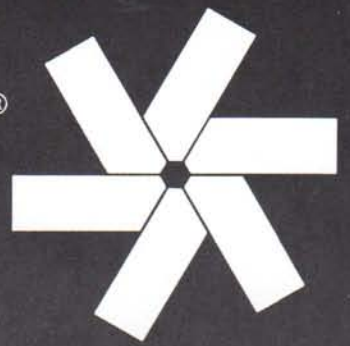


REMark[®]

May 1991



The Official Zenith Data Systems Computer Users Magazine

Zenith Data Systems Announces . . .



the

MastersPort

Watch for a Closer Look at this Outstanding Laptop
In an Upcoming Issue!

A decorative border of stylized flames in yellow, orange, and red surrounds the central text. The flames are jagged and layered, creating a sense of heat and intensity.

HADES II

It's HOTTERR than ever! Jam-packed with new features, HADES II still remains the easiest-to-use disk editor ever! Just look at some of the features:

- Sector Display/Editing
- Sector HEX/ASCII String Search
- File Display/Editing
- Physical and Logical Cluster Display
- File HEX/ASCII String Search
- Drive Parameter Display
- 512 MegaByte Drive Size Limit
- File Attribute Display/Edit
- Automatic Erased File Recovery
- Manual Rebuild File Recovery
- Works with Headerless MS-DOS Disks
- PC-Compatible or H/Z-100

HADES II is still only \$40, and original HADES owners can upgrade their distribution disk for only \$15. Call HUG today at: (616) 982-3463.

REMark[®]

May 1991



The Official Zenith Data Systems Users Magazine

Resources

Software Price List 2

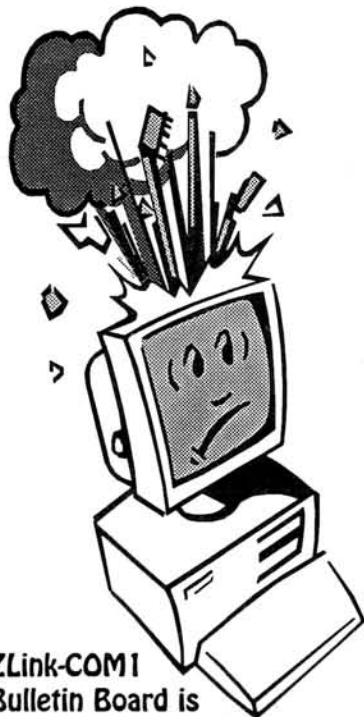
Advertising

Reader
Service
No.

Page
No.

250 B.U.D. Unlimited	22
104 FBE Research Co., Inc.	19
193 QuikData, Inc.	6
153 Surplus Trading Corp.	26
149 W.S. Electronics	48

Hardware Problems?



ZLink-COM1
Bulletin Board is
Your Best Source for
Technical Information!
Modem: (616) 982-3956

PC Compatibles

Breaking the Churlish Bounds of DOS

Part 3

David W. Lind

13

The World of WP50 and Its Wonders

Part IV

Salli Brackett

27

Z-100 & PC Compatibles

On the Leading Edge

William M. Adney

9

ENABLE Revisited — Part 1

George P. Elwood

35

Programming the Function Keys in MS-DOS

Pietro A. Carboni

39

General

Port-O-Call: COM1

Laura White

5

Do You Want to Write Another Program?

Part II

Gil Hoellerich

7

Stay Cool, Dude!

Jack W. Bazhaw

23

Introduction to C++ - Sixth Installment

Lynwood H. Wilson

29

Search for the Perfect Word Processor

Laszlo M. Vesei

33

Planes, Hotels, and Lizards

Stephen M. Kristan

43

Index to REMark: 1990

Jan Axelson

46

Managing Editor
Jim Buszkiewicz
(616) 982-3837

Software Engineer
Pat Swayne
(616) 982-3463

Production Coordinator
Lori Lerch
(616) 982-3794

Secretary
Lisa Cobb
(616) 982-3463

COM1 Bulletin Board
(616) 982-3956
(Modem Only)

ZUG
Software Orders
(616) 982-3463

Contributing Editor
William M. Adney

Printer
Imperial Printing
St. Joseph, MI

Contributing Editor
Robert C. Brenner

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

Software

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM		DESCRIPTION	PRICE
		H8 - H/Z-89/90			
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 PKG	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		CHKBK 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 PKG	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 SECT SUPPORT PKG	885-1121	HDOS		UTILITY	30.00
HDOS PROG. HELPER	885-8017	HDOS		UTILITY	16.00
HOME FINANCE	885-1070	HDOS		BUSINESS	18.00
HUG DISK DUP UTILITY	885-1217-[37]	CPM		UTILITY	20.00
HUG SOFTWARE CATALOG	885-4500	VARIOUS		PROD TO 1982	9.75
HUGMAN & MOVIE ANIM	885-1124	HDOS		ENTERTAINMENT	20.00
INFO SYS AND TEL. & MAIL SYS	885-1108-[37]	HDOS		DBMS	30.00
LOGBOOK	885-1107-[37]	HDOS		AMATEUR RADIO	30.00
MAGBASE	885-1249-[37]	CPM		MAGAZINE DB	25.00
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 DEC '80	20.00
RUNOFF	885-1025	HDOS		TEXT PROC	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 PKG	885-1127-[37]	HDOS		UTILITY	20.00
STUDENT'S STATISTICS PKG	885-8021	HDOS		EDUCATION	20.00
SUBMIT (Z80 ONLY)	885-8006	HDOS		UTILITY	20.00
TERM & HTOC	885-1207-[37]	CPM		COMMUN & UTIL	20.00
TINY BASIC COMPILER	885-1132-[37]	HDOS		LANGUAGE	25.00
TINY PASCAL	885-1086-[37]	HDOS		LANGUAGE	20.00
UDUMP	885-8004	HDOS		UTILITY	35.00
UTILITIES	885-1212-[37]	CPM		UTILITY	20.00
UTILITIES BY PS	885-1126	HDOS		UTILITY	20.00
VARIETY PACKAGE	885-1135-[37]	HDOS		UTILITY & GAMES	20.00
WHEW UTILITIES	885-1120-[37]	HDOS		UTILITY	20.00
XMET ROBOT X-ASSEMBLER	885-1229-[37]	CPM		UTILITY	20.00
Z80 ASSEMBLER	885-1078-[37]	HDOS		UTILITY	25.00
Z80 DEBUGGING TOOL (ALDT)	885-1116	HDOS		UTILITY	20.00
H8 - H/Z-89/90 - H/Z-100 (Not PC)					
ADVENTURE	885-1222-[37]	CPM		GAME	10.00
BASIC-E	885-1215-[37]	CPM		LANGUAGE	20.00
CASSINO GAMES	885-1227-[37]	CPM		GAME	20.00
CHEAPCALC	885-1233-[37]	CPM		SPREADSHEET	20.00
CHECKOFF	885-8011-[37]	CPM		CHKBK 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 CAT UPDT #1	885-4501	VARIOUS		PROD 1983 TO 1985	9.75
KEYMAP CPM-80	885-1230-[37]	CPM		UTILITY	20.00
MBASIC PAYROLL	885-1218-[37]	CPM		BUSINESS	60.00
NAVPROGSEVEN	885-1219-[37]	CPM		FLIGHT UTILITY	20.00
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
X-REFERENCE UTIL FOR MBASIC	885-1231-[37]	CPM		UTILITY	20.00
ZTERM	885-3003-[37]	CPM		COMMUNICATIONS	20.00

	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 ZUG Product List for availability of bound volumes of past issues. Requests for magazines mailed to foreign countries should specify mailing method and include appropriate, additional cost.

Send Payment to: Zenith Users' Group
P.O. Box 217
Benton Harbor, MI 49023-0217
(616) 982-3463

Although it is a policy to check material placed in REMark for accuracy, ZUG 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 Zenith Data Systems Computer Centers.

ZUG is provided as a service to its members for the purpose of fostering the exchange of ideas to enhance their usage of Zenith Data Systems equipment. As such, little or no evaluation of the programs or products advertised in REMark, the Software Catalog, or other ZUG publications is performed by Zenith Data Systems, in general, and Zenith Users' Group, in particular. The prospective user is hereby put on notice that the programs may contain faults, the consequence of which Zenith Data Systems, in general, and ZUG, in particular, can not 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 Zenith Users' Group, St. Joseph, Michigan.

Copyright (c) 1991, Zenith Users' Group

Price List

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
H/Z-100 (Not PC) Only				
CARDCAT	885-3021-37	MSDOS	BUSINESS	20.00
CHEAPCALC	885-3006-37	MSDOS	UTILITY	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
DUNGN & DRAGONS (ZBASIC)	885-3009-37	MSDOS	GAME	20.00
ETCHDUMP	885-3005-37	MSDOS	UTILITY	20.00
EZPLOT II	885-3049-37	MSDOS	PRINTER PLOT UTIL	25.00
GAMES (ZBASIC)	885-3011-37	MSDOS	GAMES	20.00
GAMES CONTEST PACKAGE	885-3017-37	MSDOS	GAMES	25.00
GAMES PACKAGE II	885-3044-37	MSDOS	GAMES	25.00
GRAPHIC GAMES (ZBASIC)	885-3004-37	MSDOS	GAMES	20.00
GRAPHICS	885-3031-37	MSDOS	UTILITY	20.00
HELPSCREEN	885-3039-37	MSDOS	UTILITY	20.00
HUG BKGRD 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
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 PKG	885-3018-37	MSDOS	VARIOUS SPRDST	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
ZPC II	885-3037-37	MSDOS	PC EMULATOR	60.00
ZPC UPGRADE DISK	885-3042-37	MSDOS	UTILITY	20.00
H/Z-100 and PC Compatibles				
ADVENTURE	885-3016	MSDOS	GAME	10.00
BACKGRD PRINT SPOOLER	885-3029	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 II	885-3040	MSDOS	UTILITY	40.00
HEPCAT	885-3045	MSDOS	UTILITY	35.00
HUG EDITOR	885-3012	MSDOS	TEXT PROCESSOR	20.00
HUG MENU SYSTEM	885-3020	MSDOS	UTILITY	20.00
HUG SOFTWARE CAT UPD #1	885-4501	MSDOS	PROD 1983 - 1985	9.75
HUGMCP	885-3033	MSDOS	COMMUNICATION	40.00
ICT 8080 - 8088 TRANSLATOR	885-3024	MSDOS	UTILITY	20.00
MAGBASE	885-3050	VARIOUS	MAG DATABASE	25.00
MATT	885-8045	MSDOS	MATRIX UTILITY	20.00
MISCELLANEOUS UTILITIES	885-3025	MSDOS	UTILITIES	20.00
PS' PC & Z100 UTILITIES	885-3052	MSDOS	UTILITIES	20.00
REMARK VOL 8 ISSUES 84-95	885-4008	N/A	1987	25.00
REMARK VOL 9 ISSUES 96-107	885-4009	N/A	1988	25.00
REMARK VOL 10 ISSUES 108-119	885-4010	N/A	1989	25.00
REMARK VOL 11 ISSUES 120-131	885-4011	N/A	1990	25.00
SCREEN DUMP	885-3043	MSDOS	UTILITY	30.00
UTILITIES II	885-3014	MSDOS	UTILITY	20.00
Z100 WORDSTAR CONNECTION	885-3047	MSDOS	UTILITY	20.00
PC Compatibles				
CARDCAT	885-6006	MSDOS	CAT SYSTEM	20.00
CHEAPCALC	885-6004	MSDOS	SPREADSHEET	20.00
CLAVIER	885-6016	MSDOS	ENTERTAINMENT	20.00
CP/EMULATOR II & ZEMULATOR	885-6002	MSDOS	CPM & Z100 EMUL	20.00
DUNGEONS & DRAGONS	885-6007	MSDOS	GAME	20.00
EZPLOT II	885-6013	MSDOS	PRINTER PLOT UTIL	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
LAPTOP UTILITIES	885-6014	MSDOS	UTILITIES	20.00
PS' PC UTILITIES	885-6011	MSDOS	UTILITIES	20.00
POWERING UP	885-4604	N/A	GUIDE TO USING PCs	12.00
SCREEN SAVER PLUS	885-6009	MSDOS	UTILITIES	20.00
SKYVIEWS	885-6005	MSDOS	ASTRONOMY UTIL	20.00
TCSPELL	885-8044	MSDOS	SPELLING CHECKER	20.00
ULTRA RTTY	885-6012	MSDOS	AMATEUR RADIO	20.00
YAUD (YET ANOTHER UTIL DSK)	885-6015	MSDOS	UTILITIES	20.00

This Software Price List contains all products available for sale. For a detailed abstract of these products, refer to the Software Catalog, Software Catalog Update #1, or previous issues of REMark.

Now Available!

ZUG software is now available on 2" disks. Just put a "-90" at the end of the part number (i.e., 885-6014-90). Also add \$3.00 to the purchase price of the software (i.e., \$20.00 + \$3.00 = \$23.00).

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

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

ORDERING INFORMATION

For VISA, MasterCard, and American Express phone orders, telephone the Zenith Users' Group directly at (616) 982-3463. Have the part number(s), description(s), 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: Zenith Users' Group, P.O. Box 217, Benton Harbor, MI 49023-0217. VISA, MasterCard and American Express require minimum \$10.00 order. No C.O.D.s accepted.

Questions regarding your subscription? Call Lisa Cobb at (616) 982-3463.



...the other cats get to sing along! And now, HEPCAT does Windows!



up over Windows™ 3, even when it is running in the standard (286) or 386 enhanced modes.

What Is HEPCAT?

HEPCAT (Handy Engineer's and Programmer's Calculation Tool) is a floating point calculator with several scientific/engineering features built in, and a binary (programmer's) calculator combined into one tiny, powerful program. HEPCAT is a memory resident program that "pops up" on your screen whenever you activate it by typing a special "hot key" sequence.

The Other Cats Can Sing Along

Unlike other pop-up calculators, HEPCAT is concurrent. That means that when

you pop it up over a running program, the program can continue to run. For example, if you pop it up while Lotus™ is busy loading a huge spread sheet, it will continue loading while you perform your calculations. And HEPCAT always pops up in the current video mode, rather than forcing the screen into a text mode like other pop-ups do. HEPCAT can pop up in any standard CGA, EGA, VGA, or Hercules™ graphic mode, as well as in any text mode. It can even pop up in some non-standard graphic modes (but it may not clear its window when you exit).

HEPCAT Works Harder

The floating point calculator in HEPCAT includes the following built-in functions: powers, pi, factorial, square root, sine, arc sine, cosine, arc cosine, tangent, arc tangent, log (natural and base 10), e^X and 10^X , and it does rectangular-to-polar and polar-to-rectangular coordinate conversion. It also includes several built-in US-metric and metric-US conversions. The binary calculator works in these number bases: binary, tetral (base 4), octal, split octal, decimal, and hexadecimal; and it supports these operations: MOD, AND, OR, XOR,

SHL, SHR.

The HEPCAT floating point calculator supports 8 significant digits and can display numbers four ways: floating point, fixed point, scientific notation, and engineering notation. Numbers are handled internally in BCD format to eliminate binary round off errors in addition and subtraction.

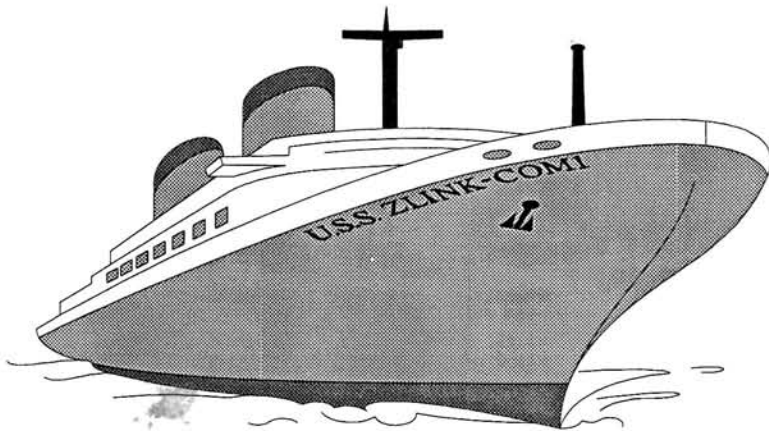
HEPCAT Eats Less

HEPCAT uses less than 18k of memory — less than any other pop-up calculator that we know of. It also uses less than 14k of disk space, so you don't have to worry about where to put it on a small system. HEPCAT is easier to learn, too, with commands that make sense.

If you are tired of pop-up calculators that can only sing solo, or calculators that can do DOS but not Windows (or Windows but not DOS) give HEPCAT a try. HEPCAT is available from ZUG as part no. 885-3045 for \$35.00 (plus S/H). It works on any Zenith Data Systems computer that runs MS-DOS or Z-DOS (including the Z-100 series), and on most PC-compatibles. If you have HEPCAT version 1, send in your original distribution disk and \$10 to upgrade to version 2. ✻

Port-O-Call:

COM1



Laura White
759 Polfus
Benton Harbor, MI 49022

One of the latest laptop lines put out by Zenith Data Systems is the ZL-1 and ZL-2. These laptops, known as notebooks, have 1Meg and 2Meg of RAM, respectively. This portable comes with an 8 bit 4.77/8 MHz 80C88 processor. The minisport also has an internal 2" floppy disk drive. In addition, the ZL-1 and ZL-2 both come with a 9-pin serial port, a parallel printer port, a 9-pin CGA color monitor port and, a 20-pin mini connector (called a HONDA connector) for an external floppy drive. They also have a slot where you can slide in a modem card that is used in this computer line only and is not the same as ones used in other ZDS portables. The MinisPort comes with DOS 3.3+ which is loaded onto a ROM (Read Only Memory) chip. Even though the computer appears to boot to a C:> prompt, the C:> drive cannot be used for storing any data. In addition to MS-DOS, the ROM includes a copy of Fastwire LX. This is a self-cloning file transfer program can be copied to another computer using either a 7-pin or a 3-wire NULL serial cable. The additional memory beyond the 640K MS-DOS environment can be used as a battery backed-up RAM (Random Access Memory) disk for temporarily storing files and/or programs during use. It is not recommended that you leave any valuable information on this RAM drive, because if the batteries go dead, the information will be lost! (more on this later). The MinisPorts also have two lithium cells to protect this memory while the main battery is being changed or, when it's lost its charge, but they are only designed to protect this memory for a very short duration.

There are however, some complications which have occurred in the early releases of the MinisPort. One prevalent problem was that when people were powering up their computer, scrambled messages were appearing, and the fix was to run through the SETUP program again. The reason for this scrambled message was that the lithium back-up batteries weren't charged enough. If replaced on a regular basis, the machine runs fine and there is no need to run through the SETUP program again.

Huggies found that there were power up problems. A number of people called into the bulletin board to inform others that they fixed their problem by, what else?? That's right—take it apart! By getting into their laptops, some Huggies found that there was corrosion on several connections and chips. They recommended giving the inside a thorough cleaning with steel wool or, by using extreme caution, you could also clean the connections with freon TF and a tooth brush. One huggie also found that when he took his computer apart, he had a loose connector on the main board that was giving him only a partial connection.

Yet another approach to some of their problems was to actually read the manual. It was found that, as recommended, if you hold the power button in for about two seconds, the power up problem was solved.

The COM1 Message Center was also filled with questions and comments on installing an external drive, or possibly, a hard drive. Installing a hard drive has already been taken care of by ZDS with the introduction of the MinisPort HD. Unfortunately, neither the ZL-1 or ZL-2 can be

upgraded with an internal HD; however, installing an external floppy drive is an easy task. As was mentioned in the beginning of this article, the MinisPort has an external floppy port already built-in. What huggies were questioning was not how to hook it up, but which cable was necessary for installation. As it turns out, because the MinisPort has a Honda connector, the cable needed is Zenith Data Systems model number HCA-77.

A few huggies actually made their own cable. The specifics for this cable were available before the bulletin board lost all its messages, but if anyone is interested in this information, just let me know by either sending me a message on COM1 or, by mail (My address is at the beginning of this article). If enough people are interested in the details, I will include it in a future article.

Yet another area of confusion for huggies seems to be the D: drive. The D: drive uses the memory beyond the 640k used by MS-DOS, up to 1 or 2 megabytes of memory—enough to hold a program like MS-Word or Wordperfect. You must remember that this RAM drive is a far cry less than what most huggies are used to. You also need to know that the D drive must be formatted before you can access it. By using `FORMAT D: /S`, you will instruct the computer to read D: as a system drive. Expanding on this, Pat Swayne added, "then you alter SETUP to have the system boot from it." If you create an `AUTO-EXEC.BAT` or a `CONFIG.SYS` file on the D: drive, the MinisPort is smart enough to look there for them no matter which drive it boots from.

