) AND (Index < RightX) DO
     BEGIN
        X := X + CellWidth;
        GoToXY ((X DIV 8), 1);
        j := 10000;
        WHILE j >= 1 DO
        BEGIN
           IF XDiff > Clean (1/j) THEN IndInc := Clean (1/j);
           j := j DIV 10
        END;
        Index := Index + IndInc;
        IF X < (ScreenWidth + Offset - 5) THEN
        BEGIN
           WRITE (Formatted (Index));
           VerLine (X, SUCC (CharHeight), ScreenDepth - 2, Blue)
        END
     END
  END; {Procedure VerBars}

  PROCEDURE HorBars;
  BEGIN
     Y := CharHeight;
     IF LeftCellWidth < ScreenWidth THEN
       HorLine (Mark, Y, ScreenWidth + Offset, Blue);
     Rate := StartRate;
     REPEAT
        Y := Y + CharHeight;
        Rate := Rate + (Increment * CharHeight);
        GoToXY (1, Y DIV CharHeight);
        WRITE (Rate:0:5);
        HorLine (Offset, Y-2, ScreenWidth + Offsett, Blue)
     UNTIL (Y >= ScreenDepth) OR (Rate >= 4.0)
   END; {Procedure HorBars}

BEGIN {Procedure DrawGrid}
   XDiff := Clean (RightX - LeftX);
   j := 1000;
   WHILE j >= 1 DO
   BEGIN
      IF XDiff >= Clean (1/j) THEN CellCount := XDiff * j;
      j := j DIV 10
   END;
   VerBars;
   HorBars
END; {Procedure DrawGrid}

PROCEDURE DrawGraph;
VAR
   FullWidth : INTEGER;
   Pop, EffWidth, Scaler : REAL;
BEGIN
   Rate := StartRate;
   FullWidth := ScreenWidth + Offset;
   EffWidth := Screenwidth * 10 / (RightX - LeftX);
   Scaler := LeftX / 10;
   Pop := 2/3;                                    {Initial Value}
   SetColor (Cyan);
   REPEAT
      GoToXY (63, 25);
      CLREOL;
      WRITE (Formatted (Rate));           {Display current rate}
      FOR i := 1 TO ShakeDown DO
         Pop := Rate * Pop * (1 - Pop);  {Shake function down}
      FOR i := 1 TO Generations DO
      BEGIN {For Loop}
         Pop := Rate * Pop * (1 - Pop);      {Get a population}
         IF (Pop * 10 < RightX)                      {In range?}
         AND (Pop * 10 > LeftX) THEN                  {Plot it}
         IF XVal > 11 THEN
         BEGIN {It in range}
            xVAL := ROUND (EffWidth * (Pop - Scaler)) + Offset
            IF XVal < Fullwidth
              THEN IF Point (XVal, YVal - 2) = Black
                THEN Plot (XVal, YVal - 2, Green)
         END {It in range}
      END; {For Loop}
      Rate := Rate + Increment;           {Increase growth rate}
      YVal := SUCC (YVal)                     {Move to next row}
   UNTIL (Rate >= 4.0) OR (YVAL > ScreenDepth) OR (KeyPressed)
END; {Procedure DrawGraph}

PROCEDURE CleanUp;
VAR
   Ch : CHAR;
BEGIN
   READ (Kbd, Ch);                        {Clear buffer}
   REPEAT UNTIL KeyPressed;               {Hold picture}
   GoToXY (1, 25);
   CLREOL;
   GoToXY (1, 1);
   SetColor (White);
   SetCursor (On)
END; {Procedure Cleanup}

BEGIN {Main}
   GetInputs;
   DrawGrid;
   DrawGraph;
   Cleanup
END.
```

*



EGAPaint 2005

A Product Review

Fred Kent
1057 Lake Road
Conneaut, OH 44030

As a treat to my retirement days from the practice of dentistry, I recently gave my trusty Z-100 to my grandson (after first removing the new 20MG hard drive) and bought myself the new Zenith state of the art Z-386 with hi-resolution screen. After installing the hard drive into this new cadillac of computers, I was ready to reload all old programs from the Z-100 and update any that would not run in this new model.

After a number of crashes and other exasperations along with a number of consults with the able people at the Cleveland store, they sent me an updated ROM chip which seemed to solve nearly all the difficulties and the machine proceeded to delight me with dazzling speed etc. A sort of 4000 items in nine seconds for example.

The 386 features a setup program called with Ctrl-Alt-Ins which should only be needed once in a great while. However, I have a nine year old granddaughter who has managed to teach herself how to run the setup to set the selectable speed to slow. This, so she can keep up with several games I have in the computer for her benefit mostly. Unfortunately, slow to the 386 is megahertz so one has to be quite nimble to keep up with the action. Most computer real time games were apparently timed to older 4.77 megahertz clock cycles. Of course this kid never replaces the slow order, so I wind up having to interrupt my first program of the day to reset the high speed cycle and reboot.

My Z-100 had been upgraded with a UCI board which was an excellent piece of hardware making the Z capable of CPM, Z-DOS or MSDOS capable. One of the peculiarities of the UCI was that it could not recognize the serial ports, so I had become accustomed when loading modem programs and drawing programs to using the Z-100 mode. The program called Palette by Software Wizardry served very well as a drawing program in the Z-100 as did Prodriver for a nice modem program. I would download anything I wanted to the Z-100 and place it on a floppy. Then I would reboot as an IBM and copy the program onto the IBM hard drive.

However, Palette is not IBM compatible and my brand new 386 never heard of Z-DOS, so I started a search for a drawing program. I must have downloaded ten drawing programs from the HUG bulletin board alone, all of which were quite unsatisfactory to one who is accustomed to Palette. None of them could handle the many colors available in high resolution. The 386 can select 640 by 480 screen resolution in 16 colors. The latest GWBASIC has screen 7, 8, 9, 10 and 11 available for various purposes as well as screen 0,1 and 2 which were the old IBM standard.

The Cleveland store is long on hardware sales but short on IBM compatible software. The boys there did say to look for PCPaint as a decent drawing program, however they did not have it. Oddly at Christmastime when I bought the 386, the store had 15 or 20 various Zenith computers on the floor and stacked to the ceiling were about 150 Apple computers in their packing cases, so many in fact, that one could not see through the storefront window. Peculiar for a Heath/Zenith store don't you think? As luck would have it, I just returned from a two week visit with my son Fred, a purchasing manager for Hughes Aircraft in Los Angeles. While visiting, he mentioned that he occasionally bought software from a store called Egghead Software and furnished his car and directions to find the place one day while he was working.

Egghead Software turned out to be one of a chain of such stores out west specializing in IBM and Apple compatible software only. Not only did they have PCPaint, but upon inquiry as to the type of computer I was using, they recommended EGAPaint by RIX Software and were kind enough to give me a thorough demonstration. In ten minutes I was convinced that I had come across a program far in excess of Palette and bought the software for $79.95 along with a Microsoft optical mouse with three buttons. Then I had to wait another week to get a plane back home where I could try out this rather marvelous program.

Which Brings Us to the Discussion of EGAPaint

EGAPaint is a mouse-driven full color hi-resolution drawing and picture program, although a number of lesser resolutions are available when one runs the EGASetup file. Also accompanying are EGAPrint for the printer setup, EGAShow and EGASlide for making shows similar to photographic slide shows. It comes on two 360K floppies or as an option, on 3-1/2 inch disks.

Minimum hardware requirements are:
an IBM-compatible computer,
384K RAM,
EGA card,
either monochrome card

or color card
or enhanced color display

Recommended are:
mouse & MS compatible driver,
enhanced color display,
fixed disk drive

EGAPaint makes very clever use of little popup menus which appear each time the right-hand mouse button is depressed. The left and middle buttons work in harmony for toggling various drawing tools into or out of action and sizing of objects. The brush is always just under the cursor wherever it may apear on screen. Selecting brush leads to one of five or six popup submenus for width of brush, color, shape and various color mixtures and palettes. Selecting color for instance gives 32 colors to select from, half of which are composite colors and of course selecting palette or mix offers various other mixtures. A little experimenting can produce a nearly infinite color spectrum for use.

Eraser mode from the main menu is unique in that one can elect to not only erase an individual color but replace the erased portions with another color selection.

Auto mode makes provision for lines, boxes, circles, ellipses and curves. Depression of the middle mouse button after selecting one of these options then allows one to adjust the size or length of the object from as little as one pixel to the edge of the full screen. A simple or compound curve can be generated by fixing the two end points as a straight line and then bending that line on screen into a curve tangential to the two fixed end points and the position of the cursor during adjustment. One can bend a line into a curve and with the click of the mouse button, jump the other end and bend that end into a compound french curve in any direction. Also on this submenus is a full mode to fill any desired enclosure on screen with a selected color. The color will proceed to replace the color under the cursor on initiation with the new color which expands to fill the enclosure until the new color encounters any other existing color. Lord help you if your enclosure has a leak. A leak of even one pixel will allow the fill to obliterate the entire screen. Fortunately, there is a keep and an undo function on all submenus. Frequent use of the keep as one is satisfied with his progress will fix the picture development at that point. If a booboo occurs later, just select undue and the errors will be removed back to the last keep.

The next selection on main menu is image which has such selections as move, copy, adjust, resume and transparent available. Move invites you to surround the desired area with an adjustable square box after which, with the flick of a button, you can drag it anywhere on the screen with mouse movements. Copy starts the same way but leaves the original in place. Multiple copies can be laid down anywhere on the screen.

Adjust allows selection of horizontal flip, vertical flip, tilt, rotate and size. Again one surrounds the area to be adjusted with a box to delineate the area and then proceeds with adjustment. Rotation of an image up to one half screen size is possible through 360 degrees. This takes a little time as the math to accomplish this within the program is colossal. I have seen rotations take ten seconds even on my very high speed 386. Once the adjustment is complete, the object can then be pulled from its original spot with the mouse to any other location for deposit, leaving the original in place.

Size starts the same way but as you expand or contract the box with the mouse, the object can be made much smaller or larger as desired. Tilt will distort an object either left or right while maintaining height and width.

Within the image menu you can export and import small images if you have them on disk from other files such as PCQuikart or EGAPaint image files. Available image names are displayed on a suitable menu for selection if one selects file type. Images can be reduced in size for disk storage to 67%, 80%, 50%, 33%, or 25%.

The next selection from the main menu is screen which contains about 11 options. Zoom 4 and zoom 8 will magnify a selected area four or eight times and allow placement or alteration of an image on a pixel by pixel basis. Zoom is the choice for those color fills that bleed into undesired areas. The magnification allows one to repair the leaks which can be as small as one pixel by placing a selected color at any precise pixel location.

Clear will blank the screen to prepare for a new picture. Grid will place a dotted grid of 10 to 80 pixels on screen as a transparent overlay to aid in precise locating of objects. File will allow loading and/or saving of picture files and print will induce your printer to give you a hard copy of whatever is on your screen at the moment.

Next from the main menu is text. One loads the text buffer with the print characters or words desired and then selects font which offers a choice of some 35 various fonts in differing sizes. Next select size to set the size of each letter, style which offers underline, reverse color, proportional spacing and italics, any or all of which may be active. Spacing of letters is selected by space selection and finally three choices of weight which are normal, bold and bolder. Finally, the desired color is selected after which the characters may be trial displayed anywhere on the screen. They won't be laid permanently until a mouse button is depressed so one can try several locations if desired without penalty.

RIX Software has thoughtfully included several flowcharts on screen to illustrate the flow of their program. I am including the main menu flowchart at this writing to aid in your understanding of this program. After printing it out, I noticed in fine print that the picture itself is copyrighted and am, therefore, sending a quick letter to RIX asking their authority to reproduce it for this article. I believe they will agree, however if you do not see a flowchart in this article you will know that RIX would not allow it.

Bear in mind that while there are even more popups than show in this primary flowchart, that there is never more than one of them present on the screen at any given time. Also, all popups are removed during any drawing or filling sequences so that they will not obscure the work at hand. This has one more advantage over programs which have icons around their borders in that the full width and height of the screen is available for larger drawings since the icon displays necessary use of a considerable portion of the available pixels.

The above represents a sampling of the many and varied options available in EGAPaint but by no means have all of them been mentioned. It is very difficult to adequately describe the advances that have been made with some of these new programs. The only way I can think of to experience the thrill of modern computer drawing and design is to actually work with programs like EGAPaint and AutoCad and other such programs. It is awesome to watch and imagine the prodigious math going on inside these little boxes we call computers, especially the newer high speed models with nearly unlimited RAM memory capabilities. Ten years ago my H-11 had a mighty 16K of RAM available but it did a lot of useful work given enough time. My first large scale sort of some 1400 items took this machine 18 hours to complete. Now I am doing the same work in ten seconds with a machine that costs fewer dollars than the H-11 costs and in 32 colors or more instead of one color to boot.

I have worked EGAPaint steadily for a week now some ten hours or more a day and have failed to find any significant bugs. It is not software protected but does carry the usual admonishments about software piracy. I believe it is battle-ready and am enthusiastic about recommending it to those who have the hardware to take advantage of it. It would appear to rival the MacIntosh which I always thought was perhaps the most user friendly machine available and of course it can handle the color which I have not seen the MacIntosh do. With screen reso-

Continued on Page 25

*** 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/calander. $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 768K. 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 or 768K 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. EasyI/O-1, One Serial Port $91.00. EasyI/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 compatable 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, Keytronics 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/calander, 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 RRL hard disk drive $1126.00

Example #3
IBM AT Compatible 80286 12 Meg CPU, 2 Meg RAM, clock/calander,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

MasterCard

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

VISA

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.

Circle #117 on Reader Service Card

A Software Fix for the Misplaced Caps Lock and Ctrl Keys

Pat Swayne
HUG Software Engineer

It seems that every time IBM comes out with a new keyboard design (which most of the clone makers will eventually copy), they insist on putting some of the keys in the wrong place. In the past, Zenith was brave enough to fix the mistakes, as when they put a well-placed L-shaped Return key on the Z-151 keyboard, while IBM's Return key was so far away from the "home" keys that you had to have a concert pianist's fingers to reach it. When IBM came out with their new 101-key keyboard, they did a better job on the Return key, but they must have thought that it would be cute to have two Ctrl keys, and to place them symmetrically on the keyboard. So they put the Caps Lock key where the Ctrl key should be, and put two Ctrl keys on the bottom row. This time, unfortunately, Zenith went along with that change.

In the November 1987 issue of REMark, Jim Buszkiewicz presented a hardware method of swapping the positions of the Caps Lock and left Ctrl keys, so that Ctrl was in its "proper" location. His modification is probably the best solution to the problem, but it involves removing two key switches, cutting some traces, and installing jumper wires, and not everyone is willing or able to make such modifications to computer hardware. I have heard rumors of a chip replacement from Zenith that can swap the keys, but I do not have the part number of such a chip, if it exists, and it certainly isn't listed in the Heathkit catalog.

There is another solution to the misplaced key problem, which involves no hardware modifications at all. When the new 101-key keyboard was introduced by IBM, they also introduced additional support for keyboards in the ROM BIOS (the Basic Input/Output System that controls the computer's peripherals). This new ROM support, which, thankfully, was added by Zenith to their systems, makes it possible to alter the codes produced by keys as they are pressed. It is therefore possible to write a program that swaps the codes produced by the left Ctrl and Caps Lock keys, and I have written a program to do just that.

My key swapping program is called CAPCON.COM, and you can generate a copy of it by typing in and running this BASIC program:

```
10 PRINT "CREATING CAPCON.COM
20 OPEN "O",1,"CAPCON.COM":L=100
30 FOR I=1 TO 17 :C=0:FOR J=1 TO 11
40 READ B:C=C+B:PRINT #1,CHR$(B);:NEXT J:READ S
50 IF S<>C THEN PRINT "TYPING ERROR IN LINE";L:STOP
60 L=L+10:NEXT I:CLOSE #1:SYSTEM
100 DATA 235,91,67,65,80,67,79,78,32,49,46,889
110 DATA 48,1,0,0,0,0,0,128,252,79,508
120 DATA 116,5,46,255,46,13,1,46,128,62,12,730
130 DATA 1,1,117,55,60,224,117,8,46,198,6,833
140 DATA 18,1,1,235,43,80,36,128,46,162,17,767
150 DATA 1,88,36,127,60,29,117,12,46,128,62,706
160 DATA 18,1,1,116,10,176,58,235,6,60,58,739
170 DATA 117,2,176,29,46,10,6,17,1,46,198,648
180 DATA 6,18,1,0,207,180,82,205,33,38,139,909
190 DATA 71,254,142,216,161,1,0,140,202,59,194,1440
200 DATA 115,47,142,192,190,2,1,139,254,30,14,1126
210 DATA 31,185,5,0,243,167,116,13,31,161,3,955
220 DATA 0,64,140,219,3,195,142,216,235,217,50,1481
230 DATA 192,128,62,93,0,68,116,2,254,192,38,1145
240 DATA 162,12,1,205,32,14,31,184,21,53,205,920
250 DATA 33,137,30,13,1,140,6,15,1,184,21,581
260 DATA 37,186,19,1,205,33,186,93,1,205,39,1005
```

To use CAPCON.COM, just enter

```
CAPCON
```

at the system prompt, and press Return. CAPCON will load itself into memory and remain there, and the system prompt will be redisplayed. To verify that it is working, press the left Ctrl key, and the light on the Caps Lock key should change state (go out if it was on or vice versa). Hold down the Caps Lock key and type A, and ^A, representing Control-A, should appear at the system prompt. Use Backspace to delete the Control-A after the test.

Even though CAPCON can swap the functions of the Caps Lock and Ctrl keys, it does not alter which keys click when they are pressed. So the Caps Lock key will still click when you press it while it is being used as a Ctrl key, and the Ctrl key will not click while it is serving as the Caps Lock key. Except for the failure to swap the click, the swapping action of CAPCON is just about as good as a hardware swap. For example, if you want to reset your computer while CAPCON is active by pressing Ctrl-Alt-Del, and you want to use the left Ctrl key, you must press the mapped Ctrl key, not the original one.

There may be a few programs that access the keyboard hardware interrupt directly for determining the Caps Lock state, etc. If you ran such a program, the keys would revert to their original state while the program was running. However, I have not found a program like that. Even Perks and SideKick (an old version), two memory resident programs that use the keyboard interrupt, recognized the mapped keys.

To disable CAPCON and return the keyboard to its original state, type

```
CAPCON D
```

at the system prompt, and press Return. The Caps Lock and Ctrl keys will now function as they are marked. To re-enable CAPCON, type CAPCON and press Return. You can disable and enable it as often as you want to. It loads itself into memory the first time it is run, and every time thereafter it just sets or clears a flag, depending on whether you enter D on the command line or not.

If you have already performed Jim's modification to your keyboard, you may still find CAPCON useful, because while it is active, it will restore your keyboard to its original condition. That is, the key to the left of the A key will perform the Caps Lock function, and the key below the left Shift will perform the Ctrl function. If someone wants to use your computer who, for some strange reason, prefers the keys in the wrong place, you will be able to accommodate him using CAPCON.

If you want a copy of CAPCON without having to type in a listing from this article, get HUG disk 885-3052. On it you will find an expanded version of CAPCON which has the ability to remove itself from memory in addition to the capabilities of the program in this article.

Below is the assembly source code for CAPCON.COM.

```
        PAGE ,132
;       CAPCON -- A SOFTWARE FIX FOR THE MISPLACED
;       CAPS LOCK AND CONTROL KEYS.  TO USE THIS PROGRAM,
;       ENTER
;
;       CAPCON          TO INSTALL OR ENABLE IT
;       CAPCON D        TO DIS-ABLE IT
;
;       BY P. SWAYNE, HUG SOFTWARE ENGINEER  18-AUG-88
;       COPYRIGHT (C) HEATH USERS  GROUP 1988.
;       ALL RIGHTS RESERVED.

CODE    SEGMENT
        ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE

;       DEFINE SOME MEMORY LOCATIONS

        ORG     1
PSPSEG  LABEL   WORD        ;PROGRAM SEGMENT PREFIX ADDRESS
        ORG     3
MCBSIZE LABEL   WORD            ;MEMORY CONTROL BLOCK SIZE
        ORG     5DH
FCBARG  LABEL   BYTE            ;FCB ARGUMENT
        ORG     100H

START:  JMP     SHORT SETUP     ;GO SET UP PROGRAM

SIG     DB      'CAPCON 1.Ø'    ;PROGRAM SIGNATURE

ENFLG   DB      1               ;ENABLE FLAG
INT15V  DW      Ø,Ø              ;INT 15H VECTOR
KEYHB   DB      Ø               ;KEY HIGH BIT
EØFLG   DB      Ø               ;KEYCODE EØ FLAG

;       INT 15H PROCESSOR

INT15:  CMP     AH,4FH          ;IS THIS A KEY PRESS?
        JZ      GOTKEY          ;YES
        JMP     CS:DWORD PTR INT15V     ;ELSE, GO ON
GOTKEY: CMP     CS:ENFLG,1      ;CAPCON ENABLED?
        JNZ     INT15X          ;IF NOT, EXIT
        CMP     AL,ØEØH         ;WAS CODE EØ?
        JNZ     NOTEØ           ;NO
        MOV     CS:EØFLG,1      ;ELSE, MARK IT
        JMP     SHORT INT15X    ;AND EXIT
NOTEØ:  PUSH    AX              ;SAVE KEY CODE
        AND     AL,8ØH          ;ISOLATE HIGH BIT
        MOV     CS:KEYHB,AL     ;SAVE IT
        POP     AX              ;GET KEY CODE
        AND     AL,7FH          ;STRIP HIGH BIT
        CMP     AL,1DH          ;CTRL KEY?
        JNZ     NOTCTL          ;NO
        CMP     CS:EØFLG,1      ;RIGHT CTRL KEY PRESSED?
        JZ      FIXHB           ;IF SO, DON T ALTER IT
        MOV     AL,3AH          ;REPLACE WITH CAPS LOCK
```

```
        JMP     SHORT FIXHB     ;AND FIX HIGH BIT
NOTCTL: CMP     AL,3AH          ;CAPS LOCK KEY?
        JNZ     FIXHB          ;NO
        MOV     AL,1DH          ;REPLACE WITH CTRL
FIXHB:  OR      AL,CS:KEYHB     ;RESTORE HIGH BIT
        MOV     CS:EØFLG,Ø      ;CLEAR EØ FLAG
INT15X: IRET                    ;RETURN WITH FIXED KEY
ENDRES:                         ;END OF RESIDENT CODE

;       INSTALL CAPCON IN MEMORY OR DISABLE/ENABLE IT

SETUP:  MOV     AH,52H
        INT     21H             ;GET MCB ADDR.
        MOV     AX,ES:[BX-2]    ;GET FIRST MCB SEGMENT
        MOV     DS,AX           ;POINT TO IT
FNDLP:  MOV     AX,PSPSEG       ;GET PSP SEGMENT
        MOV     DX,CS
        CMP     AX,DX           ;IN THIS SEGMENT?
        JAE     NOTFND          ;IF SO, CAPCON NOT FOUND
        MOV     ES,AX           ;ELSE, PSP SEGMENT TO ES
        MOV     SI,OFFSET SIG
        MOV     DI,SI
        PUSH    DS              ;SAVE MCB SEGMENT
        PUSH    CS
        POP     DS               ;PUT DS HERE
        MOV     CX,5            ;5 WORDS IN SIGNATURE
        REPZ    CMPSW           ;IS CAPCON HERE?
        JZ      GOTCC           ;WE FOUND CAPCON
        POP     DS              ;RESTORE MCB SEGMENT
        MOV     AX,MCBSIZE      ;GET MCB SIZE
        INC     AX              ;CORRECT IT
        MOV     BX,DS
        ADD     AX,BX           ;CALCULATE NEXT MCB SEGMENT
        MOV     DS,AX           ;POINT TO IT
        JMP     FNDLP           ;TRY AGAIN
GOTCC:  XOR     AL,AL           ;ASSUME DISABLE WANTED
        CMP      FCBARG,'D'      ;DISABLE CAPCON?
        JZ      SETFLG           ;YES
        INC     AL              ;ELSE, ENABLE CAPCON
SETFLG: MOV     ES:ENFLG,AL     ;SET CONDITION
        INT     2ØH
NOTFND: PUSH    CS
        POP     DS              ;FIX DS
        MOV     AX,3515H
        INT     21H             ;GET INT 15H VECTOR
        MOV     INT15V,BX       ;SAVE IT
        MOV     INT15V+2,ES
        MOV     AX,2515H
        MOV     DX,OFFSET INT15
        INT     21H             ;INSTALL NEW VECTOR
        MOV     DX,OFFSET ENDRES
        INT     27H             ;EXIT WITH CAPCON RESIDENT
CODE    ENDS
        END     START
```

ON THE LEADING EDGE

FELIX, HFS-III, Z-248, Z-386

WILLIAM M. ADNEY
P.O. BOX 531655
GRAND PARIRIE, TX 75053-1655

Pointing devices (e.g., mice) are one of the hottest items on the market today, and it seems that most of the programs that you find today can use a mouse. Indeed, some programs are to the point that they almost require a mouse in order to do things easily and efficiently. For example, one of my favorite graphics programs is GEM Draw because it is easy to use and helps me prepare graphics on overhead slides that I use for presentations.

For those of you who are too young to remember the Huckleberry Hound cartoon show, perhaps it is appropriate to quote the words of the immortal Snagglepuss (cat) when he said: "I hate meeces to pieces." In that cartoon series, Pixie and Dixie were the two mice that he was referring to, and they always managed to aggravate him in a variety of amusing situations. I suppose nearly everyone has seen a Tom and Jerry cartoon which presents similar situations.

As I mentioned last month, a mouse is a pesky critter which requires some room to play. Unfortunately, that playing room is desk space. By their nature, mice require a flat surface to play and roam around, and that is not always convenient because desk space always seems to be at a premium. And so, we must go to another cartoon character who was also after mice: Felix, the cat (the wonderful, wonderful cat).

Felix

Felix is a unique pointing device made by Lightgate. If the real estate on your desk is at a premium, then you will appreciate that Felix only needs a 6" by 6" space. Felix is a friendly device, and if you feel so inclined, you can even move him to your lap. I have found that he seems to be quite comfortable, and he is easy to use in that position, too. But before we get too far along, take a look at Felix, as shown in Photo 1.

Photo 1
Felix

Even though Felix needs the separate power supply shown in Photo 1, there has obviously been considerable thought put into that requirement. Although it is not obvious from the photo, the power supply actually plugs into the DB-25 connector so that you don't have a number of wires strung out all over your working area. And even though I generally don't like to have a separate power supply for this kind of peripheral, this is the best approach I have seen yet. In addition, Lightgate has also included a 9- to 25-pin adapter cable so that you can use Felix with a Z-248, Z-386 or laptop that has a 9-pin serial port. Felix can also be easily configured to work with either COM1 or COM2.

Of course, you also get a disk containing appropriate software and a manual that are not shown in Photo 1. Lightgate has thoughtfully included two disks: a 5.25" disk with support software for the PC compatibles, as well as a 3.50" disk that supports the IBM PS/2 series. For this review, I checked out Felix on my Z-248 and Z-386 with its version 1.04 support software for PC compatibles.

Setting up Felix is easy. Plug Felix into the COM1 or COM2 port, plug in the power supply, and power on the computer. Then, run the INSTALL program to create a Felix subdirectory and copy the appropriate files to the hard disk. The INSTALL program also updates the PATH command in the AUTOEXEC.BAT file with the appropriate drive and path (e.g., C:\FELIX). When that is complete, you should run the FLXTEST program to test the operation of FELIX and make sure it is installed properly.

If you look at Photo 1, you will see that Felix has a single button on top of the "control handle." That button performs just like the left button on a standard mouse. To activate a function that usually requires the right button on a mouse, you can move to the upper left or upper right corner (a "hot" spot), press the button once, return to the prior location on the screen, and press the button again to activate the hot spot. Other hot spots on Felix can be activated in a similar manner, and you can define your own hot spots with the included software. When described in words, this sounds awkward, but I did not find it so in using Felix. The cursor control basically moves in a one-inch square on Felix, and although that sounds rather small, I found it effective and fairly easy to use. All you have to remember is that the one-inch square represents the screen, and Felix uses an absolute movement within that square to represent any position on the screen.

I tried Felix on a number of different kinds of programs, including Word, WordStar, SuperCalc, GEM Draw, and Generic CADD, and found it to be easier to use once I got used to it. In addition, I found that it was much faster because of the small space which is used to represent the screen. The major point is that Felix uses an absolute reference to the screen, and a mouse uses a relative reference. From a user point of view, this really means that you cannot move the cursor beyond a single screen like you can with a mouse, but it is easy to set up hot spots to perform the PgUp and PgDn functions for this purpose. By the way, I should mention that the program disk includes Felix programs for all of the applications I tested, plus a number of others. The program disk also includes a special program for Lotus 1-2-3 users called a Navigator, and a special template is also included for that specific application. You can also obtain special software that will allow you to develop additional applications for use with Felix, but I did not receive that.

Like most new kinds of devices, whether you like Felix or not strictly depends on your own preferences. I do, primarily because it is faster than any mouse I have ever used. While it may take some time to get used to its operation, especially if you have used a mouse, I think that it is worth it because it can speed up your cursor movement and your work. Although my primary test system was the Z-248, I also checked out Felix on my Z-386, and it worked fine as expected. I expect that Felix would work fine on any Heath or Zenith PC compatible, but of course, it will not work on the Z-100.

I found Felix to be an effective and much faster alternative pointing device than a mouse. If you are used to a mouse, you may find that it will take some practice to really be able to use Felix because of the different implementation of "buttons", but I think you will find you like it. Overall, Felix is highly recommended as an alternative to a mouse.

HFS-III

One of the more obvious uses for a computer system is a financial application. Indeed, financial applications were just about the first ones that were placed on mainframe computers, and today, many data processing organizations still report to financial management because of that history. Even though it is an obvious application for a microcomputer too, many people have tried and failed. One reason for that is that financial applications can be slow and difficult to learn, particularly if you do not have any background in that area. I have seen some programs that are unnecessarily slow because they are written in BASIC. And some programs are difficult to set up because you must define your Chart of Accounts before you even begin to make your first entry. The accounting profession has a complex jargon of its own just like computers. The world of debits and credits, journals and ledgers, and payables and receivables is every bit as confusing as the weird world of computer jargon, too. Fortunately, there is at least one program that can help you manage finances without requiring that you also be a Certified Public Accountant, too.

Jay Gold Software has the Home Finance System version 3 that is commonly called HFS-III that can help you solve the problem of financial management, even if you are not a financial wizard. One of the best features is that HFS-III helps you set up a financial management system using the standard accounting techniques, but most everything is described in common language, except where it is absolutely necessary to use simple accounting terms like debit and credit.

HFS-III comes with a professional-looking, three-ring binder and three disks, as shown in Photo 2.

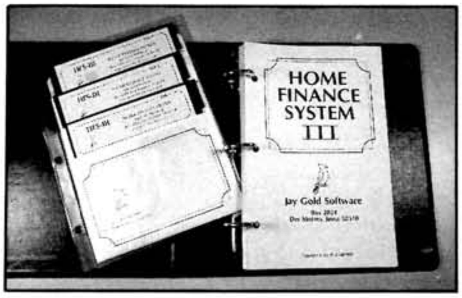


Photo 2
HFS-III

The manual is well written, and I was impressed with the documentation, as well as the program. For purposes of this review, I checked out HFS-III on my Z-248 using MS-DOS 3.21 with a NEC MultiSync monitor and a Video 7 Vega EGA card. The first part of the manual tells you how to install HFS-III, and I should note that you can run HFS-III on a Z-100, as well as any of the other Heath/Zenith computers that run MS-DOS. You don't have to worry about what to install because the INSTALL program takes care of that for you. INSTALL even checks to see if you are running a Z-100 or PC compatible. And the manual describes each of the files, and how the programs and files will be installed.

After installation, you must then set up HFS-III for your own use. HFS-III is menu-driven and is rather easy to set up. This includes defining your printer and setting up a password if you wish. Then, you set up your own expense and deposit codes for financial tracking, and if you are careful to develop "account" names that have unique characters in the first four positions, you can use these names in your transactions instead of numbers. This effectively sets up a Chart of Accounts, and there is an excellent list in Chapter 9 of the manual in case you need some help or ideas. One of the nice features in HFS-III is that you can set up just about anything from the main menu, and it is much easier to do than with other programs.

HFS-III allows you to manage up to 100 checking, savings or other types of bank accounts, but I only set up two. You can also define up to 100 expense codes, which should be more than adequate for most users. And you can print all kinds of reports to help you organize financial data and summarize it. If you are so inclined, you can also set up HFS-III to print checks in just about any format, but I did not test this.

Why would you want to use a program like HFS-III? Perhaps the most obvious reason is to help you organize financial information for tax purposes, and that, of course, will be necessary in a couple of months. This program can help ease that burdensome chore if you keep everything up to date.

I am quite impressed with HFS-III, and it is an extremely useful and well thought-out program. And although it is easy to use, I should note that you will need to take some time and thought in setting up your own financial management operation. Like all programs of this type, you must still enter data in order to take advantage of its features. I think its best feature is that it is easy to use, even for non-accountants. If you are looking for a financial management program, you will be hard-pressed to find a better one than HFS-III. Highly recommended.

AUTOEXEC.BAT and CONFIG.SYS

I just received a letter from a Huggie (whose name I will not mention) that suggested there were two interesting bugs in version 3.20 and 3.21 of Zenith's MS-DOS. In particular, he said the bugs were that, if a bootable disk was place in drive A and the hard disk (drive C) contains both the AUTOEXEC.BAT and CONFIG .SYS files, neither one was executed. If these are bugs, it is interesting to note that they have occurred in every single DOS release that provided these features, and these so-called "bugs" are actually not bugs at all. What really concerns me is that this Huggie also mentioned that he had seen this "bug report" in another publication some time ago, and he included a copy of the page that discussed the "problem." Unfortunately, the user who reported the problem did not understand how any version of DOS works with the CONFIG.SYS file, and this report was published anyway. Lest you be misled, this is not a bug at all, but how this works may not be clearly presented in some or all DOS manuals.

The significant point that this particular user did not understand is that both the CONFIG.SYS and AUTOEXEC.BAT files *MUST* be in the root directory of the *BOOT* drive. In other words, you cannot boot a floppy disk in drive A and expect DOS to go searching for these files on other drives (e.g., B or C). In his letter, the Huggie who brought this to my attention noted that he had observed this to be the case, but did not understand why. This illustrates the reason it is important for all computer users to have a fundamental understanding of some of the technical parts of DOS.

To understand why both the CONFIG.SYS and AUTOEXEC.BAT files must be in the root directory on the boot drives, let's see what happens when DOS is booted. Without getting too much into the technical details, system control is transferred to the system ROM when the computer is powered-up or CTRL-ALT-DEL is pressed. In particular, the boot loader in the ROM takes control and loads the BIOS (usually IBMBIO.COM) followed by the System Kernel (usually IBMDOS.COM). I suppose it is obvious that both of these files must be in the root directory on the boot drive. Then DOS looks for a CONFIG.SYS file in the root directory on the boot drive. If a configuration file is found, all of the valid commands are executed.

Last, but not least, the three portions of the Command Interpreter are loaded: resident, transient, and initialization. The initialization portion of COMMAND .COM is only loaded during system boot, and it is this part of the Command Interpreter that is specifically responsible for executing the AUTOEXEC.BAT file if it exists in the root directory of the boot drive. Again, I suppose it is obvious from this description that both the COMMAND .COM and AUTOEXEC.BAT files must both be located in the root directory of the boot drive. The important point is to recognize that the initialization portion of the Command Interpreter will not go looking for an AUTOEXEC.BAT file in subdirectories or on other disk drives.

You can easily demonstrate that the commands in the CONFIG.SYS file are executed before the initialization part of the Command Interpreter is loaded and executes COMMAND.COM. To see how this works, simply add a DEVICE= command to your CONFIG.SYS file with any device driver that displays an information message. If you don't know of any device drivers that display a message when they are loaded, you can use the VDISK.SYS device driver that is included in all current DOS versions to see a sign-on message. In most DOS versions, you can simply add a VDISK to CONFIG.SYS by including the following command:

```
   DEVICE=VDISK.SYS
```

If you decide to use this example, be sure that the VDISK.SYS file is located in the root directory on your boot drive. Now you can reboot the system and observe that the VDISK device driver displays a message like the following:

```
VDISK Version 3.08 virtual disk E:
Buffer size:         64 KB
Sector size:         128
Directory entries: 64
Transfer size:       511
```

If you do not have an AUTOEXEC .BAT file, the initialization portion of the Command Interpreter will display prompts for the DATE and TIME before showing the drive letter prompt. If an AUTOEXEC.BAT file is found in the root directory of the boot drive, then the initialization portion of the Command Interpreter will bypass the built-in DATE/TIME prompts and AUTOmatically EXECute the commands in the AUTOEXEC file. That is why I usually recommend that the DATE and TIME commands be the first two lines of an AUTOEXEC.BAT file unless you have a computer with a real-time clock, such as the '248, the '386 or any of the new computers that use an 80286 or 80386 CPU.

At this point, I have found that a number of users don't know whether their computer has a real-time clock in it or not. Fortunately, you do not have to be a computer hardware expert or disassemble your computer to find out. All you have to do is rename your AUTOEXEC .BAT file to another name, such as AUTOEXEC.SAV. Then, reboot the computer and check the DATE when the prompt is displayed. If the date is correct, then it is reasonably safe to assume that you have a real-time clock in your system. !f the date is displayed as the default of 01-01-80, then it is reasonably safe to assume that you do NOT have a real-time clock in your system, and you will need to add the DATE and TIME commands to your AUTOEXEC.BAT file. For ease of keeping track of which files were created when, I recommend you make sure that your system DATE and TIME are correct.

My Z-386 Configuration

Several people have written me asking what kind of hardware configuration I am using in my Z-386, so I will spend a moment on that. At the present time, my '386 has my old 28 ms 40 MB hard disk (the one that crashed last year and was reconditioned) with the original 1 MB of main memory, plus an additional 4 MB Z-515 memory board for a total of 5 MB. I have also added the Z-525 Cache card to help speed up disk I/O. The '386 still has the old Z-449 video card, and I now have a ZCM-1490 FTM monitor for it so that I do not have to keep switching the NEC MultiSync between the two systems.

For those of you interested in hardware-specific information about '386 systems, that's about it. Since the '386 is generally PC or AT compatible, with the exception of the memory boards, I will be discussing some of the more interesting things that I have found in the next few months. Here's one example.

Several people have told me that their version of Zenith GW-BASIC crashes their '386 systems. One even told me that it was a fact that the '386 was not PC compatible with BASIC in this respect which turns out NOT to be the case at all. This is one situation where you absolutely MUST understand the relationship between the hardware in your system and the software you are using. And although there is some truth in the report that BASIC can crash a '386 system, it is important to understand what has happened.

After this "problem" was brought to my attention, I was also able to consistently crash my '386 with GW-BASIC. But the only times that the crash occurred was when I was using an old version 2, and that's the point. When using the latest version 3.2, everything works just fine, and there are no crashes. What is going on? Is this really a Zenith problem? I don't think so.

All of the older Zenith '386 systems I have seen are equipped with the usual Z-449 video card that provides EGA and VGA compatibility among other things. And GW-BASIC version 3.2 provides high-resolution support for these kinds of video cards. As far as I can tell, the crashes are strictly due to the fact that the old versions of BASIC did not support this kind of high resolution video, and programs crash — mostly programs that use graphics. I found that all programs that I had ran just fine under version 3.2, although some of them did consistently crash when using any GW-BASIC version 2.

You can also find some problems if you are running programs which use the latest version's EGA display commands under version 2 which, of course, does not have them. That can cause all kinds of strange and unpredictable errors depending on what new features of version 3.2 are used in the program. For the most part, programs written under version 2 should be upward compatible with version 3.2, but if you have any problems with older programs, be sure to check the code.

Powering Down

In order to round out the discussion of pointing devices, next time I will be talking about what I think is the best mouse you can find: the Logitech C-7 three-button mouse. In addition, I will discuss pointing devices, in general, and include some of my thoughts on each: the MOUSETRAK, Felix, and the Logitech mouse. Each has its advantages and disadvantages, and I will share some of my experiences with you next time.

For help in solving specific 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 specifically 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

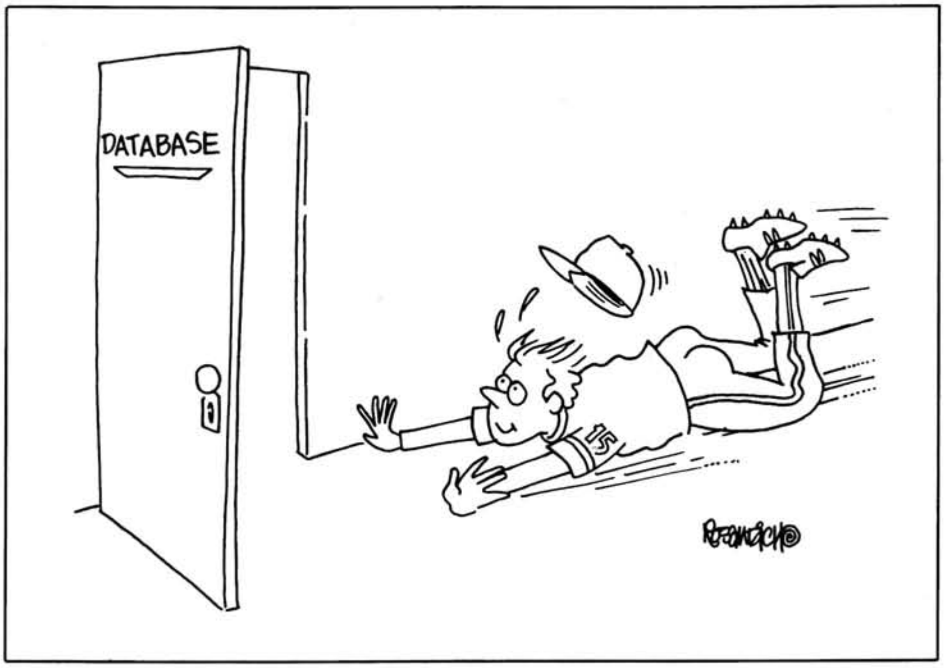
Felix $199.00
Lightgate
6202 Christie Avenue
Emeryville, CA 94608
(415) 596-2350

HFS-III $49.95 (S&H)
Jay Gold Software, Inc.
P.O. Box 2024
Des Moines, IA 50310
(800) 541-0173

HS-386 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
FTM Monitor (ZCM-1490)	999.00
EGA GW-BASIC Version 3 (MS-4164-11)	99.00

Heath/Zenith Computer Centers
Heath Company Parts Department
Hilltop Road
St. Joseph, MI 49085
(800) 253-7057
(Heath Catalog orders only)

*



Circle #153 on Reader Service Card

Circle #191 on Reader Service Card

Z-100

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

SURVIVAL KIT

Hello again! I was hoping my introduction to "Z-100 Survival Kit" in the last issue would arouse your interest. Glad to see you're back! We didn't cover much serious ground in that first installment, but now that we have the formalities out of the way, we're ready to get down to business.

As promised, we are going to take an overall look at the question of PC compatibility for the Z-100. Over the past year or two, the goal of being compatible with, and emulating the PC's has received a lot of attention. Is it really that important? Why is it so difficult to achieve? Is it worth the effort?

Beyond compatibility, we'll look at software that runs on both the native Z-100 and PC compatibles. I'll also show you how you can write your own programs that run on either type of machine. There's a lot to talk about, so let's get started . . .

The PC Compatibility Question

To someone unfamiliar with the personal computer scene, it must seem strange that one of the main pursuits is trying to get your computer to do the same thing that everyone else's does. Particularly when the machine you have is superior to that owned by your neighbors. And from a purely hobbiest viewpoint (programming being the hobby), it is ridiculous. Give me a Z-100, and I'll write a program that will run rings around the nearest PC/XT clone.

To understand the quest for compatibility in the proper light you must realize that most computer users aren't interested so much in how wonderful or powerful their computers are, but in what the machine can do to help them with practical day-to-day problems. Things like balancing the books, keeping track of inventory, analyzing statistical data, or even drawing pictures. You quickly find that the power behind the machine is in the software. If nobody has written a program for your computer that does what you need done, then you're just out of luck.

A little history . . . In the beginning (circa 1975) most personal computers were created about equal. Hardly any of them would run the same software, because they were all different. The only unifying influence was the CPM operating system, but it was far from a standard (by today's standards). For the most part, if you wanted software to do a specific task, you had to write it yourself, or hire it done. As the years went by, the situation really didn't change much until IBM announced their own personal computer. And at that moment, the computer industry forever changed (for better or worse). Amid protests that IBM had stymied technological innovation, a new standard had been set. My use of the word "standard" here should not be taken as in "standard of excellence", but should be understood as meaning "something to conform with". Whatever your viewpoint about the IBM-PC revolution, you will have to admit that it caused the computer industry to start making computers which all looked and acted vaguely the same.

One of the main consequences of this turn of events is that software companies now had a stationary target to aim at. Where once there were just a few medium-sized software companies (mostly companies selling operating systems and development tools), now new companies sprang into existence almost overnight. Companies like Microsoft and Lotus Development began to show Wall Street that computer software was big business. As software development became more sophisticated to meet the needs of an ever-demanding audience, small companies began to be edged out of the software business because they couldn't compete.

Back to the present, we find that most of the flashy high-powered software we would like to use is sold by million dollar software companies (they're the only ones who have the resources to produce it). And, naturally, being in business for a profit, they only write software for computers that have a substantial user base. Which means mainly PC Compatibles and the Apple MacIntosh. All others need not apply, including the Z-100.

Why Do We Want Compatibility?

The obvious reason for wanting PC compatibility is to take advantage of programs that aren't available for the Z-100. There are literally tens of thousands of software titles available for the original IBM-PC and its descendants, while the list of commercial software written specifically for the Z-100 is composed of only a few hundred programs.

But there are other reasons you might consider compatibility an issue. One of the most important of these is the ability to use the software interchangeably on a Z-100 and PC compatible computer. Many of you now own a PC compatible computer, as well as a Z-100. You'd like to be able to use the software on both machines, right? (We won't mention software license agreements here). There are two ways of doing this . . . buy programs that run on either machine, or fix the Z-100 so it can emulate a PC.

If you are considering buying a new PC compatible in the future, this gives you another reason to seek PC compatibility with your present Z-100. If your Z-100 could run PC software, you could go ahead and start using the programs you will use on the new machine. And you won't have to buy all new programs when you make the switch.

Emulating the PC Compatibles

The Z-100 is not very similar to the IBM-PC. As a matter of fact, you have to really look hard to find any similarities at all. The bus is different, the video layout isn't even close, the I/O chips aren't the same, the keyboard is different, the hard disk interfacing is incompatible. About the only things in common between the two machines are the 8088 CPU chips. Actually, it is a credit to the MS-DOS operating system that the Z-100 and IBM-PC run any of the same programs.

The secret to conquering this apparent list of incompatibilities lies in successful emulation of the PC's features. This is accomplished with more or less success by using special software, special hardware, or both.

Hardware Emulation

There are two alternatives for hardware PC compatibility with your Z-100. The GEMINI Board and the UCI Easy-PC. Both of these emulators plug into the Z-100 and provide fairly good results. I'm not going to spend any time describing how they work, because that has already been done. Reference the following articles for more information . . .

GEMINI Board	REMark January 1986, November 1986 SEXTANT #24
UCI Easy-PC	REMark June 1986, August 1986 SEXTANT #25

Even though these hardware solutions offer very good PC compatibility, there are some rough edges. For instance, neither will solve the problem of the incompatible Z-100 serial ports. Any program that communicates directly with the PC COM ports will not work using the GEMINI or UCI boards. UCI does have an optional (extra cost) Easy-I/O board available which solves this problem by providing the needed PC compatible serial ports.

Software Emulation

Emulation of the IBM-PC can also be accomplished to an amazing level without any additional hardware. The popular ZPC program available from HUG is the best and most widely used example of what can be done through software emulation. Some of you newcomers may not know that there were predecessors to ZPC (like PCEM.COM) which established the feasibility of software PC emulation on the Z-100.

ZPC doesn't provide the level of compatibility achieved with the hardware emulator boards, but it comes close enough to be a contender when you consider the difference in price. Software emulation will usually require that some changes or patches be made to the program before it will run. These are required where the PC program is trying to communicate directly with a port or other hardware that doesn't have a counterpart in the Z-100.

The ZPC software approach can be helped along by adding a little hardware. The ZHS circuit described by Pat Swayne (author of ZPC) helps by making more PC programs run with fewer (or no) patches. The board may be homemade, or is commercially available from Scottie Systems. Another help is the addition of an IBM style COM port for serial communication.

Limitations to Emulation

There is one important thing to consider about all the PC compatibility solutions presently available for the Z-100. They are all already absolete! One of the main reasons we are trying to emulate the PC compatibles in the first place is to be able to run all that flashy, state-of-the-art software. But have you considered that to effectively use a lot of PC software you need EGA or VGA graphics — neither of which is supported by ZPC or the GEMINI or UCI boards. It's only a matter of time before you won't be able to find PC software that uses the old CGA modes.

The Ultimate Emulation

We're about to wrap up our conversation about emulation here, but one important consideration has been left unsaid until now. The best way to run PC programs is to have a PC compatible computer on the desk next to your Z-100. By the time you purchase one of the hardware solutions to compatibility described above, you could have made a substantial down payment on a real PC clone. (I guess maybe I should say Zenith PC!)

Programs That Run on The Z-100 and PCs

This installment of "Z-100 Survival Kit" deals primarily with the need to run PC programs on a Z-100. We've discussed the standard solution, which is trying to turn the Z-100 into a near-PC compatible. But that's only half the story — especially if you do any programming. A lot of software is available that will run on both machines (without emulation). And if you know what to look out for, you can also write your own software that runs on the Z-100, as well as PC compatibles.

Programs That Don't Care

Any program that uses MS-DOS function calls whenever it communicates with a peripheral device (including the screen) should run equally well on a Z-100 or a PC compatible. Such a program is called a "Generic MS-DOS" program (sometimes referred to as "well-behaved", because it doesn't deal directly with the hardware). Until the recent advent of windowing environments, most assemblers and language compilers were generic MS-DOS programs. For instance, Microsoft's MASM assembler will run equally well on a Z-100 and a PC clone. So will the Microsoft 'C' compiler (but not Quick-C).

One big problem with this approach is in screen control. MS-DOS doesn't establish any standards for doing things like clearing the screen, positioning the cursor, etc. This means that a "well-behaved" program must be content with a scrolling type of text display, and no graphics. A partial solution to this problem is for a program to use the ANSI.SYS screen driver. The ANSI.SYS driver (called ANSICON .DVD on the Z-100) provides a standard set of screen and cursor control escape sequences.

Any PC program that expects to use the ANSI.SYS driver will specifically mention this fact in its documentation. If it does, there's a good chance it may run on the Z-100 (or any MS-DOS computer). To try it, do the following . . .

1. Copy the file ANSICON.DVD from your MS-DOS disk into the root directory of your boot-up disk.
2. Edit (or create) your CONFIG.SYS file by adding the following line;
   ```
   DEVICE=ANSICON.DVD
   ```
3. Be sure the CONFIG.SYS file is in the root directory of your boot-up disk. Then, reboot the computer to load the ANSI driver.
4. Now you're ready to try the program. Having the ANSI driver loaded won't make any difference unless the program uses the special ANSI escape sequences for screen and cursor control.

Want to write your own programs that use ANSI escape sequences for portability? Here are some commonly used ANSI escape sequences;

ASCII Command	Hex Values	Function
Esc[n;mH	1B 5B n 3B m 48	Move cursor to row 'n', column 'm'.
Esc[nA	1B 5B n 41	Move cursor up 'n' rows.
Esc[nB	1B 5B n 42	Move cursor down 'n' rows.
Esc[nC	1B 5B n 43	Move cursor right 'n' columns.
Esc[nD	1B 5B n 44	Move cursor left 'n' columns.
Esc[2J	1B 5B 32 4A	Clear screen, move cursor to top left
Esc[K	1B 5V 4B	Erase to end of line

The 'n' and 'm' letters indicate an ASCII decimal number. The Programmer's Utility Pack and most good MS-DOS reference texts have more information about the ANSI escape sequences.

The main problem with programs that use MS-DOS for everything is that they are noticeably sluggish when writing to the screen (with or without ANSI.SYS). The only way to avoid the slow screen output of DOS is to use BIOS routines or write directly to video memory. And if the program uses graphics at all, the programmer has no choice, because MS-DOS doesn't do windows (or graphics).

Programs That Differentiate

If a program uses true graphics (lines, circles, filled areas, etc.), or writes text directly to the video memory, then it must have access to at least two unique sets of graphics routines. One set for the Z-100, and another for the PC compatible. There are several general schemes that could be used to determine which routines should be used. Here are some examples:

1. The graphics routines for each machine could be kept in seperate object libraries or files. When the program is configured for a particular computer the appropriate set of graphics routines is linked with the main program to make the final executeable program. This method results in the fastest execution time and smallest code size, but is difficult to implement.
2. The graphics routines could be organized into a device driver or memory resident library. This method is not easy to implement with high-level languages, and suffers from slow calling times for routines in the library.
3. The program could contain all the graphics routines required for any configuration, and decide which routines to use at run time through conditional branching. A program which uses this method will have the advantage of running on either machine without reconfiguration, but will suffer somewhat in code size and speed.

Regardless of which of these methods you might choose for your own programs, they all have one requirement. At some point, they need to know the host computer type so they can choose the appropriate graphics routines. If the program will be configured ahead of time for a certain computer (as in 1 or 2 above), the obvious way to determine the host computer is to ask the user. But if the program will need to determine the host computer type each time it is run, a more convenient way would be appropriate.

I have found that a reliable way for a program to determine its host computer type is to check two bytes of system memory located at 0040:0000H and 0040:0003H. If this is a Z-100, the area beginning at segment 40H is the BIOS jump table, and bytes 0H and 3H will both be 0E9H (a jump instruction). If this is a PC compatible, the area at segment 40H is the BIOS data area, and 0E9H's will not be present. Here's how it looks in assembly language:

```
          MOV      AX, 40H                      ; get BIOS segment
          MOV      ES, AX                       ;
          MOV      SI, 0                        ;
          CMP      BYTE PTR ES:[SI], 0E9H       ; jump instruction?
          JNE      PC                           ; no, go to PC routine
          CMP      BYTE PTR ES:[SI+3], 0E9H     ; check another place
          JNE      PC                           ; no, go to PC routine
Z100:     ...                                   ; Z100 graphics routine
          JMP      COMMON                       ; join common code
PC:       ...                                   ; PC compatible graphics rou
tine
COMMON:   ...                                   ; common code resumes here
```

Writing Your Own Programs General Guidelines

If you want to write your own programs that run on the Z-100 and PC compatibles, here are some tips; Keep it as simple as possible. If your application doesn't require fancy graphics, don't worry about it. Generic DOS programs are definitely the best bet when practical. If necessary, use the ANSI device driver for more flexibility in text screen displays. Don't forget that an easy way to clear the screen using MS-DOS is to issue 25 line feeds.

Stay away from using keyboard input from special function, keypad, and arrow keys. There are no standard ASCII definitions for these keys. If you must use input from keys other than ASCII characters, symbols, and numbers, then you will have to write a keyboard input routine that recognizes the difference between the Z-100 and PC keyboards.

Use MS-DOS function calls for all disk input/output and communication to serial and parallel ports. Most high level languages use MS-DOS functions, but avoid using any language features that depend on BIOS calls or direct hardware contact.

If you program in BASIC, the easiest way to write programs for the Z-100 and PCs is with interpreted GW-BASIC. Most programs will operate correctly on either machine. But beware of the SCREEN command. You will need to take into consideration that the colors and screen resolution are different between the two types of computers.

Wrapping It Up

We've taken a good look at the question of PC compatibility in this installment of "Z-100 Survival Kit". It is an issue which is important to many Z-100 users. However, I don't feel like compatibility holds any secrets to the continued popularity of the Z-100. Running PC software on a Z-100 is just another way of saying "I wish I had a Z-248 instead".

If the Z-100 is to have a lasting user support base, it will be achieved through the strengths of the machine itself — running software designed to take full advantage of its features. In future columns, we'll look at ways to take advantage of some of those powerful features.

Continued from Page 15

lutions like 640 by 480, we are starting to get some pretty good looking circles. The earlier machines always coughed up something which looked like a football at best.

I should note here that EGAPaint maintains its screen aspect ratio such that vertical and horizontal resolutions appear balanced on the screen and on the printer as well. Aspect ratios are selectible during setup but I am comfortable with the recommended aspect ratio of RIX Software.

EGAPaint
RIX Softworks, Inc.
18552 MacArthur Blvd.
Suite 375
Irvine, CA 92715
(714) 476-8266

Microsoft Mouse PCPaint
MSC Technologies
2600 San Tomas Expressway
Santa Clara, CA 95051
(408) 988-0211

Pallette
Software Wizardry
8 Cherokee Drive
St. Peters, MO 63376
(314) 477-7737

A SIMPLE FILE COMPRESSION/EXPANSION PROGRAM IN C

WILLIAM S. HALL
3665 BENTON STREET, #66
SANTA CLARA, CA 95051

Introduction

When I was an Associate Editor at Mathematical Reviews in Ann Arbor, we made heavy use of dBASE II on Z-100 computers. Even with Winchester disks, however, we were at a premium for space, and it was necessary to take some action with our database files when they were not in use. After evaluating several alternatives, we decided to try making the files smaller by the simple technique of replacing runs of a single character in a file by a string describing the run. In the process, we wrote programs in C for both CP/M and MS-DOS. We present both versions since the first is simple, and the latter shows an application of block read and write to reduce disk activity and to speed up processing. Of course, the corresponding expansion companion pieces are also provided.

To check the effects of compression, we wrote a version of the UNIX program, wc, which counts characters, words, and lines in a text file. We have included it here, as well.

If you would rather not type in the program, I can supply you with an MS-DOS disk containing all source files for $10.00. Unfortunately, I can no longer make up CP/M disks, and the only copy I have left of the CP/M version is on a hard-sectored disk.

Background

Many files, such as print images and dBASE II files, frequently contain long runs of a single character, often a blank or null. Considerable space can be saved by using a filter to replace such runs by a short descriptor specifying the length of the run and the character replaced. Such a technique has been described in the well-known books by Kernighan and Plauger, "Software Tools" and "Software Tools in Pascal"[1-2]. Here, we have adapted the programs in [2] to two versions of C in wide use on microcomputers, C/80 from the Software Toolworks [3] and Microsoft C, version 5.0. The application of these programs has been dramatic; on some dBASE II files, we are able to reduce their size by 50 to 70 percent. When combined with an archive program, such as ARCHIVE-80 and ARCHIVE-86 from Generic Software[5], it is possible to store several such files in a small amount of space.

At Mathematical Reviews, dBASE II is used to drive a Reviewer Assigning program. With 66 areas, each of which can require up to 500K of storage, file management, even on a Winchester disk, can be a problem. Several approaches are possible. One is to use the supplied BACKUP and RESTORE programs, but this has proved to be an unpleasant task when the only removable media are 360K diskettes. A second method is to use the dBASE II 'copy SDF delimited' command to store the files as text with trailing blanks removed and fields delimited with a character, such as the single or double quote. This is relatively effective, and does save space, but index files cannot be compressed in this way. In addition, commands in dBASE II execute fairly slowly. A third approach is to use the compression technique described above. We find that on the average, each area can be shrunk to 70% of its original size, and because the compression and expansion programs are written in a compiled language, execution time is greatly reduced.

The basic idea of the program is relatively simple. The next character obtained from the file is compared with the previous one. If they are the same, then a counter is incremented. If not, then the counter is checked to see if it exceeds a threshold, in this case 4. If so, a three-character string is printed in the form "~XC", where the tilde is a flag indicating a compression string, 'X' is a single character describing the number of characters compressed, and 'C' is the character itself. Of course, the program needs to do something sensible with end-of-file and when the count exceeds the maximum length allowed by the program. Finally, it must handle the tilde itself as a text character; it does this by always putting it out in the encoded format just described.

The details of the compression program are discussed quite nicely in [2], Chapter 2, Section 2.3. The expansion phase is covered in Section 2.4, and as expected, it performs the inverse operation to compression. The program language used is Pascal. In addition, runs of only 26 characters are allowed. Some reasons for this restriction are to keep the files printable on a wide variety of terminals, including those with only upper case and on computer systems using EBCDIC. Thus, 'A' means a run of 1 character, 'B' a run of two, etc. We have extended our maximum allowed run to 93 by using the printable ASCII set from '!' to '}'. We found this more than adequate for our purposes.

By the way, the same approach to compression and expansion is used in the well-known public domain Kermit file transfer program.

Of course, compression methods as we have described will always make some files longer rather than shorter. (Consider, for example, one consisting of alternating tildes and blanks.) Of the four files needed for each area in our reviewer assigning program, one of the index files grows about 1000 bytes when it is 'compressed', and we do not carry out the operation on this one. But, the overall savings has been worth the effort; after a session, the user runs a batch file to compress a given area and to erase the corresponding database and index files. The packed files can then be left on the Winchester and restored later when needed. As an added benefit, the compression often makes the files small enough to fit on a 360K floppy, and if necessary, they can then be conveniently removed.

For more sophisticated data compression programs using Huffman coding, the Software Toolworks has a reasonably priced one called "Crypt and Pack".

Data compression is a vast, complicated subject not usually covered in elementary textbooks. A suitable beginning level reference is Tennenbaum and Augenstein[6].

If you are interested in details on the assigning program used at Mathematical Reviews, see [7].

The C/80 programs

The C/80 version of COMPRS/EXPAND (Figure 1) wraps a shell around the functions compress, putrep, and expand given (in Pascal) in [2]. In fact, the two programs differ only in the functions called; the outer wrapping is the same for both. For this reason, we are able to have both programs in one file by using conditional compilation. By setting the constant COMPRS to 1, we can compile the compression program. After changing the value to zero, the expansion program can be generated.

An alternative approach to conditional compilation is to have all procedures in one program and to use a switch on the command line to select the appropriate action. We prefer having two programs, but having only one would save even more space!

The following discussion applies only to C/80 version 2.0, which by now may be a bit dated. If you have a later version for CP/M, some of the comments which follow may no longer be true.

After the necessary 'printf.h' header file is included and some definitions of constants given, two global variables, the handles for the input and output files are declared. The main program opens with an error check to see that enough items were specified on the command line. If not, the program exits with a short statement about how to use the program.

Next, the input and output files are opened in binary mode; the program simply exits with an error message if something is wrong. Compress or expand is then executed until end-of-file is detected, the output is closed, and the program terminates. Compress calls putrep, as needed, to print the compression strings.

Whereas getc is used to get a character from the input, a function putcerr writes to the output. This latter is simply putc with an exit if an error is detected. It is an undocumented feature of C/80 to return -1 if an error occurs during a write. Detecting this error and exiting prevents one from writing to a full disk without warning.

The companion program, expand, is obtained by changing the value of COMPRS and recompiling, perhaps using a different file name. We always use the -m switch in C/80 and assemble the resulting output with Macro-80. If the original C file is called comprs.c, then the link commands *comprs, *printf, *stdlib/s, *clibrary, *comprs/n/e will produce the compression program. After setting COMPRS to 0 and recompiling with C/80 and the same link commands, this time with the last line reading *expand/n/e will generate the expansion program. Of course, the AS assembler can also be used. In this case, printf.c replaces printf.h. See the C/80 documentation for more details.

Figure 1

```
/*
COMPRS/EXPAND - C/80 version.  Programs to compress and expand binary
and text files by replacing runs of a single character by the string
'~XC', where the tilde indicates the beginning of a compression, X is a
character whose ASCII value minus '!' indicates the length of the run,
and C is the character itself.  Tildes are always represented as runs.