Continued on Page 8

QUIKDATA - 15 YEARS OF H/Z SUPPORT!

YOUR H/Z RAM & DRIVE EXPERTS!

ACCELERATE YOUR PC!

From Sota Technologies, Inc., the fastest and most proven way to breath new life in your Heath/Zenith PC/XT computer, giving it -AT compatible speeds! Turn your turtle into a -286 rabbit with a 12.5 Mhz 80286 accelerator board. Complete with 16K on-board CACHE.

There are other ways to speed up your H/Z PC/XT computer, but you spend too much and get too little. Our EXP12 **286i** is the effective solution, making your H/Z150/160/150/158/159 series of computers, or any general PC/XT computer faster, in many cases, than a standard IBM AT type computer! Simple half-card plug-in installation with switchable speed control. **You won't believe your stop watch!**

EXP-12 - \$295

EXP386 - \$469 Much faster 16Mhz 80386 SX version

MEMORY UPGRADES

Note: All memory upgrades come without memory chips. 150ns 256K DRAMs are \$1.79 as of this printing.

Z150MP - \$17 Will allow you to **upgrade your H/Z150/160 to up to 704K** on the main memory board, using up to 18 256K DRAM chips.

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

EAZYRAM - \$89 Upgrades EaZy PC from 512 to 640K

ZMF100 - \$55 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 - \$55 Similar to ZMF100 above, but for new motherboards with p/n 181-4918 or greater.

Z159 EMS LOGIC UPGRADE includes the PAL chip, logic chips and one bank of 256K (9) chips. Up to two banks (18) can be installed for a total system RAM of 1.2 meg on the Z159 main board.
Z315-1 - \$79!!

3MB RAM BOARD for Z241/248 computers is an excellent memory card. Will backfill your 512 to 640K, and provide both extended and expanded RAM, all can coexist. Uses 100ns M256-15 RAM chips, 36 per megabyte desired. Minimum of 18 DRAM chips required.
EVATRD - \$119

Z248/12, Z286LP RAM UPGRADE Z605-1 consists of 2MB SIMM 80ns RAM kits to upgrade your H/Z systems.
Z605-1 - \$169

Z386/20, Z386/25, Z386/33, Z386 EISA 2MB SIMM 80ns UPGRADE to add increments of 2MB to these systems. Two required.
ZA3600ME - \$95

ZA3800MK - \$389 4 megabyte SIMM upgrade for Z386/20, /25, /33, /33E. Must have 4-1 meg SIMMs installed first.

WINCHESTER UPGRADE KITS

PCW20 - \$249 Complete MFM winchester setup for a H/Z150, 148, 158, 159, 160, PC etc. Includes **21 meg** formatted half-height Segate ST-124 37ms drive, controller, cable set, doc.

PCW30 - \$295 32 meg with 28ms Segate ST-138-1.

PCW40 - \$329 42 meg with 28ms Segate ST-251-1.

PCW80 - \$595 80-meg with Segate ST-4096 full size drive.

We also have the DTC controllers (\$59) and daughter board expansions (\$65) to place a hard drive in the **H/Z148** computers.

BARE WINCHESTER DRIVES/CONTROLLERS

ST-124 - \$209 21 meg 37ms 3.5" drive in 5" frame

ST-138 - \$249 32 meg 38ms autopark 3.5" drive in 5" frame

ST-251-1 - \$289 42 meg 28ms autopark HH drive

ST-4096 - \$549 80 meg 28ms autopark FH drive

WDCON - \$59 PC/XT hard drive controller board

WDATCONF - \$129 1:1 interleave HD/floppy controller for AT's

FLOPPY DRIVES

T28M - \$159 2.8MB 3.5" floppy

MF501 - \$71 5" 360K DS/DD drive

MF504 - \$81 96 TPI 1.2 meg AT/Z100 drive

MF353 - \$79 720K 3.5" drive in 5" frame

MF355 - \$89 1.4 meg 3.5" AT drive in 5" frame

SIEM-R - \$39 40 track SS refurb (H8/H89 type)

TM100-4R - \$75 80 trk DS refurb (H8/H89 type)

TM100-2R - \$69 40 trk DS refurb (H8/H89 PC type)

ANY DRIVE IN YOUR PC/XT/AT

With the CompatiCard, you can install up to four additional drives, of any type in your PC/XT/AT computer. Add a 1.2 meg 5" floppy, or a 1.44 meg 3.5" floppy, or any other drive, including 8" to your system. The CompatiCard (CCARD) will handle up to four drives, and the CompatiCard II (CCARD2) will handle up to 2 drives. CCARD4 has boot ROM to allow it to be used as primary boot controller in systems that allow you to remove floppy controller in some systems. Also handles 2.8MB 3.5" floppy drives. Additional cables and external enclosures may be required.

CCARD2 - \$79 **CCARD** - \$99 **CCARD4** - \$125

ADD ANY FLOPPY TO ANY LAPTOP OR PC WITH A PARALLEL PORT

easily and inexpensively. With Backpack, you simply connect the external unit to your parallel printer port (do not lose printer function), install software, and away you go! No expansion boxes needed for laptops, and no slots required. Too easy to be true, but it is. Want a 2.8mb/1.4mb/720K floppy on your MinisPort? Want to add a 1.2 meg to your laptop? Want to add an additional drive to your desktop? Plug it in and go. 2.8MB 3.5" version will read and write 2.8 megabyte format, 1.4 meg format and 720K format. 1.2 meg version will read 1.2 meg and 360K floppies, and write 1.2 meg format.

BPACK2.8 - \$319 2.8 Mb version

BPACK1.4 - \$269 1.4Meg/720K version

BPACK1.2 - \$269 1.2/360K version

BPACK360 - \$269 360K only version

8-BIT/Z100

We carry a full line of replacement boards, parts and power supplies for the H/Z89/90 and Z100. We also have some H8 boards available. We continue to fully support and carry a full line of hardware and software products for the H8/H89/90 and Z100 computers including almost the full line of Software Toolworks items. Other items we carry include diskettes of all types, printers, modems, hard drives, etc.

OTHER STUFF

Quikdata also carries BIOS ROM upgrades for most H/Z PC/XT/AT computers, spike protection filters, backup power supplies, tape backup units, modems, printers, cables and ribbons, disk drives and diskettes of all types, external hard drive and floppy drive enclosures, cables and connectors, laptop batteries, CMOS batteries, video monitors and video cards, memory cards, memory chips and ICs, joysticks, accessory cards, serial and bus mouse, a variety of useful and most popular software and much more! **Need a PC/XT/AT computer?** Tell us what you want and we will quote you a price on one of our custom assembled QD computers made up to your exact specifications! Price is lower than you think for a high quality system.

Call or write in to place your order, inquire about any products, or request our **new** 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. FAX: (414) 452-4344.

QUIKDATA, INC.

2618 PENN CIRCLE
SHEBOYGAN, WI 53081-4250
(414) 452-4172

Do You Want To Write Another Program?

Part II

Gil Hoellerich
2617 Country Way
Fayetteville, AR 72703

If you read "Do you want to program?" in a previous issue and want to prepare more sophisticated programs, here is your chance.

Since I have a rabbit-tail memory (short and fuzzy), I always like to review. First, I would like to remember that the main task of the previous article was to print a label with your name, street address, city, state, and zip.

While this illustrated well how a program could be prepared (coded) from prescribed tasks, the program was not very practical. As written, for each different label we want to prepare, we would need a different program. For each copy of the same label, the program would have to be run again. In this article, we will select our tasks to make this program more practical.

Also, the previous program illustrated only one function that a program could do: output. In this article, we will see that the computer can also accept input from the keyboard, make some decisions, and repeat the same function as often as the user wishes.

At this point, we also need to discuss the syntax of the BASIC programming language. The English language requires a noun-verb syntax. BASIC programming language also has a syntax; a keyword and optional argument or parameter. We saw two English-like keywords in the previous article: PRINT and LPRINT. We learned that without any argument or parameter this produced a blank line either on screen (PRINT) or on paper (LPRINT).

In most cases, we need more than the keyword, we need an argument. With these two keywords (PRINT and LPRINT), the argument tells what to place on the paper in the printer or to display on the screen and sometimes where; an argument of characters (letters, words, or sentences) in quotes following these keywords would be displayed or printed. Example: PRINT "Gil Hoellerich". In this article, you will use another argument.

We will modify the original program so that your label can be printed as many times as you wish. Next, we will modify the program so that the user can input the name, street address, city, state and zip of the label to be printed.

Since we plan to allow the user to input data to the program from the keyboard, we need to learn the appropriate keyword. Helpfully, the keyword for input is INPUT. The data received by the program from the keyboard must be stored for later use.

To do this, we use a variable. For example, the command

```
10 Input N
would wait for an numeric input and store it in the variable N. Notice the new keyword INPUT is very English-like and very descriptive of the action since this keyword will allow the user to enter a value. The command
20 Print N
would then display on the screen the value stored in the variable N, which is the value that the user inputs. This is an example of a different argument: a variable.
```

If you would like to see this simple program operate, enter the program:

```
5 CLS
10 INPUT N
20 PRINT N
30 END
```

When the program is ready for you to enter the value of N, a question mark will appear on the screen and the computer will stop!

So the user knows what to expect, we could add

```
7 Print "Please enter a number"
However, we can combine lines 7 and 10 into
```

```
10 INPUT "Please enter a number"; N
Now we understand the syntax for receiving input from the keyboard, storing that input in a variable and using the value in that variable. Let's return to modifying the program to print multiple copies of the same label.
```

For this modification, our tasks could be:

1. Inform user about purpose of program.

2. Inform user to enter a number.
3. Use that number to repeat the function.
4. Inform the user that labels are being printed.

Remember the code which we used in the previous article?

```
10 LPRINT "your name"
15 LPRINT
20 LPRINT "your street address"
25 LPRINT
30 LPRINT "you city, state, zip"
```

To do task 1 above, we can add:

```
10 PRINT "This program will print several
copies of your label for you."
20 PRINT "You must enter a number at the
appropriate time."
```

Before we do, let's re-number the original lines 10 to 30 so they are now lines 70 to 110. So our modified program is:

```
10 PRINT "This program will print several
copies of your label for you."
20 PRINT "You must enter a number at the
appropriate time."
70 LPRINT "your name"
80 LPRINT
90 LPRINT "your street address"
100 LPRINT
110 LPRINT "you city, state, zip"
```

To accomplish task 2, we need to obtain the number of times the user wishes the label to be printed. We could modify our original program by inserting the command:

```
30 INPUT "How many times should I print
your label";N
```

Now the number of times to print the label is in storage in the variable N; in the short program above, we decided to display the number in storage on the screen. Since task 3 requires that we will use this number to repeat the function of printing the label. To do this, we will use a loop.

One type of loop in BASIC is the FOR NEXT loop; we will use that here. Our commands for this would be

```
60 For I = 1 to N
130 Next I
```

For some beginners, comparing the FOR NEXT loop construction to a sandwich is helpful. Consider lines 60 and 130 the bread; just as in making a sandwich, there is some variety, but not much! The contents of the sandwich can vary much more.

Just as in almost all cases, the bread must be around the contents, the command(s) that we wish to repeat must be placed between line 60 and 130. Lines 70 through 110 are the lines which we wish to repeat. To provide some space between the labels, let's add line 120 which prints 3 blank lines.

To further enhance this program we have added a command to clear the screen in line 40 and a command in line 50 to tell the user the labels are being printed. We will also add line 140 to tell the computer when it has reached the end of your program; this will become very important later.

Our modified program is:

```
10 PRINT "This program will print several
copies of your label for you."
20 PRINT "You must enter a number at the
appropriate time."
30 INPUT "How many times do you wish to
print your label";N
40 cls
50 PRINT "Please wait your labels are
being printed"
60 For I = 1 to N
70 LPRINT "your name"
80 LPRINT
90 LPRINT "your street address"
100 LPRINT
110 LPRINT "your city, state, zip"
120 LPRINT:LPRINT:LPRINT
130 NEXT I
140 END
```

For a bit of review, we have now learned:

1. The syntax of BASIC commands.
2. How to get data from the keyboard.
3. How to store this in memory until we need to use it.
4. How to use it.
5. How to repeat a function.
6. How to make the program more user-friendly to the user.

Now suppose we are preparing to send Christmas cards and want to prepare a simple program to print the labels for the

envelopes. For each label we will want a different name, street address, and city address. We will get this from the keyboard as we print the labels.

Our tasks for this program would be:

1. Display message on screen to indicate program.
2. Instruct operator to enter label name.
3. Instruct operator to enter label street address.
4. Instruct operator to enter label city, state and zip.
5. Clear screen.
6. Tell user that a label is being printed.
7. Have printer print label.
8. Check if more labels are to be printed.
9. If not, end program.
10. If so, go to step 2.

Let's start with task 1:

```
10 PRINT "This program prints labels. You
will be asked"
20 PRINT "to enter the necessary informa-
tion. This will be"
30 PRINT "the name, street address, city
and when to stop."
```

For tasks 2, 3, and 4 we will use statements (lines 40, 50, 60) similar to the one used in line 30 of the previous program. However, since we want to store characters instead of numbers, we need to add a \$ at the end of the variable name. Also since we want to input a comma between city and state, we use the keyword LINE INPUT instead of INPUT in that line.

```
40 INPUT "Please enter name"; THE.NAME$
50 INPUT "Please enter street address";
STR.ADDR$
60 LINE INPUT "Please enter city, state,
zip ";CITY.STATE.ZIP$
```

For tasks 5 and 6 we will enter:

```
70 CLS
80 PRINT "Please wait, your labels will be
printed"
```

For task 7:

```
90 LPRINT THE.NAME$
```

```
100 LPRINT STR.ADDR$
110 LPRINT CITY.STATE.ZIP$
120 LPRINT:LPRINT:LPRINT
```

For tasks 8, 9, and 10, we will enter lines 130 and 140

```
130 INPUT "do you wish to print more
labels. Enter Y or N";ANS$
140 IF ANS$="Y" OR ANS$="y" THEN 40
150 SYSTEM
160 END
```

First, we need to have the user tell the program if they wish to continue. This is done with line 130, which we have done before. The user will input either a Y or N; this will be stored in the variable ANS\$.

Previously, we either printed the contents of the variable or used it in a FOR-NEXT loop. This time, we will match to predetermined values (Y or y) in an IF-THEN-ELSE structure. If there is a match, then action will be taken. In this case, the program loops back to line 40 and the user is allowed to print another label; if there is no match, the program executes line 150. Instead of just stopping the program, we will send the user back to the operating system prompt by the keyword SYSTEM in line 150.

Perhaps you noticed that the program stops even if an N was not entered. Suppose you want to make sure that N or n was entered to stop. You have the necessary background to do this. TRY IT!

Hint: Your tasks would be to first compare the character enter with N or n after the Y or y comparison. If there is no match, give an error message. If there is a match, return user to the operating system.

One more step can be taken to make this program more practical. That is to have the Christmas card list from the previous year stored on a disk. Then you would not need to enter them each time from the keyboard. I will cover this in my next article. ✨

Continued from Page 5

Finally, there seems to be a problem with locating the small 2" disks necessary for the MinisPort. Following are two possible sources for the little disks:

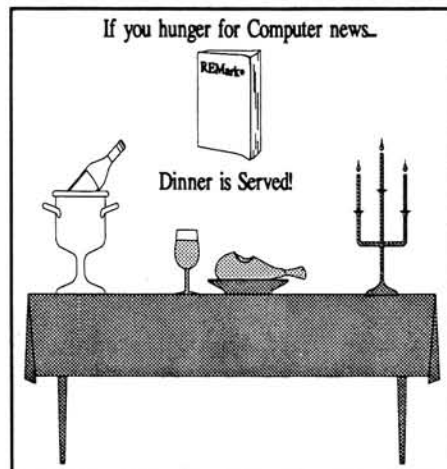
Fry's Electronics
California
(415) 770-3797
\$60.00/Box

Greenleaf Computer Products
19 Cobb Parkway, Suite C-4
Marietta, GA 30062
\$67.50/Box

That just about does it for this month's valuable information. I would, however, like to take a moment to point out a few things here. First of all, the MinisPort is quite an impressive piece of equipment. As you will notice, the "problems" cited in this month's article are really not serious. A number of these things are covered in the manual; however, most huggies who had questions received their computers from the Bargain Centre minus the manuals. The MinisPort line is quite impressive, espe-

cially the MinisPort HD. It is the same size as the other MinisPorts, but comes equipped with a 20MB internal hard drive.

As usual, I would like to thank this month's contributors, Ryan Hansen, Andy Hass, Scott Hellman, Pat Swayne, Dave Wenzel and Wally Wallace for their helpful hints from COM1. ✨



On the Leading Edge

William M. Adney

D. O. Box 531655

Grand Prairie, TX 75053-1655

Copyright (©) 1991 by William M. Adney. All rights reserved.

Physical Memory, Expanded versus Extended Memory, Memory Managers, QRAM, Manifest, QEMM386, DESQview, DESQview386

There still seem to be a number of questions related to expanded and extended memory based on the letters I receive, so we will take a look at that in this article. Most everyone seems to understand conventional memory in the usual 640 K DOS environment, but these two additional types of memory still seem to cause problems, so we will take another look at both. For those who need an introduction to the three types of memory, you can refer to Chapter 10 (Adding More Memory to Your Computer) in the original *Powering Up* book.

Adding Physical Memory

Adding physical memory to a computer is usually the least of the problem, but it still can get complicated. For example, there are three different ways to buy memory "chips": DIPs, SIPs, and SIMMs. **DIP**, or Dual Inline Pin, memory chips are the way that early computer users added memory. These are small units, typically about an inch long and 3/8" wide with two rows of pins (that's why it's called a dual inline pin) that fit into a corresponding socket on a memory board. Each DIP memory chip is has a capacity rated in kiloBITS, and typical capacity might be 64 Kb (notice the small **b** for **bit**) or 256 Kb. And since a byte consists of 8 bits, a minimum of 8 DIPs are required to add memory to our computers. Most of today's computers also have a parity bit (used to verify memory accuracy) which means that they generally have 9 DIPs in each bank of memory.

A **BANK** of memory, consisting of either 8 or 9 DIP memory chips, is rated in kiloBYTES. For example, one can add 64 KB (notice the capital **B** for **BYTE**) with a bank of 64 Kb chips or 256 KB with a bank of 256 Kb chips.

SIMM, or Single Inline Memory Module, memory is generally the standard for current desktop computers. Because this is a module, it includes either 8 or 9 "chips" on a very small printed circuit board that is typically about four inches long and

about an inch and a half wide. Like a printed circuit board, it has an edge connector that plugs into a special SIMM socket, just like any add-on board, except that SIMM sockets are only used for memory of course. If you need to buy SIMMs, you will find that they are rated in terms like: 1 MB x 8 or 1 MB x 9, which means that the SIMM has a capacity of 1 megabyte and is the equivalent of 8 or 9 bits respectively. Most current computers use the parity bit, so you will probably need something like a 1 MB x 9 SIMM for your computer if it uses SIMMs.

SIP, or Single Inline Pin, memory only has a single row of pins. It has not been especially popular, and because I don't know of any Zenith Data Systems (or Heath) computers that use it, we will ignore it here.

For computers that do not use SIMMs, there are several ways that memory can be added. Most current ZDS laptops (and most other laptops for that matter), for example, use special memory modules that are unique to each computer, and one must buy the appropriate memory module for a specific laptop. Some computers, such as the eaZy PC, have a very special memory module that can add enough memory to provide 640 KB of conventional memory, but neither expanded nor extended memory can be added. Other computers, like the Z-171 portable, have enough sockets to provide 640 KB of conventional memory, but they also do not allow any extended or expanded memory.

For older systems, such as the Z-241, Z-248, and Z-386/16, there are two ways to add memory. One way is to buy the appropriate ZDS memory board, such as the Z-505 or Z-515 (if you can still find one) for the Z-386/16. Another way is to buy a third party memory board, but that has the disadvantage that anything other than a ZDS memory board will slow memory access down because wait states are automatically added for non-ZDS memory products. For the Z-241 and Z-248 sys-

tems, the ZDS memory boards had the disadvantage that they were only configured for extended memory, and we will look at one way to "install" expanded memory using those boards later in this article.

One other point about adding physical memory with DIPs, SIPs or SIMMs. These memory units are also rated in terms of speed, and typical values range from "slow" 200 ns (nanosecond) chips to "fast" 70 ns units. Many of today's fast computers use SIMMs that are rated at least 80 ns. For additional information about memory, you might again want to refer to Chapter 10 in the original *Powering Up* book.

And last, but not least, an editorial comment on price. Prices for Zenith Data Systems memory products, including SIMMs, are simply outrageous and are not competitive as I have mentioned many times in this column. For example, you should be able to buy memory at a maximum cost of about \$100 per megabyte, so that you could reasonably expect a 4 MB SIMM to cost about \$400 or less. Other memory expansion products, such as those for laptops, also cost less from non-ZDS sources, and I recommend you check with *REMark* advertisers (e.g., Payload Computer) to find out current prices.

Configuring Memory

Once the physical memory is installed in a computer, it may need to be configured in one way or another. For example, one must set an appropriate switch on a memory board for the Z-386/16 to define whether memory is to be expanded or extended. Newer computers, such as the SupersPort 386SX laptop or Z-386/33, use the ROM-based SETUP command to specify whether expanded or extended memory is defined. If you have questions about how to configure additional memory for your system, you should check your Owner's Manual for your specific computer.

There are always at least two steps in adding memory. First, you must add the

physical memory. Second, you must usually configure that memory by using either a switch or the SETUP command, depending on what model of computer you have. But, if you want to configure the system to use expanded memory, a third step is required. You must install an expanded memory manager, such as EMM.SYS, in the CONFIG.SYS file using the DEVICE= command. And if you expect to use expanded memory with any other device driver (e.g., VDISK.SYS), remember that the DEVICE= command line containing expanded memory manager must precede a device driver that uses expanded memory.

In summary, the addition of physical memory always requires two steps: adding the memory and configuring it. That is all you need to install extended memory. But if you want to use expanded memory, you must add the third step of installing the appropriate expanded memory device driver. Now let's take a look at the differences between expanded and extended memory.

Extended Memory

Although extended memory is not nearly so popular, it is somewhat easier to talk about because it has limited usage and applications. First, extended memory only can be installed on an 80286-based system or later (e.g., 80386 and 80486). That's because the 8088 CPU (Central Processing Unit) only had an address space of one megabyte, and extended memory is that memory which "extends" beyond one megabyte capability of the 8088. Another problem was that extended memory could generally be used when the CPU was in the protected mode, which is a special mode ONLY available on 80286 and later CPUs. The 8088 CPU does not have protected mode. Another technical problem was that most hardware and software manufacturers had some difficulty in gracefully switching into and out of protected mode, but that problem has been mostly solved as experience was gained.

You may hear the term XMS, which is short for eXtended Memory Specification. This is a technical definition of extended memory and how it can be used by application programs.

Extended memory generally has limited usage by itself in today's computers. Few applications can use extended memory, but a couple of device drivers usually provided in DOS can use it. VDISK.SYS and ZCACHE.SYS can use either extended or expanded memory. Extended memory IS used (and in fact required) by some operating systems such as OS/2, but that is generally the only real use of extended memory.

However, there is one other way to use extended memory that may be of special interest to Z-241/248 owners. One can use a software program, such as

Quarterdeck's QRAM, to "convert" extended memory into expanded memory. Given the limited usefulness of extended memory that was provided on the ZDS memory products for those two systems, this is an important enhancement that will be discussed later in this article. Quarterdeck's QEMM386 performs a similar function for 80386-based systems, and it is particularly useful especially if you only have the standard one megabyte of memory available. We'll also take a look at QEMM386 later in this article.

As of now, extended memory has not been particularly popular for use in an application. I believe that is mostly because it cannot be used at all with an older 8088-based system, and it does not have all that many uses on any of the later systems either. Although extended memory is required for the OS/2 operating system, that is one of the few uses for which it is required. As I have mentioned before, the ZDS MS-DOS User's Reference for version 4.0 (and some previous versions too) suggests that there may be a problem when VDISK.SYS is defined in extended memory (page D.24). If a program uses a number of frequent interrupts, some interrupts may be "lost" which may result in data loss. On the occasions that I need to use a VDISK, I always use expanded memory because of that information, although the manual suggests that you may be able to adjust some of the VDISK parameters to eliminate the problem.

All things considered, extended memory is probably a better overall solution to additional memory requirements for technical reasons, but its implementation has not fully matured to the point it is generally acceptable. Besides, most software manufacturers would rather develop a program that will run on nearly all systems, including the 8088, rather than developing two versions: one for 8088-based systems with expanded memory and one for 80286 or later systems with extended memory. And since expanded memory can be added to nearly all systems, it ends up saving money for both manufacturers (in development cost) AND users. Not a bad trade-off.

Expanded Memory

Expanded memory, commonly called EMS (Expanded Memory Specification), was originally developed in 1985 by Lotus, Intel, and Microsoft (LIM). You may see it identified as LIM EMS because of the companies involved in the development of the standard. Although EMS was really developed to provide a scheme to add memory to the relatively limited 8088 CPU, it has been enhanced, changed, and modified to provide additional capabilities, especially for systems with newer CPUs. As a result, there have been several different versions of EMS, and software is generally designed to take advantage of the features in at least

one of those versions. Some of the newest software may require EMS 4.0, so you need to check each of the application program manuals you have to verify what EMS version you need. For example, the expanded memory manager EMM.SYS is included in ZDS MS-DOS 4.0, and the manual clearly states that this version supports EMS 4.0. As I have mentioned in other articles, this device driver can be used with any ZDS computer which uses SIMMs for memory expansion (e.g., Z-386/25 or Z-386/33); otherwise you must use a ZDS memory board such as the Z-505 or Z-515 for the Z-386/16. I will also note that EMM.SYS will NOT work with the ZDS memory boards for the Z-241/248 because those units only supported extended memory and cannot be hardware configured for expanded memory.

Expanded memory is quite popular, and many application programs can use it if it is available. That includes virtually all spreadsheets and even some word processors, not to mention all kinds of other utilities and applications. But there is one question that always seems to arise. That is, why should you spend the money to add any memory to your system in the first place? This question was specifically discussed in Chapter 10 of the original *Powering Up* book, but it is worthwhile to revisit the subject because I continue to get a lot of questions about it.

Adding Memory and Other Stuff

One of the most common questions I receive goes something like: "I have an eaZy PC with 640 K of memory, and I need more memory. How can I add memory to my computer?" The most direct answer is: "You can't." But there is more to it than that. Perhaps the most important point is that nearly all of these kinds of letters fail to mention any particular reason for adding memory. Some letters include a statement that "I need more memory to make my system run faster" or something similar to that. Unfortunately, more memory will not necessarily make a system run faster, and that's a fact. To understand why, let's take a look at a user who bought an eaZy PC.

Despite its limitations, the eaZy PC is really a good computer for many users since they don't really need a lot of the power of more expensive systems. For example, how much memory do you really need to write a one-page memo or even a 10-page term paper? How much memory do you need for a two-page spreadsheet? How much memory do you need for a data base, even dBase IV? And how much memory do you need to run utilities or games? The answer to all of these questions is that a maximum of 640 K MAY be needed, and in many cases, the eaZy PC's standard 512 K is quite sufficient for many users. And although Win-

dows 3.0 will not run effectively on an eaZy PC (because you cannot add additional memory), the eaZy PC would not really be a very good system for Windows anyway because it is far too slow (So is my Z-386/16!).

The whole point is that the eaZy PC and many other 8088-based systems are quite adequate to perform many common tasks such as those noted above. And for a lot of users, that is really all that is necessary. Why spend thousands of dollars when all you need to be able to do is work with small spreadsheets and perform small word processing tasks? More importantly, it is significant to note that adding expanded memory will probably not help improve system performance because the eaZy PC still uses an 8088-compatible CPU. And that is the whole point.

As I've said many times, be sure you know WHY you think you need more memory (or any other accessory for that matter) before you even begin to check on the feasibility and cost. Don't be misled by the assumption that more of something in a computer (e.g., memory, clock speed, CPU, hard disk space, etc.) is better. More of something also costs more, and it may not improve the performance or usability of your system much, if at all. And that goes for any computer, not just an eaZy PC. If, for example, you use a word processor to write documents that are generally less than 10 kilobytes long, an 80386-based system with five megabytes of memory and a 330 MB hard drive is considerable overkill. Sure it runs faster, but it generally has a faster hard drive in addition to a more powerful CPU. That kind of performance costs considerably more, much more than the actual value of the performance improvement when judged on cost-benefit criteria. It may cost four or more times as much to get a 10% performance improvement in a common task, and by any measure, that is not a very good investment.

Although it is not related to adding memory, I did receive a letter from one eaZy PC owner who wanted to add VGA to the system. Of course that is not possible, but it struck me as somewhat odd that he would want to spend nearly twice as much money for a display system (video card and monitor) as the entire computer system originally cost. Sure, it looks nice, but that is kind of like having a 4-cylinder engine in a Cadillac limousine—it may look nice, but it is REALLY slow and the overall performance is terrible. But enough of additions to memory and other hardware; let's take a look at some software that you may find helpful.

Memory Managers

How useful any memory manager is depends on exactly what kind of system you have and how much memory it has. For example, the EMM.SYS memory man-

ager that is included with MS-DOS can be used on a system that has 1 MB of installed memory to provide access to a modest amount of expanded memory. Unfortunately, EMM.SYS only works with appropriate Zenith Data Systems expanded memory hardware (or SIMMs on late model computers), so this memory manager cannot be used on older systems like the Z-241 and Z-248. Fortunately, there is a "cure" for that, and I will mention a software package you can use on those Z-200 computers to convert the memory on the extended memory boards (i.e., the Z-405 and Z-415) to expanded memory.

If you have an 80386 computer, MS-DOS 4.0 includes a memory manager that can "convert" extended memory to expanded memory. Although this memory manager -EMM386.SYS- is included in DOS, it will not work with some programs, such as Windows 3.0, that run in the protected mode. In general, a program that runs in the protected mode basically supports multitasking, so EMM386.SYS is not any particular help for protected-mode programs like Windows.

HIMEM.SYS is another memory manager that is included with MS-DOS 4.0. It provides support for the eXtended Memory Specification (commonly called XMS) which provides hardware-independent access to extended memory. Among other things, XMS also allows MS-DOS to use the 64 KB region just above the 1 MB boundary (i.e., 1024 KB to 1088 KB) for both program code and data. There seem to be a variety of versions of this device driver, and I have found that the version supplied with my version of ZDS MS-DOS does not work with the ZDS Windows 3.0 or any other Windows version.

As you can see, there are a number of memory managers which are supplied with MS-DOS 4.0, but I have found a couple of other memory managers that solve specific problems that these "free" ones do not. Both of the memory managers are available from Quarterdeck Office Systems; one is for 80286 systems and the other is for 80386 systems. Let's take a look at the 80286 program first.

QRAM

For those of you who have a Z-241 or Z-248 computer with one or more of the ZDS memory boards (i.e., the Z-405 or Z-415), Quarterdeck's QRAM is probably something you have been looking for. As you probably know, these ZDS memory boards only supported extended memory. The problem is that very little software is able to use extended memory, so these memory boards were not a whole lot of help if you needed expanded memory. Some people (including me) spent a bunch of money on these memory boards only to find they were not particularly useful, but there is a solution.

Quarterdeck's QRAM is a device

driver that "converts" extended memory into expanded memory, which can be used by most of today's software. Even the installation is simple and automated, so that is not much of a problem even if you are not interested in the technical details. QRAM is specifically for an 80286-based system with extended memory, and it will not work on anything else, such as the Z-150 series, the eaZy PC or the Z-386/486 series. QRAM works with virtually any 80286-based system, and it helps you to recover some of that investment in the Z-241/248 memory boards by allowing you to use that memory hardware as expanded memory which may be needed by your software. If your software can use expanded memory, this may be the solution you have been searching for. Although I specifically recommend QRAM for Z-241/248 systems with ZDS extended memory cards, you may find it useful on other 80286 systems as well. But there is more to QRAM than just the capability to convert your extended memory to expanded memory.

Manifest

Manifest is an extremely useful diagnostic and reporting program that is generally included with QRAM and QEMM386 (more on that in a minute). Although its main purpose is to help you fine tune your system for the memory managers, Manifest provides lots of other useful information as well. Aside from information about conventional, expanded, and extended memory usage, Manifest displays information about the CPU type, ROM, video card and memory, drive types (hard and floppy), and serial/parallel ports. Manifest also provides memory testing and benchmarks to help you see how your system performs. Even if you are not technically inclined, Manifest has some beautiful displays that can help you learn more about your system. And while it is generally included with both QRAM and QEMM386, you can also find Manifest as a single program if you do not need a memory manager. Manifest is highly recommended for any 80286 and 80386 system.

QEMM386

Like QRAM, QEMM386 "converts" extended memory to expanded memory, but QEMM386 is specifically for 80386-based systems. I am still using the Z-386/16 as my "production" system, and it is equipped with a total of 5 MB of memory with Z-515 and Z-505 ZDS memory cards. For testing purposes, I changed the DIP switches on both cards so that I have 1 MB of base memory and 4 MB of extended memory. My testing indicated that all of my popular software, including Quattro Pro version 2.0, works just fine with the QEMM386-defined "expanded" memory. I tested QEMM386 with ZDS

memory. I tested QEMM386 with ZDS MS-DOS 4.0.

I also tested QEMM386 with the ZDS version of Windows 3.0 because QEMM386 is advertised to be compatible with Windows. Older versions of QEMM386 are apparently not compatible with Windows 3, but I used QEMM386 version 5.1 which is. Most of you probably know that I am not much of an advocate of Windows, so my testing on that was relatively limited.

QEMM386 can also help "recover" memory within the first megabyte of base memory, and it works quite well. With QEMM386, you can also specify that some memory-resident programs, such as SHARE, are to be loaded into memory above 640 KB (and under 1 MB) to help provide more space for DOS and applications. I should note that QRAM will also do the same thing if you have an 80286-based system that came equipped with 1 MB of base memory. Although QRAM can also help recover and use some additional memory for systems that do NOT have 1 MB of base memory (e.g., the Z-241/248), it is limited on those systems as a result of the old memory configuration because those systems did not use 1 MB memory chips (DRAMs) or SIMMs. But if you have an 80386-based system with 1 MB of base memory, QEMM386 is quite effective and will help you use as much as an additional 256 K, depending on your system. QEMM386 is highly recommended.

Now that we have various memory-related problems under control, let's take a look at a program that provides the best alternative to Windows 3.0.

DESQview

DESQview is a program that will provide multitasking on just about any PC compatible computer. That includes the Z-150 series, the eaZy PC, and all of the later 80286 and 80386 systems. DESQview even works on systems that only have floppy drives; a hard drive is NOT required (unlike Windows 3.0). Of course DESQview runs faster on newer systems with a hard drive, but I think the fact that it also runs on older systems with 640 K of memory and no hard drive is nothing short of a miracle. And if you want to have multitasking on something like a Z-150, Windows 3 can't do that at all because it is limited to the real mode on older 8088 compatible systems.

As you can tell, there are several particular advantages that DESQview has. First, it runs on virtually any PC compatible system. Second, and perhaps more important, DESQview provides multitasking capabilities on those systems. From a cost perspective, DESQview is extremely cost effective because it works just fine with your current software and hardware. For example, you do not have to buy new software that runs under DESQview be-

cause it was designed to work with the "old" applications under DOS. On the other hand, if you want to fully utilize the capabilities (including multitasking) of Windows 3.0, you must buy new software that is specifically designed for Windows 3.0. You may also need to buy new hardware because Windows 3.0 requires a hard drive (with about 6-8 MB of free space for installation). And although Windows 3.0 works with a CGA video system, I've noticed that at least EGA is required for the best display. DESQview works just fine with CGA.

Although DESQview does not provide the graphical type of display that Windows 3.0 does, many users really don't need that because the requirement is to be able to perform multitasking with their current application software. Sure, you can use your current software with multitasking under Windows 3.0 if you have an 80386 system (running in 386 enhanced mode), but that requires you to buy an 80386 if you don't have one.

I checked out DESQview version 2.3, and except for the graphical interface (and additional hardware cost), it has virtually all of the features you are likely to need. DESQview even does windows to the point that it "out-windows Windows 3.0." With DESQview for example, you can even have a Windows 3.0 window. Now *that's* flexibility!

In case you are not too familiar with what DESQview's multitasking can do for you, let's say that you frequently use the Word Perfect word processor and the Quattro Pro spreadsheet. Once you load both programs under DESQview, you can quickly switch between them just like they were memory-resident programs. You can also copy data (not graphics) between the two programs, such as copying spreadsheet data into the word processor. And of course you can be doing that while you are also repacking a data base, compiling a C program, and downloading an interesting file from a bulletin board. Keep in mind that all of this can be done with your existing software and hardware, and you don't need to spend a fortune for new software and hardware to do it. DESQview does it all.

One point to keep in mind is that the speed of these tasks on your computer is strictly limited by the existing hardware, just like any multitasking process. Although you can perform these kinds of tasks on just about any system, newer and more powerful systems with a hard drive will be considerably faster. That's because multitasking means that although multiple tasks can be performed, *they are NOT performed simultaneously*. Each task takes as long as it normally would if it were running alone, so you will likely notice that your system is "slowing down" as you add more tasks. Remember that you only have one CPU, and it can still only do one task

at a time.

If you have an 80386 system, you may be interested in DESQview386 which is, of course, designed to take advantage of that CPU's features. I decided to buy and test the standard DESQview software because it had the widest capability for just about any PC compatible system.

I found no problems running DESQview with a wide variety of programs during my testing on the Z-386/16 under MS-DOS 4.0. I also bought a combination package that included DESQview, QEMM386, and Manifest, and that is available at considerable savings from the individual applications. From a user perspective, DESQview loads and runs much faster than Windows 3.0 on my system, and I like that. If you need a very cost-effective way to do multitasking without spending a fortune, DESQview is highly recommended.

Quarterdeck Software Summary

If you need a diagnostic and reporting program for your computer system, Manifest is highly recommended. It works with CPUs ranging from the 8088 and 8086 to the 80486. Manifest can be purchased alone, and it is also included in combination packages with QRAM and QEMM386.

QRAM is specifically for 80286 systems, and it solves the problem of converting ZDS extended memory to expanded memory. If you have a later 80286 system, QRAM can also "recover" some unused memory for some DOS programs. QRAM is highly recommended for all 80286-based systems, especially the Z-241 and Z-248 with ZDS extended memory cards. Although QRAM can be purchased alone, it is also included in combination packages with Manifest and DESQview.

QEMM386 is specific to 80386 systems, and it also allows you to "recover" some unused memory for some DOS programs. QEMM386 is highly recommended for all 80386-based systems. And QEMM386 can be purchased alone or in combination packages with Manifest and DESQview.

DESQview provides a multitasking capability on virtually any PC compatible CPU from the 8088 to the 80386. By the time this article is published, DESQview may also be available for the 80486 as well in a later version or perhaps the current version (2.3) I have will work, but I have not tested that. DESQview is highly recommended for multitasking and is available in combination with the two memory managers as mentioned above.

Even if you are not interested in all of the technical details, I think you will find at least one of Quarterdeck's programs will be quite valuable. And you don't need to be a technical wizard to install them either. For example, QEMM386 will check out

Continued on Page 38

Breaking the Churlish Bounds of DOS

Part 3

David W. Lind
RR 1, Box 3114
Bar Harbor, ME 04609

The first two articles in this series discussed means of handling programs too large for the 640K byte limit of IBM PC compatible computers through overlaying. This article discusses the use of expanded memory to store large amounts of data and code. Recall that expanded memory is not directly addressed by the computer's microprocessor. Instead, expanded memory boards contain special circuits which transfer selected blocks of expanded memory, usually called banks, in and out of a segment of memory that the microprocessor can address. This segment of memory is not unlike the overlay segment described in Part 2 of these series of articles. The expanded memory replaces the segment of conventional memory, which makes the process much quicker and more efficient than overlaying memory from a mass storage device.

The promise of expanded memory's speed and efficiency encouraged its development for the first Intel 8088 and 8086 microprocessors. Unfortunately, engineers used various approaches to interface the expanded memory system with applications programs. Many of these approaches involved the direct manipulation of registers on the memory boards through specific ports. Often, these approaches were not well documented, which discouraged wide spread use of expanded memory. Some sort of standard interface with programmer-friendly commands was necessary.