Both programs are contained in the same source file.  Set the constant
COMPRS to 1 to compile comprs and to 0 to compile expand program.

Usage:  compress/expand infile outfile

Written by:  William S. Hall
             Associate Editor
             Mathematical Reviews
             611 Church Street
             Ann Arbor, MI 48104

Completed:  08/20/84
*/

#include <printf.h>             /* needed for printf in c/80 */
#define COMPRS 1                /* set to zero to compile expand program */
#define WARNING 126             /* text compression marker '~' */
#define MAXREP  93              /* maximum length of compressed text */
#define THRESH  4               /* threshold to start compression */
#define NULL    0
#define EOF     -1
#define stderr  0
int in, out;                    /* file handles */

main(argc, argv)
int argc;
char *argv[];
{
    if (argc != 3) {
        fprintf(stderr, "Usage - comprs/expand infile outfile\n");
        exit();
    }

    if ((in = fopen(argv[1], "rb")) == NULL) {
        fprintf(stderr, "Unable to open output file\n");
        exit();
    }

    if ((out = fopen(argv[2], "wb")) == NULL) {
        fprintf(stderr, "Unable to open output file\n");
        exit(1);
    }
#if COMPRS
    compress();
#else
    expand();
#endif
    fclose(out);
}

/*
The following three routines, compress(), putrep(), and expand() are
adaptations to C from the book by Kerhighan and Plauger, "Software Tools
in Pascal", Addison-Wesley, Reading, MA, 1981, Chapter 2, Sections 2.3
and 2.4.
*/

#if COMPRS
compress()
{
    int n, lastc, c;

    n = 1;
    lastc = getc(in);
    while (lastc != EOF) {
        if ((c = getc(in)) == EOF) {
            if ((n > 1) || (lastc == WARNING))
```

```
                putrep(n, lastc);
            else
                putcerr(lastc);
        }
        else if (c == lastc)
            n += 1;
        else if ((n > 1) | (lastc == WARNING)) {
            putrep(n, lastc);
            n = 1;
        }
        else
            putcerr(lastc);
        lastc = c;
    }
    return;
}

putret(n, c)
int n, c;
{
    while ((n >= THRESH) || ((c == WARNING) && (n > 0))) {
        putcerr(WARNING);
        putcerr(min(n, MAXREP) - 1 + '!');
        putcerr(c);
        n = n - MAXREP;
    }
    for (n = n; n >= 1; n--)
        putcerr(c);
    return;
}
#else
expand()
{
    int n, c;

    while ((c = getc(in)) != EOF) {
        if (c != WARNING)
            putcerr(c);
        else if (((c = getc(in)) >= '!') && (c <= '}')) {
            n = c - '!' + 1;
            if ((c = getc(in)) != EOF)
                for (n = n; n >= 1; n--)
                    putcerr(c);
            else {
                putcerr(WARNING);
                putcerr(n - 1 + '!');
            }
        }
        else {
            putcerr(WARNING);
            if (c != EOF)
                putcerr(c);
        }
    }
    return;
}
#endif

/* Putc with error detection and exit */

putcerr(ch)
int ch;
{
    if (putc(ch, out) == EOF) {
        fprintf(stderr, "Write error!\n");
        exit(1);
    }
    return;
}
```

The Microsoft C Program

One problem with the C/80 program as we have written it is the relative slowness of getc and putc. It is more efficient to read and write in large chunks. Although the read and write primitives are available in C/80, they can be a little more troublesome to use when blocks which are not a multiple of 128 are needed. It is much easier in Microsoft C, and since the technique does not appear often in elementary texts on C, we would like to show you how it can be done. The reduction in running time is significant; files of 100,000 bytes can be compressed in less than half the time taken when using getc and putc, and disk accesses drop markedly.

As in the C/80 version, both the compression and expansion programs are written as one by using conditional compilations. The task of making the two different programs is made much simpler, however, by the use of a make file (Figure 3). In this way, the process can be automated. In particular, the constant COMPRS can be defined on the compile command line. So we have eliminated the line #define COMPRS 1 (or 0), and have changed the #if statements by #if defined(COMPRS). By the way, in the pending ANSI standard for C, this replaces the older preprocessor command if #ifdef COMPRS. Similarly, #if !defined(COMPRS) replaces #ifndef COMPRS.

The MAKE utility is supplied with the Microsoft package. Thus, to create the two programs, it is enough to give the command 'make comprs'.

The next new difference in the MS-DOS source (Figure 2) is the constant BLKSIZE, which we have set to 16384 = 32*512. Any other value can be used, but a multiple of 512 is best in MS-DOS. How large a block to use depends on available memory and the 65,536 byte limit on the function malloc.

In Microsoft C, several constants defined explicitly in the C/80 source, such as EOF, stderr, etc. are available in the numerous header files supplied with the compiler. So, only those constants local to the program need be defined here.

Next, we define BYTE as an unsigned integer. In the pending ANSI standard for C, characters are now regarded as signed. In particular, a character value of 0xFF is extended to the integer -1. Such problems are avoided if we use unsigned characters appropriately, and the BYTE typedef provides a convenient synonym.

You may be also surprised at another major change, ANSI C, and that is how functions are coded. The obsolescent method is:

```
foo(a,b)
int a;
char b;
{
        ...some code
}
```

The new method, and the one we use here is:

```
foo(int a, char b)
{
        ...some code
}
```

Pascal programmers will feel right at home. By the way, if you are still using Microsoft C, version 4.0 or earlier, you have to do things the 'old' way.

Next, several global variables are declared, including pointers to the beginning of the read and write blocks and corresponding pointers to advance through

Figure 2

```
/*
comprs.expand - Microsoft C version.  A file compression/expansion program
which compresses text and binary files by replacing runs of the same
character with the string "~XC", where 'X' represents the count (ASCII value)
minus '!' of the run and 'C' is the character itself.

comprs is compiled if COMPRS is defined.  Leave it undefined to comple
the expand program.

Written by:  William S. Hall
             3665 Benton Street, #66
             Santa Clara, CA 95051

Written in Lattice C, version 3, 08/20/84
Updated to Microsoft C, version 5.0, 05/07/88
*/

#include <stdio.h>
#include <stdlib.h>

#define WARNING 126             /* '~' marks beginning of compression string */
#define MAXREP  93              /* maximum compression allowed */
#define THRESH  4               /* threshold to begin compression */
#define BLKSIZE 16384           /* read and write primitives' block size */

/* global variables */

typedef unsigned char BYTE;

FILE *in, *out;                 /* file pointers */
BYTE *rblock, *rptr;            /* pointers to read and write block heads */
BYTE *wblock, *wptr;            /* pointers to characters in the blocks */
size_t rcount, wcount;          /* count remaining in blocks */

/* local function declarations */
void compress(void);
void putrep(int, int);
void expand(void);
int getcmem(void);
void putcmem(int);
void flush(void);
size_t fill(void);

main(int argc, char (argv[])
{

    if (argc != 3) {
#if defined(COMPRS)
        fprintf(stderr, "Usage - comprs infile outfile\n");
#else
        fprintf(stderr, "Usage - expand infile outfile\n");
#endif
        exit(1);
    }

    if ((in = fopen(argv[1], "rb")) == NULL) {
        fprintf(stderr, "Input file not found\n");
        exit(1);
    }

    if ((out = fopen(argv[2], "wb")) == NULL) {
        fprintf(stderr, "Unable to open output file\n");
        exit(1);
    }

    if ((rblock = malloc(BLKSIZE)) == NULL) {
        fprintf(stderr, "Not enough memory for input block\n");
        exit(1);
    }

    if ((wblock = malloc(BLKSIZE)) == NULL) {
        fprintf(stderr, "Not enough memory for output block\n");
        exit(1);
    }

    rptr = rblock; wptr = wblock;

    if ((rcount = fill()) == 0) {
        fprintf(stderr, "Input file empty\n");
        exit(1);
    }

#if defined(COMPRS)
    compress();
#else
    expand();
#endif

    if (wcount != 0)
        flush();

    if (fclose(in) == EOF) {
        fprintf(stderr, "Unable to close input file\n");
        exit(1);
    }

    if (fclose(out) == EOF) {
        fprintf(stderr, "Unable to close output file\n");
        exit(1);
    }
}

#if defined(COMPRS)
void compress()
{
    int n, lastc, c;

    n = 1;
    lastc = getcmem();
    while (lastc != EOF) {
        if ((c = getcmem()) == EOF) {
            if ((n > 1) || (lastc == WARNING))
                putrep(n, lastc);
            else
                putcmem(lastc);
        }
        else if (c == lastc)
```

the blocks. Counter variables are also declared. Finally, we declare all functions at the top of the file. This is a good practice which facilitates error checking during compilation. Note also the use of the type size_t. This is simply a typedef for an unsigned integer which is used in the functions fread, fwrite, and malloc.

As in C/80, the files are opened in untranslated mode to prevent the program from ignoring carriage returns during reads and to not add superfluous carriage returns during writes. Next, memory blocks are obtained and the read and write pointers are initialized. The read buffer is then filled for the first time, and compress or expand are called as appropriate. Any remaining characters in the output block are then flushed, the files are closed, and the program exits.

In place of getc and putc, the functions compress, expand, and putrep now call getcmem and putcmem (get and put a character in memory). Getcmem, for example, simply returns the next character in the read buffer and advances its pointer while decrementing the read buffer counter. If the count drops to zero, it calls fread to fill the buffer again. If fread returns a value less than the number requested, then a check is made for end-of-file or an error. Our technique for handling errors is simple; we exit with a warning message.

Putcmem, on the other hand, waits

```
            n += 1;
        else if ((n > 1) || (lastc == WARNING)) {
            putrep(n, lastc);
            n = 1;
        }
        else
            putcmem(lastc);
        lastc = c;
    }
}

void putrep(int n, int c)
{
    while ((n >= THRESH) || ((c == WARNING) && (n > 0))) {
        putcmem(WARNING);
        putcmem(min(n, MAXREP) - 1 + '!');
        putcmem(c);
        n = n - MAXREP;
    }
    for (n = n; n >= 1; n--)
        putcmem(c);
}
#else
void expand()
{
    int n, c;

    while ((c = getcmem()) != EOF) {
        if (c != WARNING)
            putcmem(c);
        else if (((c = getcmem()) >= '!') && (c <= '}')) {
            n = c - '!' + 1;
            if ((c = getcmem()) != EOF)
                for (n = n; n >= 1; n--)
                    putcmem(c);
            else {
                putcmem(WARNING);
                putcmem(n - 1 + '!');
            }
        }
        else {
            putcmem(WARNING);
            if (c != EOF)
                putcmem(c);
        }
    }
}
#endif

int getcmem()
{
    int c;

    if (rcount-- > 0)
        c = *rptr++;
    else {
        rcount = fill();
        rptr = rblock;
        if (rcount == 0)
            c = EOF;
        else {
            rcount--;
            c = *rptr++;
        }
    }
    return(c);
}
void putcmem(int ch)
{
    *wptr++ = ch;
    wcount += 1;
    if (wcount >= BLKSIZE)
        flush();
}
void flush()
{
    size_t wnum;

    wnum = fwrite((BYTE *)wblock, (size_t)sizeof(BYTE), wcount, out);
    if (wnum != wcount) {
        fprintf(stderr, "Write error!\n");
        exit(1);
    }
    wcount = 0;
    wptr = wblock;
}

size_t fill()
{
    size_t numread;

    numread = fread((BYTE *)rblock, (size_t)sizeof(BYTE), (size_t)BLKSIZE, in);
    if (numread < BLKSIZE) {
        if (!feof(in)) {
            fprintf(stderr, "Read error!\n");
            exit(1);
        }
    }
    return(numread);
}
```

Figure 3

```
compr s.exe : comprs.c
    cl -DCOMPRS comprs.c

expand.exe : comprs.c
    cl -Foexpand -Feexpand comprs.c
```

for the buffer to be filled, then writes the entire block. Of course, after EOF is detected, there will most likely be characters still to be written; this is why flush is called just before closing the output file.

Here is where C/80 becomes a bit awkward to use and explains why we stuck to getc and putc. Its write primitive allows only a multiple of 128 bytes to be written. Suppose we want to flush wcount bytes, wcount <= BLKSIZE. Let n = (wcount/128)*128 and perform write(out, wblock, n). We still have r = wcount % 128 bytes remaining to pick up. Set the pointer wptr to wblock + n. It now points to the first character remaining, which has not yet been written. Now, r calls to putc will flush out the remaining bytes. If you want to experiment, try rewriting the C/80 version using reads and writes and the hint above.

Note that it was fairly easy to adapt the C/80 program to block read and write because the functions compress, expand, and putrep knows nothing about how I/O is accomplished. Except for main's call to fill and flush, the contents of getcmem and putcmem could be replaced by getc and putc itself, and we would essentially recover the C/80 version. In fact, with a bit more code in getcmem and putcmem, even the initial call to fill and the final call to flush could be eliminated. Such structured programming makes it easy to revise individual modules without propagating changes in other modules, which should know only what is necessary to accomplish their tasks. This kind of organization is possible in BASIC, FORTRAN, COBOL, and assembly, but it takes additional effort as compared to writing in C, Pascal, Algol, and other similar languages.

One final note: Any error exit in the MS-DOS program is at level 1. Batch files can make use of this information as needed. For example, if the file compress.bat contains the lines

```
comprs %1 %2
if not errorlevel 1 del %1
```

and if the command 'compress review .dbf review.pak' is given, an error in compression will not result in having the original file deleted anyway.

Figure 4

```
/*
wc - count lines, words, chars in text files
From Kernighan and Ritchie, "The C Programming Language"
     Prentice-Hall, Englewood Cliffs, NJ, 1978, p 18.

Adapted to Software Tools C80 and to Microsoft C by:

        William S. Hall
        3665 Benton Street, #66
        Santa Clara, CA 95051

Usage:  wc <infile {>outfile}.  Output is number of lines, words, characters.

To make the executable in Microsoft C use the command line cl -Ox wc.c.
Follow the documented procedures in C/80.
*/

#define YES 1
#define NO  0
#define MS_DOS YES                              /* Change to NO for C80 */

#if MS_DOS
#include <stdio.h>
#include <fcntl.h>
#else
#include <fprintf.h>
#define EOF -1
#endif

main()
{
    int c, inword;
    long int nl, nw, nc;

#if MS_DOS
    setmode(fileno(stdin), O_BINARY;
#endif

    inword = NO;
    nl = nw = nc = 0;
    while ((c = getchar()) != EOF) {
        ++nc;
        if (c == '\n')
            ++nl;
        if (c == ' ' || c == '\n' || c == '\t')
            inword == NO;
        else if (inword == NO) {
            inword = YES;
            ++nw;
        }
    }
    printf("%ld  %ld  %ld\n", nl, nw, nc);
}
```

Checking the Results

To see if it is worth compressing a text file, use the program, wc, which we have provided (Figure 4). It has been modified to accept compilation by C/80 or Microsoft C, depending on the value of the constant MS__DOS. We assume you have the Mathpak for C/80. If not, change all long declarations to unsigned, replace fprintf.h by printf.h, and hope nothing ever exceeds 65,535 in size. Note that wc may not work very well on some binary files in CP/M, since it will exit at the first Control-Z (0x1a). And, the byte count will not include any carriage returns. These problems are bypassed in the MS-DOS version by opening stdin in binary mode.

It is equally important to see if the file you get back after successive applications of comprs and expand is anything like what you started with. In MS-DOS, use the fc utility with the binary switch to check on a byte-by-byte basis. In C/80, use the program cmp.c provided on the distribution disk. You may find that in C/80, the expanded file may be bigger than the original. This is because C/80 always writes 256 byte chunks, which is 2 logical CP/M records. Hence, if your original program has an odd number of records, is compressed, then expanded, the record size will be one larger. We have had no problem, at least in dBASE II, with the slightly larger files since the position of the Control-Z in the file does not move. However, it is always a good policy to use programs, such as comprs and expand, with caution until you have tested them extensively on the types of files you use.

Conclusion

Even simple file compression techniques can save secondary storage space

Continued on Page 36

We Have What You Want - At The LOWEST Prices!

And we'll get it to you quik, or our name isn't QUIKDATA!

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 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)

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/Z150160 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.

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 85ms drive, WD WX1 winnie controller board, cable set, documentation.

PCW40 - $449 Same as above but with 37ms Segate ST-251.

PCW80 - $695 Same, but 28ms Segate ST-4096 full size drive.

EWIN20 - $510 Complete 21 meg winchester package for the Z100. Includes S100 host adapter card, WD winchester controller, Segate ST-221 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.

BARE WINCHESTER DRIVES

ST-225 - $229 85ms 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.

ACCELERATE YOUR 241/248

With the AMI ZX-386 20Mhz 0-wait state CPU/IO/RAM board. Remove 3 old boards and plug in the new ZX-386 board and in 15 minutes you can be enjoying a computer that's faster than the H/Z386! Can populate the on-board memory to 1 or 2 megs using 256K 100ns DRAMS, or up to 8 megs using 1 mb DRAMs.

This one-board replacement will free up slots, use less power than the H/Z method and give you one parallel and two serial ports. Has a place for a 20Mhz 80387 co-processor ($695 as of this printing). See H-SCOOP #103, or request a copy for a report on this board, or inquire for more information.

We'll even give you a trade-in on your old CPU and I/O board if you desire, and if you need the M256-10 DRAMs, we'll even give you a deal on them!

AMI386 - **SPECIAL** - **$1395** 0K RAM

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-1080I	80 column 120 cps parallel std	$199
KXP-1091I	80 column 160 cps parallel std	$225
KXP-1092I	80 column 240 cps parallel std	$359
KXP-1595	132 col 240cps ser/par std	$490
KXP-1524	132 col 240 cps 24pin par std	$589
KXP-1124	80 col 192 cps 24pin parallel	$395
KXP-3151	132 col daisy 22cps par std	$539

MODEM

We carry a full line of internal and external modems and laptop internal modems:

ZOOMXL	PC internal 1200 baud w/clock/features	$185
ZOOMXL2	PC internal 2400 baud w/clock/features	$275
ZOOMHC	Zoom 2400 baud no frills half-card	$169
H1200	Hayes 1200 baud external	$295
SA24	Practical Peripherals 2400 external	$220

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.

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

Circle #130 on Reader Service Card

J. Stolarz
220 Yost Avenue
Park Ridge, IL 60068

QuickBASIC 4.0* - BASIC "Grows Up"

Some Background

The Z-158 I got in November of '85 was not my first computer and I was already quite fluent in BASIC so I ordered the GW BASIC Compiler at the same time. The reason I bought the "PC Compatible" was so I could send compiled versions of programs for statistical analysis to fellow professors and former students scattered throughout the country. They all had access to IBM-PC's (or compatibles). You can imagine my fury when the dozens of programs I sent out would not run on non-ZDS machines. The Heath/Zenith version of the compiler produced code that ran only on their machines. I had to send out code in BASIC.

At HUGCON '86 I brought this up at a software "bull-session" and someone suggested that I get a copy of Microsoft's QuickBASIC. A vendor on the floor below was selling this at a good price so I added it to the pile of stuff we all used to buy at HUGCON (whether we needed it or not). I compiled those same programs and they worked on all kinds of machines all over the country. I then started using the "programming environment" QuickBASIC provided.

Before I go further, please do not get me in the middle of the competition currently underway between Microsoft and various other companies producing "programming environments." I know there are other excellent programs around and I do not wish to belittle them. I write programs. I do not collect programming languages. I only know what I have used and I am not even sure I know what a "turbo" is.

As I used QuickBASIC 2 [QB2] I found that it was as easy to use as an interpreter. Since labels (or line numbers) were needed only by lines which were targets of GOSUB or GOTO statements, I left them off but fundamentally my programming style did not change much. This is still a feature of QB. It will accept (virtually) all of the code you write for GW BASIC. In a short period of time, the editor with its "pull down menus" became quite familiar.

To my surprise I got a free update to Version 2.1 and I found that it corrected some of the minor errors I found in the compiler. Some months later I was offered an update to Version 3.0 for less than I used to pay for blank disks and I found some of the features of "C", PASCAL, and FORTRAN were added. I could use actual subroutines (not "GOSUBs") which used local variables. Features like SELECT CASE, DO-LOOP, WHILE/WEND, and Block IF statements allowed me to put some logic and structure into my code. Then I ordered the update to QuickBASIC 4.0 and it ushered in a whole new ball game.

Interpreted BASIC

Fundamentally when using a BASIC interpreter you create a single massive program. Since it is difficult to insert substantial blocks of code where it should be (to keep the logic clear) because of the constraints of line numbers, the temptation is to put the code anywhere using GOTOs to jump there and back. Yes, it is possible to renumber, insert, renumber, etc., but this interferes with one's concentration during programming. After adding the needed GOSUB's into this mix, you soon have the kind of code for which BASIC is famous (infamous?).

This, however, is not a limitation of the language. It is a limitation imposed by the way the language has been implemented. The statements used by BASIC and the way they describe the computer operation are very clear and logical. I will go so far as to say that when properly written, BASIC programs are the most readable programs around. When they are arranged in the way interpreters have caused us to arrange them they can be the most atrocious mess in the mundus digitus (if there is such a thing).

This is an important point which is too often overlooked. FORTRAN, C, and PASCAL are compiler languages. There are interpreted versions of these and programming environments for them but they did not start out that way. Apart from its structure, reading someone else's C code can be a slow and difficult process. I say this even though any criticism of this "devinely inspired" language leads me to the fear of being treated as a heretic. There are some routines that are best written in C, but others are best written in other languages. As features are added to BASIC, the difference between it and C are being narrowed. Also, the syntax makes more sense. If you don't know C, what do you guess a function called "scanf()" does? How about having to write a function to clear the screen in text mode?

Programming Environments

When you talk about "compilers" many people still have visions of the way we used to write programs in the past. I spent many years punching my code into cards, one statement per card (yes, N=0 used one card) and bringing them to the computer center. The sequence of compile, test, compile, test, etc., took weeks. But those machines were not as powerful as our Z-100 series computers and if one were to describe a Z-386 back then it would have been considered a fairy tale.

In the "programming environment" an intelligent editor is wedded to an in-memory compiler. As soon as the program is written it can be compiled and run with a few keystrokes (or mouse movements). If a "run time" error is encountered, the program returns to the editor with the line which caused the error "flagged" in some way. All of this happens in seconds. I am currently working on a program that is over 65 pages long using the listing program to be discussed later. It compiles in seconds and I am still using that Z-158 at 8 MHZ and 640K RAM.

Using one of these environments, BASIC programming is as fast as using an interpreter. The real benefit, however, is that once you have the program working you can request an executable, "stand alone" version. This version is a standard ".EXE" program which runs at a much greater speed and will run on any of the millions of PC's around.

Programming environments are available for other languages as well. I have

Microsoft's C 5.0 Optimizing Compiler which includes QuickC which is a programming environment for C. I find it speeds up programming in that language. Now, if they could only do the same thing for Assembler (QuickASS?) .... But I digress.

Structured BASIC

There is no doubt that QuickBASIC 4 is very much influenced by the C language. It is quite unlike previous versions. If you have used Versions 3 or under and have not seen Version 4, this entire discussion might be confusing. If you start programming with it, the structure of the environment leads you to write more readable and more structured code almost automatically. At first glance, the three manuals seem intimidating to the beginning or intermediate BASIC programmer but it is well worth the trouble to learn to use this method of programming.

Actually, one of the books in the package is a discussion of "Selected Topics" in BASIC programming and it is surprisingly readable. It has many very useful examples of code to illustrate its points and if you are like me, you will find yourself patterning your code after these. There are good sample programs on the disks which also contain sections you can modify. The support for graphics in CGA, EGA and VGA is excellent.

All programs entered into the environment are divided into "module level" code and "procedures." As I stated before, you can put all of your code in the module level using GOSUBs and line numbers if you want to. This is sort of like playing "chopsticks" on a concert grand piano. Procedures are either subroutines (like FORTRAN subroutines) or functions. The discussion of "modules" is somewhat unfamiliar and is related to the structure of the 8086 series CPU chips. It would have been easier had the writers of the program used the command PROC rather than SUB. No doubt there was much discussion of this option during its development.

When you define a subroutine (as in SUB PageHeader) the editor puts it in a different place. Subroutines and Functions do not appear on the screen when you are looking at the module level code. You must select these using the drop down menus and dialog boxes. (Get used to this; windowing displays are in the future whether we like them or not.) Variables created in procedures are local to the procedures and must be explicitly declared global or SHARED.

By using menus and the Select Case or Block IF structures, most of the code in programs ends up in procedures leaving the main module code to contain calls to procedures and a menu (perhaps) along with statements declaring variables, creating data types, setting up COMMON storage for programs with multiple modules, etc. I will try to illustrate some of this in the program to be discussed later which, by the way, is also quite useful since QB4 does not provide a separate print utility like QB3 did.

Other Features

Because of the structure of the 8086 series CPU chips in our machines, each "segment" of code is limited to 64 K bytes in size. Another reason for breaking programs into more that one "module" is so we can write programs longer than 64 K in size. QB4 is specifically designed to make this easy, but in a single article the details would take too much time to describe. It is possible, however, to write a program that occupies all of the available memory in the machine for which it was intended.

For example, I am currently writing an elaborate program to keep track of checking accounts. It uses lots of fancy screen effects, color, and an interactive dialogue. Because of its size, it must be well organized and contain quite a bit of comment and blank space. When I load QB4 and inquire as to free program space before loading any program, I find I have 318,172 bytes of free space for program code (and other things like the actual machine code, the heap, etc.). This would differ somewhat on your machine depending on the number of TSR's (terminate and stay resident utilities), declared stack space, file buffers, etc., you have in memory at the time. After loading the two modules of code, I use about 87 kb of space and I am still not finished with the program.

This is the program which covers over 64 pages when listed by the print utility program to be described later. There are over 30 procedures in two modules. When running the program within the environment for testing, a "run time" error branches back to the editor which displays the procedure in which the error occurred with the line displayed in green and an error message in a dialogue box. This is actually more convenient than an interpreter. If the cause is not immediately apparent (sure, one can dream) it is easy to have QB4 print only that procedure to allow for more extensive "head scratching."

There is an intense competition in progress in the development of "programming environments" and QB4 has too many features to describe here. The math coprocessor is supported if it is present; emulated if it isn't. Long integers are now supported along with standard IEEE math support. There is an outstanding "debugging" facility. Data types (as in C) are supported making random access I/O a "breeze." Programs by other suppliers also have extensive features, more being added almost monthly. BASIC as a language is no longer shackled to the limitations imposed by an interpreter.

An Illustration

Perhaps the best way to illustrate how all this works is with an actual program. Rather than show a "trivial" example, I am presenting a program that I use to list programs currently under development with QB4. While QB4 will list a program module (or procedure or "selected text") on the line printer, the listing goes across page perforations and it is not conducive to study. If you "burst" the output and mix it up, getting it back in the right order can be a pain. Listings should also be date and time stamped and page numbered. After a long programming session there are outputs everywhere. The smart money is in companies that sell computer paper.

While QB4 normally saves programs on disk in its own version of compressed binary code it will, on request, save programs in ASCII code so that they can be read by other programs. I decided how I wanted the listing to be arranged and I wrote a short utility to do the job. The listings are so arranged that the output can be burst, perforated, and put in a notebook. It assumes, of course, that you use a printer with a tractor feed. Anyone who does more than a little "hacking" buys one of these and also finds a good source of cheap paper and ribbons.

Using the Program

When you enter a procedure in QB4 it allows you to enter comment lines before the "SUB" statement. Only comment lines are allowed in this position. In the listing I wanted procedures and functions to appear at the top of a new page or separated from the remaining text on a page by a blank space so they can be quickly located. When the procedure is longer than a page in length I also wanted the name of the procedure to appear as a "subtitle" in the listing. To accomplish this I start the first comment line with a "plus sign" after the apostrophe in column one. For example:

```
'+======< Name >======
```

Note that in this line I include the name of the procedure or function.

When it first starts, the program assumes the first section of code is the main program and calls it such in the subtitle at the top of the page. This is a safe assumption. QB4 "saves" modules with the main module placed first and all procedures and functions follow this in alphabetical order. The listing program prints the main program first and continues until it finds a line with an apostrophe in column 1 and a "+ sign" in column 2.

When this is detected, the line is

parsed and the procedure (or function) name is "teased out" and placed in the page subtitle. It assumes all other characters on the line are not alphanumeric. If there are less than 20 lines of code present on the page it skips 3 lines and continues. If there are more than 20 lines present it generates a "line feed" and titles a new page.

The program can easily be modified to use another "symbol" or another strategy to produce page skips. This program matches my programming style and this is a personal matter with programmers. While program formatting is almost a religious matter with C programmers, we "BASIC types" are more relaxed about it. Yes, perhaps too relaxed about it.

Some Comments on the Program Code

Modules in QB4 will almost always start with a list of DECLARE statements for all procedures and functions called or used in the module. These are generated by QB4 when the program is saved and placed in this position. A compiler needs to know that a name is not a variable but identifies a subroutine or function when it encounters the procedure name. BASIC does not require that you declare all variables before you use them and a procedure name is written exactly like a variable name.

All global variables (those which retain their identity throughout the module) should be declared as such before they are used. They can be declared as SHARED in the procedures in which they are used, but this can make the code harder to read. Note that the variables Title, PageNo, and Tb are used in the procedure called PageHeader. Once these are declared as SHARED their values can be altered by any procedure or function in the module. This makes writing long programs much easier. Anyone who has written more than a few BASIC programs has had to track down "bugs" caused by "variable aliasing" where somewhere down the line you redefined the value of a variable by either creating one with the same name or by spelling the name wrong. With a known number of global variables, the "search" function provided by the compiler can track these down faster.

The statement "ON ERROR GOTO ErrorHandler" activates error trapping. This is the only time in QB4 where you must use a GOTO (it is, after all a four letter word) and generate a label. This activates error trapping in all procedures in the module and the routine to handle errors must be at the module level. For an "in house" utility such as this, I was interested only in printing errors. Program listings can get quite long and anything which stops the printer in mid stream could cause you to start over. Disk file access errors would occur here only at the

```
DECLARE SUB PageHeader (Ti$)
 ==============================| PPRINT2 |============================
    Lists ASCII QB4 files on Line Printer.
        CALLS:  PageHeader(Ti$)
 =====================================================================
DIM SHARED Title AS STRING * 30
DIM SHARED PageNo AS INTEGER
DIM SHARED Tb AS INTEGER
Lpage = 1
WIDTH LPRINT 96

ON ERROR GOTO ErrorHandler       'Error trapping for printer

    CLS 0
    PRINT : PRINT
    PRINT TAB(10); "Enter Filename : ";
        LINE INPUT "", F1$
    OPEN F1$ FOR INPUT AS #1
    PRINT
    PRINT TAB(10); "Enter TITLE (30 spaces max): ";
        LINE INPUT "", Temp$
        IF LEN(Temp$) > 30 THEN
            PRINT TAB(15); "Too Long; Re-Enter :";
            LINE INPUT "", Temp$
        END IF
        Title = Temp$
    PRINT
    PRINT TAB(10); "Enter left margin (Col #): ";
        INPUT "", Tb
    PRINT
    PRINT TAB(10); "Enter Starting Page Number: ";
        INPUT "", PageNo
    PRINT
    PRINT TAB(10); "Be sure printer is ready.  Press any Key"
        Ch$ = INPUT$(1)

 *** ----- Start printer output
    Ti$ = "Main Program"
    CALL PageHeader(Ti$)

 *** ----- Begin File read and print loop
    DO UNTIL EOF(1)
        LINE INPUT #1, Ch$
        IF LEFT$(Ch$, 2) = "+" THEN

 *** ----- Extract Proc/Func name from title bar
        Ti$ = ""
            FOR i% = 1 TO LEN(Ch$)
            Tmp$ = MID$(Ch$, i%, 1)
                IF Tmp$ >= "@" AND Tmp$ <= "z" THEN
                    Ti$ = Ti$ + Tmp$
                ELSEIF Tmp$ >= "0" AND Tmp$ <= "9" THEN
                    Ti$ = Ti$ + Tmp$
                END IF
            NEXT i%
        Ti$ = "PROC/FN: " + Ti$
 *** ----- Name is in Ti$

                IF Lpage < 20 THEN
                    LPRINT : LPRINT : LPRINT
                    Lpage = Lpage + 3
              ELSE LPRINT CHR$(12);
                CALL PageHeader(Ti$)
                Lpage = 1
            END IF

        ELSEIF Lpage = 55 THEN
            LPRINT CHR$(12);
            CALL PageHeader(Ti$)
            Lpage = 1
        END IF
        LPRINT TAB(Tb); Ch$
        Lpage = Lpage + 1
    LOOP
    LPRINT CHR$(12);

END
```

start of the program when a "re-start" takes only a few seconds.

Note how the SELECT CASE and Block IF statements make ErrorHandler an easy to read section of code. Remember, it is not a procedure even though it is placed after the END statement. Code that is easy to read is also easy to write and to maintain. If you should wish to trap another error such as, say "92 - Printer just died" all you would need to do is create another ELSEIF. This is a brief and simple exposition; error trapping is not all that easy to use in this environment.

The rest of the code is straightforward. I used the DO UNTIL - LOOP commands for the read and print loop. A WHILE-WEND loop would have worked just as well. A FOR-TO loop is not appropriate here. In similar cases in the past, I used to create a loop using logical "IFs" and a GOTO.

```
 --------------------------- ErrorHandler ---------------------------
ErrorHandler:
    SELECT CASE ERR
        CASE 24, 25, 27
            PRINT
              IF ERR = 24 THEN
                PRINT "Device Timeout"
              ELSEIF ERR = 25 THEN
                PRINT "Device Fault"
              ELSEIF ERR = 27 THEN
                PRINT "Out of Paper"
            END IF
            PRINT
            PRINT "Check printer to correct fault"
            PRINT
            PRINT "Press 'A' to Abort; any other key to continue"
            Ch$ = INPUT$(1)
              IF Ch$ = "a" OR Ch$ = "A" THEN
                  END
                ELSE RESUME
              END IF
        CASE ELSE
            ON ERROR GOTO Ø
END SELECT
 +==========================< PageHeader >============================
    Prints heading for each new page of printer listing.
        GLOBAL Variables:   PageNo Title Tb
 ====================================================================
SUB PageHeader (Ti$) STATIC
    LPRINT : LPRINT
    LPRINT TAB(Tb); "Title: "; Title; TAB(Tb + 71); "Page"; PageNo
    PageNo = PageNo + 1
    LPRINT TAB(Tb); Ti$; TAB(Tb + 69); DATE$
    LPRINT TAB(Tb + 71); TIME$
    LPRINT TAB(Tb + 44); "Microsoft QuickBASIC Compiler V4.ØØ"
    LPRINT : LPRINT

END SUB
```

Note that, except for ErrorHandler, there are no labels or line numbers needed and without much ado, you find it is not necessary to use GOTO or GOSUB statements. There are times, however, when you have deeply nested decision structures and you need to repeat a long section of code on request from the user that a GOTO with a label can actually clarify the code rather than obfuscate it. But avoiding labels is a good idea because they are global to the entire module even though the manuals for some reason do not point this out. While QB4 will catch duplicate labels, it may still be possible to branch to a label in a procedure by mistake. You name the "bonehead" mistake and I have made it.

The subroutine PageHeader uses the word STATIC in the line which declares it. This is not needed here. Using this designation local variables in the procedure (or function) will retain the values assigned to them between successive calls to the procedure. If this word is omitted, a subroutine can call itself. This is called "recursion" and QB4 supports it. This was a difficult concept for me when I first encountered it and I have not written any recursive routines so far. It does allow for some very powerful algorithms when properly used and it can provide some interesting problems when not properly used.

Summary

While using programs such as QB4 requires one to learn some new habits, I feel that anyone with an interest in BASIC will profit from their use. BASIC is a very logical "higher level" language. I know its limitations as well as its strengths. I have written in ALGOL, FORTRAN, COBOL, PL I, C, and MASM. It is possible to link procedures written in other languages to the code you write in QB4. The actual machine code produced by QB4 runs very fast. I noticed, for example, that the screen updates are much faster in this version.

At the end result, the user of your programs has no idea what language you used to produce them and could care less. But despite the clarity and logic to BASIC, we are being pushed to using C or PASCAL. Microsoft's Windows Development Kit was described in a brochure with a statement that said you could use "any of the major microcomputer programming languages including C, PASCAL and MASM." I always considered BASIC as "a major microcomputer programming language" so I bought it. No such luck. You can use only the three languages mentioned. With the advent of the "new" windowing operating systems, will BASIC be no longer supported? I hope this is not the case.

* QuickBASIC, GW BASIC, QuickC are trademarks of Microsoft Corp.

Continued from Page 31

effectively. And providing that compatible programs are available at both ends, such methods can reduce file transmission time between computers and decrease the probability of errors with a small increase in processor time.

References and Sources

[1] B. W. Kernighan, P. J. Plauger, "Software Tools", Addison-Wesley, Reading, MA, 1976.

[2] B. W. Kernighan, P. J. Plauger, "Software Tools in Pascal", Addison-Wesley, Reading, MA, 1981.

[3] The Software Toolworks, 15233 Ventura Blvd, Suite 1118, Sherman Oaks, CA 91403, 818-986-4885.

[4] Lifeboat Associates, 1651 Third Ave, New York, NY 10128.

[5] Generic Software, P. O. Box 790, Marquette, MI 49855, 906-249-9801.

[6] A. M. Tennenbaum, J. J. Augenstein, "Data Structures Using Pascal", Prentice-Hall, 1981.

[7] A software bridge to success (with M. Rabindranathan), Ashton-Tate Quarterly (January/February/March, 1985), 27-33, 64-68.

Getting Started With . . .

Microsoft® Word

Jack W. Bazhaw
900 - 13 Street
Bellingham, WA 98225

Earlier (May 1988 REMark), we took a look at the idea behind Microsoft Word's style sheets. Now we are going to create one. Style sheets allow us to attach complex formatting to all or a portion of a document with a couple of keystrokes, allowing complicated documents to be generated at different times or by different people with a uniform format. Style sheets are one of the features that has set Word apart from most other word processors.

Unfortunately, a lot of people using Word bypass this powerful feature. Like many other skills, once learned, you will wonder how you ever got along without it. To help you, here is an in-depth look at style sheets.

Even if you never create a style sheet you are still using one. Word will, absent specific instructions to the contrary, attach the style sheet normal.sty to any document you create. If you choose Gallery and look at normal.sty, it will appear to be blank. However, it has several styles already defined. Your division (page) formatting will be set to top and bottom margins of one inch, left and right margins of 1.25 inches, running heads will print one-half inch from the top and bottom, paper dimensions are 8.5 inches by 11 inches, page numbering is turned off, but set to print one-half inch from the top and 7.25 inches from the left, footnotes will print on the same page and line numbering is turned off. Paragraph formatting will be block style with no added spacing before or after paragraphs and a character or font will be assigned to all text. The exact font assigned will depend upon which print driver you have loaded; it will be the first listed font after pressing ALT-F8. If you load the HPLASER3 print driver, you will be able to follow along and match on screen what happens.

There are several other styles that Word defines. The standard styles are visible if you press F1 in the "variant" field of Gallery Name. The standard styles all have a name instead of just a number; two have "standard" as part of their name.

Let's develop a style sheet for a hypothetical document to demonstrate the creation and power of style sheets.

Our project will be a report with chapter headings, footnotes, index, table of contents, and two special paragraphs in the main body. Word 4.0 users also have the option of creating the styles with Format Stylesheet Record; all versions can follow the steps in this example.

Start Word with no document loaded. Before creating any text, we will make up the style sheet. Select Format Stylesheet Attach (press ESC F S A) and type in the name of our style sheet, "project" (Word will add the extension ".sty").

Bring up the style sheet by selecting the Gallery (press ESC G); it will be blank. In the lower right corner, "PROJECT.STY" should be displayed.

First, we will define the standard division, paragraph and character formats, then the more specialized formats.

Start with the standard division; choose Insert. Press TAB to go to the "usage" field (key code will be left blank), then press D to select Division, press TAB to go to the "variant" field, then press FT for the list. "Standard" will be highlighted, which is our selection; press TAB to go to the "remark" field and type in "STANDARD DIVISION". Press ENTER and you will see all the default formatting Word applies to the division.

The boss likes a lot of white space, so select Format Margins and change the top and bottom margins to 1.25 inch and the left and right margins to 1.5 inch. For a laser printer, change the page width to 8.0 inches. All the other defaults are fine, so press ENTER.

Now on screen you will see the formatting that you have applied to the division.

Choose Insert again for doing the standard paragraph. Once again, leave the key code blank, select paragraph from the "usage" field, TAB to the "variant" field and press F1. "Standard" is again our choice, so TAB to "remarks" and type "STANDARD PARAGRAPH", then press ENTER.

Notice that Word has already assigned the standard character to this paragraph; default is Courier in 12 point. Our standard will be Times Roman in 10 point, so press ALT F8 to change the font. Press F1 to see the list of fonts, highlight TMSRMN, press TAB, then F1 again to see the list of sizes, highlight 10 and press RETURN.

You should see the selected font name, TMSRMN, followed by its generic name, roman a, followed by a fraction "10/12". It really is not a fraction, but shorthand notation for the selected point size, 10 point and the line spacing, 12 point. Word uses a line spacing of 1 line (12 points or 6 lines per inch) as the default line spacing.

Set up the paragraph format by choosing Format Paragraph. Set the "first line" indentation to 0.3 inch and "space after" to 1 line. In most cases, 10 point type set on a 12 point line, which gives 2 points of leading which would be adequate. But the boss wants white space so change "line spacing" to 13 points by typing "13 pt" and pressing RETURN. (REMark articles are usually set in 9 point Optima on a 10 point line.)

Our report will have footnotes, so we will define the standard characters to be used for the reference mark and the footnote text.

Choose Insert; since the "usage" field already has "character" selected, press the down arrow to go to "variant" and press F1 for the list. Highlight "Footnote ref", press TAB, type "FOOTNOTE MARK" and press RETURN. We want the footnote mark to be 8 point superscript Helvetica, so press ALT F8 and select the type face (HELV), size and superscript.

For the footnote text choose Insert, select "paragraph" for "usage" and press F1 in the "variant" field to see the list. Highlight "Footnote", press TAB, type "FOOTNOTE PARAGRAPH" and press RETURN. Then use Format Character and Format Paragraph to set up an 8 point Helvetica font, justified paragraph, zero indentations and space before and after, with a line spacing of 9 points.

You may be wondering why we have not assigned "key codes" to any of the formats we've defined. These are the standard formats which will automatically be applied to the document. By leaving off the key code assignment, the style bar will show a code only when we depart from the standard format and makes the display less cluttered. A drawback to this method is that Word turns off the style bar key code display if you change anything by the keyboard formatting keys. So if you change the formatting of either a standard paragraph or a special paragraph, they will both have a blank key code shown in the style bar and could be mistaken for a standard paragraph. If you make all your changes with the style sheet, this is not a problem; also, you can add key codes at any time.

The remainder of our formats will have a key code defined because they are not defaults, but must be applied with the key codes.

Chapter headings need to be defined as a centered paragraph, with 2 lines of spacing after and 18 point line spacing with "keep follow" set to yes; the font is to be 14 point bold Helvetica. This time, after choosing Insert, type in "ch" as the key code for chapter headings. In the "usage" field, select "paragraph", press F1 in the "variant" field to see the list. We have already defined the formats for "Standard" and "Footnote" — that is what the "( )" after each is telling us. Click on "1" with the left mouse button, press TAB, type "CHAPTER HEADING" in "remarks" and press RETURN. Now use Format Character and Format Paragraph to set up the desired formatting.

In a similar fashion, set up the following four paragraphs:

Quotations	Key code: QT Alignment: Left Indents: Left 0.4; First 0; Right 0.4 Line Spacing: 10 pt; space before 0; space after 1 Font: Times Roman, 8 pt
Hanging	Key code: HP Alignment: Left Indents: Left 0.4; First -0.4; Right 0 Line spacing: 13 pt; space before 0; space after 1 Font: Times Roman: 10 pt
Contents	Key code: TC Alignment: Left Indents: Left 0; First 0; Right 0 Line spacing: 13 pt; space before 0; space after 1 Font: Times Roman, 10 pt Tabs: 0.4 inch, right flush, with leader dots
Index	Key code: IN Alignment: Left Indents: Left 0; First 0; Right 0 Line spacing: 9 pt; space after 0, space before 1 Font: Times Roman, 8 pt

For all except the index paragraph, when you click on the "variant" field, you can use the selection that Word proposes. For the index entry, select "Index level 1" for the variant.

There remains one more format to specify — a second division or page. The index will consist of lines with only a few words and would look better set up in more than one column on the page. Choose Insert, type in a key code (I often use a "/" for the second symbol in divisions); for our sample I'll use "i/". Select "division" for "usage", keep Word's proposed "1" for "variant" and in "remarks" type, "INDEX DIVISION". The margins etc. from our standard division will be set; all we need to do is change the "Layout" to two columns. Leave the space between at 0.5 inch.

At this point, the styles for our style sheet are complete, but let's add a name tag to the sheet. Although Word can print your style sheet, it does not include the name. By assigning one style that is really just a name for the sheet, we will be able to identify the style sheet in its printed form. Create a character style; I use "variant" 25 as it is doubtful a single document would use that many font changes (unless it's a ransom note). In the "remarks" section, type in the name of the style sheet. If you make this the first (or last) style assigned, you can look in the same place on every printout for the identification of the particular sheet. Table 1 is a printout of our sample style sheet.

Now the fun begins. Turn on Word's style bar (choose Window Options and select "yes" for style bar) and type in your report. You can also load an existing document; you may see some differences due to existing formatting. Be sure the PROJECT.STY is attached to the document. Each paragraph should have the formatting you selected for the standard paragraph. Since no key code is assigned, the style bar will be blank.

Create a footnote (choose Format Footnote and press RETURN, then type the footnote text). The footnote reference mark should be a superscript character and the footnote text should be the smaller font and justified. Again, the style bar should be blank.

Select a paragraph that is to be a hanging paragraph and press ALT HP. Now the style bar should show "HP" and the first line should hang to the left over the other lines.

Use the same procedure to format other paragraphs that are not the standard paragraph. In each case, you should see the key code displayed in the style bar.

The index material will need division formatting in addition to the paragraph formatting. Go to the start of the index material. Start a new division by holding CTRL and pressing RETURN at the same time. You should see the division mark which is a line of colons (:) across the screen. Make sure the cursor is still in the index material (after the division mark just created). Then, press ALT I/. Another division mark should appear at the end of the index material and the style bar should show "I/".

"What", you ask, "did we gain by all this 'extra' work?" "What is wrong with just typing in the text and applying the styles as I go?" I thought you would never ask. Picture this: Friday afternoon, the boss comes in with your project and says he can't possibly write in his edits unless it is triple spaced. In addition, he would like to see what it would look like if it was justified text instead of ragged right. And also, the printer wants us to use Helvetica instead of Times Roman. What will that make it look like?

Had you used paragraph-by-paragraph formatting you might be there until Monday morning making these changes. With style sheets, it takes only a few minutes. Here is how to do it.

Load your document and choose Gallery. Save the style sheet, PROJECT .STY, under another name, say PROJECT2.STY. Make the style changes needed (don't forget the name tag style) and resave PROJECT2.STY. Exit the Gallery, attach PROJECT2.STY to your document and print. Repeat as often as needed, using a different style sheet for each change; go home at 5 PM.

Glossary

Font. A complete assortment of a given size of type.

Hanging indention. The first line of a paragraph set flush left and subsequent lines indented.

Indent. To set a line of type so that it begins or ends inside the normal margin.

Justify. To space out lines of type to a specified measure.

Leading. Extra spacing between lines of metal type. A lead (pronounced led) is a thin strip of metal the length of the line, 1, 2, or 3 points thick. By extension, the spacing between lines set by other than casting in metal.

Measure. The length of the line in which type is set.

Point. The printer's basic unit of type measurements — 0.0138 inch (approximately 1/72 of an inch).

(From The Chicago Manual of Style, Thirteenth Edition)

HEPCAT is here! *HEPCAT* is here! *HEPCAT* is here! So what is *HEPCAT*, you may ask? Why it's just another Pat Swayne SUPER-UTILITY. *HEPCAT* is an acronym for *HUG Engineer's and Programmer's CAlculation Tool*. Just what we don't need, another memory resident calculator, right? Wrong! With *HEPCAT*, you can throw away the rest and use the best. *HEPCAT* only uses two partial lines on your screen, and best of all, does NOT cause existing programs to stop executing! That means, while your computer is grinding numbers internally, you can be grinding them externally. Order ***HUG P/N 885-3045-37***.

Table 1

1	I/	Division	INDEX DIVISION	Page break. Page length 11"; width 8". Page # format Arabic. Top margin 1.25"; bottom 1.25"; left 1.5"; right 1.5". Top running head at 0.5". Bottom running head at 0.5". 2 columns; spacing 0.5". Footnotes on same page.
2	IN	Paragraph Index Level 1	INDEX PARAGRAPH	TMSRMN (roman a) 8/9. Flush left, space after 1 li.
3	TC	Paragraph 4	TABLE OF CONTENTS	TMSRMN (roman a) 10/13. Flush left, space after 1 li. Tabs at: 4" (right flush, leader dots).
4	HP	Paragraph 3	HANGING PARAGRAPH	TMSRMN (roman a) 10/13. Flush left, Left indent 0.4" (first line indent -0.4"), space after 1 li.
5	QT	Paragraph 2	QUOTATIONS	TMSRMN (roman a) 8/10. Flush left, Left indent 0.4", right indent 0.4", space after 1 li.
6	CH	Paragraph 1	CHAPTER HEADING	HELV (modern i) 14/18 Bold. Centered, space after 2 li (keep with following paragraph).
7		Paragraph Footnote	FOOTNOTE PARAGRAPH	HELV (modern i) 8/9. Justified.
8		Character Footnote Reference	FOOTNOTE MARK	HELV (modern i) 8 Superscript.
9		Paragraph Standard	STANDARD PARAGRAPH	TMSRMN (roman a) 10/13. Flush left (first line indent 0.3"), space after 1 li.
10		Division Standard	STANDARD DIVISION	Page break. Page length 11"; width 8". Page # format Arabic. Top margin 1.25"; bottom 1.25"; left 1.5"; right 1.5". Top running head at 0.5". Bottom running head at 0.5" Footnotes on same page.
11		Character 25	PROJECT STYLE SHEET	TMSRMN (roman a) 10.

IMPROVE YOUR C PROGRAMS WITH LINKED MODULES

Clement S. Pepper
12938 Orangeburg Ave.
San Diego, CA 92129

Some time ago I wrote a program of some 23,000 bytes in length. Later I wanted to add some new features. I found picking up where I had left off to be harder than I would have believed. I should have known better than to write that particular program in one continous chunk. It was definitely time to modularize.

Writing our C programs as modules to be compiled and linked does have it plus and minus sides. It means devoting thought and planning to what we really have in mind for this program before cranking out reams of code. Of course, this plus mostly looks like a minus when we are itching to get our hands on a keyboard.

Not every program can or should be modularized. Length is but one of several factors, and is not neccessarily of greatest import. Some compilers are not readily amenable to linking program modules. The Software Toolworks CP/M C/80, for example, will provide linkable .REL modules, but I had to purchase a macro assembler to make use of it. MS-DOS has been more generous in that the distribution disks include two files, LINK.EXE and LIB.EXE, that permit module linking and library creation. (But to program in assembly I had to purchase the assembler.)

Why Modularize?

The best reasons from my point of view are two. The first is the easing of organizational type problems when putting together a multi-faceted program. The second is the assistance afforded to its understanding later when memory has grown cold.

Many short programs have a way of growing. Planned growth takes a little longer up front but can reduce the pain later as we struggle to integrate the inevitable hot new feature. Programs constructed as functional modules are more readily modifiable for new applications.

We often find ourselves using the same functions repeatedly. These can be put into utility modules to be linked in as appropriate.

Each change to a C program, no matter how trivial, requires re-compiling. Long programs not only take a longer time, they make more demands on our disk drives. We do not have to do a lot of programming for this to have a negative effect on their life expectancy.

Tracking down a small bug in a large program can be a major exercise. Locating the elusive little critters is a lot easier in the limited search area of a module.

What Is A Module?

A module is not defined by the number of bytes it contains. It may be as brief as a single function. The definition I like is a sub-task that can be compiled and run on its own when linked in with a defining module containing main() and, in general, a utility header module.

With this approach, the defining module carries the name of the final program. The utility module has the conventional header extension ".h" to distinguish it from a program segment. Each module has a name identifying its role in the program. If the program name is NEW_YORK.C its utility could be CITY_HAW.H with modules QUEENS.C, STATEN_C, and BROOKLYN.C.

In developing a new module, I often construct it as a short stand alone program until it is performing as I require. I then replace main(), with a function name appropriate to the module's role in the program. This call is then added to the function main() in the defining module. When globals from other modules are involved the header file is modified (if neccessary) and linked in.

How To Write A Modular Program

Every C program must have one, but only one, function main(). It is reasonable to put main() in the module having the program's name. The role played by main() in this module is to call the leading function in each of the modules. In other respects, the coding of a module is no different from any program we write, except for the care that must be taken with global declarations.

Local variables are not a problem as they are known only within the function in which they are declared. Globals used only in a single module and declared in that module are not a problem either. A potential difficulty arises when a global is declared in two or more modules. A solution is to declare the global as "extern" (the quotes are mine) in those modules following that in which it is initially declared.

If globals are declared in this manner, then we end up with declarations strung out from module to module. It is surprisingly easy to encounter bookkeeping problems with this approach. There is a safer way for variables likely to be used in more than a single module. That is to make all such declarations in a single header file.

The header file can include any needed utilities. Examples are functions we may wish to use for cursor positioning, character attributes, screen displays, and the like.

With this approach, our program consists of three modules. Each module is compiled to provide a relocatable module. So, what is a relocatable module?

A relocatable module is one that can be linked with other similar modules to produce an executable file. With CP/M, relocatable modules have the extension .REL. With MS-DOS, the extension is .OBJ. CP/M executable files have the extension .COM. With MS-DOS, the extension is either .COM or .EXE. In general our MS-DOS C programs will execute as .EXE files.

With MS-DOS, all C compilers will provide a relocatable module as an .OBJ file. This is a first requirement for linking with library modules to provide the .EXE file. Such is not the case with CP/M compilers. C/80, for instance, has its own assembler, AS.COM, which reads the .ASM file provided by the compiler and produces the executeable .COM file with no intermediate code. However this, and similar compilers, provide an option by which .REL modules can be delivered. In this situation no executable file is provided by the compiler.

Now that we have our relocatable files, we require a means of linking them together to form the executeable. In a conceptual sense, the requirements are the same for CP/M or MS-DOS based systems. In reality, distinct differences exist.

Using Microsoft MACRO-80 with CP/M programs

I got my start with the C language back in 1982 using The Software Toolworks C/80 Version 2.0 with my Heath H-89 computer. I am indebted to Dr. Grant Gustafson, (Viking Software), for much of my know-how with C. It was Grant who introduced me to MACRO-80.

Before attempting to use MACRO-80 or another macro assembler with your programs, review your compiler manual carefully for procedures to be followed.

Using The Software Toolworks C/80 manual as a reference:

1. Normal compiling of a C/80 source program provides a file coded in assembly with the extension ASM. The C/80 distribution disk contains an assembler, AS.COM, an absolute 8080 assembler for the assembly of the C/80 compiler's output files. The output of AS.COM is the executable .COM file.
2. To perform assembly with the Microsoft MACRO-80 or Digital Research's RMAC requires a switch in the command line entry.
   a. For compatibility with MACRO-80, the switch is -m.
   (yourprog.c -m [other args ...]
   b. For compatibility with RMAC, the switch is -m2.
   (Note that C/80 can be configured for either switch as the default.)
3. Assembly by M-80 or RMAC produces a file with the extension REL. REL files are linked by L-80 (LINK if using RMAC) to create the COM file. The last file to be linked must be CLIBRARY.REL; the replacement for CLIBRARY.ASM. (When the .ASM assembler is used, CLIBRARY.ASM is linked in automatically.)
4. The primary advantage in using a MACRO ASSEMBLER is the ability to construct our programs as consecutive modules which are individually compiled and linked to comprise an executable file. There are, however, concerns of which we must be aware.
   a. We must be familiar with the constraints of the assembler/linker-loader we are using. For example, the length of global variable names with MACRO-80 is restricted to six characters except, as warned by the C/80 manual, version 3.43 of MACRO-80 which employs 7 characters and is not compatible with the earlier versions.
   b. When the same global variable is employed in multiple modules, the first module in which the variable is used provides the initial declaration. Succeeding modules do not simply repeat the declaration. Instead the term "extern" (without the quotes) is to precede the declaration. Each linked module in which the global is used must include the extern declaration.
   c. When we use C/80 with AS.COM with a program calling PRINTF, we must have the line #include "printf.c" at the beginning of our program. When we assemble with a macro assembler we must change the include to one of the following: #include "printf.h" or #include"stdio.h." This can vary between compilers, so follow the dictates of whatever your compiler manual specifies. Frequently the double quotes are replaced with < and >.
5. It is often to our advantage to link frequently used functions into a single library function. We may or may not wish to include our sources's library in this. Procedures for constructing libraries will vary between compilers and with some it may not be possible to include the library. MACRO-80's LIB-80.COM and C/80 do permit construction of our own private libraries. LIB-80 warns us that it is possible to destroy libraries and we should always have a backup copy of existing libraries before attempting operations with LIB-80. A number of switches are used with LIB-80 to provide specific capabilities.
6. When the MACRO-80 command line includes the /C, the assembler opens a cross reference (.CRF) file in lieu of a .PRN file. The .CRF file contains symbol cross references that are often helpful in tracking down errors in our programs.

A Summary of the MS-DOS LINK Program

LINK is an MS-DOS utility that lets us combine one or more of our object (.OBJ) files into a single relocatable load module. .OBJ files are intermediate between .ASM and .COM (or .EXE) in MS DOS. (With CP/M, the intermediate file has the extension .HEX.) Library files can be included as well as .OBJ. LINK will also produce the executable run file when instructed to do so.

Our MS-DOS manual provides a detailed description of LINK. The following summary, excerpted from the manual, illustrates features helpful to C program module linking.

LINK is involved by one of three methods.

1. COMMAND LINE PROMPT. Type LINK then press RETURN. LINK prompts us for instructions. (The options are listed in method 2 following.)
2. COMMAND LINE ENTRY. Type LINK followed by options as shown below.