The Lotus Development Corporation, Intel Corporation, and Microsoft Corporation joined forces to develop such a standard which became known as the Lotus-Intel-Microsoft Expanded Memory Specification (LIM EMS). There are other specifications, but the LIM EMS is the most commonly used and is the only specification discussed in this article. The LIM EMS evolved to a powerful, but complex, tool. Only the basics are discussed in this article

although all the non-reserved functions are listed. The reader can delve into the LIM EMS in great depth by reading the appropriate sections of references numbered 1 and 6 in the list of references at the end of the article. However, enough detail is covered in this article to enable a programmer to use LIM EMS in most common applications.

Expanded Memory Basics

Expanded memory is accessed through a program called a driver which is normally accessed through interrupt 67h (103 in decimal notation). The driver is usually placed into memory by the configuration system process when the system is booted. This driver is a set of subroutines to which interrupt 67h points. Recall that an interrupt is a number representing a system program. This number is used with the microprocessor's INT instruction (machine language code CDh) to transfer execution control to the system program. Interrupt 67h will not operate unless the driver, i.e., the expanded memory program, is installed. When the driver is installed, it places the correct program address into the interrupt vector table at vector 67h.

To use interrupt 67h one places the function and subfunction number in register ax, any other necessary information in other registers, and then executes INT 67h. For example, to obtain the status of the expanded memory manager (the expanded memory driver) one places 4000h in ax and executes INT 67h. If the driver is working properly, the number zero is returned in register ah. Otherwise an error code is returned. Table 1 is a listing of these error codes. Any LIM EMS function/subfunction may return one of these error codes.

A glance at Table 1 may reveal terms which are unfamiliar to most readers. The first term that should be defined is "page." A page is a block of memory normally 16 kilobytes in size. Starting with LIM EMS

Version 4.0, the page sizes can be different and other default factors can be changed. For the purposes of this article, the limitations of earlier versions will be assumed. The reader who has Version 4.0 or later and wishes to use the more flexible features of that version will find detailed discussions of these features in the references previously cited. Logical pages are numbered blocks of expanded memory. The numbering starts at 0 for each handle. Handles are discussed later. Recall that expanded memory is a hardware memory bank switching scheme. The logical pages are stored in the memory banks and selected by the hardware on the memory boards for transfer to the computer's system memory range under the direction of the driver program.

A 64 kilobyte section of the microprocessor's memory address space is also divided into 16 kilobyte physical pages. This segment is called the "EMS Page Frame" and is normally located in an unused portion of the memory space between 640 kilobytes and one megabyte. There are four physical pages numbered from 0 to 3. Keep in mind that the physical pages are in system or conventional memory. The logical pages are in expanded memory.

When a portion of expanded memory is to be accessed, the expanded memory circuits "map" up to four logical pages to the physical pages in the EMS Page Frame. That is, as many as four logical pages in expanded memory become the physical pages in the system memory. These physical pages are totally accessible to a program by a long jump or far call. Contrary to what some folks believe, the EMS Page Frame may contain either code or data or both. More about that later.

Mapping associates a specific logical page with a specific physical page. The programmer can specify this map, thereby determining what logical pages in expanded

CODE	MEANING
00h	No error
80h	Internal driver error
81h	Malfunction of expanded memory hardware
82h	EMS manager is busy
83h	Handle number not defined
84h	Function not specified
85h	Maximum number of handles in use
86h	Mapping context error
87h	Exceeded maximum number of pages
88h	Insufficient pages available
89h	No pages requested
8Ah	Invalid logical page number
8Bh	Invalid physical page number
8Ch	Hardware state area is full
8Dh	Page-mapping storage attempt failed
8Fh	Requested an undefined subfunction
90h	Undefined attribute type
91h	Requested function is not supported
92h	Part of the source region was overwritten
93h	Destination and source sizes do not match
94h	System and expanded memory overlaps
95h	Offset is outside logical page
96h	Region size exceeds one megabyte
97h	Source and destination have same handles
98h	Source or destination type undefined
99h	Reserved
9Ah	Specified alternate map register set is not supported
9Bh	All alternate map register sets are allocated
9Ch	Alternate map register sets not supported
9Dh	Alternate register set is not defined
9Eh	Dedicated DMA channels are not supported
9Fh	Specified DMA channel is not supported
A0h	Specified handle name is not defined
A1h	Duplicate handle name requested
A2h	Attempted overlay of conventional system memory
A3h	Corrupted source array or segment count is too large
A4h	Operating system denied access

Table 1. LIM EMS Error Codes.

memory are currently available to the program. Reading or writing the physical page is equivalent to reading or writing the mapped logical page. There is one more important concept that must be discussed, the concept of a "handle."

A handle is a number which identifies a specific entity. Open data files are assigned unique handles to make references to them easier. Likewise, expanded memory applications are assigned handles by the expanded memory manager. When a programmer requests a specific number of logical pages to be allocated to an application, the expanded memory manager will return with a handle number if the allocation was successful. From that point on, the programmer must identify the set of logical pages, thus allocated, with the handle number. The reason for this process is clear if one recognizes that there may be more than one program using expanded memory. For example, there may be a virtual disk program and several terminate-but-stay-resident programs using expanded memory. Each program will have unique sets of logical pages identified by a unique handle number assigned by the expanded

memory manager.

Using Expanded Memory

The easiest way to use expanded memory is to use a driver or program which is designed to use expanded memory. For example, most operating systems for computers capable of expanded memory have drivers or programs which create files in memory equivalent to disk drives or hard disk partitions. To use any of these programs or drivers with expanded memory, the Expanded Memory Manager (EMM) program must be installed. Normally, the EMM program has the file name EMM.SYS and is installed through the CONFIG.SYS file by the command, DEVICE=EMM.SYS. The appropriate switch in the applications program or driver must also be set. The memory boards may also need jumper and/or switch changes.

After the EMM is installed, any program can access expanded memory through interrupt 67h. Tables 2 through 4 are listings of the functions available through this interrupt. These functions are available to applications programmers who wish to use expanded memory. The assembly language

program at the end of this article shows how to use the basic functions to process data and execute code stored in expanded memory. Assembly language is the most direct of the common programming languages. Therefore, one can better show the processes associated with expanded memory with Assembly Language. Many higher level programming languages can access the EMM in a similar way.

The demonstration program uses only the EMS Version 3.0 functions. The additional functions in more current versions permit more control and flexibility, but don't introduce basic capabilities. The program demonstrates all the important EMM functions. Not all functions demonstrated are required in all programs. However, it is good programming practice to include most of the functions. The reader may wish to delete some of the functions in an attempt to execute the program. Many functions were included, e.g., the Get EMS Version function, to show how the function is used - not because they are necessary. Of course, the reader may wish to delete the comments when transcribing the program. It is instructive, though not necessary, to execute the program under control of a program like Microsoft Code View.

EMMPROG.EXE begins by shrinking the memory allocated to the program to the minimum amount necessary. This process prevents possible memory problems when the program attempts to call the overlay, OVERLAY3.OVL, later. These shrinking and overlay processes were described in detail in Part 2 of this series of articles. The OVERLAY3.OVL program is exactly the same as the program of the same name in Part 2 and must be in the same directory as the EMMPROG.EXE file. The source code for OVERLAY3.OVL is listed at the end of this article. Recall that these two files must be assembled and linked independently.

Once the program's allocated memory is shrunk, the presence of the EMM is ascertained. This step is not necessary if one is certain that the EMM is present, but it is prudent. A call to interrupt 67h may establish the presence of the EMM. If the EMM is not present, the results of a call to interrupt 67h are unpredictable. The code in the demonstration program looks at the header of the interrupt 67h code. If the data at the expected location of the header is EMMXXXX0, then the EMM is present and meets the LIM EMS. Be careful not to use the vector offset at interrupt 67h in the interrupt vector table. That offset is the executable entry point to the program, not the beginning of the header. The program begins at the segment address with offset 0. The header information begins at offset 0Ah (10 in the decimal system).

If the EMM is present, the status of the EMM is determined. This step is usually unnecessary, but will help determine if the

Function	Description
40h Get Status	Returns a 0 or an error code in register ah.
41h Get Page Frame Address	Returns EMS Page Frame segment address in register bx.
42h Get Number of Logical Pages	Returns total number of logical pages in register dx, the number of unallocated logical pages in register bx.
43h Allocate Handle and Logical Pages	Register bx must contain the number of logical pages to be allocated. The function returns the handle number in register dx.
44h Map Expanded Memory Page	Register al must contain the number of the physical page to be mapped. Register bx must contain the number of the logical page to be mapped to the physical page or -1 if an existing map is to be released. Register dx must contain the handle number.
45h Release Handle	Register dx must contain the handle number.
46h Get EMS Version	Returns the major EMS version in upper half of register al and the minor version in the lower half.
47h Save Page Map	Register dx must contain the handle number. This function saves maps of 64K page frames only.
48h Restore Page Map	Register dx must contain the handle number. This function restores maps of 64K page frames only.
4Bh Get Number of Assigned Handles	Returns number of assigned handles in register bx. NOTE: Handle numbers are zero based. Thus, if the number of handles is 3, the highest numbered handle is 2.
4Ch Get Number of Pages for a Handle	Register dx must contain the handle number. The function returns the number of pages assigned to the handle in register bx.
4Dh Get Number of Pages Assigned to All Handles	Registers es:di must be the address of a buffer to receive the requested data. The function returns the total number of assigned handles in register bx and a double word entry in the buffer for each handle. The first word is the handle number and the second word is the number of pages assigned to that handle.

Table 2. EMS Version 3.0 Functions.

EMM is functional. The status is determined by calling interrupt 67h with 40h in register ah. If the EMM is functional, a zero is returned in ah. Otherwise, an error code is returned. If one has an error code returned at this point, re-boot the system to clear any problems generated by previously run programs. If that action does not clear the problem, there is a conflict between the EMM and another program or the EMM is defective.

The EMS version number is found using function 46h. Note that the major version number (the integer before the decimal point) is a binary number (actually a binary coded decimal number which is the same as binary for numbers under ten) in the upper four bits of al. The minor version

number (the integer after the decimal point) is a binary coded decimal number in the lower four bits of al. These numbers are separated, stored, converted to ASCII characters, and displayed. Storage of these numbers in this program is not necessary, but is included as a good practice.

The program then determines the Page Frame segment address. This information is vital. Function 41h places the Page Frame segment address in register bx. The segment address is then stored for later use. The program also displays this address in hexadecimal form using the procedure "binhexdis." Most readers will probably find that the Page Frame segment begins at segment address D000h.

The total number of logical pages and the

number of unallocated logical pages are determined by function 42h. This information is displayed and a check is made to determine if there are enough pages for subsequent requests. If there are not enough pages, the program terminates. The procedure "bindis" is written with the assumption that there are no more than 99 logical pages. If there are more, the procedure must be revised to handle larger numbers.

The next part of the program determines the number of handles currently assigned. That number is converted to an ASCII character and displayed by the procedure "bindis." This procedure is written with the assumption that no more than 99 handles are assigned. If more handles are assigned, the procedure must be expanded to display greater numbers.

Now that the characteristics of the expanded memory are determined, the program requests five logical pages and a handle number for subsequent use of expanded memory. This request is accomplished by placing the number of requested logical pages in register bx and calling function 43h. If the function is successful in assigning the pages and handle, the handle number is returned in dx. The handle number is stored in memory and displayed on the console. At this point, the number of handles increases by one and the number of unallocated pages decreases by five. However, this information is not displayed.

The next step is to assign logical pages to physical pages. Since there are only four physical pages, one can assign only four logical pages at a time. The demonstration program will not use all the physical pages, so not all the physical pages will be mapped. The program maps logical page numbered 3 to physical page numbered 0. This process requires that the physical page number be placed in register al, the logical page number be placed in register bx, the handle number be placed in register dx, and the function number (44h) be placed in register ah before interrupt 67h is executed. Keep in mind that for each handle the logical pages are numbered from 0 to n, where n is one less than the number of pages requested. Thus, the logical page number along with the handle number identifies a unique block of expanded memory. Until the mapping is released, anything written to or read from the physical page address will be written to or read from the logical page in expanded memory mapped to the physical page.

Now one is ready to use expanded memory. The program uses the move string instruction to move data from the data segment to physical page 0. This process requires that the Page Frame segment address be placed in register es, that is, the Page Frame segment temporarily becomes the extra segment. When this data transfer is complete, the mapping between the

physical page and the logical page is released. This process involves calling function 44h again in the same way as before except that -1 is placed in register bx.

Now logical page 3, containing the data, is mapped to physical page 2. At this point, the data in expanded memory logical page 3 is available through physical page 2. To demonstrate that this assertion is true, the data is read and displayed. In order to use INT 21h-Function 09h, Display String, the Page Frame segment must be redefined as the data segment. Note that the offset address is 8000h which is the beginning of page 2 of the Page Frame. Recall that each page is 16 kilobytes long. Physical page 0 will begin at offset 0. Physical page 1 will begin at an offset of 16 kilobytes or 4000h. Physical Pages 2 and 3 begin at offsets of 8000h and C000h, respectively.

The techniques used to store and retrieve data to and from expanded memory are straight forward. Generally, expanded memory was developed to handle massive data storage in memory. But, can one store code in expanded memory and execute it? The answer is yes. The program demonstrates the techniques. However, only one page is used. Normally, one would use the entire Page Frame for code or transfer several pages of code segments to a part of conventional memory where the code can be executed. The latter technique may be necessary if the code interferes with EMM operations, e.g., if the code re-maps expanded memory.

The program maps physical page 1 to logical page 4. Then a procedure is overlaid onto physical page 1 which is equivalent to placing the procedure in expanded memory. Note that the Page Frame segment address is temporarily increased by 400h paragraphs (16 kilobytes) to start the overlay at physical page 1. Keep in mind that the operating system overlays by paragraph numbers (segment addresses) not byte locations. Then the procedure is executed by a far call. Refer to Part 2 of this series of articles for more details on the overlay and call.

Finally, the program releases the handle and expanded memory. This step is crucial. If the handle and memory are not released before the program terminates, the handle and memory remain intact and may be used/released by another program or will remain active until the computer is reset. Thus, a programmer should release handles and memory when their utility is finished. On the other hand, not releasing expanded memory is a handy way to pass data or code from one program to another. The reader, at this point, might reflect on the possibilities of combining overlay techniques with expanded memory to make large and complex programs viable on a personal computer.

In the last article of this series, extended memory techniques are discussed. Ex-

Subfunction	Description
00h Save Page Map	Register ax contains 4E00h. Registers es:di is the address of the buffer to hold the mapping information. Upon return, the function has placed the map information in the buffer.
01h Restore Page Map	Register ax contains 4E01h. Registers ds:si contains the address of the buffer holding the map information. Upon return, the indicated map is restored.
02h Save and Restore Page Map	Register ax contains 4E02h. Registers ds:si contains the address of the buffer for the restored map. Registers es:di contain the address of the buffer for the saved map. The function saves the current map and restores the indicated map.
03h Get Size of Page Map Information	Register ax must contain 4E03h. The function returns the size of the map buffer in register bx.

Table 3. LIM EMS Version 3.2 New Function 4Eh Subfunctions.

tended memory is only available on computers using the 80286 or more advanced microprocessor. For readers who have computers using the 8088/86 microprocessor, the discussion of circumventing the memory limitations of DOS ends with this article. The reader is encouraged to experiment with the techniques with a reasonable amount of caution. It will not only help the reader understand how some commercial software works, but may enable the reader to apply these techniques to solve difficult problems.

Before closing this discussion, the capabilities of the EMM introduced by the newer versions of LIM EMS should be briefly discussed. Table 3 lists the LIM EMS enhancements introduced in version 3.2. These enhancements are all called through function 4Eh and permit the save of more than one map set. The save and restore map functions were not used in the demonstration program. EMS Version 3.0 allowed the programmer to save one page mapping, re-map memory, and then restore the original page mapping with one function call. EMS Version 4.0 introduces the capability to save and restore partial page maps, functions to map more than one page at a time, reallocation of pages, handle attributes, handle names, raw pages, and the ability to manipulate hardware. Table 4 lists the functions and subfunctions added in version 4.0. The reader can review this list to determine if any of these functions may be useful in some application. More details on the use of these functions are contained in the references previously cited. Of course, some of the new terms should be defined to make sense of Table 4.

There are two types of handle attributes, volatile and non-volatile. Volatile handles and memory are released when the computer is reset (e.g., Ctrl-Alt-Del key sequence or "warm boot"). Nonvolatile handles and

memory are maintained when the computer is reset. This capability requires a hardware system specifically designed for this purpose in addition to EMS Version 4.0.

Handle names are eight-byte strings which enable handles to be referenced by name. This feature is useful when sharing data between programs. It is not necessary to find a means to pass a handle number to another program when a handle name is specified in the program which establishes the handle. Other programs can request the EMM to identify the handle number given the handle name.

EMS Version 4.0 contains map functions which also call or jump to executable code. This feature is useful for code segments which will not fit in the Page Frame.

Some expanded memory systems have the capability to use pages which are other than 16 kilobytes in size. These pages are called "raw pages." Their size can be obtained through the Get Hardware Configuration function, Function 59h - Subfunction 0.

There are version 4.0 functions which permit task switching by operating systems, e.g., Microsoft's Windows. Generally, these functions are not used by applications programmers. Such functions include alternate mapping and functions involving Direct Memory Access (DMA) registers.

Summary

This article discussed the use of expanded memory in some detail. When using expanded memory, the programmer should first ascertain that the Expanded Memory Manager is installed and operational. One should then determine if there is enough expanded memory available for the intended application. Then the necessary number of expanded memory pages can be allocated and a handle number ob-

**Function/
Subfunction Number**

Description

4Fh/00h	Save partial page map.
4Fh/01h	Restore partial page map.
4Fh/02h	Get size of partial page map.
50h/00h	Map multiple pages by page numbers.
50h/01h	Map multiple pages by physical page segment addresses.
51h/00h	Reallocate pages for existing handle.
52h/00h	Get handle attribute.
52h/01h	Set handle attribute.
52h/02h	Determine attribute capability.
53h/00h	Get name of specified handle.
53h/01h	Give a name to a handle.
54h/00h	Get names of all handles.
54h/01h	Get handle number given name.
54h/02h	Get total number of handles.
55h/00h	Map by page numbers and jump to executable code.
55h/01h	Map by physical page segment addresses and jump to executable code.
56h/00h	Map by page numbers and call executable code.
56h/01h	Map by physical page segment addresses and call executable code.
56h/02h	Find stack space required for map page and call.
57h/00h	Move memory region.
57h/01h	Exchange memory regions
58h/00h	Get addresses of physical pages.
58h/01h	Get number of physical pages.
59h/00h	Get hardware configuration.
59h/01h	Determine number of raw pages.
5Ah/00h	Allocate handle and standard (16Kb) pages.
5Ah/01h	Allocate handle and raw pages.
5Bh/00h	Get alternate map register save area.
5Bh/01h	Set alternate map register save area.
5Bh/02h	Get size of alternate map register save area.
5Bh/03h	Allocate alternate map register set.
5Bh/04h	Deallocate alternate map register set.
5Bh/05h	Allocate DMA register set.
5Bh/06h	Enable DMA on alternate map register set.
5Bh/07h	Disable DMA on alternate map register set.
5Bh/08h	Deallocate DMA register set.
5Ch/00h	Prepare for warm boot.
5Dh/00h	Enable EMM functions 59h, 5Bh, and 5Dh. Assign access key if not already assigned.
5Dh/01h	Disable EMM functions 59h, 5Bh, and 5Dh. Assign access key if not already assigned.
5Dh/02h	Release access key.

Table 4. LIM EMS Version 4.0 New Functions.

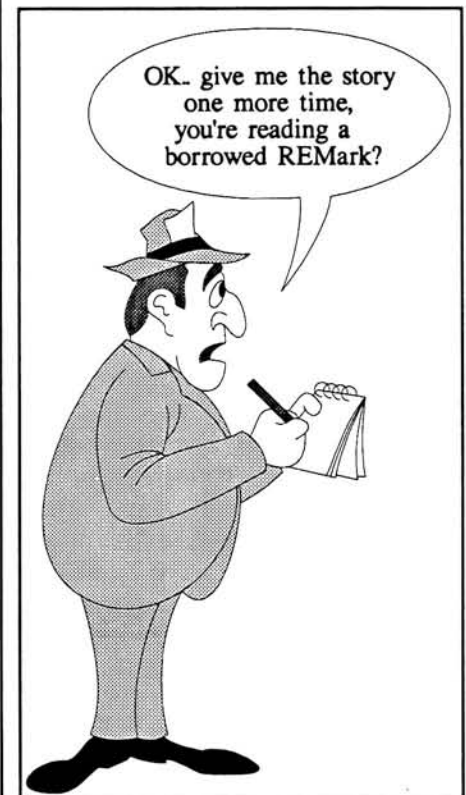
tained. The expanded memory can then be used to hold data or code.

References

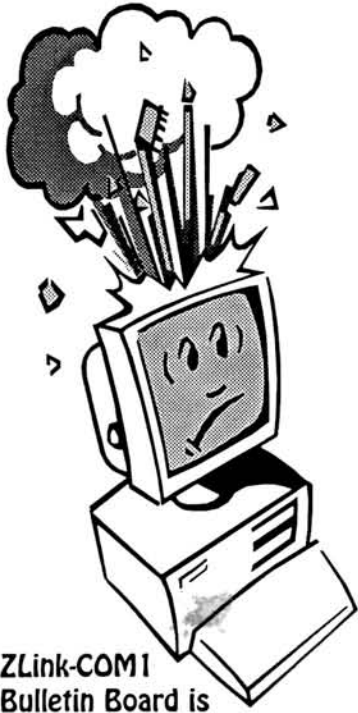
1. Duncan, Ray, Extending DOS, 1990, Addison-Wesley Publishing Company, Inc., Reading Massachusetts.
2. Microsoft Corporation, Microsoft Macro Assembler 5.0 Microsoft CodeView and Utilities, 1987, Microsoft Corporation, Redmond, Washington.
3. Microsoft Corporation, Microsoft Macro Assembler 5.0 Programmer's Guide, 1987, Microsoft Corporation, Redmond, Washington.
4. Microsoft Corporation, Microsoft MS-DOS Operating System Programmers Reference, 1984, Microsoft Corporation, Bellevue, Washington.
5. Phoenix Technologies Ltd., System BIOS for IBM PC/XT/AT Computers and Compatibles, 1989, Addison-Wesley Publishing Company, Inc. Reading, Massachusetts.
6. The Waite Group, The Waite Group's MS-DOS Developer's Guide, second edition, 1989, Howard W. Sams & Company, Indianapolis, Indiana.

Trademarks

IBM is a trademark of the International Business Machine Corporation.
 Intel is a registered trademark of the Intel Corporation.
 Lotus is a registered trademark of the Lotus Development Corporation.
 Microsoft, CodeView, and Windows are registered trademarks of the Microsoft Corporation.



Hardware Problems?



**ZLink-COM1
Bulletin Board is
Your Best Source for
Technical Information!
Modem: (616) 982-3956**



ACCOUNTING & TAX

Not sure if you need the expensive 'Chinese Flower 1-2-3', or 'Spanish Numeral Four' spreadsheet programs? Then find out for only \$20! "CheapCalc" will do double precision addition, subtraction, multiplication, division, power, SUM, and roots (using fractional powers). CheapCalc has many other functions too numerous to mention (just like the expensive spreads)! CheapCalc is available for all Heath/Zenith computers and operating systems. For more information, check out page 58 of the Software Catalog Update #1, or call HUG and order your copy today.

COMMENT * EMMPROG.EXE

This program demonstrates the basic Expanded Memory Manager functions of the Lotus-Intel-Microsoft Expanded Memory Specification (LIM EMS).

Copyright (C) 1990 by David W. Lind

No expressed or implied warranties are made for this program with respect to merchantability or fitness for a particular purpose. The user assumes all responsibility for any damages resulting from the use of this program. *

```

data SEGMENT ;Program Data Segment
testdata DB ' Test data for demonstration. This information was '
          DB ' displayed from logical',10,13,' page 3. '
          DB 14 DUP (84,69,83,84,32),10,13,'$',1024 DUP(?)
majver DB ? ;Storage location for major version number
minver DB ? ;Storage location for minor version number
COMMENT * The following two words must be together and in the order
        shown. *
pfoff DW 0 ;Storage location for Page Frame offset address
pfseg DW ? ;Storage location for Page Frame segment address
handle DW ? ;Storage location for handle number
header DB 'EMMXXX0' ;Expected header data
lfcrl DB 10,13,'$' ;Line feed/carriage return characters
ovname DB 'OVERLAY3.OVL',0 ;File name of overlay procedure
mess0 DB ' ERROR - could not shrink memory.',10,13,'$'
nodriver DB ' ERROR - Expanded Memory Manager not present. ',10,13,'$'
bademm DB ' ERROR - Expanded Memory Manager defective.',10,13,'$'
notenough DB ' ERROR - Insufficient expanded memory to continue. ','$'
nohand DB ' ERROR - Could not allocate handle and pages.',10,13,'$'
emmok DB ' Expanded Memory Manager responding. Version ','$'
numhand DB ' Number of active handles is ','$'
segaddr DB 'h is the EMS Page Frame segment address. ',10,13,'$'
totalpages DB ' Total number of logical pages is ','$'
unalpages DB ' Number of unallocated logical pages is ','$'
handnum DB ' Allocated logical pages with handle number ','$'
data ENDS
esdata SEGMENT
segaddr DW ? ;Address of overlay segment
relfact DW ? ;Relocation factor for overlay
esdata ENDS

code SEGMENT ;Program Code Segment
ASSUME cs:code,ds:data,es:esdata
start: ;Program Entry Point

mov ax,data ;Put address of data segment in ax
mov ds,ax ;Put address of data segment in ds

COMMENT * The following section of code shrinks the program memory
        space to that which it actually requires. This step may be
        necessary to ensure the overlay will load properly. *

mov cx,es ;Put PSP segment address in cx
mov bx,seg pgmend ;Put address of last segment in bx
sub bx,cx ;Find length (in paragraphs) of this program
mov ax,4A00h ;Put Set Block function number in ax
int 21h ;Shrink memory allocation
jnc memok ;If carry flag is clear (no error), continue
mov ah,09h ;Else, put Display String function number in ah
lea dx,mess0 ;Put message offset in dx
int 21h ;Display error message string
jmp xit ;Terminate program

memok:
mov ax,esdata ;Put address of extra segment in ax
mov es,ax ;Initialize es register

COMMENT * The following code determines if an LIM EMS Expanded
        Memory Manager is present. The extra segment is set to
        the beginning of the Expanded Memory Manager code and the
        header information at offset 0Ah is examined. If the
        header contains EMMXXX0, the Expanded Memory Manager
        exists and conforms to the LIM EMS. NOTE: One may be
        tempted to use INT 21h - Function 35h, Get Interrupt
        Vector to find the address of the code. That function

```

may return the correct segment address, but the offset will be the executable entry point, not the beginning of the code. *

```

push    es                ;Save the es register
mov     ax,0              ;Put 0 in ax
mov     es,ax             ;Make es the beginning of the interrupt table
mov     ax,es:414        ;Find segment address at int 67h
mov     es,ax             ;Put segment address of EMM in es
mov     di,10            ;Put offset of EMM header in di
mov     cx,8             ;Initialize loop count to 8
mov     bx,0             ;Initialize bx to 0

loop1:
mov     al,ds:header[bx] ;Put character of expected header in al
cmp     al,es:[di]       ;Compare to character in header
jne     noemm            ;If not the same, no EMM is present
inc     di               ;Increment the header offset address
inc     bx               ;Increment the comparison string offset address
dec     cx               ;Decrement the loop counter
jnz    loop1            ;Jump if not zero to loop beginning
pop     es               ;Restore es register
jmp     cont1           ;Jump over error code

noemm:
lea     dx,nodriver      ;Put offset address of message in dx
mov     ah,09h          ;Put Display String function number in ah
int     21h             ;Call display string
jmp     xit              ;Exit program

cont1:

```

COMMENT * The Get Status function in the following code determines if the Expanded Memory Manager is responding properly. *

```

mov     ax,4000h         ;Put Get Status function number in ax
int     67h             ;Call Get Status
cmp     ah,0            ;Check for no error
jne     emmbad          ;If error, go to error code
jmp     cont2           ;Else, continue

emmbad:
lea     dx,bademm       ;Put offset address of message in dx
mov     ah,09h          ;Put Display String function number in ah
int     21h             ;Call display string
jmp     xit              ;Exit program

cont2:

```

COMMENT * The following code obtains and displays the EMS Version number. This information is important if the programmer plans to use the more sophisticated functions in more advanced EMS versions. *

```

lea     dx,emmok        ;Put offset address of message in dx
mov     ah,09h          ;Put Display String function number in ah
int     21h             ;Call display string
mov     ax,4600h        ;Put Get EMS Version function number in ax
int     67h             ;Call Get EMS Version function
mov     bl,al           ;Put compressed version number in bl
mov     cl,4            ;Put number of bits to shift in cl
shr     bl,cl           ;Shift bl right four bits
add     bl,48           ;Convert BCD to ASCII
mov     majver,bl       ;Store major version number
mov     bl,al           ;Put compressed version number in bl
and     bl,0Fh         ;Mask upper four bits
add     bl,48           ;Convert from BCD to ASCII
mov     minver,bl       ;Store minor version number
mov     dl,majver       ;Put major version number in dl
mov     ah,02h         ;Put Display Character function number in ah
int     21h             ;Display character
mov     dl,'.'         ;Put period character in dl
mov     ah,02h         ;Put Display Character function number in ah
int     21h             ;Display character
mov     dl,minver       ;Put minor version number in dl
mov     ah,02h         ;Put Display Character function number in ah
int     21h             ;Display character
lea     dx,lfcrl        ;Put message address in dx
mov     ah,09h          ;Put Display String function number in ah
int     21h             ;Call Display String

```

COMMENT * The EMS Page Frame segment address is necessary for the

Quality Enhancements!

EaZy PC Products

EZM-128: Expand 512K base memory to 640K. Simple, plug-in installation. \$95.00
EZCLOCK: Calendar/Clock. Piggy-back add-on for EZM-128. \$35.00

No Slot Clock/Calendar

FBE SmartWatch: Automatic date/time on bootup. Installs under BIOS/Monitor ROM. Ten year battery. Works with all Heath/Zenith MSDOS computers. For PC's \$32.00. Z-100 \$33.00 Module: \$25.00

H/Z-148 Expansions

ZEX-148: Adds one full-size and one half-size expansion card slot. \$79.95
ZP-148: PAL chip expands existing 640K memory to 704K. CGA/MDA only! \$19.95

Configuration Control

CONFIG MASTER: Menu-select active CONFIG.SYS during bootup. Software for PC/Z-100 MSDOS. \$29.95

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

VCE-150: Eliminate video card. Install EGA or VGA card. All plug in. Includes circuit board, SRAM and RM-150. \$49.95
ZP640 PLUS: PAL chip to expand standard memory card to 640/704K with 2 banks of 256K RAM chips (not included). \$19.95
ULTRA-PAL: Three PAL chips: MR150 for 704K + 512K RAM Disk; MR150T for 640K + 512K RAM Disk; LIM150 for 640K + 512K (32 pages) of simulated v3.2 Lotus/Intel/Microsoft Expanded Memory. With software. Install on standard memory card. No soldering. Needs 45 256K RAM chips (not included) for maximum memory configuration. \$39.95
COM3: Change existing COM2 to COM3. Put internal MODEM at COM2. Don't lose serial port. With software. \$29.95

H/Z-100 Modifications

ZMF100A: Expand "old" motherboard (p/n 181-4917 or less) using 256K RAM chips (not included). No soldering. \$65.00
ZRAM-205: Put 256K RAM chips on your Z-205 board. Get 256K plus 768K RAM disk. Contact us for data sheet before ordering. Without RAM chips. \$39.00

H/Z-89 Add-Ons

H89PIP: Parallel printer 2 port interface card. With software. \$50.00 Cable \$24.00
SLOT4: Add fourth expansion slot to right-side accessory bus. \$39.95

Order by mail, FAX, telephone, or see your dealer. UPS/APO/FPO shipping included. VISA/MasterCard. WA residents add 8.1% tax. Hours: M-F 9-5 PST. We return all calls left on our answering machine!

FBE

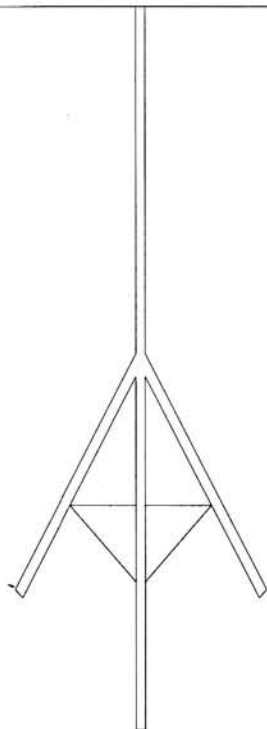
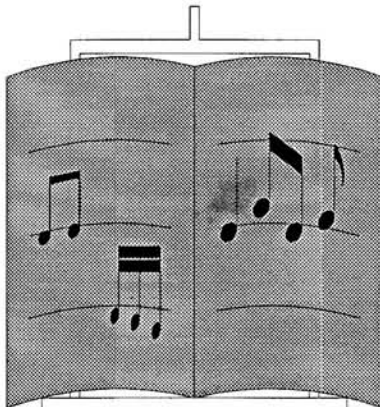
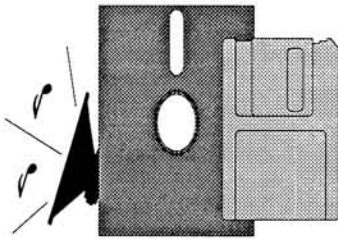
FBE Research Company, Inc.

P.O. Box 68234, Seattle, WA 98168

206-246-9815 Voice/FAX TouchTone Selectable

Reader Service #104

New Software Product!



The Electronic Clavier
P/N 885-6016

transfer of data to and from expanded memory. This address can be specified when the Expanded Memory Manager is installed. Normally, the Expanded Memory Manager determines this address. *

```

mov ax,4100h ;Put Get Page Frame Address function # in ax
int 67h ;Get segment address of EMS Page Frame
mov pfseg,bx ;Store Page Frame segment address
mov dl,20h ;Put ASCII space character in dl
mov ah,02h ;Put Display Character function number in ah
int 21h ;Call Display Character
mov al,bh ;Put upper half of segment address in al
call binhexdis ;Convert to hexadecimal and display
mov al,bl ;Put lower half of segment address in al
call binhexdis ;Convert to hexadecimal and display
lea dx,segadd ;Put offset of message in dx
mov ah,09h ;Put Display String function number in ah
int 21h ;Display string

```

COMMENT * The following code determines the total number of logical pages (pages of expanded memory) available to the Expanded Memory Manager and the number of logical pages not as yet allocated. If fewer than five logical pages are available, the program terminates. *

```

lea dx,totalpages ;Put message offset in dx
mov ah,09h ;Put Display String function number in ah
int 21h ;Display String
mov ax,4200h ;Put Get Number of Pages function # in ax
int 67h ;Get number of logical pages
mov ax,dx ;Put number of logical pages in ax
call bindis ;Display binary number
lea dx,unalpages ;Put message address in dx
mov ah,09h ;Put Display String function in ah
int 21h ;Display string
mov ax,bx ;Put number of unallocated pages in ax
call bindis ;Display binary number
cmp bx,5 ;Compare number of unallocated pages to 5
jae enough ;If 5 or above, go to enough
lea dx,notenough ;Put offset address of message in dx
mov ah,09h ;Put Display String function number in ah
int 21h ;Call Display String
jmp xit ;Terminate program

```

enough:

COMMENT * The following code determines the number of handles currently in use and displays the result. *

```

lea dx,numhand ;Put message offset in dx
mov ah,09h ;Put Display String function number in ah
int 21h ;Display String
mov ax,4B00h ;Put get number of handles function # in ax
int 67h ;Get number of active handles
mov ax,bx ;Put number of active handles in ax
call bindis ;Display binary number

```

COMMENT * A request for five logical pages of expanded memory and a handle number is made by the following code. An assumption is made that enough handles are available. Previous code determined that enough pages are available. After this call to the Expanded Memory Manager, the number of handles increases by one and the number of unallocated logical pages decreases by five. *

```

mov bx,5 ;Put request for 5 pages in bx
mov ax,4300h ;Put Allocate Handle and Pages function # in ax
int 67h ;Allocate handle and pages
cmp ah,0 ;Success?
je cont3 ;If successful, go to cont3
lea dx,nohand ;Put offset of message in dx
mov ah,09h ;Put Display String function number in ah
int 21h ;Display string
jmp xit ;Terminate program

```

cont3:

```

mov handle,dx ;Store handle number
lea dx,handnum ;Put offset address of message in dx

```

```

mov    ah,09h          ;Put Display String function number in ah
int    21h            ;Display string
mov    ax,handle      ;Put handle number in ax
call   bindis        ;Display handle number

```

COMMENT * The following code causes the expanded memory logical page numbered 3 to replace the physical page numbered 0 in the EMS Page Frame of conventional memory. *

```

mov    ah,44h        ;Put map page function number in ah
mov    al,0          ;Put Page Frame page number in al
mov    bx,3          ;Put logical page number in bx
mov    dx,handle     ;Put handle number in dx
int    67h          ;Call Map Expanded Memory Page

```

COMMENT * The instruction movsb, move string by bytes, is used to transfer the data located at testdata in the data segment to expanded memory. The data is transferred to the Page Frame segment in conventional memory, but the data is actually sent to expanded memory by the Expanded Memory Manager per the mapping function just executed. *

```

mov    cx,1024       ;Put number of bytes to be moved in cx
lea    si,testdata  ;Put address of testdata in si
mov    ax,pfseg     ;Put Page Frame segment address in ax
mov    es,ax        ;Make extra segment Page Frame segment
mov    di,0         ;Make Page Frame offset 0
rep    movsb       ;Move test data to expanded memory
mov    ax,esdata    ;Put address of esdata in ax
mov    es,ax       ;Restore extra segment

```

COMMENT * The following code releases the map between physical page 0 and logical page 3. Note that only the physical page is specified. *

```

mov    ah,44h        ;Put map page function number in ah
mov    al,0          ;Put Page Frame page number in al
mov    bx,-1         ;Put -1 in bx to release map
mov    dx,handle     ;Put handle number in dx
int    67h          ;Call Map Expanded Memory Page

```

COMMENT * The following code maps the logical page in expanded memory containing data (logical page 3) to physical page 2. *

```

mov    ah,44h        ;Put map page function number in ah
mov    al,2          ;Put Page Frame page number in al
mov    bx,3          ;Put logical page number in bx
mov    dx,handle     ;Put handle number in dx
int    67h          ;Call Map Expanded Memory Page

```

COMMENT * Now the data in physical page 2, which is actually logical page 3 in expanded memory, is read and displayed. Note that the physical page begins at an offset which is the number of the physical page (0,1,2, or 3) multiplied by 4000h. The Display String function requires that the data segment be re-defined as the EMS Page Frame segment. *

```

mov    ax,pfseg     ;Put Page Frame Segment address in ax
push   ds          ;Save data segment address
mov    ds,ax       ;Make Page Frame Segment the data segment
mov    dx,8000h    ;Put physical page 2 offset in dx
mov    ah,09h     ;Put Display String function number in ah
int    21h        ;Call Display String
pop    ds         ;Restore data segment

```

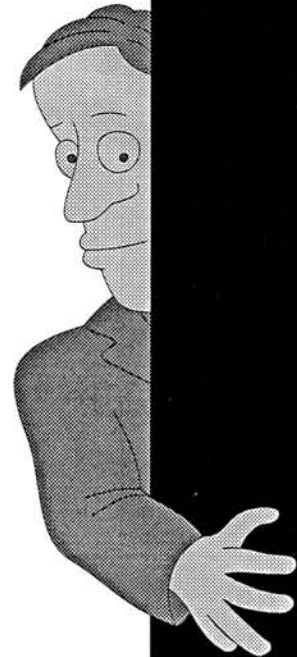
COMMENT * The following code maps physical page 1 to logical page 4. *

```

mov    ah,44h        ;Put map page function number in ah
mov    al,1          ;Put Page Frame page number in al
mov    bx,4          ;Put logical page number in bx
mov    dx,handle     ;Put handle number in dx
int    67h          ;Call Map Expanded Memory Page

```

COMMENT * Now the procedure in OVERLAY3.OVL is overlaid on physical page 1. See Part 2 of this series of articles for details of this process. Of course, logical page 4 in expanded memory will contain this overlay. *



We are still trying to get caught up on our issues.