```
LINK Kobject-list,runfile,listfile,lib-list[
```

   where LINK invokes the LINK command;
   * object-list is a list of object modules. A drive and/or path name specification are optional inclusions. A + sign is inserted between modules. The .OBJ extension may be omitted.
   * runfile is the name we include for the executeable file. Drive name and/or path name specification are optional inclusions.
   * listfile is the name we include for an optional list file. Drive name and/or path name specification are optional inclusions.
   * lib-list is a list of library files. Up to 10 may be included. They must have been created by a library utility such as LIB. A drive and/or path name specification are optional inclusions. A + sign is inserted between modules. The .OBJ extension may be omitted.
   * /x represents any one of the following optional switches:
   /DISALLOCATE - Load data at high end of data segment.
   /HIGH - Place run file as high as possible in memory.
   /MAP - List all global symbols with definitions.
   /LINE NUMBER - Include line numbers in the list file.
   /PAUSE - Pause linker session.
   /STACK:number - Set fixed stack size in run file.
   /NO - No default library search.

We must use CTRL-BREAK to recover from an erroneous response. We have pressed the RETURN key and then recognized a mistake in our entry. We can use CTRL-BREAK to exit LINK and start over.

3. RESPONSE FILE METHOD. Type LINK @filespec.
   LINK invokes the LINK command;
   @ tells LINK that all prompt responses are contained in filespec;
   filespec is the name of our file containing the LINK instructions.

These are written with a text editor in filespec in the same sequence as otherwise entered with method 2. Placing commonly used instructions in a response file saves the repeated typing.

LINK is a relocatable linker designed to link together modules of 8086/8088 object code. The output (run) file can be loaded and executed at any convenient memory address. LINK employs a dictionary-indexed library search method which minimizes library search time. Modules totalling up to 900K bytes can be linked. LINK resolves external references between reference modules and can search multiple library files for external references not otherwise resolved. The

```
/* FRIENDS.C  An example program for module linking  */

/* ==  function prototypes (optional)  == */
void f_cmptr();
void hpy_usr();
void lament();
void remedy();

main()  /* Begin Program */
{
  f_cmptr();    /* == FMOD1 function call == */
  hpy_usr();    /* == FMOD2 function call == */
  lament();     /* == FMOD3 function call == */
  remedy();     /* == FMOD4 function call == */
   exit(Ø);
}

/*  F-UTIL.H   Utility for FRIENDS.C  */

/* == Global Declarations == */
char *MESS1 = "\n\nHi! I'm you're friendly computer.  Who're you?\n";
char *MESS2 = "\nA. Happy User. Pleased to make your acquaintance.\n";
char *MESS3 = "Anything worth an interrupt going on?\n";
char *MESS4 = "\nJust looping about hoping for a call.\n";
char *MESS5 = "\nI'm a C programmer. Here's betting I'll clear \
your problem.\n";

/*  FMOD1.C  First module to link in program FRIENDS.C */

#include <stdio.h>
extern char *MESS1;

void f_cmptr()  /* Friendly computer introduces self */
{  printf(MESS1);  }

/* FMOD2.C  Second module to link in program FRIENDS.C */

#include <stdio.h>
extern char *MESS2;
extern char *MESS3;

void hpy_usr()  /* A. Happy User introduces self */
{  printf(MESS2);
   printf(MESS3);  }

/*  FMOD3.C  Third module to link in program FRIENDS.C */

#include <stdio.h>
extern char *MESS4;

void lament()  /* Friendly computer's lament */
{  printf(MESS4);  }

/* FMOD4.C  Fourth module to link in program FRIENDS.C  */

#include <stdio.h>
extern char *MESS5;

void remedy()  /* Happy user's remedy */
{  printf(MESS5);  }
```

Listing 1
Six program modules we can link to obtain the run file FRIENDS.EXE.

```
frnd_lbb
+a:friends +a:futil +a:fmod1 +a:fmod2 +a:fmod3 +a:fmod4
frnd_lst
```

Listing 2
The response file I used with LIB.EXE to link the six modules into the single file FRND__LBB.LIB.

optional list file shows the external references resolved.

All errors other than "NO STACK SEGMENT" cause the session to abort. Error messages are provided.

The Software Toolworks TOOLWORKS C is an example of a C compiler utilizing LINK for combining .OBJ modules to obtain a runnable file.

TOOLWORKS C varies significantly from its CP/M predecessor, C/80, in its processing of our source programs. Where C/80 performs its compiling in a single pass, TOOLWORKS C requires two. PASS1 reads a .C source file and outputs a temporary file having the extension .$$$. PASS2 reads the .$$$ file and outputs a file with the extension .OBJ. In the process the .$$$ file is erased. The command line for both passes may include switches for a variety of compiler options.

The TOOLWORKS C distribution disks contain two files of importance to the linking of compiled source modules: CRT0.OBJ (zero, not O), and CLIBRARY.LIB. We must use these with the MS-DOS LINK program to produce a runnable file. The command line for linking a single .OBJ file is:

```
LINK CRTØ+OURFILE,OURFILE,NUL,CLIBRARY
```

CRT0.OBJ is a TOOLWORKS C file that performs certain stack and memory environment operations. The NUL term is in the LIST FILE location. Replacing NUL with a filename will give us a .MAP file that can be helpful with resolving program difficulties.

If our system utilizes two floppies or a hard disk and a floppy we may have to insert their directory names; A:LINK, for example. This is a frequent requirement when using a ramdrive.

The above command line will link CRT0.OBJ with OURFILE.OBJ and the CLIBRARY to yield OURFILE.EXE. To link in additional modules we continue the + sequence - that is,

```
LINK CRTØ+OURFILE+MODULE1+MODULE2,OURFILE,
                                  NUL.CLIBRARY
```

combines three source modules to produce OURFILE.EXE. If a command line ends with a + sign, the prompt will reappear on the next line to enable our continuing with module entries.

As with C/80 and MACRO-80, we must keep track of globals with proper sequencing in our linking and use of the "extern" declaration where the variable is used in more than a single module.

A Summary of the MS DOS LIB Program

LIB is a utility provided with the MS-DOS system that lets us create or modify library files. These must be compatible with Microsoft's LINK utility. Using LIB we can combine existing .OBJ files to construct libraries unique to our specific program needs. Our libraries may range from the general for use with the majority of our programs, to specific creations for specialized application.

If your C compiler uses the MS-DOS LINK.EXE, it can use LIB as well. Even a compiler that provides its own version of LIB may function with LIB.EXE. For instance, Turbo C .OBJ files made with either version (1.0 or 1.5), can be combined into library files with LIB. This is a real plus in that version 1.0 does not provide a library program. Of course with compilers such as the Software Toolworks TOOLWORKS C which routinely utilize LINK, we must employ LIB if we are to combine our .OBJ files to create new LIB modules or modify those we have.

C, by the way, does not restrict us to a single library. While we may link libraries of our own in with those provided by our compiler or other sources there is no requirement to do so.

Five basic actions LIB performs are:
1. Creation of a new library file.
2. Replacement of an existing library module.
3. Changing .OBJ files to library modules. These may be appended to an existing library file.
4. Extracting modules from an existing library file to create an .OBJ file. (The module is not deleted.)
5. Module deletion from an existing library file.

As a final step, LIB creates an index for use by LINK in identifying modules and PUBLIC symbols. (C globals are PUBLIC.)

When requested LIB will write a cross-reference listing of the public symbols in the library.

The first step in using LIB is to invoke it. This may be done by one of the following three methods:
1. Command prompt LIB
   A simple command line entry LIB. With this method we must provide answers to specific queries made by LIB. The prompts enable LIB to ascertain our requirements and to configure itself to respond to them.
2. Command line LIB library operations list, [listing]
   As above, except that desired options are included on the command line before pressing ENTER. This will often be the method of choice. [listing] is the file name you give in response to the query following the operations list entry. The listing is of the module names and the globals in two groupings.
3. Response file LIB @filespec
   A "response file" is a text file listing of the responses to the LIB command prompts. LIB is then invoked by entering LIB @filespec, where filespec is the file name we assign. @ is a pointer to inform LIB that the responses exist in the file name following. We create filespec using any ASCII editor. A new text line is used for each response. Responses must be listed in the sequence in which they appear. In general, the method follows the same rules as the command line method. This method is advantageous when developing a large program in which frequent library changes must be made.

LIB recognizes the following command characters: (+, - and * are command characters for manipulating modules.)

+ The + sign followed by an .OBJ file name appends file as the last module in the library named in response to the Library file prompt.

- The - sign followed by an .OBJ file name deletes that file from the library. LIB then closes up the gap left by the deletion. New modules, replacements included, are always added to the end of the library.

* The * followed by the module name extracts that module from the libray and places it in a separate object file. The module remains in the library. The module name is used as the file name. LIB adds the default drive name and the file extension .OBJ to the module.

These command characters may be entered in any order.

Other command characters:

; When followed by RETURN, instructs LIB to use default responses for the remaining prompts.

& Use this sign to extend the command line when required with the operations prompt. This sign, followed by RETURN provides an addition operations line.

CTRL-BREAK Aborts the library session, returning the system to DOS. LIB must be re-invoked to continue.

Error Messages:

LIB provides a message related to the error. Fatal errors require aborting LIB and starting over following the error correction.

Using Turbo C's LINK and TLIB

Just about anywhere we look in the computer arena we observe a surge of growth verging on the unbelieveable. The C language is right in there trekking along with, if not in front of, the rest of the industry. A comparison with version 2.0 of C/80, circa February 1982, my introduction to the language, with version 1.5 of Turbo C is easy as there is so little in common to compare. With an extremely important exception: all the basics of the original Kernighan and Ritchie's "The C Programming Language" reference have been preserved. Many of the new ANSI standard features to be released are already present.

Keep in mind this writing is by a hobbyist - very powerful versions of C have been available for years, but these have been largely out of reach for those of us who have programmed for enjoyment. But Turbo C and its Microsoft Quick C counterpart are professional quality releases at affordable prices. These are far more than C compilers of course, they constitute language systems.

What follows is based on my experience with Turbo C, but from reviews of Quick C and conversations with a user, what is true for one holds pretty much for both.

With these we may opt to program in an "Integrated Development Environment." In this mode we utilize a dedicated editor provided with the system for writing our programs. Working within the environment one can write the source code, compile it, identify and clear errors and execute the completed run file. A variety of compiler and linker options are available and a built-in project-make facility.

The distribution also includes a full complement of command line options for compiling, linking, making and running programs.

I prefer the command line approach. In part, I suppose, because it conforms to past practice and I am comfortable with it. I have an editor I enjoy using (PC Write). I do my compiling with a ramdisk using elements of ineptness and all.

I don't want to get deeper in this. My reason for bringing it up is that the procedures I describe are based on the command line approach as that is my way of working.

The three files of particular application to this article are the compiler, TCC.EXE; the linker/loader, TLINK.EXE; and TLIB.EXE (version 1.5 only). There are two other file types of import: INCLUDES, both .C and .H; and LIBRARY, .OBJ and .LIB. For these two directories are created: TURBOC\INCLUDE and TURBOC\LIB.

If we are working in our default directory (that is, the directory containing these files) these are all we need. If we are working from two directories with the INCLUDE and LIB sub-directories separated from TCC and TLINK, we must provide an information file, TURBOC.CFG contains the directory information for these.

I do my programming with four directories. This, in part, because my hard disk is partitioned into the directories C:, D:, E: and F:. Beyond that, I don't like keeping my data on the hard disk, so I use a floppy in drive A: for that. I do all the compiling, linking, and library creation in a ramdrive, directory G: with the work saved on the floppy. .BAT files are used wherever possible. Sounds complicated, but it works out very well. With this arrangement I can switch between TOOLWORKS C or Turbo C with very little change in procedure.

When all of our program's source code is within a single file (or TURBOC.CFG is in residence) we can use TCC alone with the command line:

```
tcc our_prog.
```

```
fmod1!.......fmod1          fmod2!.......fmod2
fmod3!.......fmod3          fmod4!.......fmod4
friends!.....friends        futil!.......futil
_f_cmptr.....fmod1          _hpy_usr.....fmod2
_lament......fmod3          _main........friends
_MESS1.......futil          _MESS2.......futil
_MESS3.......futil          _MESS4.......futil
_MESS5.......futil          _remedy......fmod4

friends      Offset: 200H   Code and data size: 14H
 friends!      _main

futil         Offset: 400H   Code and data size: FDH
 futil!        _MESS1        _MESS2        _MESS3
 _MESS4        _MESS5

fmod1        Offset: 600H   Code and data size: 9H
 fmod1!        _f_cmptr

fmod2        Offset: 800H   Code and data size: 11H
 fmod2!        _hpy_usr

fmod3        Offset: A00H   Code and data size: 9H
 fmod3!        _lament

fmod4        Offset: C00H   Code and data size: 9H
 fmod4!        _remedy
```

Listing 3
The list file for FRIENDS from LIB.EXE.

We can include the .c extension if we like, but it is not needed. If all is as it should be, the compiler creates the .OBJ file and calls the linker to finish up.

Should there be a fatal error in our source code the compiler will list the error and exit. Sometimes there is something questionable in our code and the compiler will issue a warning, then continue on with the linking. I have learned there is substance to these warnings and it is smart to check them out before proceeding.

When our program is constructed of modules, the command line to produce just the .OBJ file includes the switch -c preceding the file name:

```
tcc -c our_prog.
```

We can still use TCC to compile one or more .C modules, link together previously compiled .OBJ modules, read in functions from the INCLUDE and LIB directories and deliver a fresh, smoking hot .EXE file. The command line for all of this has the format:

```
    tcc -enew_york new_york.c city_haw
          .obj boroughs.lib brooklyn.obj
```

where

```
    boroughs.lib consists of manhat.obj,
          queens.obj, .....
```

-e is a switch instructing the linker to produce an .EXE file from the filespec immediately following. Note there is no space between this switch and the filespec.

If we screw up on declaring our globals we may get an incredible can't find or multiple definition complaint list from TLINK.

The rules for using TLIB are very similar to those for LIB.EXE. I haven't attempted to create a .LIB file from TOOLWORKS C .OBJ modules and haven't any plans to. However, according to the Turbo C description we should be able to.

From the description given in the manual we can use TLIB to:

* create a new library from a group of object modules
* add object modules or other libraries to an existing library
* remove existing modules from an existing library
* replace existing modules from an existing library
* extract object modules from an existing library
* list the contents of a new or existing library

When modifying an existing library, TLIB will first make a backup copy of the original. This file has the .BAK extension.

The procedures for TLIB are very similar to those described for LIB. That is, the command prompt method, TLIB; the command line method, TLIB followed by the lib file name, the operations, and the optional listfile name. And the response file method.

TLIB maintains a directory of all the public symbols (globals) used in the library. If we attempt to add a module containing duplicate declarations, TLIB displays a message and refuses the module. If the duplicate differs in case, TLIB can be forced to accept it by use of a /c option. Without this option, TLIB will reject an upper case duplicate of a lower case global.

I have skipped over one other feature of Turbo C, which is the command line use of TLINK. The linker can be used in a stand-alone mode for linking of object and library files to produce the .EXE file. Its primary advantage, as I see it, is for the optional parameters available. A variety of switches can be used to produce map files, source code with line numbers, initialize segments, provide detailed segment maps, warn of duplicate symbols, etc.

These are valuable features, in particular, the production of map files. To date, TCC and TLIB have met my requirements very well. Perhaps that will change at some future time.

A Summary of What Has Preceded

We have learned there are several advantages to writing our C programs as modules. These are independently compiled and linked with each other and external library files to provide the executable run file. The advantages are improved organization of our source files, reduced compile time, and reduced operation of our floppy and/or hard disk drives. A disadvantage for CP/M programs is the need to purchase a macro assembler.

For those of us still programming in CP/M, the Microsoft MACRO-80 and Digital Research RMAC are macro assemblers providing a relocatable assembler, a linker/loader and library maker. A cross reference program is also available with these.

For programming in MS-DOS, the distribution disks include LINK.EXE and LIB.EXE which provide linking/loading and library equivalents of the macro assemblers. Recent releases of Turbo C from Borland and Quick C are affordable language systems including these capabilities.

The use of modules in combination with these highly integrated systems permit very large programs to be efficiently organized and rapidly compiled with well defined error checking. The resulting object modules are available with no further compiling needed unless subsequent program developments require a revision. In this event, the affected module may be extracted from a library, revised, and reentered with minimum interaction to other components of the program.

A Modular Program Example

It is only fitting to include an example of a linked modular program. Listing 1 provides six modules for linking experimentation. Right off, let me say, be prepared for a certain amount of exasperation when working with linked modules. From my point of view, C is an experimenter's language. We learn by doing and sometimes it takes a lot of doing before we make our way to the summit.

I can only provide the example in relation to my own systems. Though the example was developed using the Zenith version of MS-DOS 2.11 for my Z-161, it should run with most compilers having provision for module linking. You must, of course, take into account any differences between your compiler and what has been described here.

I compiled and ran the program under two differing sets of instructions. In the first, I created .OBJ files for all six modules and linked them using Turbo C's TCC command line procedure:

```
tcc -efriends friends.obj futil.obj fmod1.obj fmod2.obj fmod3.obj
fmod4.obj (this will fit in a single command line)
```

The correctly linked and compiled FRIENDS.EXE, when run, displays:

```
Hi! I'm you're friendly computer. Who're you?
A. Happy User. Pleased to make your acquaintance.
Anything worth an interrupt going on?
Just looping about hoping for a call.
I'm a C programmer. Here's betting I'll clear your problem.
```

An alternative is to link all of the .OBJ modules into a single library file. The program was written using Turbo C and the .OBJ then files created by:

```
tcc -c filename.c
```

The object files from these were combined into a single library file using LIB.EXE. Then, again using TCC, the run file was obtained from:

```
tcc -efriends frnd_lbb.lib
```

FRND__LBB.LIB is the library file created by LIB. So how did I get from the .OBJ files to the .LIB file?

I used a response file, FRND__LBB, (Listing 2). It reads:

```
frnd_lbb
+a:friends +a:futil +a:fmod1 +a:fmod2 +a:fmod3 +a:fmod4
frnd_lst
```

The resulting list file is displayed in listing 3.

To obtain FRND__LBB.LIB I used the command line:

```
E:LIB @A:FRND_LBB
```

Keep in mind that my system is configured differently from yours. I work in a RAMDISK, G: and my program files are kept on a floppy in A:. In making the transition from my compiler to yours, some heartburn may occur. But persistence will pay off programs more efficient in their organization, easier to upgrade and abstract from in future applications.

The following sources were referred to in preparing this article:

1. User's Manual: The Software Toolworks TOOLWORKS C/80, version 3.1, February, 1984. Walt Bilofsky
2. Microsoft MACRO-80 Assembler, CP/M Version, the Software Reference Manual for HEATH/ZENITH 8-bit digital computer systems. Heath Company, Benton Harbor, MI, 1981.
3. User's Manual: The Software Toolworks TOOLWORKS C for MS-DOS, version 3.2, March 1985. Walt Bilofsky and Michael E. Duffy.
4. Microsoft MS DOS Version 2, Zenith Data Systems Corporation, St. Joseph, MI 49085, 1984. Chapter 13, LIB; chapter 14, LINK.
5. TURBO C Reference Guide, IBM Version, Borland International, Inc., 4585 Scotts Valley Drive, CA 95066. 1987. Appendix D. Turbo C Utilities.
6. TURBO C Version 1.5 Additions and Enhancements. Borland International, Inc., 4585 Scotts Valley Drive, CA 95066. 1987. Appendix B. TLIB: The Turbo Librarian.
7. Brodie, Jim "Microsoft C 5.0 With Quick C." Computer Language, February, 1988, Vol 5, No. 2, pp 120 - 135.

Registered Trademark Copyrights:

1. CP/M is a trademark of Digital Research.
2. Microsoft is a trademark of Microsoft Corporation. MS-DOS and the Microsoft logo are trademarks of Microsoft Corporation.
3. Turbo C is a trademark of Borland International, Inc.

Previously published material on the C language by C. Pepper:

1. C: THE UNKOWN LANGUAGE WORTH KNOWING ABOUT
   ACCESS Magazine, Nov/Dec 1983
   Abstract: A description of the syntax, operators, etc of C using The Software Toolworks C/80 for reference.
2. USING "C" FOR FAST ACTION SCREEN GAMES
   REMark Magazine, Issue #43, 1983
   Abstract: Basics of C with application of C/80 to Heath H-89 screen action game creation.
3. A monthly column in the San Diego Heath Users Group (SDHUG) newsletter Dup & Dump on the C language. This column has been maintained for the past two and one-half years.

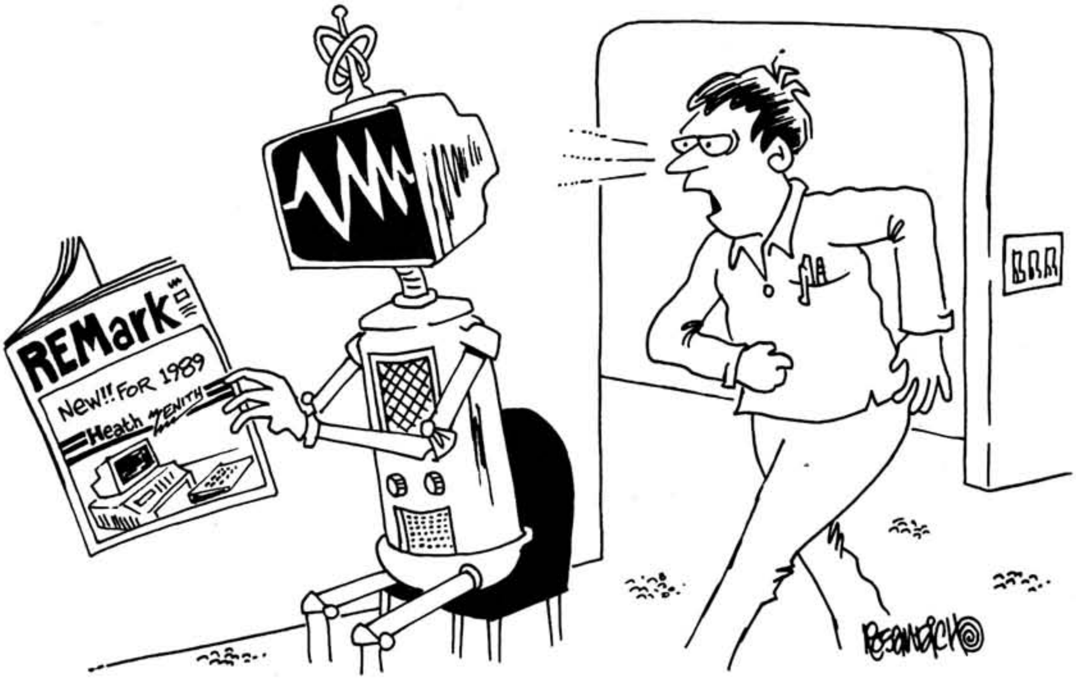
A SUMMARY OF MICROSOFT'S MACRO-80 FEATURES AND APPLICATION

OPERATING SYSTEM - CP/M

Disk Files: M80.COM L80.COM LIB.COM CREF.COM
Manual: Three ring binder with four major divisions:

1. MACRO-80 Assembler Reference
   - Chapter 1. Using the MACRO-80 Assembler:
     - Source file formats
     - Command formats
     - Error codes/messages
   - Chapter 2. Expression Evaluation
     - Arithmetic and logical operators
     - Modes:
       - External
       - Opcodes as operands
   - Chapter 3. Pseudo-Opcodes/Assembler Directives
     - Pseudo-Opcodes
     - Listing control pseudo-operations
     - Conditional pseudo-operations
     - Relocatable pseudo-operations
   - Chapter 4. Macros and Block Pseudo-Operations
     - Macros and block pseudo-operations
2. LINK-80 Loader Reference
   - LINK-80 command strings
   - Format of LINNK-80 compatible object files
3. LIB-80 Library Manager
   - LIB-80 command strings
   - Modules
   - LIB-80 switches
   - LIB-80 listings
   - Sample LIB session
4. CREF-80 Cross Reference Facility
   - Using the cross reference facility
   - Example

✱



"HEY!! GIMME THAT!!"

Zenith Makes the Most of the Fastest-Growing PCs

This article was reprinted with permission from the Monday, December 19, 1988 edition of the Detroit Free Press.

William J. Mitchell
Staff Writer
Detroit Free Press

St. Joseph — Heading south along Michigan's western shore, there's not much chance of mistaking the lakeside landscape for Silicon Valley, that high-tech mecca tucked below the mountains surrounding San Jose, Calif. "This is called San Jose East," jokes Heath Co. President Bill Johnson, noting that the English translation of the California city matches the name of the place where his firm has been headquartered since 1936.

More striking than the coincidence of names, however, is the emergence of this southwest Michigan community on the map of big-time computer production.

"Most people think of America as shaped like a dumbbell, with all the brains on the east and west coasts and very thin in the center," says James Magid, a senior advisor with the New York investment banking firm of Needham and Co. "But it's not true."

Zenith Electronics Corp. of Glenview, Ill., parent company of Heath and Zenith Data Systems operations in St. Joseph, earlier this month held off a strong challenge from an investor group pushing management to sell off the firm's money-losing television manufacturing arm.

The company won't discuss profits of individual operations, but it appears clear that the profits of the computer operation are partially offsetting the TV losses and enabling the company to remain the sole U.S.-owned TV maker.

Pointing to Zenith's profitable computer operation, Magid says: "People ask, 'How can that be when they're (manufacturing) in Michigan?' They don't understand how good Michigan is."

It's impossible to know for certain, since manufacturers guard such statistics like trade secrets, but the company's nondescript plant on Hilltop Road here may be producing more laptop computers than any plant in the world.

Assembled by a work force represented by the United States Steelworkers, Heath/Zenith computers are the only union-label personal computers in the world, says Andrew Czernek, ZDS's Vice President of Product Marketing.

Jobs at the 500,000 square foot plant in St. Joseph have doubled — from about 1,000 to about 2,000 — in the nine years since Zenith took over. But automation has limited further employment gains despite recent spurts in laptop sales.

Sales have grown about 50 percent a year in its computer segment, but Zenith has yet to capture more than five percent of overall personal computer market. IBM, Apple and Commodore are all expected to win bigger chunks of the 10.2 million PCs to be sold this year.

But Zenith is the leader among laptops — those foldup marvels of design and technology that pack millions of pieces of information into four- to 15-pound packages. And laptops are the fastest growing segment of the personal computer industry.

Zenith, which also makes larger desktop models in St. Joseph, believes laptops will soar from their current slice of seven percent of the PC market to 25 to 33 percent over the next few years. "And we think that's a conservative estimate," says spokesman Matt Mirapaul.

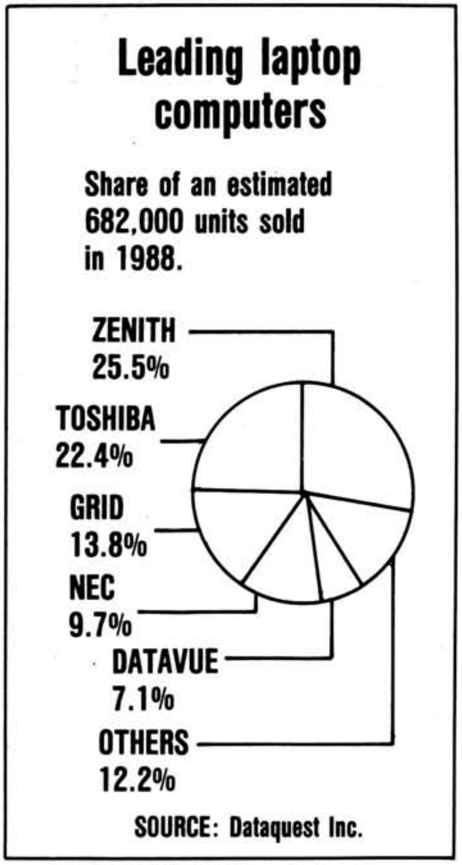
Industry analysts aren't quite that bullish, but they do forecast big growth.

Robert Charlton, an analyst for Dataquest Inc., a San Jose market research firm specializing in high-tech, project a doubling of market share — to 15.5 percent — by 1992.

"If people can get the same power and display with a smaller unit," he says, "it seems reasonable to me that they will replace a lot of desktops."

Until recently, laptops were of limited usefulness to those who rely on computers for massive storage of records or for complex tasks, such as spreadsheet analyses. The portables' small screens were also hard to read, and their batteries — if they had them at all — offered limited life before demanding a new charge.

Now Zenith, Toshiba and other companies are selling portables that have replaced the limited technology of floppy disks with so-called hard drives that can hold 20 to 40 million characters. The new screens are clear and sharp and beefed-


Leading laptop computers

Share of an estimated 682,000 units sold in 1988.

SOURCE: Dataquest Inc.

MARTHA THIERRY/Detroit Free Press

up batteries will operate four hours or more.

Pointing to such product segments of the automobile market as mini-vans, Czernek predicts that similar mini-markets will open up for computers.

"The personal computer business is only 11 years old," he says. "We're going to see more and more profound changes until, maybe 10 years from now, we won't have just a personal computer segment anymore. Perhaps we'll have an educational computer business, a notebook computer business."

Zenith has carried the automotive analogy directly into its marketing strategy with the names for its portables: the top-of-the-line TurbosPort, which offers a 40 megabyte configuration for up to $8,499, and the more streamlined SupersPort, which starts at $2,399 for a 10.5-pound model equipped with two floppy disks.

Heath has begun offering one of the higher-priced SupersPorts in kit form. Says Mirapaul: "We're saying you can put it together in an evening."

The company is still assembling some SupersPorts in Japan, but installed a new assembly line in St. Joseph earlier in 1988 to handle the bulk of the production. Other laptop producers, some of them reacting to tariffs placed on the products by President Reagan last year, are also building in the United States.

Says Johnson: "I love the idea of moving a laptop computer line from Tottori, Japan, to St. Joseph, Michigan, for economic reasons."

Johnson acknowledges that the strength of the Japanese yen was a factor in the decision, but argues that Michigan is no longer considered the kind of high-cost state to be avoided by high-tech manufacturers.

Danny Kauffman, President of Steelworkers Local 8850, echoes some of Johnson's sentiments: "We're very happy they decided to bring it here."

But he also points out that Zenith relies on a significant share of imported parts for the machines it assembles in St. Joseph.

The company refuses to say how much of the computers' content is imported, but says it regards the SupersPort as "mostly American" in terms of the value of its parts.

According to Dataquest, Zenith took the lead in the laptop sweepstakes in 1986 and is expected to close 1988 with a narrow lead over Toshiba.

If the machines themselves are high-powered, so too is the competition — and the profit potential.

Mike Hakeem, Zenith Data Systems' Vice President for Engineering, works on a 12-month product cycle, and is pushing to get weight out of the laptops at the same time he tries to pack in more performance.

Says Hakeem: "We're looking for tenths of pounds."

He won't discuss specific plans for future models, but industry analysts say the first manufacturer to produce a reliable color screen — perhaps within the next year — will reap big rewards.

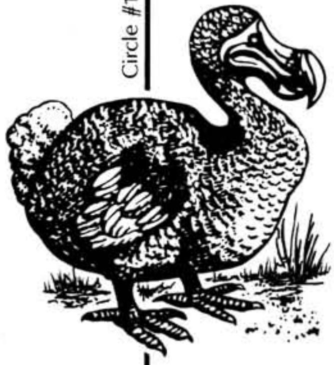
Heath, a company known for decades for its radios, oscilloscope kits and build-it-yourself model airplanes, had just gotten into the computer business when Zenith acquired it in 1979 as a way of expanding its base of TVs and radios to the promising new world of bits, bytes and microprocessors.

As the competition grew fierce for an exploding retail market in the early 1980s, Zenith figured it could survive only if it headed for the niches.

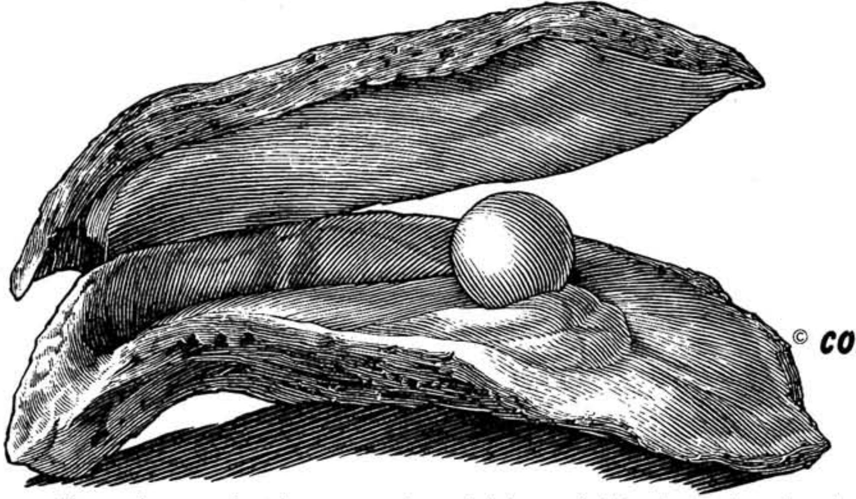
As its latest niche turns the corner toward becoming a major market, Zenith is facing the kind of problems that often accompany growth.

During a Quality Circle meeting at the plant last week, Credit Manager Larry Slutz told staffers that customers were raising legitimate complaints about Zenith's failure to respond quickly when equipment arrives defective or is shipped to the wrong place.

"Everyone has problems," he said of Zenith and its competitors. "But if we fail to service our customers right, we're going to have more problems. And we don't sell the only computers in the world." *



A POWER USER'S SHELL



KEVIN R. GRANTHAM
5 DITTON ROAD
SURBITON
SURREY KT6 6RE
ENGLAND

After a long and arduous search, a power user has found a mouse compatible DOS shell that stays out of the way until it is wanted. Norton Commander version 2 can be used by regular DOS users, but really performs in the hands of an experienced operator.

I began an intensive search for DOS shell and menu programs shortly after I purchased my first mouse. I looked for programs that would let me use my mouse to gain the productivity benefits of the Macintosh user interface, but still allow me the power and flexibility of the DOS command line. I wanted a menu system so that I could take shortcuts to my various development environments and applications, and so that the other casual users in my household could be spared the agonies and frustrations of navigating around my 6 disk system.

When I added a mouse to my Z-248, I was very excited. Finally, I could run all the mouse-oriented programs I had, without the cumbersome cursor key commands. I was impressed by the menu utilities provided with the mouse, and felt that I could build a sophisticated menu to handle my usual DOS needs. Well, after tinkering with it for several months, I concluded that the "pop-up" menus available with the mouse (or with any mouse I know of) didn't really do for me what I had hoped.

In order to make a reasonably informed selection, and to reduce the cash outflow, I decided to produce a simple list of requirements.

First, as the primary user of the system, came my technical needs. I wanted a mouse interface, that would allow me to point and click on applications, or point and change to different directories and disks. I wanted the ability, as exists in GEM, to point at a file with a .DOC extension, for example, and automatically load Microsoft Word, passing the document name as a parameter. I liked the ability in Bill Niedert's excellent shell ("F") to view two directories at once, and mark for action files to be deleted, copied, moved, etc.

I also wanted to be able to drop out of the shell instantly to view the results of a DOS operation, and to turn panels off and on similar to the Macintosh interface. Finally, I wanted a shell that was thrifty with memory, but was memory resident. I specifically did not want a shell that had to be reloaded at the end of each application or command. Of course, a "friendly" version, one that wouldn't conflict with SideKick Plus or other TSR (Terminate and Stay Resident) utilities was desirable.

Secondly, I wanted a menu system that would reduce the keystrokes required for me to do my usual activities, and would allow my family to go into the ckeckbook program, various games, the word processor (in the FAMILY subdirectory), etc. Another consideraton was the reduction or elimination of the number of batch (.BAT) files in the system. A word of explanation might be in order there.

DOS operates on chunks of data called clusters. The size of the cluster depends upon the type of disk being used and other factors, but can range from 512 to 8k bytes. It doesn't matter if the actual data in a file is less than the size of one cluster, it still takes up one cluster of space on the disk. Hard disks typically are set up with cluster sizes from 2K to 8K. With over 100 batch files in use on my system, I was looking for a way to combine them into one or more larger files. This might speed up the time to find them, and if designed correctly, could speed up the time to execute them. It also would reduce the amount of wasted disk space allotted to them.

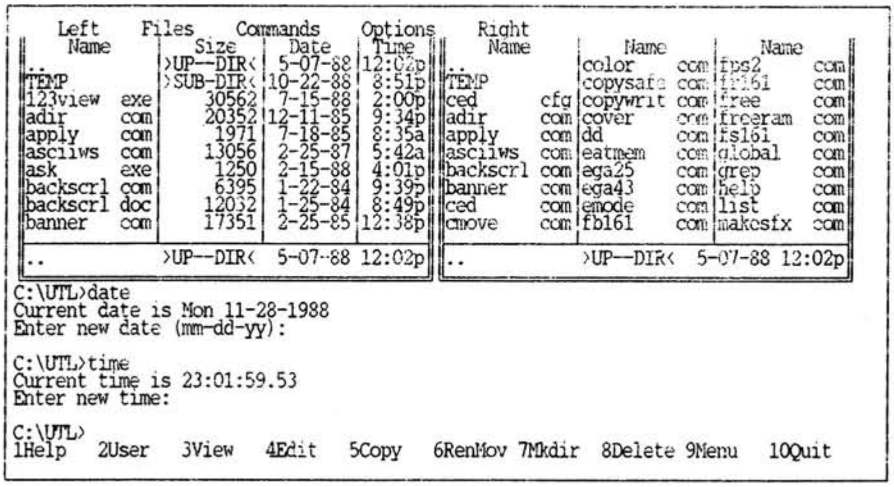
A good menu system would let me place the instructions in the menu item rather than in separate batch files, and since the menu file would already be loaded, the time to execute the commands should be reduced.

I tried a number of shareware, freeware and commercial software packages. Each package generally had some highly desirable feature, while being cumbersome or inapplicable in some other way. I tried COMMAND.COM replacements, as well as add on packages. During my travels, I acquired a copy of Norton Commander version 1.00. It came close to what I wanted, had good mouse support, but was just a little too clunky and complicated for my family. When Norton offered me an upgrade to version 2, I went ahead and tried it. I didn't have great expectations, however.

Commander surprised me. While there are a few things that it does not do, and could use some improvement, I found in it finally a useful shell that met most of my needs. For the remainder of this article, I will describe the Commander and some of its better (and weaker) points.

Commander comes on two disks, one each of 5.25" or 3.5". Installation is simple. Four files are copied over to the hard disk, into a subdirectoy included in the PATH. These files are NC.EXE (the main Commander Program), NCSMALL .EXE (a smaller version for hard disk users, it requires less memory), 123VIEW.EXE and DBVIEW.EXE (utilities that allow Lotus 1-2-3 or compatible and dBase III and compatible files to be viewed in a coherent formatted manner, instead of in hex). There is also a READ.ME file with the latest information, and a demonstration program of the Norton Guides, another Norton product.

The Commander is started with either of the commands NC or NCSMALL.