We would like you to know how much we appreciate your patience.
Thank You!

RAPID INDEX: A computer program that lists titles of magazine articles by subject or in alphabetical order.

CPM or PC/MS-DOS 2.1 or later.

DRIVE:

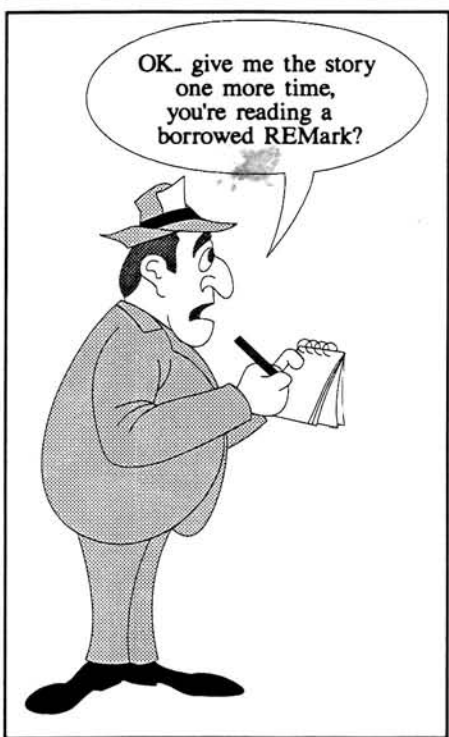
COMPUTER: 5.25"DD..\$21.00
 Z-100 Models 5.25"HD..\$23.00
 IBM Compatable 3.5"DD....\$23.00

Magazines currently available:
 Elect. Service & Tech. (1984 - 89)
 REMark computer magazine (1985-89)
 Calif. residents add 6.75% sales tax.
 Send check or money order (US funds) only to:

B.U.D. Unlimited,
P.O. Box 503, Elverta, CA 95626.

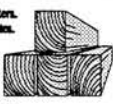
REMark is a trademark of Heath. Zenith User's Group.

Reader Service #250




Splinters. termite.

With this board all you have is a stick in the mud!



Message. Ovens.

With this board, you're really talking!



Which is more useful?

Zenith Data Systems

```

mov ax,pfseg ;Put page frame segment address in ax
add ax,400h ;Adjust to beginning of physical page 1
mov segaddr,ax ;Put segment address in parameter block
mov relfact,ax ;Set relocation factor
lea dx,ds:ovname ;Put overlay file name offset address in dx
lea bx,es:segaddr ;Put parameter block offset address in bx
mov ax,4B03h ;Put function and subfunction number in ax
int 21h ;Call Execute Program function

```

COMMENT * The following code executes the procedure overlayed on the EMS Page Frame to demonstrate a technique of executing code stored in expanded memory. See Part 2 of this series of articles for details of this process.
 Note: Saving pfseg is not necessary in this program, but it is a good practice. The code must be executed with an offset of 0 because one is entering an EXE program. *

```

push pfseg ;Save pfseg
add pfseg,400h ;Adjust pfseg to beginning of code
call DWORD PTR pfloff ;Call the far procedure
pop pfseg ;Restore the pfseg

```

COMMENT * The following code releases the handle and related logical pages (expanded memory). This operation is important because the Expanded Memory Manager will keep the handle open and all related expanded memory active even after the program is terminated unless the release function is called. With some hardware and EMS Version 4, expanded memory and the handle can be made nonvolatile which means the handle and memory remains active even after a warm boot (Ctrl-Alt-Del). Normally, a warm boot will release all handles and expanded memory. *

```

mov dx,handle ;Put handle number in dx
mov ah,45h ;Put Release Handle function number in ah
int 67h ;Release handle and allocated pages

```

```

xit:
mov ah,4Ch ;Call terminate function
int 21h

```

```

bindis PROC NEAR

```

COMMENT * This procedure converts a binary number in al to a pair of ASCII numbers in ax and displays these numbers on the standard console device. The registers ax, cx, and dx are used by this procedure, but their values are preserved. The Display Character function (INT 21h - Function 02h) and Display String function (INT 21h - Function 09h) are used. A line feed/carriage return string must be defined in the current data segment. The binary number in al must be less than 100. This procedure can be modified to handle larger numbers. *

```

push ax ;Save ax
push cx ;Save cx
push dx ;Save dx
aam ;Convert from binary to BCD
mov cl,al ;Save al
mov dl,ah ;Put high order digit in dl
add dl,48 ;Convert to ASCII
mov ah,02h ;Put Display Character function number in ah
int 21h ;Display character
mov dl,cl ;Put low order digit in dl
add dl,48 ;Convert to ASCII
mov ah,02h ;Put Display Character function number in ah
int 21h ;Display character
lea dx,lfcrr ;Put message address in dx
mov ah,09h ;Put Display String function number in ah
int 21h ;Call Display String
pop dx ;Restore dx
pop cx ;Restore cx
pop ax ;Restore ax
ret ;Pop instruction pointer

```

```

bindis ENDP

```

```

binhexdis PROC NEAR

```

Continued on Page 32



Stay Cool, Dude!

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

Not too many months ago, I replaced my Heath/Zenith H/Z-150 with a faster AT clone. The '150 was still a useful machine and it seemed wasteful to sell it for a pittance. The kids had some games and things they liked so I put the Zenith video card back in and dug out the amber monitor it had come with; I had taken the EGA card and monitor to the AT. With 640K, a game port and 20M hard drive I figured the Zenith would be something they could have fun with, I wouldn't cry too much if it went up in smoke, plus not only would it insulate my software from possible gremlins but I no longer needed to keep their stuff on my crowded hard disk.

I promptly moved all the games off my '286 machine onto the 150's disk. That gave me a little breathing room on my drive. Which, of course, I promptly filled back up with material I had stripped off when running short of room. The old axiom "You can never be too rich or too beautiful" now includes "Or have too large a fixed disk".

Activity on the Zenith seemed to drop off, until after replacing the original monochrome monitor on the H/Z-150 with a used CGA monitor, the kids started taking more interest in using the computer. I certainly cannot fault them for not liking the 'yellow' screen, because that is the first thing I wanted to change.

A friend at work suggested the youngsters might like a sound card and let me try out his Sound Blaster over the weekend. It was an instant hit. Nicole (5 years old) particularly liked the organ music player. She would select a tune, open a song book and sing along with Bradley (3) watching. The talking parrot sounded slow-witted running on the 8088, but the kids didn't complain. Sunday night, when I removed the borrowed card, there were two un-

happy faces watching. Monday I checked out Computer Shopper for the best prices and ordered a card.

Thursday found me installing our card into the Zenith; it was a newer version of the Sound Blaster then we had borrowed. But, horror of horrors, it wasn't working! It didn't work in the AT, either. The parrot talked, and a new program, included with this card talked, but the organ program would not run. More unhappy faces watching as I again removed the card. Fortunately, I had tried the first card in both machines and had no problems, so I was sure the problem was with the card.

While waiting for the card to be fixed, an update notice for Mother Goose and the original Kings Quest (by Sierra) arrived. Both now provide support for the Sound Blaster. Another phone call and another wait for UPS. The rub with the new games was that both need a CPU faster than the 8088 in the H/Z-150.

Of course, I could move the games back to my '286; but that meant the sound card had to be moved, too. And there was hardly any room left on my drive. The two games needed 4M of space on the disk and the sound card needs over 1M for its software. Since I was cramped for space anyway, I decided to add a second drive. To simplify matters, it would be identical with the first, a Seagate ST-296N SCSI. Surely 168 M of storage would hold things for some time. Although I am not totally enamored with the 296 drive, it is a very economical storage device.

(If anyone has any experience using a SCSI adapter other than the Seagate ST01/02 series with these drives, I would be interested in hearing about it. The drives are slower than expected and reformatting with different interleave factors has only made matters worse. The WD FASST se-

ries has been suggested, but they cost almost as much as the drive.)

With the increase in available storage the problem of backup needed to be addressed. Currently, I am using Fastback Plus for backup. I already had a ton of 1.2M floppy diskettes sitting around in the form of three backups to the 84M on the current drive. The thought of sitting and stuffing one hundred floppies in and out for a backup of two drives was not appetizing.

"Why bother with backups?" you ask. Scrimping on backups had almost cost me enormous effort when a ST-4096 had failed 18 months ago. It had been a month since any backup had been done and I never recovered some of the work. When a 40M IDE drive failed at work, a current backup cost me only the time to make the restore. So I was sold on keeping backups current. Tape backup seemed more convenient than floppy based backups and an economical way to go. I had experience with a DC-600 tape system on our Unix system. The friend with the sound card had a tape unit on his newest PC; it had proved reliable and easy to use when his new drive failed within 90 days of receipt. A Colorado Memory System unit, identical to my friend's unit and capable of 120M per tape was ordered.

While installing the second hard drive a most unpleasant thought crossed my mind. These drives run quite warm, almost hot. Stacked one above the other the bottom one could fry the top one! With two floppy drives, a tape drive, two fixed disks, the sound card, video card, three I/O cards and 4M of RAM, there was a lot of heat being dissipated inside the box and I was filling most of the air pockets. The increased temperatures could cause early failure of more than just the second drive. One option to remove the extra heat was

to replace the power supply with one having more fans. Several companies advertised AT power supplies designed for removing additional heat dissipation. This seemed the most expensive way and put a perfectly good power supply out of service.

Add a fan! I recalled seeing an advertisement for an add-in fan. It was for a full length card with two muffin fans powered from the computer buss. The cost was around \$75 and it took a whole slot. I had two slots left, so that was not a problem, but the \$75 was more than I wanted to put into the project. At this point I am already in for a hard drive, tape drive, tapes, sound card (and a mike) and updates to two games.

I had a fan which I had removed from the H/Z-150. I had used it when some heat related problems with the original Zenith power supply cropped up. It was no longer needed since the supply had been replaced. Alas, the 4" muffin fan was too big to fit inside the AT case. Not only too big but it needed 115 volt AC power. In the H/Z-150 I had cobbled together a fitting on the back to supply power, but that took a slot, looked messy and probably voided the Class B certification.

One slot in my '286 machine was designed for a special memory card and had an additional 62-pin socket along side the disk drive cage. The idea of a small 12 volt DC fan fastened to a circuit board stuck in this socket looked feasible.

Radio Shack obligingly stocks a 3-inch fan that requires 2 watts of 12 volt DC power (catalog number 273-243B). A small circuit board and the fan brought the cost up to less than \$25.

The circuit board was for a 72-pin socket and too tall but a couple whacks with a nibbling tool made it fit the 62-pin socket; a few more whacks along the top and it was short enough to fit in the case. I was careful when cutting down the socket end to keep the solder tabs aligned with the contacts in the socket. It didn't appear the tabs were wide enough to bridge two contacts, but I've been wrong before. Melting traces on the motherboard from shorts was not part of this project.

After drilling the circuit board for the mounting holes for the fan, I marked the cut out needed for the fan blades and made that with the nibbling tool. A socket and Y-adapter for the disk drive power cables was on hand from the '150 fan project and pressed into service again.

Photo 1 shows the fan mounted on the circuit card. The five contacts on the left

```
@echo off
:tapeback.bat
rem *G stands for the bell
if "%1"==" " echo Backing up ALL files *G
if "%1"==" " echo Do you need to ERASE the tape? *G
if "%1"==" " SET MESSAGE=_All_Files
if "%1"==" " goto OK
if "%1"==" /m" echo Backing up MODIFIED files *G
if "%1"==" /M" echo Backing up MODIFIED files *G
if "%1"==" /m" SET MESSAGE=_Modified_Files
if "%1"==" /M" SET MESSAGE=_Modified_Files
if "%1"==" /m" goto OK2
if "%1"==" /M" goto OK2
rem fall through if wrong flag
goto BADEND
:BADEND
echo Usage: %0 [/M] (will backup only modified files)
goto END
:OK
echo CTRL-Break to exit and erase tape
pause
:OK2
cls
rem First timestamp logfile and remove TSR programs
rem cr.dat contains only a carriage return to answer the prompt from date & time
date | c:\dos\find "Current" < c:\batch\cr.dat >> c:\util\tape\tapelog.txt
time | c:\dos\find "Current" < c:\batch\cr.dat >> c:\util\tape\tapelog.txt
echo Backing up %MESSAGE% >> c:\util\tape\tapelog.txt
C:
cd \f
fontres /off
cd \pcpanel
pcpanel /unload
cd \util\tape
if "%1"==" " echo *GInsert tape for Drive #1 . . . *G
if "%1"==" m" echo *GInsert tape for backing up %MESSAGE% . . . *G
if "%1"==" M" echo *GInsert tape for backing up %MESSAGE% . . . *G
pause
time | c:\dos\find "Current" < c:\batch\cr.dat >> c:\util\tape\tapelog.txt
rem First do drive 1
rem Backups are with compression and automatic comparison with disk
tape backup c:\.* %1/S/A/C/I/J/K/-P/T="Drive_1_Vol_1%MESSAGE%" >> c:\util\tape\tapelog.txt
tape backup e:\.* %1/S/A/C/I/J/K/-P/T="Drive_1_Vol_2%MESSAGE%" >> c:\util\tape\tapelog.txt
tape backup f:\.* %1/S/A/C/I/J/K/-P/T="Drive_1_Vol_3%MESSAGE%" >> c:\util\tape\tapelog.txt
time | c:\dos\find "Current" < c:\batch\cr.dat >> c:\util\tape\tapelog.txt
echo Drive 1 backup finished >> c:\util\tape\tapelog.txt
if "%1"==" " echo *GInsert tape for Drive #2 . . . *G
if "%1"==" " pause (continues)

rem no tape change needed if only modified file backup
time | c:\dos\find "Current" < c:\batch\cr.dat >> c:\util\tape\tapelog.txt
tape backup d:\.* %1/S/A/C/I/J/K/-P/T="Drive_2_Vol_1%MESSAGE%" >> c:\util\tape\tapelog.txt
tape backup g:\.* %1/S/A/C/I/J/K/-P/T="Drive_2_Vol_2%MESSAGE%" >> c:\util\tape\tapelog.txt
tape backup h:\.* %1/S/A/C/I/J/K/-P/T="Drive_2_Vol_3%MESSAGE%" >> c:\util\tape\tapelog.txt
date | c:\dos\find "Current" < c:\batch\cr.dat >> c:\util\tape\tapelog.txt
time | c:\dos\find "Current" < c:\batch\cr.dat >> c:\util\tape\tapelog.txt
echo Drive 2 backup finished >> c:\util\tape\tapelog.txt
echo ----- End of Backup ----- >> c:\util\tape\tapelog.txt
rem search tapelog file for error message line and display
grep -n "Error [0-9]" c:\util\tape\tapelog.txt
rem when problem corrected edit tapelog.txt and change "Error" to "error"
rem so that error message does not display next time
:END
```

Figure 1

end of the finger contacts, lower left in the photo, were removed. The object in the upper right is a brace to fasten to the drive cage.

If you are going to use one of the standard card slots, you may take power directly from the buss and omit the Y-adapter business. All the power connections are made on the B side of the connector. Ground is available on B1, B10 and B31. The +12 volt buss is on B9 (a -12 volt supply is available on B7).

No color code for the fan wiring was available; the rotation was marked on the case and red seemed a likely choice for the plus side. With a 50-50 chance, I was right the first time.

The acid test was how much added noise the fan was going to make. The previous PC I had at work had such a noisy fan I had to put it on the floor. When sitting on my desk the sound reflecting off the wall bounced right back at me and was very irritating.

I could tell very little difference in added noise from the additional fan but there was an annoying rattle. As a matter of fact, the whole cover was vibrating. I leave the cover screws out of my unit because hardly a month goes by that I don't need to do something inside. With the back unfastened the whole top of the case was able to make like a big sheet metal drum. For a quick fix I put the modem and print buffer

I could tell very little difference in added noise from the additional fan but there was an annoying rattle. As a matter of fact, the whole cover was vibrating. I leave the cover screws out of my unit because hardly a month goes by that I don't need to do something inside. With the back unfastened the whole top of the case was able to make like a big sheet metal drum. For a quick fix I put the modem and print buffer

I could tell very little difference in added noise from the additional fan but there was an annoying rattle. As a matter of fact, the whole cover was vibrating. I leave the cover screws out of my unit because hardly a month goes by that I don't need to do something inside. With the back unfastened the whole top of the case was able to make like a big sheet metal drum. For a quick fix I put the modem and print buffer

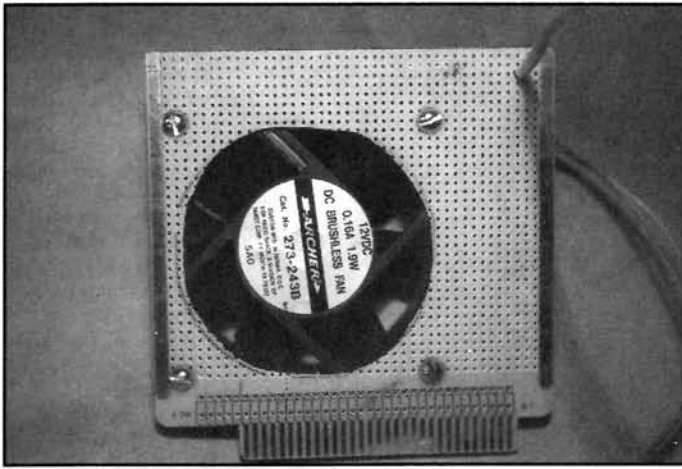


Photo 1

on top of the case. Later, I ran a brace from the top of the fan card to the drive cage.

Photo 2 is the fan and card mounted in the AT and shows the other side of the assembly. The drive cage for the AT is just behind the fan (the sound card is the one sticking out just to the left of the fan). The fan card is short enough to clear a half-length card in the normal slot but not a three-fourth-length card, like the sound card.

After several hours of running, I removed

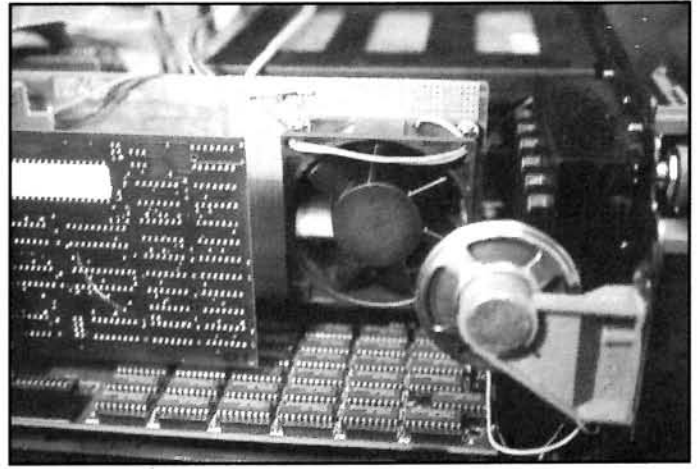


Photo 2

the cover and checked the two hard drives. The top drive, the one I was afraid of cooking, is cooler than the single drive ever was. I have no real reference point for temperature in the remainder of the PC, however nothing seems to be warm to the touch which is encouraging. Since heat more than anything else can cause early failure of equipment, I believe I've given this PC a shot of the 'fountain of youth'.

Initially the backup fit on one tape and a small batch file automated doing the six

partitions. Later (not much later) the data overflowed one tape and I now put one drive per tape which requires a tape change for a complete backup. To keep track of what happens during the backup, I have the batch file write to a log file all output from the backup program that came with the CMS tape drive. Figure 1 is the batch file. In Figure 1, the seven lines that end with the double right redirect symbol (>>) are continued on the following line. Both lines should be typed as one. *



Power Supplies
Complete Board Assemblies

SURPLUS COMPUTER PARTS

20 Meg. Hard Drives
From \$5.00

- o Largest Surplus Electronics Dealer in Western Michigan
- o In Business Over 40 Years
- o Over 12,500 square foot Store and Warehouse

Examples of Zenith Data Systems Salvage Products:

386-16 1 Meg. Memory Board	190.00
Z-100 Power Supplies	From 15.00
Z-100 Winchester Controller & Data Separators	Ea. 35.00
Z-449 Video Boards	75.00
AT Dual Controller Board	115.00
Laptop Keyboards	From 5.00
20 Meg. Tape Backups	From 50.00
8087 & 80287-6 Co Processors	75.00
Numeric Keypads for Laptops	20.00
Laptop Modems	From 5.00
PC & AT 84-key Keyboards - Used	From 5.00
Z-148 Power Supplies	15.00
Zenith Data Systems 101-key Keyboards	From 10.00
286 Laptop Motherboards	500.00
Power Supply Fans	From 5.00
40 Meg. Hard Drives for 286 SuperPort	300.00
Laptop Replacement Batteries	From 6.00

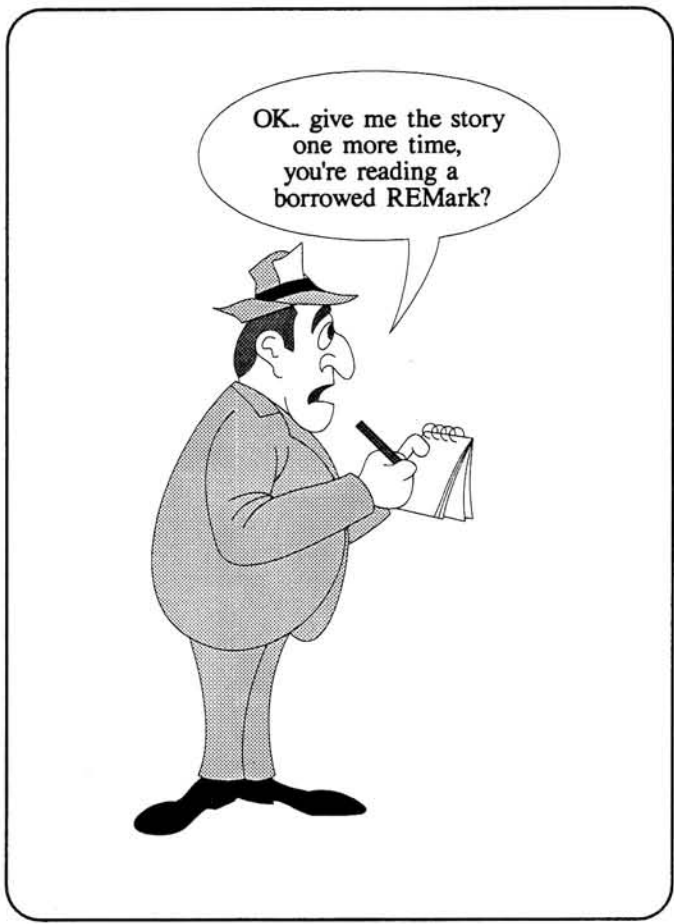
Many Other Zenith Data Systems Salvage Parts Available
All Surplus/Salvage Sold AS IS - WHERE IS
- NO GUARANTEE -

No Catalog - Please Call for Quotes
We Ship U.P.S.-C.O.D. (ONLY) Sorry - No Foreign Shipments

Surplus Trading Corp.

THE HOUSE OF EVERYTHING "ALMOST"
2700 N. M-63, P.O. BOX 1082
BENTON HARBOR, MI 49022

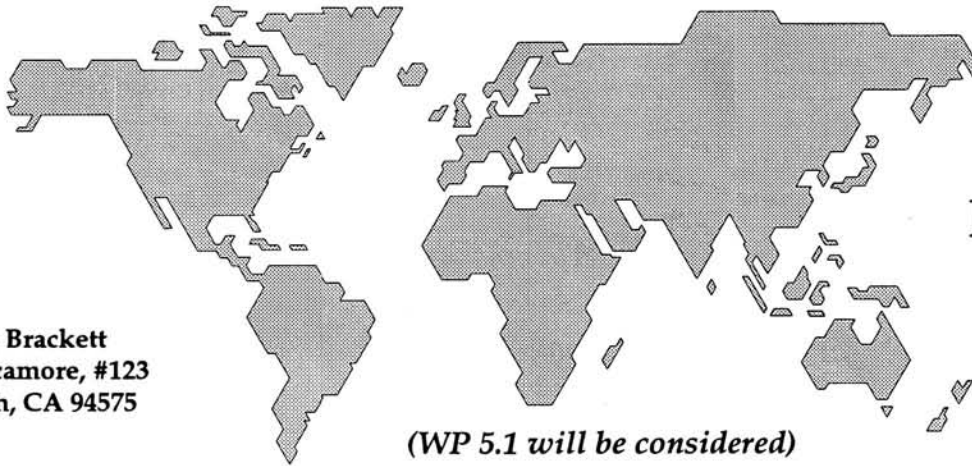
CALL: LES - (616) 849-1800
FAX - (616) 849-2995 (Orders Only)



Reader Service #153



The World of WP50 and Its Wonders



Part IV

Salli Brackett
2201 Sycamore, #123
Antioch, CA 94575

(WP 5.1 will be considered)

The column function is a real asset in handling many columnar situations, but there can be some confusions when using it. The following article explains how the column function works and gives some tips on using it effectively. To use the method in this article, you need to change the 'Pos' display on your status line to units of measure (columns and lines). Press Shift F1, u (in WP5.1, press e, u), d, u, s, u, F7.

First, you must define the columns. There are two types of columns, newspaper and parallel. With newspaper columns, the text you type wraps to the next column. You would use this if you wanted the text to read like a 'newspaper article.' With parallel columns the text is side-by-side on a column, like the sample below. The parallel with block protect keeps both parallel columns together at the end of a page even if there are not the same length. I have found problems with this feature, so this article will concentrated on just parallel without the block protect. Below is a description of the definition menu. Read the descriptions; following will be an explanation of how best to enter data in columns, using these definitions as a sample.



Feature

Type

Number of Columns

Distance Between Columns

Margins

Function

There are two types of columns, newspaper and parallel.

You can create as many columns as the paper will hold. WordPerfect will automatically set the column margins based on the number of columns, and distance between columns, and the left/right margin.

The default distance between columns is 5. This can be changed.

The margins can be changed to fit specific columns. Left refers to the beginning of the column and right refers to the end of the column. For example: You want your first column to be 20 spaces and the second column to be 50. For the first column: the left margin would be the left margin of the paper, 11 and the right margin would be 31. Using the default distance between columns (5), the left margin of the second column would be 36 and the right margin would be 86. These figures are based on a Helvetica 12 point font.

Below is a description of how to define and create columns using the sample above. If you have any problems entering this data, check page 3.

Application

To Define Columns: (This is ONLY a definition, it can be at the beginning of your document. You only need to *define* the columns once, then you can turn them on and off throughout the document.)

To Begin Document

To Turn the Columns ON

To Create Columns

Required Keystrokes

Press Alt F7, c(column) d(efine)

Press t(ype)

Press p(arallel)

In this sample, we will use the default number of columns and margins.

Press F7, twice

Type and center the phrase: This is a list for columns. Press ENTER twice.

Press ALT F7

Press c(column on/off) (5.1 - press

c(column), o(n)

Type the first column heading from the sample, Feature.

Press ENTER twice

To create the 2nd column, press Ctrl ENTER. (A hard page code [Hpg+] will appear.

Type the 2nd column heading, Function.

Press ENTER twice

Then turn the columns off by pressing Alt F7, c(column)

To Enter Data

on/off) (5.1 - press c(olumn), f(off). You *always* turn your columns off at your last column. I recommend typing in the longest column first, then placing the short column in place.

To Move from Column to Column

Type all the paragraph's in the second column of the sample starting with 'There are two types...' and ending with 'This is based on...'. Use the GOTO key (Ctrl End, left or right arrow keys).

Return to the first column to enter the data in column one by pressing GOTO, left arrow.

Cursor down two and start typing.

The secret to working with columns is to use the GOTO key and be careful what you delete. I suggest putting on the reveal codes before deleting anything so that you won't delete the hard page code. If you cursor too far, you will end up in the next column. Remember that hard page code delineates columns.

The information below has tips on problems that may occur. If you delete the hard page code, your screen flips out.

Replace the hard page code and your columns are restored.

If you need to go to another page and your information breaks improperly

Press ENTER until the beginning of the information is on the next page.

Make sure that each column has the same number of lines. In other words, if data in column one stops before the data in column two and you go to page two, your page break line will look like this:

If you put carriage returns in column one to make up for the missing lines you will have:

Occasionally this method doesn't give you the desired results with the page break line. If you still have:

check, for each column, the line number from your status line that the data on the second page is on. If they are on the same line, they will print properly, even though the page break line is not connected.

When you turn the columns off at your last column, your text will return to normal.

Cursor to the right of the [Col Def:] code
Press F7, c(olumn), d(efine), m(argin)
Press ENTER to go to the right margin of the 1st column.

Based on the Helvetica 12 point, Type size 36.
Press ENTER to go to the left margin of the 2nd column.

Type 41 (the distance between columns is 5).
Press F7 twice.
Delete the old [Col Def] code.

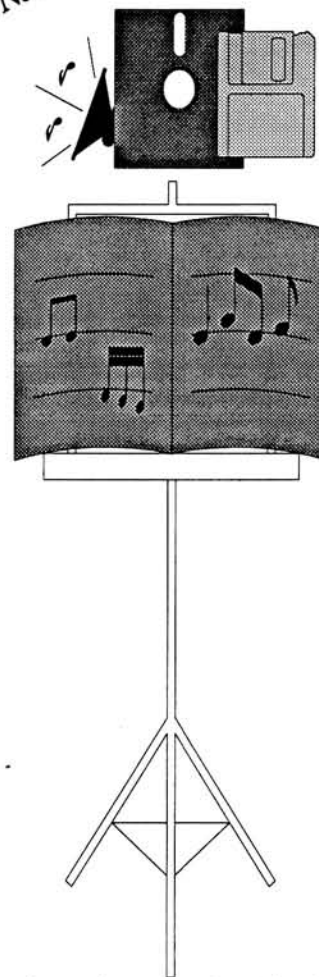
Always check at the beginning of the column. Particularly if you have more than one page, the misalignment may have occurred further up on the page and does not show on the screen.

Even though the manual will give you another method of creating columns, I find this the best method for input. It also has advantages when you need to move or edit. The key factor is to remember the *Hard Page Code* is now the column delineation NOT a page delineation. If you are going to use columns frequently, I also recommend creating two macros for the cursor movement between columns. I found that Alt Z and Alt A work well, because they enable you to use the macro with one hand without losing your place on the keyboard (a real advantage for typists).

I recommend using the column function for similar text as the sample. For columnar occurrences using numbers, use tabs. My next article will involve setting up tabs. A few tricks I have learned will help take the frustration and confusion out of using tabs.

✱

New Software Product!

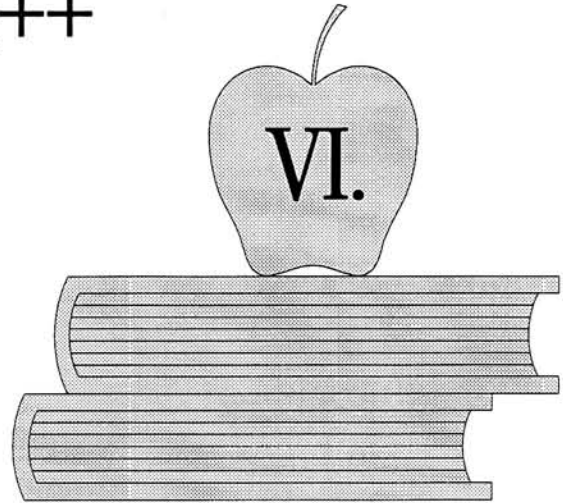


The Electronic Clavier
P/N 885-6016

Introduction to C++

Sixth Installment

Lynwood H. Wilson
2160 James Canyon
Boulder, CO 80302



Since last month I have written a data entry program of about 2000 lines in C++. This is my first program bigger than a few pages, and everything went quite well. I could have written it in straight C in perhaps 20% less time, but it wouldn't have been as easy a program to modify and debug. The customer needed a bunch of changes after it was running and that part went a bit faster than it would have with a similar program in C. When I have gathered more experience with C++ I expect the time saving to be substantial. I also expect to be able to reuse the objects I created next time I need a data entry routine. This will be the big time saver.

Most of the details of the language are such logical and reasonable extensions of C that those of you with C experience will have little difficulty with them. For those of you without, it's no harder to learn than any other language. The difficult part, perhaps more difficult for those of us with experience, is the new paradigm. The whole idea of objects, of combining code with data, could be a problem. However I didn't find it so and others have said the same. Object oriented programming may be more like the way we represent problems and their solutions in our minds than ordinary procedural programming and that could explain the ease with which many people seem to adopt it.

I will have more to say about this as I accumulate more experience. After all 2000 lines is not a very large program, and I have a lot more to learn. In the meantime let's get on with some more of the fundamentals of the language.

Resources

The Waite Group's C Programming using Turbo C++ by Robert Lafore is a good book for learning C and it includes a short introduction to the added C++ fea-

tures. This book is very specific to this compiler and the PC compatible hardware, which is a very good thing if that is what you are using. There is excellent coverage of things like the ROM BIOS, the Turbo graphics library, dealing directly with screen memory and the keyboard, and the various memory models. I highly recommend this book if you have no C background, even though its coverage of C++ is limited. Read this one first and then get one of the books which is heavier on C++.

I have used two previous editions of this book as texts in the C programming class I teach, and I will be using this one next semester.

Decisions

Most programs are full of decisions. We have seen one kind of decision as part of the lesson on loops. The loop constructs have to decide whether to execute the loop or quit every time around. Now we are going to look at some more general decisions.

IF

The if in C++ (and C) is quite similar to most other languages. I have used it several times and you probably didn't even notice that I hadn't formally introduced it. One obvious difference from some other languages is that C++ does not use the keyword then. Here is an example:

```
#include <iostream.h>

main()
{
    char ch;

    cout << "Enter Y or N: ";
    cin >> ch;
    if(ch == 'Y')
        cout << "It's a Y!";
}
```

Note the similarity to the while. The if is exactly like a while which only runs

once or not at all.

The keyword if is followed by a conditional expression in parentheses. The conditional expression contains a relational operator, the double equals. This expression will evaluate to 1 (true) if ch and 'Y' are the same, and 0 (false) if they are not.

The next line is the body of the if, and is indented because it is controlled by the line above. The body of the if is executed if the conditional expression evaluates to true. If not the body is skipped.

Here is another way of doing the same thing. Here I have used an old C input function called getch() to get the character from the keyboard. Fetching the data within the conditional expression like this is quite common. Note that it saves a line of code and a variable.

```
#include <iostream.h>
#include <conio.h>
```

```
main()
{
    cout << "Enter Y or N: ";
    if(getche() == 'Y')
        cout << "It's a Y!";
}
```

This does not do exactly what the first version does since the stream input statement `cin >> ch;` sends a carriage return to the screen and `getche()` does not.

Note the addition of `conio.h` for the prototype of `getche()`.

Note that there is no semicolon at the end of the line which begins with if. The whole if construct including the body appears from outside to be one single statement and takes a semicolon at the end as above.

If you put a semicolon at the end of the line which begins with if the if construct then has a null body (an empty statement for its body) and the next line which we intended to be the body is not controlled by the if and is executed in all cases.

This is a very easy mistake to make,

and a very hard mistake to see. Remember not to be fooled by your own indenting.

The if construct expects a single statement for its body just as the for construct does. If you need more statements in the body you enclose them in curly braces, just as in the case of the for. Like this:

```
#include <iostream.h>

main()
{
    char ch;

    cout << "Enter Y or N: ";
    cin >> ch;
    if(ch == 'Y') {
        cout << "It's a Y!";
        cout << "\nSecond statement.";
    }
}
```

In this example both of the statements in the curly braces are executed if the conditional expression evaluates to true, and not otherwise. The whole thing still looks like a single statement from outside.

To my mind the syntax would be more regular if it required a semicolon after the closing curly brace, but it doesn't.

Nested IF

An if statement can go anywhere any other statement can, even inside the body of an if. Watch this:

```
#include <iostream.h>
#include <conio.h>

main()
{
    char ch;

    cout << "Enter Y or N: ";
    cin >> ch;
    if(ch == 'Y')
        if(getche() == 'e') {
            cout << "It's a Ye!";
            cout << "\nSecond statement.";
        }
}
```

The body of the first if does not need to be enclosed in curly braces even though it is four lines long because the second if looks like one statement from outside no matter how long its body. However, if we have another statement in the body of the first if it will need the curly braces.

```
#include <iostream.h>

main()
{
    char ch;

    cout << "Enter Y or N: ";
    cin >> ch;
    if(ch == 'Y') {
        cin >> ch;
        if(ch == 'e') {
            cout << "It's a Ye!";
            cout << "\nSecond statement.";
        }
    }
}
```

Once again this program will behave slightly differently than the last because of differences between the input routines.

The body of the first if now contains 2 statements, and so requires the curly

braces.

IF ELSE

Well that's all very well, but what if you want to do one thing if a condition is true and a different thing if it's false. You could do it like this:

```
if(something)
    one thing;
if(!something)
    another thing;
```

and it would work fine, but the second if seems like more control structure than we should need. And sure enough, there is the if else.

```
#include <iostream.h>

main()
{
    char ch;

    cout << "Enter Y or N: ";
    cin >> ch;
    if(ch == 'Y')
        cout << "It's a Y!";
    else
        cout << "It's not.";
```

The if first evaluates its conditional expression. If it is true, the if executes the line under the if. If it is false, it executes the line under the else. It cannot execute both under any circumstances.

The body of the else can be several lines enclosed in curly braces, just as the body of the if can. The else should be indented to match the if it belongs to. Which brings up another good point.

In more complex cases it is not always simple to see which if an else belongs to. The rule is simple, but sometimes its application is less than obvious. The rule is:

An else goes with the nearest if above it which has no else.

```
if(a)
    if(b)
        x
    else
        y
```

In this case, x is executed if a and b are true and y is executed if a is true and b is not. The else belongs to the second if. It doesn't matter in the slightest if you indent it like this:

```
if(a)
    if(b)
        x
    else
        y
```

If you want y to be executed if a is false, that is you want the else to belong to the first if, you must hide the second if within a pair of curly braces like this:

```
if(a) {
    if(b)
        x
}
else
    y
```

An else will not associate itself with an if above it which is inside curly braces unless the else is inside the same curly braces.

If elses can be nested in another way. This format is often used to select an action from many possibilities.

```
#include <iostream.h>

main()
{
    char ch;

    cout << "Enter a letter: ";
    cin >> ch;
    if(ch == 'Y')
        cout << "It's a Y!";
    else
        if(ch == 'Z')
            cout << "It's a Z!";
        else
            if(ch == 'A')
                cout << "It's an A!";
            else
                if(ch == 'B')
                    cout << "It's a B!";
                else
                    cout << "?";
}
```

This program works its way down the ifs until it finds one of the conditional expressions true and then executes the body of that if. Then it falls out the bottom of the mass of if elses and executes the next statement. In this case, the program ends. If none of the ifs are true the body of the last else is executed.

This is a common and useful construction but it is seldom formatted like the example above. As you can see, this one will indent itself off the right side of the screen with a few more cases. This does not matter to the compiler but it can make the program harder to read, particularly after it has been printed. So this form has been adopted.

```
#include <iostream.h>

main()
{
    char ch;

    cout << "Enter a letter: ";
    cin >> ch;
    if(ch == 'Y')
        cout << "It's a Y!";
    else if(ch == 'Z')
        cout << "It's a Z!";
    else if(ch == 'A')
        cout << "It's an A!";
    else if(ch == 'B')
        cout << "It's a B!";
    else
        cout << "?";
}
```

This form saves a few lines, in addition to preventing the code from growing wider. Perhaps it's habit, but I find it easier to read.

Note that each else in this construction belongs to the if directly above it. Thus the code is easy to read even though the elses are not indented to match their ifs.

Should you need a null case, a case in which nothing is to be done and so the if needs no body, it should be written like this:

```
#include <iostream.h>

main()
{
    char ch;

    cout << "Enter a letter: ";
    cin >> ch;
```



```

if(ch == 'Y')
    cout << "It's a Y!";
else if(ch == 'Z')
    cout << "It's a Z!";
else if(ch == 'A')
    cout << "It's an A!";
else if(ch == 'B')
    cout << "It's a B!";
else if(ch == 'Q')
    ; // do nothing
else
    cout << "?";
}

```

This helps preserve the symmetry and readability of the code.

The bodies of the ifs and elses can be multiple statements in curly braces, as always.

Break & Continue

Break and continue are control statements which are used with loops, and in the case of break, with the switch statement.