```
  Left     Files    Commands    Options    Right
 Name          Size      Date     Time     Name         Name               Name
..           >UP--DIR<   5-07-88 12:02p  ..            color     com  fps2      com
TEMP         >SUB-DIR<  10-22-88  8:51p  TEMP          copysafe  com  fr161     com
123view  exe     30562   7-15-88  2:00p  ced      cfg  copywrit  com  free      com
adir     com     20352  12-11-85  9:34p  adir     com  cover     com  freeram   com
apply    com      1971   7-18-85  8:35a  apply    com  dd        com  fs161     com
asciiws  com     13056   2-25-87  5:42a  asciiws  com  eatmem    com  global    com
ask      exe      1250   2-15-88  4:01p  backscrl com  ega25     com  grep      com
backscrl com      6395   1-22-84  9:39p  banner   com  ega43     com  help      com
backscrl doc     12032   1-25-84  8:49p  ced      com  emode     com  list      com
banner   com     17351   2-25-85 12:38p  cmove    com  fb161     com  makesfx   com
..           >UP--DIR<   5-07-88 12:02p  ..            >UP--DIR<   5-07-88 12:02p
C:\UTL>date
Current date is Mon 11-28-1988
Enter new date (mm-dd-yy):

C:\UTL>time
Current time is 23:01:59.53
Enter new time:

C:\UTL>
1Help   2User   3View   4Edit   5Copy   6RenMov 7Mkdir  8Delete 9Menu    10Quit
```

Figure 1

Commander is in 25 line format. At the top is the menu bar, which is normally hidden unless called with function key F9, or clicked on with the mouse. Under it are two panels. The left panel shows a FULL directory display, and the right panel shows the same directory in BRIEF. Note that the two panels could be showing different directories as easily. Below the panels is the DOS command area, showing the results of DATE and TIME commands. The two panels can be switched off at the touch of a key to reveal the results of other DOS commands. At the bottom is the function key legend, which also may be switched off.

NC is for floppy users, and loads the whole program into RAM. This allows the floppies to be changed whenever needed. The NCSMALL takes up much less RAM (12736 bytes on my system), and loads what it needs from the NC.EXE file when necessary. I placed the NCSMALL near the end of my AUTOEXEC .BAT file, before Sidekick Plus. Commander attaches no interrupts, like other TSR programs, so it does not have interrupt conflicts. It also allows easy removal from memory when desired, and can be removed even if it is not the last TSR loaded.

The Commander has five areas on the screen (see Figure 1). At the top, and invisible until needed, is a pull-down menu bar. This becomes visible when either the F9 key is pressed, or the mouse is clicked anywhere in the top line of the screen. Otherwise, anything DOS or a program has written on the top line is visible.

Underneath the menu bar are two panels, one on the left and one on the right. The panels may be set to cover the vertical space on the screen from row 2 to row 23, or they may be set to cover only half. These panels may be turned on or turned off independently. For example, I generally use only the right panel so that I can see the results of various commands and utilities I have used. I flip the left one on only when comparing directories, copying, etc. Each panel can show a different disk and directory, an information panel, or a directory tree. We'll come back to the uses and formats of these panels in a minute.

Below the panels is the DOS command line. Any DOS command can be entered on the command line immediately, and the Commander will step aside and let DOS do its things automatically. This feature was very important to me. I did not like shells or menus that require you to "EXIT to DOS" in order to execute a command, or required some other function key or control key combination. With Commander, you just type in the command as if you were not running the Commander at all, no muss no fuss.

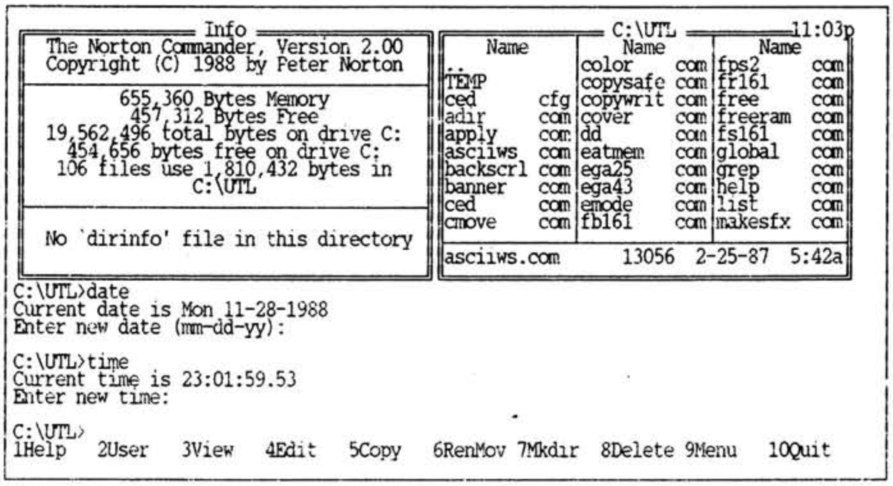
On the bottom of the screen, row 25, are function key labels. The Commander uses F1 through F10. The functions also can be executed by pointing at the label with the mouse and clicking. Function keys are used in combination with the Shift key or the right mouse button for more general versions of the same command.

When the Commander is first installed, a moment with the setup options is in order. OPTIONS are found on the menu bar, either by pointing with the mouse and clicking on the top row of the screen, then pointing at OPTIONS and clicking again, or by pressing F9 (to bring up the menu bar) and "O" for Options. The following options are available:

Color: This allows the system to be set for color, monochrome monitors running on a color adapter, monochrome monitor/adapters, and even a special mode for laptop users' LCD screens.

Auto Menus: Normally, the user menu is displayed only when asked for with the F2 key. For novice and other users, it may be desirable to have the menu displayed automatically. Any menu display can be sent away with the Escape key, or recalled with the F2 key.

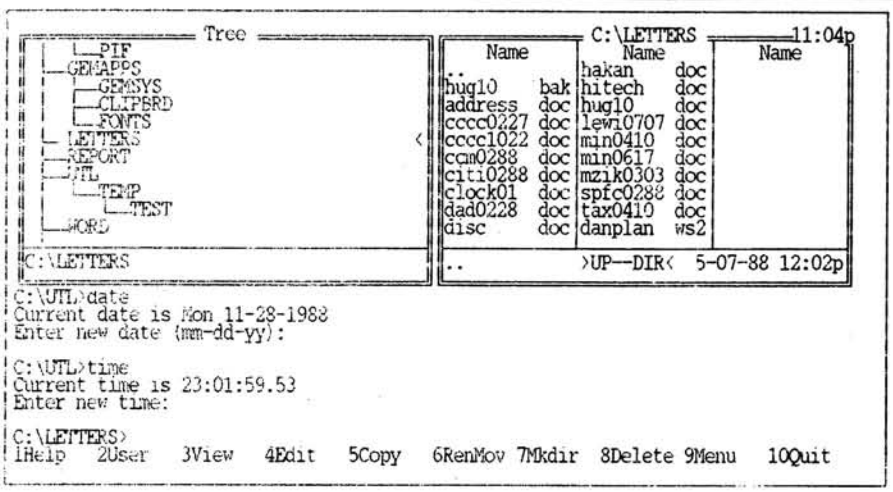
The DOS prompt can be set to the standard, or to the disk and directory format that one gets with a PROMPT PG. The function key bar can be set on or off, and the panels can be set to cover the whole screen or half screen, as described above. A mini-status line can be shown at the bottom of each panel, or disabled if more directory lines are desired. An on-screen clock can be displayed, if desired, and the ASCII editor supplied with the



```
 ========= Info ==========        ========== C:\UTL ============11:03p
 The Norton Commander, Version 2.00    Name        Name           Name
 Copyright (C) 1988 by Peter Norton   ..          color    com  fps2     com
                                      TEMP        copysafe com  fr161    com
      655,360 Bytes Memory            ced     cfg copywrit com  free     com
       457,312 Bytes Free             adir    com cover    com  freeram  com
 19,562,496 total bytes on drive C:   apply   com dd       com  fs161    com
  454,656 bytes free on drive C:      asciiws com eatmem   com  global   com
 106 files use 1,810,432 bytes in     backscrl com ega25   com  grep     com
             C:\UTL                   banner  com ega43    com  help     com
                                      ced     com emode    com  list     com
                                      cmove   com fb161    com  makesfx  com
 No `dirinfo' file in this directory
                                      asciiws.com      13056  2-25-87  5:42a
C:\UTL>date
Current date is Mon 11-28-1988
Enter new date (mm-dd-yy):

C:\UTL>time
Current time is 23:01:59.53
Enter new time:

C:\UTL>
1Help   2User   3View   4Edit   5Copy   6RenMov 7Mkdir  8Delete 9Menu    10Quit
```

Figure 2

Here, the menu bar has been hidden. The right panel shows a BRIEF directory. Notice that the subdirectory, TEMP, is in upper case and that the individual file names are in lower case. The left panel shows the INFO display showing the same information one gets from CHKDSK, and also showing the size of the directory displayed in the other panel. The statement "No 'dirinfo' file in this directory" means that I have not attached an explanation of what this directory contains.



```
========== Tree ==========            ===== C:\LETTERS =====11:04p
  │   └──PIF                        Name        Name          Name
  ├──GEMAPPS                        ..          hakan    doc
  │   ├──GEMSYS                     hug10   bak hitech   doc
  │   ├──CLIPBRD                    address doc hug10    doc
  │   └──FONTS                      cccc0227 doc lewi0707 doc
  ├──LETTERS                   <    cccc1022 doc min0410  doc
  ├──REPORT                         ccm0288 doc min0617  doc
  ├──UTL                            citi0288 doc mzik0303 doc
  │   └──TEMP                       clock01 doc spfc0288 doc
  │       └──TEST                   dad0228 doc tax0410  doc
  └──WORD                           disc    doc danplan  ws2
C:\LETTERS                          ..          >UP--DIR<  5-07-88 12:02p
C:\UTL>date
Current date is Mon 11-28-1988
Enter new date (mm-dd-yy):

C:\UTL>time
Current time is 23:01:59.53
Enter new time:

C:\LETTERS>
1Help  2User  3View  4Edit  5Copy  6RenMov 7Mkdir  8Delete 9Menu  10Quit
```

Figure 3

The left panel has been set to TREE. Notice the pointer to the current directory. Just "point and shoot" at a directory and the prompt at the bottom will change to show it, and the panel on the right will change to the selected directory.

Commander can be replaced with your own (I use Logitech's POINT, but of course the Commander manual suggests Nortons's Editor!)

The setup can be saved into a file called NC.INI, and defines the setup every time Commander is started. Note that the setup options must be explicitly saved. This lets you experiment with different options, depending upon your needs, but still have the system start up with your preferred configuration.

The next bit of setup to do is the menu and extension files. These functions live under the COMMANDS pull down menu. Point and click or F9 and "C".

The eXtensions file (saved in NC.EXT) defines the application that should be started when a data file with the defined extension is "run" (Norton calls this "point and shoot"). For example, any file that ends in .TXT, .CAP, .LOG or .ASC is an ASCII file, probably downloaded from Compuserve or produced by some other program. I set up lines such as TXT: PT !.! in the eXtension file. This tells Commander whenever I "shoot" a file that ends in .TXT, to run the Logitech POINT editor (called PT) and pass it the full name of the file, substituting the filename and the extension for the !.!. I set up the same way for .BAS (calls up Quick Basic), .ASM (calls up POINT, configured for programming), .DOC (calls up Microsoft Word), etc. Lotus 1-2-3, by the way, doesn't work this way because you cannot pass it the name of the file. So, I can have 1-2-3 start up when I shoot a .WKS or .WK1 file, but I still have to do a /File Retrieve.

There can be more than one menu file. The "main" menu file is stored in the root directory of the startup disk. The other menu files are located in any subdirectories. When you change directory, Commander looks to see if there is a menu file there. If there is, that is the one that will be displayed automatically or upon pressing the F2 key. If there is no menu file in the current directory, the main menu is used.

The menu files (NC.MNU) contain a letter, colon, and a descriptive label, followed by the commands to be issued. For example:

```
L:      Lotus 1-2-3
        pushdir
        c:
        cd \123
        lotus
        popdir
P:      Procomm/Compuserve
        pushdir
        d:
        cd \procomm
        procomm /fipss12
        popdir
```

On the menu, this would appear as:

```
L Lotus 1-2-3
P Procomm/Compuserve
```

To select an item on the menu, type the letter assigned, move the selection bar up or down using the cursor keys, then hit enter when the chosen item is selected, or point and click with the mouse.

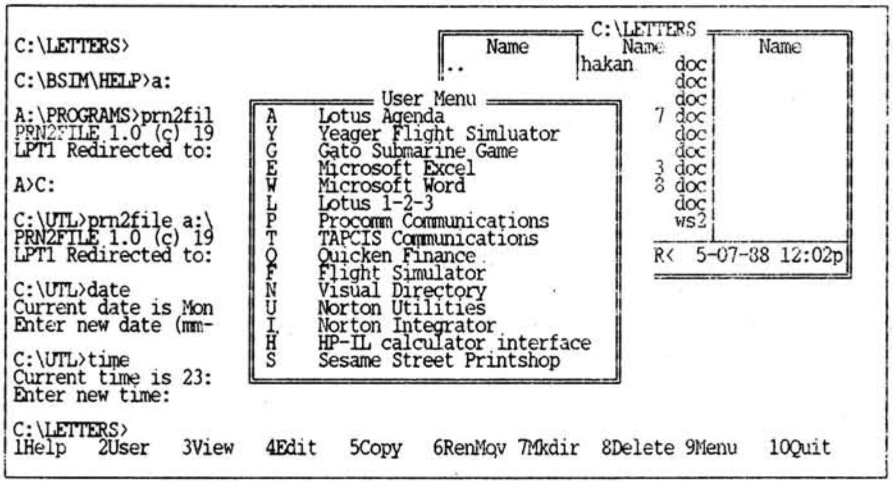
As you can see, the menu feature meets the requirements for easy-to-use application selection, but stays out of the way until it is wanted.

Now let's talk about the main functions of the Commander.

In the middle of the screen are two panels, which may be switched off or on, or changed into different formats at the touch of a key or two. Each panel can have different information displayed.

Two directory formats are available. The FULL format shows one file per line, along with its creation date, time and size. Subdirectories are shown in upper case letters, and files are shown in lower case. A selection bar in the active panel can be moved with the cursor keys or mouse, and the listing can be scrolled backwards and forwards. The BRIEF format shows three columns of files in the panel. At the bottom of the panel is the mini-status line, which shows the size and creation data of the file under the selection bar, or the total size of all the files selected.

Other formats include an information panel (which provides continuously the information available from a CHKDSK command) and a directory tree panel which shows the directory structure as a graphic tree. A change to a different directory can be chosen simply by pointing and shooting the directory name in the tree, and the other panel (if it is a directory format) will change to show the new di-



```
                                                ===== C:\LETTERS =====
C:\LETTERS>                                   Name       Name        Name
                                              ..         hakan   doc
C:\BSIM\HELP>a:                                                  doc
                          ====== User Menu ======                doc
A:\PROGRAMS>prn2fil       A   Lotus Agenda                     7 doc
PRN2FILE 1.0 (c) 19       Y   Yeager Flight Simluator            doc
LPT1 Redirected to:       G   Gato Submarine Game                doc
                          E   Microsoft Excel                  3 doc
A>C:                      W   Microsoft Word                   8 doc
                          L   Lotus 1-2-3                        doc
C:\UTL>prn2file a:\       P   Procomm Communications             ws2
PRN2FILE 1.0 (c) 19       T   TAPCIS Communications
LPT1 Redirected to:       Q   Quicken Finance              R<  5-07-88 12:02p
                          F   Flight Simulator
C:\UTL>date               N   Visual Directory
Current date is Mon       U   Norton Utilities
Enter new date (mm-       I   Norton Integrator
                          H   HP-IL calculator interface
C:\UTL>time               S   Sesame Street Printshop
Current time is 23:
Enter new time:

C:\LETTERS>
1Help   2User   3View   4Edit   5Copy   6RenMov 7Mkdir  8Delete 9Menu   10Quit
```

Figure 4

Here, you can see a User Menu. Behind this menu are the commands normally found in batch files. User Menus may be stored in subdirectories, which become active when the subdirectory is entered, and in the root directory of the startup disk. Notice here that only one panel is in use.

rectory. This is useful when you are looking through many directories. You can navigate with one panel and view the directory contents with the other.

Directory formats BRIEF and FULL always show the subdirectories first, then files. However, the names can be sorted by date, time, size, name or extension, and resorted simply by selecting a different option. This only affects the presentation in the panel, not the physical directory organization on disk. For those who have presorted their disk directories into a special order for some special purpose, Commander allows the panel to display the directory unsorted, as well.

The formats can be changed on the fly either through the pull down menus or through speedy keyboard commands. In fact, everything in the package can be accomplished with direct keyboard commands or through the menus.

Within a panel, a "speed search" is available. For example, say there are 400 files in your database directory, and you want to find the one entitled var99.dbf to look at. Hit ALT-V and the selector bar will jump to the first one whose name starts with V. Then, if that is not the one you want (maybe there are several other files that start with V), hit "A" and the selector will jump to the first file that starts with "VA", etc., until you get the one you want. This works for directories, as well.

Once a file is selected, it can be copied, deleted, renamed, viewed or edited with the press of a function key. The editor works on ASCII files only, unless you have specified your own editor in place of Commander's. The view function will show the file and let you scroll back and forth, search for a string, etc. It does not have a hex mode, however, which I find annoying. The Commander does provide two utilities to examine .WKS and .WK1 files, and .DBF files. Called into play immediately upon selecting the view function on a file with one of the extensions mentioned, the utilities show a 1-2-3 screen image that you can navigate around in (but cannot change anything) or a formatted database record with the labels and correct fields. The DBF utility lets you search for strings within the record or within the database.

I find these utilities invaluable when I am looking for a quick piece of data, or trying to find out which spreadsheet to load. It is much faster than loading the actual applications first. Commander allows new directories to be created and old ones to be renamed or deleted easily. It is compatible with Norton Change Directory (NCD), one of the Norton Utilities.

Several files can be selected at once, when mass copying or deleting is desired. Each file can be pointed to and selected with the INSert key, or deselected with the DELete key (or by pointing and clicking with the right mouse button). The gray + key will bring up a box that lets you type in a selection range, and the gray - lets you type in a gray deselection range. For example, suppose you wanted all files marked JIM????.*, where ???? are wildcards for the month/day in mmdd format. You want to include all .DOC, .TXT, .CAP, and .LOG files, but not .STY files. Simple. Hit the gray +, then type JIM????.*<enter>. All the files that meet that criteria will be selected (and will light up). Then hit the gray - key, and type *.STY<enter>. Any files with the extension .STY which had been selected will be deselected. Then hit F5 for copy, confirm with a Y, and off it goes.

When more than one file is selected, the mini-status line at the bottom of the panel will change to show the total number of bytes in the selected file. If copying to a floppy, and you run out of room before the copy is completed, Commander will prompt you to insert a new floppy, then will continue with the copying. Any multifile deletes give fair warning.

Other useful features: Commander will display 43 lines on an EGA and more on a VGA display, and you can switch back and forth between the compressed display and the 25 line display at the touch of a key. Commander provides an enhanced version of the Norton File Find command and its look-alikes. The listing of all the occurrences of a file specification is scrollable backwards and forwards and doesn't just roll off the screen as FF does. Also, you can position the cursor on the file you want, hit enter, and the Commander will automatically change to the subdirectory the file is in.

Commander will display a full screen directory tree display, with functionality similar to and compatible with the Norton Change Directory. An excellent directory comparison utility is provided that will automatically highlight the differences between two directories, so that a subsequent copy or delete will make them the same. A DOS command line editor is provided, which allows the last 10 commands to be recalled, edited and entered.

The above description provides a glimpse of the power and utility found in Norton Commander Version 2. The ease of use, the really useful functions placed at your fingertips, the intelligent command and mouse interface combine to make a product more valuable than this article can convey. For around 60 dollars street price, this is the DOS shell for power users.

Lotus, 1-2-3 are trademarks of Lotus Development Corporation.
Microsoft, QuickBasic and Word are trademarks of Microsoft. Norton, Norton Editor, Norton Guides, Norton Utilities and Norton Commander are trademarks of Peter Norton Computing.
GEM is a trademark of Digital Research.
Logitech and POINT are trademarks of Logitech, Inc.
Z-248 is a trademark of Zenith Corporation.
Macintosh is a trademark of Apple Computers Inc.
SideKick Plus is a trademark of Borland.
dBase II is a trademark of Ashton-Tate, Inc.
Compuserve is a service mark of Compuserve Information System.
Procomm is a trademark of Datastorm Technologies, Inc.

"I CAN HEAR THE CRASHING OF DATA!"

ENABLE - A Tutorial

MACROS

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

The fourteenth ENABLE article will focus on the macro capability. This advanced feature sets ENABLE one step above the rest of programs that are currently available. This article is outside the normal line of the articles to date, but this feature needs to be covered at this time. The next article will cover the menu capability of ENABLE, which when combined with the macros, makes ENABLE very friendly for people with little or no computer background. I will return to the word processor in the article after that with a discussion of version 2.15 and the changes that have been made in this version.

I want to thank all of the many people who stopped by and said "Hi" at the Dayton Computerfest in August. I did two presentations on ENABLE and was able to enlighten some people who have not used the product. I enjoy talking to people about ENABLE and I hope to get out to more shows in the nearby area as time permits and if I know they exist.

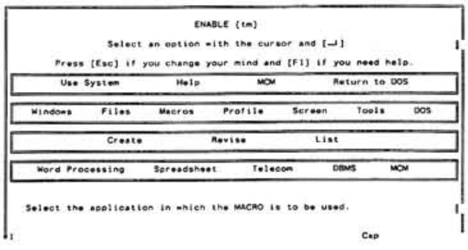
ENABLE macros permit you to develop a one keystroke set of commands that can be used in any of ENABLE's modules. They can be used for such tasks as drawing repeating lines in a word processing document, printing different sections of a spreadsheet, or developing a simple report with long "where" and "fields" statements. These are examples of macros I have built to save myself keystrokes and time. ENABLE macros can be built in one of three ways. The first method is to record the keystrokes as they are typed in. This is the "teach the computer" method and the one that I use the most. The second method is basically the same but ENABLE will not perform the steps that you type. This would required you to have the keystrokes written down if they are long. The third method is creating a macro using the MCM feature and word processing. This is the same method that LOTUS and ENABLE spreadsheet uses to record macros in cells.

The first method is to "teach the computer". This will record ALL keystrokes including the errors and the steps you took to correct them. To invoke this method of creating a macro, press "F0 F9" Backslash (Alt/F9 for the PC) and the key that you want to name the macro. Valid macro names are any character except Backslash "\", minus "-", equal "=", and any of the arrow keys. On the PC version, the "End", "PgUp", and "PgDn" are also not valid keys. Macro keys are not case sensitive so pressing the "Shift" will not add more macros. Also, the shifted numbers on the top of the keyboard cannot be used. The "Ctrl" on both the Z-100 and PC and the PC "Alt" (or F0 on the Z-100) cannot be used in conjunction with other keys to name macros. There is one exception, and that is the function keys. You can use the "Shift", "Ctrl", and "Alt" with any of the function keys except "Alt/F9" and "Ctrl/F2". Once you have named the macro, the word "MAC" will be displayed on the bottom of the screen, on the Status Line, to show that a macro is being created. Now you can press any key combination and ENABLE will record it. When you have finished the macro procedure, press "F0 F9 Shift/1" (or Alt/F9 End on the PC). The macro will be recorded on your data disk and the "MAC" will be removed from the screen. The macro will appear in the file as $ { x } .xxM. Each module can have the same macro name if it is created in that module and can be used ONLY in that module. The file name will have the extension ".WPM" for word processing, ".SSM" for spreadsheet, ".DBM" for the data base, or ".MCM" if it was created from the main menu. This means that you can have over 40 macros per module.

To run this macro press "F0 F9" (Alt /F9 for the PC) and the character you selected for the name of the macro. The macro will then run exactly as you typed it in, including the errors and the keystrokes you made to correct them. The macro will run through the entire procedure. If you want the macro to run only one step at a time press "F0 F9 -" and the character you selected for the name. The minus sign "-" is the one on the keypad and not the one on the top line of the keyboard after the numbers. If you select this option, pressing any key will cause the macro to advance one step.

The second method of creating an ENABLE macro is to type them in without running the procedure. This means that you must know the exact keystrokes as you input the macro. To start this procedure press "F0 F9 =" (Alt/F9 = for the PC) and then the name of the macro. You must be in the module that you want the macro to run in, in order to have the correct extension. When you have completed the macro press "F0 F9 Shift/1" (Alt/F9 End for the PC) to record it and end the macro procedure.

The third method of creating a macro is to use the MCM and word processor. This will permit you to type in the macro in the same manner as LOTUS. You can also use this method to correct macros that you have built using the other methods. To use this procedure, press (M)CM (M)acros (C)reate and then the application where the macro is to run and then the key that will be its name. ENABLE will then present a blank screen in the word processor where you can type in the commands. Using this procedure, you can send the macro to the printer in the normal manner (F0 F2 or Alt/F2 for the PC) if you require a hard copy. If you type in commands like "~~~~" for four <RETURN>s, the ENABLE macro generating program will save it as {4x}{~}. If you type in other commands like this, ENABLE will save them in the manner that saves space. Because this is in the word processing module, any of the word processing commands can be used. Any text that is to be inserted by the macro can have attributes added. When you are finished, the macro can be saved in the same manner as any word processing document.



ENABLE (tm)

Select an option with the cursor and [↵]

Press [Esc] if you change your mind and [F1] if you need help.

Select the application in which the MACRO is to be used.

FIGURE 1
ENABLE Macro Creation Menu

ENABLE's macros use words that reflect the operation you want ENABLE to perform. The commands are enclosed in braces ({x}). You can have ENABLE wait for a keystroke or it will even permit you to do things such as input records until you press a certain key and then it will continue. I use this procedure to permit records to be input. When the "@" is pressed, the records are then saved and the user is returned to the main menu. Figure 2 has a list of basic ENABLE macro commands.

Key	Code
Alt/F9 \	name and record macro
Alt/F9 End	end recording and save
Alt/F9 (name)	run macro
Alt/F9 Minus	step through macro
Ctrl/Break	Cancel Macro
Ctrl/I	Wait for input
Ctrl/-	wait for keystroke
F1	{F1}
F2	{F2}
F3-F10	{F3} - {F10}
ESCAPE	{ESC}
TAB	{Tab}
HOME	{HOME}
Up arrow	{Up}
Down arrow	{Dcwn}
Right arrow	{Right}
Left arrow	{left}
PgUp	{PgUp}
PgDn	{PgDn}
END	{End}
INS	{Ins}
DEL	{Del}
Backspace	{BS}
ENTER or RETURN	~
CTRL/ENTER	{LF}
{	{{}
}	{}}
~	{~}
ALT/x	{&x}
CTRL/x	{^x}
SHIFT/x	{!x}
ALT/Up arrow	{PrevW} (previous window)
ALT/Down arrow	{NextW} (next window)
ALT/END	{CloseW}(close window)
ALT/HOME	{OpenW} (open window)

FIGURE 2
Codes for Macro Keys

Note that these are the same commands that are available in the spreadsheet macros that can be inserted into cells. The above commands are for the PC version of ENABLE. For the Z-100 version, the same commands would be used except that "Alt" is the "F0" key, and the control keys are the "Shift/F0" keys, and "End" is equal to "Shift/1". If you are using the MCM Macro Create menu to create macros, you must use the PC commands as listed in Figure 2. The Z-100 equivalent commands will not work.

Additional commands that are available for macro operations are listed below. These commands will permit the macro to use other capabilities of the ENABLE program.

Error conditions	
if ENABLE error, perform command	{if error}commands
if no error, perform command	{else}commands
terminate commands	{endif}
Menu	
display named menu	{do menu}name~
make named menu default	{menu}name~
Perform named macro	{do macro}name~
Repeat	{nx} (where n is the number of times to repeat)
Beep	{beep}
Status line	
restore	{send} (r)
send message to	{send} (s) message~
Timed paused	{pause}
Video display	
freeze	{Voff}
restore	{Von}
Wait for input	{?}
Wait for input x	{^F9}x (where x is key)
Wait for keystroke	{Wait}
Window	
go to named window	{GoToW}name~
name current	{Wname}name~

FIGURE 3
Additional Macro Commands

These commands will assist you in making the ENABLE macro do exactly what you want with one keystroke. We are now ready to input our first macro. If you wish to document the macro this can be done by inserting a double semicolon in the first position on the line. Everything on the line after this will not be processed during the macro operation. Our first macro will be run from the MCM module and this should be selected when entering by typing in (M)CM (M)acro (C)reate (M)CM T. This first macro will check the files for all word processing documents, select the second one and bring it up. It will then save it to a file called "TEST" and then returns to the main menu.

```
;;test macro
UWR?~
{down}~
{&f10}c~~ntest~
w
```

FIGURE 4
Basic Macro Structure

The first line is a comment and will not affect the operation of the macro. The second line selects (U)se System, (W)ord processing and (R)evise with a "?" in the name of file block and a <RETURN>. The third line tells ENABLE to go down one line and accept this file by pressing a <RETURN>. The next line calls the save menu (F0 F10 for the Z-100 or Alt/F10 for the PC) and tells ENABLE you want to change the name to "TEST" and then closes the window after saving is completed. After typing this in the macro creation screen, you save it in the same manner as a word processing document (F0 F10 <RETURN> or Alt/F10 for the PC) and then quit to the main menu. If you make a mistake while typing in a macro, ENABLE will not permit you to save it. ENABLE will highlight the first line where it finds the error to assist you in correcting the mistake(s). You can now run this macro by press "F0 F9 T" (Alt/F9 T for the PC).

This was an introduction to a basic macro in ENABLE. These macros can get as long as necessary to complete a repeating task. This will permit you to save keystrokes during the normal business day. An example for military organizations would be the headers of messages. These headings can be fairly long with many addressees. Using ENABLE's macro capability in the word processor, one keystroke will cause the entire header to be inserted. The quickest way to do this is to teach ENABLE as discussed earlier. You can have several macros, one for each of the long addressees list. You can edit the macro using the MCM Macro Revise menu selections. See Figure 5 for this example.

A function in the macro that will permit you to stop macro execution is the "wait for keystroke." When this is added to the macro, ENABLE will stop until any key is pressed. The command that will accomplish this is the {wait} statement. ENABLE will not permit adding data or anything else as the first keystroke will cause the macro to start processing.

Another feature is the "Wait for Input" command (Shift/F0 I for the Z-100 or Ctrl/I for the PC). If you are creating a macro, this would be input as either {?} or {^i}. This command permits the macro to run to a specified point and then stop for inputting data until the <RETURN> or <ENTER> key is pressed. I use this in the data base to stop a macro while a "Where" clause is filled in. Pressing the <RETURN> key will cause the macro to restart and run until complete or another wait for input is encountered. Figure 4 and 5 have examples of the "wait for input" command.

A similar function is the "wait for a specific key." This is inserted in the macro as {^F9}x with x being the specified key. This key is case sensitive so there are many selections available. I use this feature as a means to terminate data base inputs or edits. I have selected the "@" key for this purpose, although any key can be used. The macro will run from the menu to the data base input or edit screen. The users can then input data using the <RETURN> key without problem until finished with the operation. The user then positions the cursor in a blank space on the screen and presses the "@" key. My macro then backspaces to {BS} to clear the "@" from the field, saves the record, quits the add or edit function and returns to a menu, normally the main menu. This makes it possible to develop an application for somebody that does not know ENABLE.

An example of the "wait for a specific key" would be:

```
UDIA
STUDENT~~
{^F9}@
{BS}
{F10}S
{F10}Q
{ESC}
Q
```

This macro is similar to those I use in the data base add and edit function called from a menu. The first line moves you from the main menu to the add menu in the data base. The second line selects "STUDENT" as the data base and uses the default input form as specified in the data base definition. You can now insert records as long as you wish. On the input screen I put the statement "Place cursor in blank space and press '@'." When this occurs, ENABLE will backspace to clear the "@", save the record and then quit to the main menu.

Another type of a stop command is the timed pause. This command, {pause}, will cause ENABLE to stop processing the macro for approximately one second and then continue. Several of these commands can be put together for an extended wait state. This would be inserted into the macro as {nx}{pause} where n is the number of seconds you want ENABLE to wait. As example would be {n30}{pause} which would result in a pause of 30 seconds. This command is used in the ENABLE tutorial to permit the student to read the screen where no response is required.

The {beep} command will cause the computer to sound a tone. It can be used to refocus attention on the screen while a macro is running, when an input is required or a process is completed.

When you are using macros in ENABLE it is best to insert the first letter of the command rather than using the cursor movements. If the macro is started at a location other than the designated start, the results may not be what is expected. This goes for all commands including the (P)rint command screens. If your macro requires movement between windows, use the {Wname}name~ command in macros. This will insure that the correct window is selected for the macro process. When the command is inserted in the macro, ENABLE will "name" the current window that name so you can move away from it and get back by using the {GoToW} command. Windows are opened in sequence, so they should have the same number in future macro processes; but to ensure that they are correct, they should be named. This name can be anything. It would be helpful to use a name that describes the window so that debugging would be easier along with self documentation.

Another feature of macros is the capability to display messages up to 70 characters on the bottom line of the screen or the Status Line. This is a feature that can be used to prompt the user of the macro to insert some data element. It could be used in conjunction with the {beep} command to alert the user to a requirement. To insert a message, the {Send}{S}message~ command is used. The {Send}{R} command is used to removed the message. I have used this feature in Figure 5.

Figure 5 is a example of a message format macro that can be used to fill in the top blocks on a standard DoD message form 173. This example uses several of the commands I have talked about. The first line permits this macro to run from the Main Menu and will open a

```
;;macro for test message address
UWC{?}
~
{F9}
{Ins}
2S                                           UUUU~
1              AUG88  RRRRRRRR{4X}{Left}
{2X}~
{15X}{Right}
HQ AFLC/ZZ WRIGHT-PATTERSON AFB OH~
{15X}{Right}
HQ USAF/ZZZ WASHINGTON DC~
{15X}{Right}
HQ TAC/ZZ LANGLEY AFB VA~
{15X}{Right}
HQ MAC/ZZ SCOTT AFB IL~
{15X}{Right}
HQ SAC/ZZ OFFUTT AFB NE~
{15X}{Right}
HQ PACAF/ZZ HICKAM AFB HI~
{15X}{Right}
HQ MCCLELLAN ALC/ZZ MCCLELLAN AFB CA~
{15X}{Right}
HQ SAN ANTONIO ALC/ZZ KELLY AFB TX~
{15X}{Right}
HQ OKLAHOMA ALC/ZZ TINKER AFB OK~
{15X}{Right}
HQ ROBINS ALC/ZZ WARNER ROBINS AFB GA~
{15X}{Right}
HQ OGDEN ALC/ZZ HILL AFB UT~
SUBJECT{Send}
(S)Make sure you type in the correct Month and Year.~
}{Beep}
{4X}{Pause}
{Send}
(S)Make sure you set the printer defaults.~
{Beep}
```

FIGURE 5
Sample Message Header Macro

word processing document. The "{?}" causes the macro to wait for a name to be inserted in the name of the document block and will continue when a <RETURN> or <ENTER> is pressed. The next "~" will bypass the document title block of the word processor. The "{F9{}Ins}2S" will cause the document to be placed in the double space mode which is required for messages. The next set of characters, "UUUU", indicates the security classification of the message which is followed by a <RETURN>. The next line places a "1" in the "of" page block. ENABLE then spaces over to the date time block (DTG) and inserts a default "AUG88" which can be corrected later. The rest of the line inserts the message precedence indicators. The macro will now drop down two lines and start inserting the address blocks. After the subject line, the macro will display a message on the status line and beep at the operator to ensure that the correct DTG is inserted and after a pause of four seconds, a second message to set the printer defaults correctly is displayed. The macro can be enhanced to insert the drafter name, title, office symbol, and phone number and then return to the "SUBJECT" line. Using this example, several different macros can be created for each of the standard message headers. Once completed by pressing "F0 F9" and the letter for the macro, the entire message header can filled out. Although this macro seems long, I have one that runs some 75 lines and produces ten reports from three different data bases.

Macros that run from the keyboard are limited to one character as the name. If you have many macros, it is possible to run out of names. ENABLE handles this by permitting a macro to be named and run from another macro. This second macro, the one called from the basic macro, can be up to eight characters long and have an extension and be located on another drive or in another directory. The {Do Macro}name~ command is used to invoke this command.

In a menu based program that runs with several macros, errors may have been inducted by the operator. This person may know enough ENABLE to cause a process to hang up because of an unwanted keystroke. Using ENABLE's macro error trapping capability, many of these problems can be caught before they cause problems. The three error trapping commands are {If Error}, {Else}, and {Endif}. These commands would be inserted in the macro in locations where you think an ENABLE error statement may occur. If ENABLE generates an ERROR Statement, the macro will provide the instruction on how to handle it. All statements after the {If Error} command will be processed if an error is detected. If no error was encountered, all statements after the {Else} command will be used. The {Endif} command MUST be used after the error trapping routine to return processing to normal.

The {Voff} and {Von} macro commands permit you to stop the normal ENABLE screens from being displayed. An example would be using the data base menu from the main menu. If you do not turn off the screen, each ENABLE menu and selection will be displayed on the screen. This takes time and will slow down the process slightly. By inserting a {Voff} command, ENABLE's main menu or user menu will remain on the screen until an input is required. The {Von} command will turn the screen update back on so that the steps can be followed. I normally will not insert this command in a macro until it works correctly. This way I can watch the macro run and fix errors if they occur.

An example of this type macro would be:

```
{Voff}
UDIA
EXAMPLE~~
{Von}
{^F9}@
{BS}
{F10}S
{F10}Q
{ESC}
Q
```

This macro will freeze the screen at the main menu and then load the data base add function using "EXAMPLE" as the data base using the default input screen. The rest of the macro is like the "wait for a specific key" macro.

When I use macros to run "WHERE" clauses, I use break statements to help identify the data I need in the format I need. If you use the MCM Macro Create function, you cannot type in braces like they are inserted in the data base. You must enclose each brace in braces so that the break command looks like this:

```
date{}}b1{}},instr{}}a1{}},rating{}}
      a1{}}
```

Note the way that the brace is enclosed in other braces.

One special macro that can be created in ENABLE is the "0" macro in MCM. This macro can be specified, loaded and run when you first enter into ENABLE. This is indicated by typing in ENABLE (,,,,),@0 directly or having it in your AUTOEXEC.BAT file. Again I use this macro to drop through the ENABLE Profile screen and load a menu. An example could be:

```
{End}
{^F10}M
```

ENABLE's macro capability makes this great program even better. I use these macros frequently to assist me in many functions. They have helped me to draw lines in word processing documents and "standard" reports I have completed. This helpful capability saves many keystrokes. All ENABLE macros are upward compatible, that is macros created in version 1.15 will work in 2.0 or 2.15. Macros created in version 2.0 will not work in version 1.15.

In the next article I will go into detail on ENABLE's menu capability. Using these menus along with macros will further enhance applications that you wish to develop with the program. ✱

POWERING UP

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

Understanding Video Hardware

Although there are many things that are puzzling to new computer users, particularly those buying their first computer, my experience is that there is one subject which is universally confounding — the selection of a video card and CRT monitor. Unfortunately, most people rely on advice and suggestions from sales personnel in computer stores for this selection. What many people fail to realize is that in many computer stores (except for Heath stores), the sales staff is paid on a commission basis. The higher their dollar sales, the higher their commissions. That should be no big surprise, because many sales operations are run that way. The bad news is that many computer users spend a lot of money on hardware they do not really need. Or, they may not spend enough money and end up being dissatisfied with their systems simply because the display system is not adequate.

This article will help you understand something about how the video display system works in your computer. More importantly, it will help you choose your next display system so that it better meets your needs, since I assume you already have one on your current system. In order to help you make an intelligent choice for a video display, it is important to look at some basic terms and information that are used in this area. We'll start by looking at some of the kinds of CRTs that you can get.

Basic Video Terms and Concepts

As you may have found, there are many ways to display information on your computer. More importantly, there are a number of terms that are used to describe the television-like display for a terminal. The television-like display unit can be called any of the following: CRT (Cathode Ray Tube), monitor, video display or simply, display. These terms can be combined in any reasonable order to describe the same thing. When you buy a CRT display for your computer, you should first decide what kind of resolution (i.e., picture clarity) you want.

The type of resolution that a CRT has will determine how clear and sharp the display is. When a CRT is made, a phosphor coating is applied to the inside of the "tube" that presents the display. The actual display is created by "shooting" electrons in a particular pattern, and when an electron strikes the phosphor coating, that phosphor "dot" begins to "glow" for a short time. Appropriate combinations of these phosphor dots result in the display of various characters on the CRT. As it turns out, the resolution of a CRT is specifically related to these dots of phosphor.

The resolution or clarity of a CRT (monochrome or color) designed for use with a computer system is generally rated in pixels. The term PIXEL is short for "picture element" (which is basically a "dot" of phosphor) and is the smallest element on the CRT, which can be independently assigned color and intensity. And since resolution is determined by the number of pixels on a CRT, higher numbers of pixels (which means smaller pixels) mean better resolution.

To illustrate how this works, let's look at an example using the Zenith ZCM-1490 FTM (Flat Tension Mask) CRT. It has a rating of "640 × 480" (read as 640 by 480). The 640 is the number of pixels in a single "pixel line" (the horizontal resolution) and the 480 is the number of pixels in a single vertical "pixel column". These numbers are NOT the same as the text display capability, which is normally rated in terms of character columns and rows (e.g., 80 × 25).

If you compare that resolution to the Zenith ZVM-135 monitor, it only has a 640 × 240 resolution. Because the ZCM-1490 has a 640 × 480 resolution, you would expect that the higher number of vertical pixels (i.e., 480 vs. 240) would result in a "clearer" display (and it does).

A related factor is that CRTs with finer "shadow masks" — the metal grid through which the electron beams pass — usually display sharper images. The spacing between the grid slots (called the pitch) is usually on the order of 0.43 mm or less for high resolution monitors. For example, the ZVM-135 has a pitch of 0.43 with a pixel resolution of 640 × 240. If you can compare numbers for various kinds of CRTs, a unit with a SMALLER pitch (e.g., 0.29 mm) will generally give you a clearer, sharper display if the resolution is the same. We'll use this information later when we discuss more specific details about video displays, but let's take some time to look at another fundamental choice you need to make when buying a display system — Do you want monochrome or color display? Choosing one over the other is getting easier, but it still may not be simple.

Monochrome Versus Color

After you decide what kind of resolution you want in a CRT, the next thing is to determine whether you want a monochrome or color display. Although this choice should be fairly easy, there are some things that can complicate the choice.

If you can afford it, I recommend a color monitor. The main reason for that is because most of today's software does an excellent job with color displays, and the color capability represents more than just a nice display — it can make your work easier. For example, I prefer black text on a white background for writing because I think it is easier to see. How many books or documents have you read in the last 20 years that have green or amber text on a black background? For print enhancements, I have selected blue to mean bold, green to mean underline, and magenta to mean italics. I have found that this use of color makes it extremely easy to visualize what the final document will look like, and it is much easier to see than the equivalent display on all monochrome monitors. Depending on the software and the type of monitor, you may find that ALL print enhancements are displayed the same way (e.g., in reverse video), and it is not clear at a glance exactly what the print enhancement is. With color, it's easy to see. Still, monochrome monitors are popular because they generally are less expensive, as a group, than the color units, so we'll take a look at them first.

Monochrome Monitors

Monochrome monitors are single-color displays that typically display white, amber or green characters on a black background. The choice of which color is best for you is primarily a matter of personal choice, but there is some evidence that amber is the best choice because it is easier for most people to see. Amber seems to be a somewhat "brighter" color than green, and there are some indications that amber may help reduce eyestrain over long periods of viewing simply because the brightness level adjustment does not have to be as high. When I bought my first Heath computer (an H-89), I originally selected a green screen because I was used to that on mainframe systems. When I bought my second computer (an H-100), I selected an amber monitor. Why the amber instead of green? Well, I thought that the amber was easier on my eyes. Considering the amount of time that I spend at the computer, that was a very important consideration for me. Even though I like amber best for a monochrome monitor, be sure to look at different brands of CRTs because there are some differences in the shades of green and amber between brands. In case you are wondering why I did not mention white, my experience is that a white monitor tends to be difficult to read under some lighting conditions.

Choosing a color for the monochrome display is fairly easy, but there are, unfortunately, two completely different types of monochrome displays that are identified by which kind of signal from the computer drives them. The first is generally called an NTSC or composite display, and it is easily identified by its cable that has a phono plug (like the back of a VCR) which fits into an appropriate socket on the back of your computer. For reasons that will become evident in a minute, I usually like to refer to a monochrome composite CRT as a black/white (or B/W, for short) monitor to avoid confusion. And although we will talk about the special video display cards later, it is important to note that the monochrome composite CRTs are used with the Color Graphics Adapter (CGA) card. Resolution for this kind of CRT is generally rated as 640 × 200, which is the actual number of lines displayed in the actual image you see. You may also see these CRTs rated as 640 × 240, but the "extra" 40 lines are not normally available or used by software.

The second type of monochrome monitor is sometimes referred to as THE monochrome monitor or as a monochrome TTL (Transistor-to-Transistor Link) monitor. Most monochrome TTL monitors I have seen have a 3-wire plug. Aside from the obvious difference in the physical connections, a monochrome TTL CRT also REQUIRES the use of a special video card, such as the Monochrome Display Adapter (MDA) or the Hercules Card. Because of the technical differences, a monochrome TTL unit has a much higher resolution, typically on the order of 640 × 350 or so.

When you choose a monochrome monitor of either type, first select the color, then the resolution. The next step is to select a matching video card, but first let's take a look at the differences in color monitors.

Color Monitors

Color monitors are becoming increasingly popular, which is probably a direct result of the fact that, in general, their prices are decreasing compared to what they were a few years ago. Another reason is that more software is available with color capabilities, and users are generally finding that a color display is generally easier to use because certain things are much easier to see. As I mentioned earlier, I like to use blue for bold characters on my CRT, and that stands out nicely on a white background.

As you look at ads for color monitors, you will see all kinds of strange acronyms, such as RGB or RGBI, EGA or VGA. Although you can find other acronyms for color monitors, the key difference in all these types of monitors is the type of resolution aside from some other technical differences that are not important to this discussion. Since virtually all color monitors work in about the same way, it is useful to have an understanding of what is going on.

A typical color monitor is usually called an RGB monitor because of the basic Red, Green, and Blue colors it can display. The different colors that you can see are created by an appropriate mix of these basic colors. These monitors are always capable of displaying eight colors with seven actual colors, plus black (the CRT background) as the eighth color (i.e., no color at all). For PC compatible computers, CRTs commonly use Intensity (either "normal" or "bright") to create the standard 16-color display. When you see the RGBI acronym, it means that the CRT also has a separate wire for the "Intensity".

But a typical color monitor may have several types of input or physical connections. Some have RGB (and/or RGBI) input, some have composite input, and some have both. The RGB output is basically three separate signals on separate wires which are fed into the CRT and displayed as appropriate by one of the three electron guns (red, green or blue). In a composite video signal, the three color signals are basically combined (technically called multiplexing) into one signal which is then sent to the CRT. Monitors capable of processing a composite RGB signal have a special circuit which separates the red, green, and blue components of the signal for the electron guns.

Why do you care about the difference between RGB and composite video for this "typical" color monitor? The primary reason is that RGB input generally provides better resolution so that the picture seems to be "clearer" — very important in graphics applications. And in some cases, you will find that a CRT may be supplied with only a cable for the composite connection, although you can usually buy an RGB cable at extra cost. For a PC compatible computer, you will need a Color Graphics Adapter (CGA) card for an RGB monitor. Note this is the same card that is used with the monochrome composite monitor that was discussed earlier. The resolution for an RGB monitor is typically on the order of 640 × 200.

If you want to have a better color display, the next step is to get an EGA monitor which typically has a resolution of 640 × 350. Even though the numbers may not seem like much of a difference, it does represent a definite improvement in the display quality that really can be seen.

In today's technology, the best resolution is obtained with a VGA monitor that has a resolution on the order of 640 × 480. Again, the numbers are not very good at representing the real crispness and quality that a VGA monitor has.

These are the common types of color monitors, and the real difference in these units is the resolution. And although you may not have too much difficulty in deciding which type of monochrome or color monitor is best for you, it is important to understand that selecting the monitor is not enough — you also must select a corresponding video display card to be used with that monitor.

Video Display Cards

It is unfortunate that most computer-related video discussions tend to degenerate into acronyms regarding the merits of the CGA, MDA, EGA, and VGA. That kind of discussion can get extremely confusing very quickly, but the key thing to keep in mind is that all of these acronyms refer to the resolution of the display. That's why I suggest you select a CRT first (it is something you can see), then you can worry about the type of video display card you need.

When you get to the point of selecting a video card, the most important thing to know is that the display characteristics MUST be compatible with the CRT unit you have chosen. This becomes quite critical because you can actually destroy the electronics in a CRT by plugging it into the wrong video card. Fortunately, most CRTs and video cards have boxes and manuals that are clearly labeled, so it is difficult to make a mistake on your initial purchase. But, if you move your CRT to a different computer, be absolutely SURE that the video card in that computer is compatible with your CRT. Unfortunately, it is possible to make this kind of mistake because some of the physical connectors for the various types of CRTs may be the same. The good news is that it is relatively easy to choose a video card, once you have selected a CRT you like. Here's how that works.

The most common kind of video card in a PC compatible computer is called a Color Graphics Adapter or CGA card. This card is used to supply the appropriate signals for either a composite monochrome monitor or color (i.e., standard RGB) CRT. For color CRTs, most CGA cards can run them with either a composite or RGB (or RGBI) signal, but I suggest buying an RGB cable (if necessary) as suggested earlier for best display results. Since this card is directly matched to either a composite monochrome or RGB, it can provide a 640 × 200 resolution for those monitors. For example, many of the older Z-200 computers contain the Z-409 video card for a CGA-compatible display. When the CGA is used for a color display, it is capable of displaying 16 colors based on a signal for each of the red, green, and blue lines, plus an intensity signal which is either on (for "bright") or off (for "normal").

While I'm on the subject of CGA cards, it is important to make a specific note about some of the Heath/Zenith computer systems. Many of the Heath/Zenith computers (e.g., Z-138, '148, '151, '158, and '161) contain a built-in CGA video card that is physically part of another "board" inside the computer. Although this kind of construction has some advantages (such as lower cost), its major disadvantage is that it is not easy to upgrade to a different video card that can drive a higher resolution monitor.

It is not just a matter of simply replacing the video card in these computers; so you will have some additional work to upgrade those systems. If you are really interested in doing that, I suggest that you take a look at the advertisements in REMark and look for a "video eliminator kit" that can be used to bypass the existing built-in CGA unit. Be advised that you will have to disassemble your computer to install this kit if it is available for your computer. In general, I do not recommend this approach because it may cause problems (with some software) later on, but it is important to mention it as an alternative if you have one of these systems. I also feel uncomfortable with any hardware add-on that requires a physical change to something inside the computer because it may not be reversible in the event that something goes wrong. And depending on how the change is made, it can even void the warranty for your computer. If it sounds like I am trying to talk you out of this approach, I am.

Another choice is the Monochrome Display Adapter (MDA) that is ONLY used with monochrome TTL CRTs. The MDA is not a good choice for most general applications because it will not display the standard IBM graphics characters, because its memory contains other features. In some cases, you may want to sacrifice the display of IBM graphics characters in favor of a much better display for a specific application, such as word processing. I have seen several word processors working with the MDA card, and it is interesting to SEE the underline on the screen — a standard CGA card cannot do that because it does not have sufficient resolution. But, some software (e.g., games in particular) will simply not work with the MDA card because of the program's use of graphics characters. The MDA is an excellent product, but I do not recommend it because it may or may not work with all the software you have. If you want better resolution on the order of 640 × 350 for a monochrome TTL display, this may be a good choice for text-specific applications, but be sure that you don't need graphics characters for your other programs.

But there is at least one other choice for a monochrome TTL CRT: the Hercules card. The folks at Hercules invented this card to overcome IBM's omission of the graphics characters in the MDA. That is, the Hercules card CAN display the graphics characters which is important to graphics-intensive applications, such as Lotus 1-2-3. In fact, some software (e.g., Lotus 1-2-3) has a special software driver that allows the full use of the Hercules' display features. That, coupled with the Hercules' better 720 × 350 resolution, made IBM's MDA virtually obsolete in a very short time. Because of its graphics capability and the availability of special drivers, most software is compatible with the Hercules card, but you should check your documentation to be sure.

To add color capabilities with improved resolution, the next step is the Enhanced Graphics Adapter, or EGA, that is capable of resolutions up to 640 × 350. The EGA can display 64 colors because there are four intensities for each of its color combinations. Although you can find monochrome monitors that will display this kind of resolution, most EGA-compatible monitors provide color capabilities, although you can find some less expensive ones that display monochrome only.

When IBM announced the PS/2 series computers, a new video standard called VGA (Video Graphics Array) was also introduced. The VGA system is capable of producing resolutions of 640 × 480. Because the VGA has a considerably different kind of design than the other display standards, it is capable of displaying a maximum of 256 colors at a time out of a total palette of 262,144 possible.

This summarizes the significant features and display capabilities of all major video display standards. There are others, but these are the most popular and common ones. How all this works from a technical perspective is not particularly important, but there is a facet of video display technology that I have purposely neglected up to now.

Multi-Synchronous Monitors And Scan Rates

If you take a look at the different kinds of CRTs available today, you will see some that are advertised which have the "multi-synchronous" feature. For example, I currently use an NEC MultiSync monitor with my '248 system. What does a "MultiSync" monitor really do?

In order to keep from confusing the hardware discussion, I have ignored the subject of scan rates, even though it is quite important to understanding video hardware. To understand how this works, consider that the CRT screen is nothing more than a bunch of tightly-packed "dots" (i.e., pixels) which must be illuminated in various patterns (characters) to produce the image you see. There is, however, a little more to it than that.

Pixels are arranged, as you might expect, in rows and columns, and the number of pixels is specified by the resolution. Consider that the standard CGA resolution is 640 × 200. In order to illuminate the pixels, the electron beam is rapidly moved (by electronics, not physical movement) from left to right on a single line. As each line is illuminated, or "traced", the electron beam moves to the next line and illuminates it. And so on. Pixels on each line are illuminated in this way (from left to right, top to bottom) with a horizontal scanning frequency of 15,750 cycles per second (or 15.75 KHz if you prefer). In order to produce what appears to be a continuous image, the display must be redrawn, or refreshed, at least 60 times per second. This technique relies on the human eye's persistence of vision to create the illusion of a single image on the CRT. And by the way, a television works in exactly this same way, which is one reason that many TVs were used as computer monitors in the "old" days.

The CGA horizontal scan rate is 15,750 Hz (60 Hz refresh rate) with a 640 × 200 resolution. The MDA has a scan rate of 18,432 Hz (50 Hz refresh rate) with a 640 × 350 resolution. The EGA uses a 21,800 Hz horizontal scan rate (60 Hz refresh rate) for a 640 × 350 resolution with color. And the VGA runs at a screaming horizontal scan rate of 31,500 Hz. Now consider that your video card and CRT must work together as a SYSTEM.

Because of the system concept, the video card must produce the proper horizontal and vertical frequencies that SYNCHRONIZE with the CRTs. That's why an EGA-only monitor, running at 21,800 Hz, will simply not work with a CGA card that runs at 15,750 Hz. From a hardware manufacturer's perspective, that was great because it required you to buy another video card and monitor every time you upgraded to a higher resolution system.

Then the folks at NEC (Nippon Electric Co. Ltd.) quietly developed a brilliant solution to that problem — a monitor that could automatically accept (i.e., synchronize with) an incredible range of horizontal and vertical frequencies, as well as both analog and digital intensity signals. The original NEC MultiSync monitor became almost an overnight hit because it was virtually immune to the built-in obsolescence of other single-frequency monitors. For example, my NEC User's Manual states that the MultiSync provides automatic synchronization for horizontal frequencies in the range of 15.5 KHz to 35 KHz. Needless to say, other manufacturers quickly developed multi-synchronous monitors too, but the original NEC MultiSync was (and still is) hard to beat.

At this point, you should be able to define your requirements for a video card and monitor, but there is one other consideration that is important to most of us: cost.

The Cost of Video Hardware

As you move toward higher resolution in a video system, the cost of that system also increases. To give you a general idea of how CRT prices change based on the display capability, I have shown the list prices of Zenith monitors in Figure 1.

As the capabilities and/or the resolution of the CRTs increase, so do the prices. Obviously, the ZCM-1490 has the best resolution with the widest range of video card capabilities, but it is also the most expensive for that reason. Other brands of monitors will have similar price differences, depending on their capabilities, but it pays to shop around for the best deal.

Many of the Heath and Zenith desktop computers are supplied with the Z-449 video card that can provide output for CGA, MDA, Hercules, EGA at one port and a separate port for the VGA-compatible Zenith FTM (ZCM-1490) monitor. The Z-449 has a list price of $499, which is in the same range as other similar cards that provide the capability to automatically switch to one of several modes. CGA cards can be found for around $100, if you need one. MDA and Hercules cards generally cost in the $150-200 range. EGA cards can range from about $250-400 depending on their capabilities — some provide the capability to adjust the display mode depending on the CRT used (like the Z-449). And finally, VGA cards generally have list prices in the $450-700 range.

Model	Price	Works With
ZVM-1240	169.95	MDA only
ZMM-1470-G	299.00	CGA or EGA (monochrome only)
ZVM-135	599.00	CGA (RGB), Composite
ZVM-1330	649.00	CGA (RGBI)
ZCM-1390	699.00	VGA
ZCM-1490	999.00	VGA

Figure 1
Comparison of Zenith Monitors

Since nearly all Heath and Zenith computers (except laptops) have a CGA-compatible display capability, about the only cost here is just for the monitor. If you have a Z-100 or a PC compatible system (with CGA), I can also recommend the Goldstar 12" monochrome (amber only) monitor (MBM-2105-A) that is available from the Dallas Heath store for about $150. I have received a number of letters, particularly from Z-100 owners whose old CRT has "given up the ghost", asking about a "regular" monochrome composite monitor (not TTL) because Zenith has discontinued that particular line. And if you are having difficulty locating an RGB monitor at a reasonable price, you might also want to call or write to the Dallas Heath store because they also have other Goldstar monitors available at reasonable prices.

At the low end of the resolution scale, you can expect to spend about $150 for a monochrome CGA monitor to over $600 for a CGA color monitor. At the high end, you can see that it is easy to spend more than $1,000 for the high-resolution VGA systems.

The Bottom Line

One can argue endlessly about the various merits of all of the display systems, their resolutions, and their costs; but I think there is one other idea that can help you ignore some of the more technical details. This does not eliminate the need to know some of the other information discussed in this article, but it will help you make the best choice for you and your system. To help summarize the differences among the resolutions for the various video display standards, I have summarized them in Figure 2.

I think that the absolute most important consideration is to actually SEE the results of a specific video card and CRT on a computer that has the same model number as the one you own. I don't think you would buy a chair without sitting in it first; don't buy a display system like that either. A display system that looks good on a '248 system may or may not look as good on a '386. A display system that looks good on a '158 may or may look that way on a '148. And so on. Different computer models have different internal hardware, and if you don't insist on seeing the results BEFORE you buy something, you may be disappointed when you get home. This may require your taking your computer into a store for a "test" of a display system, and while most people in sales don't like this kind of thing, you can usually get them (sometimes grudingly) to help you.

Aside from the internal differences in computer models, certain brands of video cards seem to work better with some

Type	Display Resolution
CGA	640 × 200
MDA	640 × 350
Hercules	720 × 350
EGA	640 × 350
VGA	640 × 480

Figure 2
Standard Video Display Resolutions

brands of CRTs. For example, I have found my NEC MultiSync monitor (the old version) seems to work best with the Video 7 EGA card on my '248 system. That is strictly a personal judgment, and you may not agree; but I thought that was the best combination for me when I saw it. I compared that directly with a Paradise EGA card (again using the NEC MultiSync), and I thought the colors were a little better, the display was a little sharper, and it seemed like the Video 7 card was a little faster. I bought what I liked, and I have been extremely pleased with that combination ever since. And although I have no hesitation in recommending that combination to anyone with a '248 system, I still suggest that you take some time to see for yourself if you are considering upgrading to an EGA on that specific model. Perhaps you will find a combination that you like better.

One other point about choosing a display system also needs to be made. Even if you have already decided (and can afford) to upgrade to a high-resolution display, don't discount the possibility that you can save money with a lower-cost unit. I found a C. Itoh CM-1000 monitor (now discontinued) with the standard Z-409 CGA card also worked extremely well in my '248. I decided to upgrade to an EGA display because I use my system for so much writing. If you spend 10 or more hours a day at your computer, it is not too difficult to justify that kind of cost. But if you only use your system for occasional work, such as evenings and weekends, the cost of a higher resolution display may not be justified. You may find that all you need is a different monitor that "fits" you better. Again, be sure to take a look at the different possibilities before you buy something — you may be able to save a considerable amount of money.

Next Time

After you have learned to really use your computer, you begin to see how valuable additional memory can be. And if you look at some of the popular literature, you will find all kinds of terminology that refers to various ways to add memory: RAM, extended memory, expanded memory, and EMS. Some systems have specific restrictions as to what kind of memory you can add, and we will take a look at that next time.

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

Products Discussed

VGA/MDA/EGA/VGA Video Card
(Z-449) $499.00
Monitors (See Text)
Heath/Zenith Computer Centers
Heath Company Parts Department
Hilltop Road
St. Joseph, MI 49085
(800) 253-7057
(Heath Catalog orders only)

Goldstar 12" amber CRT
(MBM-2105-A) $150.00
Heath/Zenith Computer Center
12022C Garland Road
Dallas, TX 75218
(214) 327-4835
(Ask for Allen Kern or John Mitchell)

Continued from Page 4

Here is an explanation of the files:

WSZ100.DOC — Instructions for using WordStar on a Z-100.

WSZ100.COM — This program is the parent program that runs WordStar (WS.EXE) on a Z-100. For normal operation, you would rename this file to WS.COM so that WordStar could be run using the normal WS command.

WINSTALL.COM — This is a parent program that runs WINSTALL.EXE on a Z-100.

WSCHANGE.COM — This is a parent program that runs WSCHANGE.EXE on a Z-100.

Z100.PAT — This is an auto-patch file for use with WSCHANGE. The patches in this file must be applied before WordStar will work with WSZ100.COM on a Z-100.

COLORS.PAT — This is an auto-patch file that sets up the colors used in WordStar so that they look good on a Z-100. This patch can also be used with WordStar on a PC-compatible, so that WordStar has the same colors on both computers. A color scheme is used that does not involve intense colors, which are not supported on a Z-100.

KEYS.PAT — This is an auto-patch file that sets up the keypad so that it works like the H/Z-89 version of WordStar 3.3. This method of using the keypad is more versatile than the PC layout. This patch can also be used with WordStar on a PC-compatible when it is used with the WSPC.COM parent program (described below).

WSLIST.COM — This is a parent program that allows the WSLIST.COM provided with WordStar 5 to run on a Z-100.

FIXWSL.COM — This program patches the WordStar 5 WSLIST program so that it can run under the WSLIST parent (above).

PRCHANGE.COM — This is a parent program that runs PRCHANGE.EXE on a Z-100.

PDFEDIT.COM — This is a parent program that runs PDFEDIT.EXE on a Z-100.

LSRFONTS.COM — This is a parent program that runs LSRFONTS.EXE on a Z-100.

PF.COM — This is a parent program that runs Pro Finder (PF.EXE) (supplied with WordStar 5) on a Z-100.

PFINST.COM — This is a parent program that runs the Pro Finder installation program (PFINST.EXE) on a Z-100.

READ.COM — This is a parent program that runs the README.COM program supplied with WordStar 5 on a Z-100. README.COM is used to display or print a file containing additional documentation not found in the manual.

WSPC.COM — This file contains instructions for using the WSPC.COM parent program on a PC-compatible computer to add features to WordStar.

PC.PAT — This is an auto-patch file containing a patch to WordStar to allow it to work with WSPC.COM. The patch is applied using WSCHANGE.

***.ASM** — The assembly source code files are provided for all .COM files on this disk.

TABLE C Rating: 3, 7, 10.

Classified Ads

Z-100 HARDWARE: PSS-T Card with Programmed Library: Dual 8" Drives Complete. Offer. (206) 365-1542

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

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...

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 compatibilty 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!

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 ours 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.

Upgrades Available:

FCC-150/286-UP: Z-151/2/8/9 or Z-161 to Z-286 80286 AT compatible (16 bit). $1295.
FCC-150/386-UP: Z-151/2/8/9 or Z-161 to Z-386 80386 computer (32 bit processor). $2495.
FCC-248/386-UP: Z-241/248 to Z-386 80386 model (32 bit processor). $1995.

Now... Even *More* Upgrades:

'FCC-150/CL-AT: Z151/2/8/9 or Z-161 to First Capitol 12MHz CL-AT (16 bit) Clone. $1095
FCC-150/CL38TW: Z151/2/8/9 or Z-161 to First Capitol 20MHz CL-386TW (32bit)Tower.$2295.
FCC-248/CL38TW: Z-241/248 to First Capitol 20MHz CL-386TW (32 bit) Tower. $1795.

Also Works 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

Circle #105 on Reader Service Card

BUG ZAPPING

Problem: The ZKB-2 keyboard intermittently loses contact with the computer.

Solution: On the foil side of the keyboard, the keyboard cable is routed over sharp connector pins on the board. These pins can puncture the insulation of the frame ground lead, shorting the keyboard. In some computers, the short may blow a 2.5A pico fuse (p/n 421-72) in the +5 VDC supply to the keyboard. To correct this problem, route the ground lead to the other side of the keyboard cable.

Problem: The Z-319 card won't work with MFM-150 version 3.0 or higher.

Solution: In the MFM-150 version 3.0 or higher system ROMs for the Z-100 PC, the code that supports the Z-319 card has been removed. Software that attempts to set the Z-100 mode will no longer function properly. Attempts to use the ROM routine SET_MACHINE_MODE to enter Z-100 mode now causes an error message to be displayed and invokes the ROM debugger.

Problem: The H-140 or Z-159 series computer locks up when running the monitor ROM-based memory test.

Solution: H-140: On the #181-5751 CPU/Memory board, the monitor ROM should be changed from Version 3.0 to Version 3.1C. The firmware change corrects a problem with the ROM-based memory test. During this test, the computer would lock up when the INS key was pressed. Also, the new firmware corrects problems with IBM's Spanish version of LOGO and with the video scroll routines.

Z-159: On the #181- 6836 CPU board, the monitor ROM should be changed from Version 3.0 to Version 3.1C. This ROM corrects the same problem as that of the H-140 series computer.

Problem: The H-150 keyboard seems to lock up, cause phantom characters, CAPS LOCK function reversed, keys toggle, etc.

Solution: On the foil side of the #181-5059 CPU board, install a 270pF capacitor between pin 7 and pin 9 of U202 on PCB #85-3017 or U203 on PCB #85-2969. Keep the capacitor leads as short as possible and lay the capacitor flat against the board.

Problem: On the Z-200 series computer, some 84-key keyboards generate extra keystrokes when running MS OS/2.

Solution: To correct this problem, change the keyboard ROM from a version 1.3 to a newer version 1.7A.

Problem: The Z-386 PC series computer CPU locks up during coprocessor math processing.

Solution: On the #181-7043 CPU board, the coprocessor interface PAL at U201 is changed from a part number #444-535-2 to a #444-535-3. The code was changed because the CPU would lock up when a divide by zero or an overrun error was generated.

Problem: The ZVM-1220, ZVM-1230, and ZVM-1240 monitor totally dead; power LED not lit, and the fuse is not blown.

Solution: In the B+ power supply, check for a shorted switching transistor at QX501 (Zenith #121-1142), an open winding between pins 10 and 12 in the inverter transformer at TX502 (Zenith #95-3891-01), and a burned and open resistor at TX501 (Zenith #63-10836-32).

Problem: Moving the ZCM-1490 by a small amount, causes color purity change.

Solution: When the monitor is on and its position is changed, you'll see a change in the color purity on the screen. This is caused by changing the relative position of the CRT to the stray magnetic fields present in most environments. It's recommended that if the monitor position is changed while the monitor is on, turn it off and wait at least 15 minutes for the degaussing circuits to reset. Then, with the monitor in the desired position, turn it on. An external degaussing coil, such as the one used to degauss a color TV CRT, can also be used. However, if the monitor has a serial number that ends with "NOA" or "NOB" and correct purity cannot be achieved, replace the power supply module.

Editor's Note: *As further service to our readers, I'd like to keep this column in REMark on somewhat of a regular basis. In it, we'll hopefully be able to keep you abreast on ROM versions, simple fixes for known problems, etc. If you have a fix that can benefit other HUGGIES, send in the Problem/Solution on a post card, along with a request for ANY single piece of HUG Library software, and, if we use it here in the magazine, we'll send your requested software product to you FREE!*

Continued from Page 6

running. To use the thesaurus and speller, replace the data disk in drive B with the thesaurus or speller disk. Then press alt-F1 (help-F1) to ctrl-F2 (F0-F2), respectively. WordPerfect will look for the necessary files in drive B when it doesn't find them in drive A, provided that PATH=B: as mentioned above or if you have changed the default drive to B: with the F5 key (list files). Replace the disk in drive B with the data disk when saving text.

The only problem I have had with WordPerfect 5.0 using 360k floppy disks is that the shell command (ctrl-F1) does not work because COMMAND.COM cannot be found. There is no room for it on WordPerfect disk 2. Putting it on another drive and naming that drive in the PATH command does not solve the problem. A hard drive may solve this problem. I can live without a shell. No other problems have been encountered, although I have not yet used every single feature that WordPerfect has to offer. I have yet to meet the person who has.

Larry Motyka
34 Granger Place
Buffalo, NY 14222

ZPC Update #23, October '88

Dear HUG:

The improvement to ZPC described in the October 1988 REMark (page 11), which is intended to speed up screen displays in mode 6, has a few problems:

(a) It doesn't display the bottom row of pixels. This is due to a bug, for which the fix is given below.

(b) The top left corner of the screen (where the Num Lock status is displayed) flickers whenever the Num Lock status is visible ("on" when the pixels there are turned off, or "off" when the pixels are supposed to be on). (You may be able to fix this by pressing the F11 key.)

(c) Some displays (such as text) are slower in the new version than the original. For example, a directory display that took 41 seconds using the old version (vs. 8.5 seconds in modes 3 and 7, and 3 seconds in Z-100 mode), took 46 seconds using the "improved" version. (On the other hand, it is possible that rapidly-changing graphics displays might be speeded up using the new version.)

The reason for (b) and (c) is that the new version doesn't bother to check whether the screen has changed or not, whereas the old version does. This will make the new version run more slowly whenever the screen is changing only moderately often, as in some text displays — it should run more quickly than the old version if the screen is changing rapidly.

To fix the bug which leaves the bottom row of pixels blank, replace the last two lines of code in procedure COPYGM, which read:

```
CMP  BX,396
JB   NEXT2L
```

with the following three lines:

```
CMP  BX,400
JB   NEXT2L
RET
```

I hope this is of some use.

Yours sincerely,

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

Speeding Up ZPC

Dear Pat:

After installing the patches from the latest ZPC Update for faster operation in high resolution graphic mode, I went lookng for other ways to speed up ZPC.

I found that the variable CHKMAX in the file DATA.ACM has the value 6. I don't know why you picked this particular value for the frequency of updating the Z-100 vram from the ZPC emulated vram, but I thought that less frequent updating should reduce the overhead of checking for changes in emulated vram and in copying.

I experimented with REFLEX, a database program that uses high resolution graphics mode, and WHERE IN THE WORLD IS CARMEN SANDIEGO, a color graphics game that includes some animation.

Changing CHKMAX to 12 resulted in a noticeable improvement. Changing to 255 was even better. I suppose that for some programs a lower value of CHKMAX might be better, but at least with the programs I use, the higher, the better. Programs that use text mode, like the Norton Utilities NI also go faster.

Sincerely yours,
Stanley Schwartz
60 Edgehill Road
Providence, RI 02906

Refer to "Buggin' HUG" 9-88

Dear HUG:

Reference my letter that appeared in the "Buggin' HUG" column of the September 1988 issue of REMark about incompatibility problems with WordPerfect version 5 and the Z-100. Good news as a result of a letter I recently received from Emmie Olsen of WordPerfect's Customer Support department. WP 5 will run in the IBM-mode on a Zenith Z-100 with a Gemini-PC emulator board! She pointed out that the Z-100 requires a file named NOSHELL.COM in order to run WordPerfect version 4.2. She indicated this program, included with WP 4.2, "is often effective in correcting clock resetting problems, as well as some other hardware configuration incompatibilities." (I've successfully used WP 4.2 on my Z-100 in the IBM-mode without running NOSHELL .COM, but had problems with the clock resetting to an invalid time.) In WP 5.0, the NOSHELL.COM program has been renamed FIXBIOS.COM and is found on the "Conversion" disk. She also stated that many Zenith users must use the "/NC" option (WP/NC) when starting WP 5 on their computers. I followed her suggestion to include the FIXBIOS.COM program in my AUTOEXEC.BAT file and to use the "/NC" start-up option. Success! No longer does WP 5 lock up my Z-100 when it "autosaves" a file. The tutorial program seems to work fine now, too. Admittedly, I have not tried out all of WP 5's many features on my Z-100 yet, but the biggest problems seem to be solved for now. Many thanks to Emmie and WordPerfect Corporation for their excellent customer support program. I highly recommend WordPerfect V5.0 (or 4.2) to anyone in need of a word processing program for their computer.

In his "On the Leading Edge" column in the October 1988 issue of REMark, Bill Adney mentioned the internal 30MB hard disk system for the Z-148/Z-150 available through Payload Computer Services. I have installed one of these systems in my Z-148 and it works great. Initially, I had a problem with the first drive that Payload sent me. I could not get the computer to recognize that a hard disk was installed. I took my '148 to a local Zenith dealer and they checked it out with a Zenith hard disk controller board installed and couldn't get the drive to work either. After reporting the problem to Payload, they promptly sent a replacement 30MB hard disk drive (a Seagate ST-238R) and the latest version of the Western Digital half-size RLL hard disk controller card. I've had the system in use now for several months without any problems.

With companies such as WordPerfect Corporation and Payload Computer Services, much of the uncertainty associated with computer hardware and software is gone. In my experience, they are companies you can rely on.

Sincerely,
Gary R. Evens
PSC Box 8533
APO New York 09012

HEATH ZENITH

Related Products

Jim Buszkiewicz
HUG Managing Editor

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

The WonUnder II is an innovative dual slot expansion unit that allows two full-size PC expansion cards to be added to a Zenith Z-183 laptop or eaZy pc desktop computer. It attaches to the expansion port in back of the Z-183 or eaZy pc. The WonUnder II's small size and low profile make it easy to take with or to leave behind on your desk — connecting and disconnecting takes seconds.

The WonUnder II consists of a sturdy metal expansion chassis which accommodates two standard, full-size PC expansion cards, an interface cable, and a power supply.

Expansion. Almost any PC-compatible standard expansion card can be installed in the WonUnder II. You can add cards for EGA graphics support, additional memory, tape backup, and instrumentation, extending the capabilities of your Zenith computer.

Connectivity. Ideal for connection to a network, the WonUnder II supports network interface cards, such as Token Ring, Arcnet, and others. The WonUnder II is also well suited for mainframe terminal emulation applications, supporting IRMA cards. In addition, when combined with the WonUnder II, the Z-183 or eaZy pc can be used as a remote portable terminal by installing a remote communication card.

WonUnder II $459.00
Connect Computer Company
9855 West 78th Street, Suite 270
Eden Prairie, MN 55344
(612) 944-0181
(612) 944-9298 (FAX)

The easiest way to manage your list is **WhizList** which is produced by Arrow Connection. WhizList takes charge of any type list. No matter what type of list you have — mailing list, customer list or any others — WhizList will give you complete control of it. No need to go through manuals or get involved with programming. WhizList does the job for you. With its menu-driven format you get choices, which you can command with a touch of the key. Dollar for dollar, WhizList gives you more performance at a lower cost. Competing programs that sell for up to $149.00 give you a lot less than WhizList.

Here are the features that make WhizList the best value on the market: more flexible import/export, sort on any field, merge 15 files at once, near dupe recognition, Nth record sampling, more flexible use of fields allows you to delete or add or reposition fields, more versatile find/replace with 5 kinds of wildcards, mailmerge compatible, select on multiple conditions, print in any possible format, faster editing, and much, much more.

Arrow Connection lets you try it for 10 days with absolutely no risk. Call (800) 824-2222, ext. 63. They ship quickly and bill you $49.95 plus $3 shipping. (Calif. residents add sales tax.) Or you may mail check to:

WhizList $49.95
Arrow Connection
P.O. Box 899-R2
Pollock Pines, CA 95726.

Hogware Company announces **ShowU**, a new product for Zenith Z-100 and PC-compatible computers.

ShowU allows Z-100 and PC computers to share graphics screen data. ShowU captures Z-100 or PC graphics screens and displays graphics on either computer.

Applications for ShowU include:

- Capturing a screen image, like a bar or pie chart, created on a PC; enhancing it on the Z-100 using ShowOff to add professional quality text and up to 92 colors; then, displaying the enhanced image on a PC with VGA display, and printing it for a business report.
- Creating slide shows for the Z-100 or the PC that combine graphics from several different computers with different display adapters.
- Translating clip art files designed to be used with ShowOff or desktop publishing software into a format that can be used by either the Z-100 or PC.

ShowU captures normal Z-100, high resolution Z-100, CGA, Hercules, EGA, and VGA graphics screens. ShowU also reads, translates, and displays popular Z-100, Macintosh, and PC graphics file formats.

Other products from Hogware include ShowOff, the high resolution graphics editor for the Z-100, and ZIP Image Processing software for the Z-100 and PC-compatibles.

ShowOff, ZIP Image Processing, and ShowU are trademarks of Hogware Company.

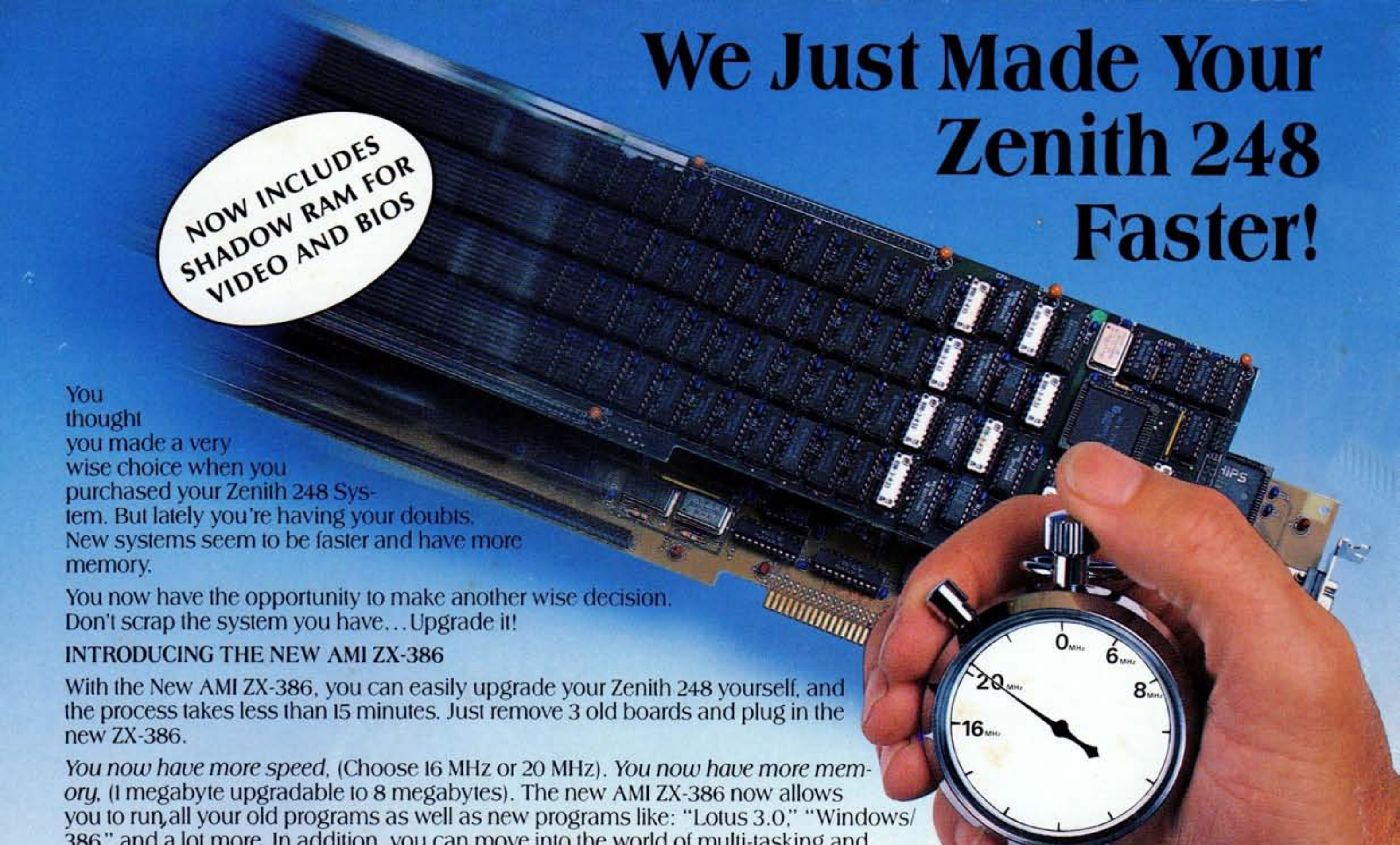
Smart Graphics is the agency of record for Hogware Company. For further information, contact:

ShowU $34.00
Janet M. Hirsch
Smart Graphics
470 Belleview Avenue
Webster Groves, MO 63119
(314) 962-7833

Lap Top Survival Kit. Electronic Specialists announce their **Lap Top Survival Kit** for the on-the-road, on-the-go Lap Top computers. Gathered in the kit are essential accessories for hassle-free computer operation in motels, hotels and phone booths.

Featuring Electronic Specialists' Pocket Protector AC Power and Communication-Computer-Modem Security units, this kit includes often needed adapters, tools and cables for the traveling computer. Packed with the Pocket Protectors in a durable plastic container are such indispensibles as screwdrivers; clip lead modular connector phone tap; RJ-11 "tee" adapter to expand a hotel phone jack into two phone jacks; 4-pin to modular RJ-11 adapter; 25-foot modular type phone extension cord; AC Power 2 prong to 3 prong adapter plug; and an AC Power triple tap outlet adapter.

Lap Top Survival Kit
Model LTP-301 $149.95
Electronic Specialists, Inc.
171 S. Main Street
Natick, MA 01760
(800) 225-4876
(508) 655-1532 (In Mass.)



Heath/ZENITH Users' Group

P.O. Box 217
Benton Harbor, MI 49022-0217

POSTMASTER: If undeliverable, please do not return.

P/N 885-2109