Break takes you out of the innermost loop immediately without reference to the loop control. The statement following the loop is executed next just as if the loop had ended normally.

This can be useful in dealing with an unexpected case which requires the immediate termination of the loop. It is usually reserved for a complex situation in which putting the rest of the loop code inside an if to accomplish the same thing would be less obvious.

Both theoretically and practically it has been found that a segment of code is easier to understand, debug, and modify if it has one entry point and one exit point. A break is a second exit from the segment of code which comprises the loop being broken out of. Therefore I tend to resist the use of break (and continue, for the same reason) unless the code is obviously clearer with it than without.

The continue statement causes the program to skip the rest of the code in the body of a loop and jump to the loop control statement which is executed next.

Here is a simple example.

```

#include <iostream.h>
#include <ctype.h>

main()
{
    char ch;
    while(1) {
        cin >> ch;
        if(ch == 'Q')
            break;
        else if (!isalpha(ch))
            continue;
        else if(ch == 'Y')
            cout << "It's a Y!";
        else if(ch == 'Z')
            cout << "It's a Z!";
        else
            cout << "?";
    }
    cout << "\nDone at last!";
}

```

While(1) is a common way to create an endless loop, as 1 will always be true. The break statement is a common way of terminating what would otherwise be an

endless loop.

The loop runs until the character from the keyboard is a Q and then the break is executed. The rest of the loop is skipped and the next code executed is the line which prints "Done at last."

If the character is not a letter the continue is executed and the program jumps to the loop control statement, the while(1), skipping the rest of the code in the loop.

The library function isalpha() is one of a number of classification functions which return true or false depending on the nature of the argument. This one returns true if its argument is a letter, either upper or lower case, and false otherwise. Thus if the input is not a letter the continue statement is executed which causes the program to skip the rest of the loop and jump to the while(1) statement.

Switch

The switch construct is similar to the case statement used in some other languages. It is a method of choosing between several actions based on the value of a single variable.

```
#include <iostream.h>
```

```

main()
{
    char ch;

    while(1) {
        cin >> ch;
        if(ch == 'Q')
            break;

        switch(ch) {
            case 'Y':
                cout << "It's a Y!";
                break;
            case 'Z':
                cout << "It's a Z!";
                break;
            default:
                cout << "?";
        }
    }
    cout << "\nDone at last!";
}

```

The switch(ch) statement determines what variable controls the selection. Each of the case statements will cause the code which follows it to be executed if the control variable, ch in this case, equals the constant which follows the case. In this example, if ch is Y then the program prints "It's a Y!".

The break causes an exit from the switch just like it does in the case of a loop. This is necessary because otherwise all the code following the case statement which corresponds to the value of the variable will be executed.

For example if we left both of the breaks out of the switch in the above program and enter a 'Y' it will execute all the code from the case 'Y'; statement to the end of the switch and print "It's a Y!!t's a Z!?". In this situation the rest of the case statements are ignored.

Note also that you can have multiple statements after a case statement without curly braces. This seems more regular if you think of the case statements as conditional labels. The program jumps from the switch statement to the first of the case labels which is true and executes from there on, ignoring any other case statements.

This means that almost all switches have breaks in them so that they will execute only the code associated with one case.

Sometimes it is useful to be able to execute one block of code for several values of the control variable.

```
#include <iostream.h>
#include <ctype.h>
```

```

main()
{
    char ch;

    while(1) {
        cin >> ch;
        if(ch == 'Q')
            break;

        switch(ch) {
            case 'Y':
                cout << "It's a Y!";
                break;
            case 'Z':
                cout << "It's a Z!";
                break;
            default:
                cout << "?";
        }
    }
    cout << "\nDone at last!";
}

```

This version of the example will identify Y and Z in upper or lower case.

The code after the default: is executed if none of the other cases are true. It is optional.

The switch statement is useful, but limited. The control variable must be a statement which produces an integral value. The values which the case statements test it against must be constants. There is no way to test for a range of values except to list them all. If you have a selection problem which the switch will not handle, you can always fall back on the else if construction which we discussed earlier.

Conditional Operator

The conditional operator is a short-handed way of choosing one of two values. It is unusual in that it has three operands.

```
#include <iostream.h>
```

```

main()
{
    int x, y, z;

    cin >> x >> y;
    z = x > y? x: y; // conditional
    cout << "The larger is " << z << ".\n";
}

```

First the expression $x > y$ is evaluated. If it is true (non-zero) then the conditional returns the value of the second expression

(in this case, x) which is assigned to z. If the first expression is false (zero), then the conditional returns the third expression (in this case, y).

Note that this is exactly the same as saying:

```
if(x > y)
    z = x;
else
    z = y;
```

The conditional saves 3 lines of code, in this case.

Let's look at this another way. Here is the simplest form of the conditional expression.

```
a? b: c
```

First, a is evaluated. If true, the value of the whole expression is b. If false, the value of the expression is c.

As with other expressions, the conditional can be used anywhere a variable or constant of the same data type can be used. The above example could be written like this:

The conditional is not used frequently, #include <iostream.h>

```
main()
{
    int x, y;

    cin >> x >> y;
    cout << "The larger is " <<
        (x > y? x: y) << '.';
}
```

but when it is called for, nothing else is quite right.

It is woefully easy to forget features of the language which you do not often use. Even with a language as small and terse as C++. For this reason, I try to re-read one of the books on the language at least once a year.

I find it even easier to forget library functions. It's not so bad when you just forget the syntax and have to look it up, but when you forget the existence of a function that you need and write a new version of it, that wastes even more time. The library manuals don't make very interesting reading, but I try to go through the ones I am currently using once a year, too.

Sources

C Programming Using Turbo C++

Robert Lafore, Howard W. Sams, 1990 *

Continued from Page 22

COMMENT * This procedure converts a binary number in register al to a hexadecimal number pair in ax. These hexadecimal numbers are then converted to ASCII characters and displayed on the standard console. This procedure uses the ax, cx, and dx registers. Data in these registers are preserved. *

```

push    ax          ;Save ax
push    cx          ;Save cx
push    dx          ;Save dx
xor     ah,ah       ;Clear ah
mov     cl,4        ;Put number of bits to shift in cl
shl    ax,cl        ;Shift ax left four bits
mov     cl,4        ;Put number of bits to shift in cl
shr    al,cl        ;Shift al right four bits
cmp     ah,10       ;Compare ah to 10
jb     nex1         ;If below 10, check al
add     ah,7        ;Else, add 7 to offset into ASCII letters
nex1:   nop
        cmp     al,10       ;Compare al to 10
        jb     nex2         ;If below, go to ASCII conversion
        add     al,7        ;Else, add 7 to offset into ASCII letters
nex2:   nop
        add     al,48       ;Convert to ASCII
        add     ah,48       ;Convert to ASCII
        mov     cl,al       ;Save al
        mov     dl,ah       ;Put most significant hexadigit in dl
        mov     ah,02h     ;Put Display Character function number in ah
        int     21h        ;Display character
        mov     dl,cl       ;Put least significant hexadigit in dl
        int     21h        ;Call Display Character function
        pop     dx          ;Restore dx
        pop     cx          ;Restore cx
        pop     ax          ;Restore ax
        ret                ;Pop instruction pointer
binhexdis    ENDP

```

```
code    ENDS
```

```
stack  SEGMENT stack          ;Program stack segment
        DW     64 DUP(?)      ;Define stack space
stack  ENDS
```

```
pgmend SEGMENT                ;Dummy segment to mark program end
pgmend ENDS
```

```
END     start                ;Mark end and define start
```

COMMENT * OVERLAY3.OVL

The following procedure displays a message and returns to the calling program. The message to be displayed is in the procedure's code segment. Therefore, the data segment is made identical to the code segment so that the Display String function can access the message. Of course, the original data segment address must be saved before the data segment address is changed and restored before the procedure returns control to the root program. *

```

code    SEGMENT
over3   PROC     FAR
        push    ds          ;Save data segment address
        mov     ax,cs       ;Put code segment address in ax
        mov     ds,ax       ;Make data segment the same as code segment
        lea     dx,messa    ;Put the message offset address in dx
        mov     ah,09h     ;Put Display String function number in ah
        int     21h        ;Call Display String function
        pop     ds          ;Restore data segment address
        ret                ;Return to the FAR code
messa   DB     '***This message is displayed by the procedure.',10,13,'$'
over3   ENDP
code    ENDS
END

```

Search for the Perfect Word Processor

Laszlo M. Vesei
420 N. Philip Rd.
Niles, MI. 49120

YAWP, Yet Another Word Processor. Yes, a full-featured and humane word processor!

My first word processor, Magic Wand, came bundled with ye olde H-89. It was a very good one at the time. I experimented briefly with another one for this machine, but never liked it. Mainly, it was much too slow, and too expensive to boot.

My next machine, an H-161 came bundled with "sWordplay" (not its real name, I can do without a defamation suit). Unfortunately, it was an "unnatural one". Unnatural means that the various commands defy any natural inclination of the human intellect, therefore difficult to learn and only too easy to screw up. (See INFO WORLD magazine, November 19, 1990 issue, article titled "The X Factor". It is about humanizing program interfaces). Therefore I opted for the successor of Magic Wand, called Peachtext, mainly because of my previous experience with the Wand.

Being multilingual, my work increasingly included foreign correspondence which needed the upper (above decimal 127) ASCII characters. To be sure, Peachtext can produce these, but only with heroic effort, using the \out\ command. Furthermore, the umlaut letters thus produced were "exceptional", that is, they did not obey the normal formatting rules. "sWordplay" handles the Umlaut letters okay, but, as mentioned before, it is inhuman, unnatural and unwieldy. Sooo... The search was on for the Perfect Word Processor.

First came the various upgrades of "sWordplay". They were, for the registered previous user, relatively inexpensive, but each upgrade became more unwieldy and more unnatural than the previous one. A redeeming fact was that version 4.xx has a marvelous Thesaurus, which is, quite as expected, missing from the subsequent upgrades. This version includes also a skeleton printer driver. It seemed to be handy, as none of my three printers had a ready-made driver on the "sWordplay" list. Unfortunately, it was also "unnatural", mean-

ing that I did not succeed to build a usable printer driver. Am I perhaps subhuman?

I tried two more Big Name Processors (with Big Name prices). Unnatural ones, full of unneeded bells and whistles. They did not fit into the memory of my machine (not then, anyway). After upgrading the memory system with FBE Research's memory stretcher I tried again without any significant success. Next came about a dozen shareware word processors. The only good thing to say about these is that their disks could be reused for something more worthwhile, albeit it would be cheaper to buy blank disks.

Not long ago I received a postcard from Ann Arbor, extolling the virtues of EXTRA, YAWP (Yet Another Word Processor). I read it with some skepticism due to all the experiences with both, cheap and expensive word processors. Yet the people sending the card offered to loan a complete word processor on approval. The price, about 100 Clams (plus the Michigan nuisance tax for Michiganders), was to be paid only if I liked what I saw, otherwise just return the package and owe nothing. This was an offer I could not refuse. Apparently they still possessed their trust in humanity. In due time the package came, three 5.25" disks. Unfortunately, my machine can use only 3.5" disks. A quick phone call and a trip to the post office plus a few days waiting fixed that. While waiting I studied the printed, bound instruction book. (Yes, a printed BOOK!). Strangely enough, the instruction book is written in plain English for plain people, not by expert programmers to impress other expert programmers. I much liked what I saw.

Meanwhile the new (and this time the correct) disks arrived. The trial package consisted of three 3.5", 720K disks. This package did not include the Thesaurus, which will come only if you decide to keep EXTRA and send payment. Later I learned, that the real disks come after payment in full. These real disks contain not only the Thesaurus but also some

improvements on the other files. A recipe sheet comes with these disks, explaining what of the old files must be replaced and how to go about it. These instructions are easy to follow. As far as "look and feel" is concerned I did not see any difference between the trial and real disks.

The #1 law of computerdom: "Thou shalt make a copy" did not quite work with the <Copy *.*> command. Ach, some kind of copy protection? However <diskcopy> did the trick. The <dir> command revealed a lot of files on the disk, more about these later. Installing the program on my hard disk was easy as a pie. Evoking the Install file on the disk will copy the necessary files (now you have a choice to make) either to your hard disk or one of the floppies. Appendix B in the instruction book explains the choices. Be ready for some disk swapping. The absolute minimum to run the program consists of two files: EXTRA.EXE and EXTRA.DRV. These two will fit even on a 360 K disk with some place left for saving your work. I find it advisable to add the MACRO.DRV file as well, if one is created, also some often used files, such as a letterhead. Now the proof of the pudding.

Upon entering the command <EXTRA> a menu window appears on the screen. It gives 5 choices: Save, Print, Create, Retrieve, or Exit to DOS. If you choose a hard disk as base instead of a floppy, you will have a lot of sample files on your disk, as mentioned before. There are letterheads, commercial and legal documents, even a skeleton resume. Lazy people (which includes me) can select one for the occasion and change it as needed. A word of caution when saving the changed one: If you want to preserve the original DO NOT USE THE SAVE COMMAND from the first (window) list! Rename and save using option N from the Saving Options list, produced by hitting Shift+F10.

At this point I decided to retrieve an existing document. Hitting <R> produced a list on screen, as well as some choices regarding what to do with these. I decided

to try "printst.doc" and "chartest.doc", as I wanted to try the printer interface. The first thing to do after loading is to install the printer. My printer is on the TEXTRA list, so I loaded it and printed these files. It worked fine as long as the text was English. The upper ASCII characters, however, did not work. At this time I called their tech support. They answered the phone on the first ring. The person answering was courteous and knowledgeable. He suggested, since my dot matrix printer can emulate several IBM and Epson models, try installing the "IBM Graphics printer" or one of the "IBM proprinters" instead of mine. Both of these worked fine with my printer but the "Proprinter" has more choice of fonts. Another occasion to call arose when I tried to use my daisy-wheel printer. It obeys Diablo commands, but when I installed any of the Diablo drivers it refused to print. They suggested to use an "Unknown ASCII Printer" and that did the trick. Naturally, daisy-wheel printers can not print characters which are not residing on the wheel. My daisy-wheel printer therefore is restricted to English text.

Talking about printer drivers: there are drivers for dozens of popular (and less than popular) printers, including mine. Unseriner (Look it up in a German dictionary) cannot afford laser printers, but their drivers are there. For laser printers this program also sports some downloadable fonts. Graphics? No problem. More than one printer? You can install four different ones and select any of these. Also it is surprisingly easy to install a new printer. You have a color printer? TEXTRA can print in color too, but it is a bit slower.

We come now to actually processing words. TEXTRA awakens in insert mode. (For those unfamiliar with this term it means that if you hit a letter in the middle of a word that letter will be squeezed into the word in front of the cursor and the entire row will be lengthened). If you prefer overtype mode, just hit the <Ins> key. (In this mode any letter you hit in the middle of the text will replace the character at the cursor). This key is a toggle, meaning that it changes from overtype into insert mode or back. The shape of the cursor shows the mode: it is a small underline in insert mode and a halfblock in replace mode. TEXTRA will take care of carriage returns, as any self-respecting word processor should. The "Return" key ends a paragraph, as usual. The editing commands are natural, meaning that the combinations of keystrokes are simple and logical, even intuitive. The Help command is always only a keystroke away. Furthermore, it is genuinely helpful. There is also a list of commands at the bottom of the screen. These are understandable and also intuitive, Praise be! There is even an Oops! feature: undelete things if the wrong button is hit accidentally. Additionally, all commands can be cancelled before being

committed for keeps. Macros? For not being an expert, in a few minutes I was able to create a goodly number of them, for the Umlaut letters. Creating indented or hanging paragraphs is a snap. Selecting the fonts is also easy. If you have a color monitor the different fonts are shown in different colors. TEXTRA knows the fonts your printer can produce and your selection is restricted to these, naturally.

TEXTRA is a WYSIWYG (What You See Is What You Get) program, that is, the screen looks much as your finished document will look, well, almost. Paragraphs on screen are hanging, indented or plain, and they will print, as you set them. There are two caveats, however:

1. The key combination <control + t> changes the looks of the screen. The program awakens in 25-line, 80-column mode. This key combination, which is a toggle, changes the screen into a 43-line, 80-column mode, or back. (It happened to me by accident and scared me witless).
2. Formatting signs show up on the screen, but these do not print.

If you want to see the finished page, even two pages, side-by-side on screen, there are commands for that. There are many advanced editing features: Style sheets, headers and footers, odd-and-even page, remapping the keyboard, using both sides of the paper (neat trick), mail merge, multiple column formats, search-and-replace, speller, Thesaurus, split screens, name address book, even a phone dialer. There are ways to generate and fill out forms, such as the infamous (Horrors!) Form 1040. Actually, there are so many features that it is difficult to list them all. As you can see this is a full featured word processor. In case of difficulties the TEXTRA people have an online help phone number. I've found it uncrowded: they answer right away. The people giving advice are friendly and knowledgeable.

So far I have found only two minor glitches: When using the background printing feature, the document goes to never-neverland. (At least it does this on my H-151). Fortunately it is not lost, it still can be saved. This glitch, if it is indeed a glitch, is something to watch for, because the program awakens with this feature on. It is easy to disengage, though. For background printing ZSPOOL will work; however, as usual. The other glitch, or, rather an unexpected thing: when you start a program by retrieving an existing document a list appears on screen. You are restricted to the list visible on the screen. You hit <PgDn> to see the next page of lists and again you are restricted the same way. For instance, if you happen to know that your document is #22 on the list, you cannot retrieve it from the first screen (each screen has only 15 files listed). You have to hit <PgDn> to switch to the second list, where it appears. The first time it hap-

pened to me the message on screen was scary. I've thought a long one was lost.

The bottom of the screen shows the momentarily valid commands, using the "F" keys. Hitting <F1> changes this list to the next set. Altogether there are four such lists. Some of these commands from these lists do their thing right away, such as "insert" or "delete". Others will bring up a window with choices, such as "new format" or "save options". Some of these lists have sublists; for example the printer selection or the customizing list.

This customizing list deserves further mention. This is where the keyboard can be remapped, or the color scheme of the screen can be changed. In fact, almost every function of TEXTRA can be changed here.

The automatic Hyphenation is a very useful feature. It works thus: every row of the document is scrutinized and hyphenated as needed to make these rows nearly equal in length. The standard variables are: 10% length of a row, a minimum of 2 letters before and 3 after the hyphen, and no change, of course. (I've found 5% a better choice).

Using the Thesaurus is simple: move the cursor to the word you would like to replace and select "thesaurus" from the command list. A window will appear with a number of synonyms to select from. You can either replace the word with any of these or leave it. The spell checker, if evoked from the command list will go thru the document and stop at a suspicious word, highlighting it. A window appears with choices: you can either leave as is, add the word to the dictionary, or select from a list of possible substitutes. The "search and replace" is semiautomatic: it goes thru the document and stops at every search string, highlighting it. You choose to leave or to replace. It would be tedious in a long document, if it could not be made automatic thanks to the customizing feature.

The name/address book is called up by the <Alt+F5> keys. As an example, the program contains a fictitious list. The Lazy guy (that's me) can delete or change names on this list. This list consists of names, addresses, phone numbers and remarks. The list can be sorted by any of these. The mail merge function takes the addresses and other variables from this list. Furthermore, if a Hayes-compatible modem is connected to the computer it can dial the phone automatically. As true to any aspect of TEXTRA, this name/address book can also be customized by adding or removing categories.

As mentioned before, the price is \$99.99. All in all TEXTRA is a bargain for the price, considering all its features. Highly recommended for a business (it will work in a LAN) or as a private processor at home. In its skeleton form it is excellent

Continued on Page 42

ENABLE REVISITED

VERSION 3.0 PART 1

GEORGE P. ELWOOD
1670 N. LADDIE CT.
BEAVERCREEK, OH. 45432

It has been a couple of years since I last wrote about Enable, an integrated software package. My last articles focused on the Z-100 version of the product. The articles were written to point out the capability of the package and how to use it with the Z-100 although the PC keystrokes were included. What follows over the next few months will be a review of new capabilities found in Enable OA.

In the time since these articles were published, The Software Group has changed its name to Enable Software and two new versions of Enable have been released. There have been no new Enable upgrades for the Z-100 and none are planned.

The first revision to Enable after 2.0 was 2.15. This was a government version that brought the product that Zenith was providing under DeskTop II up to speed. It fixed many of the problem areas that caused early government users of Enable V1.15 problems. Version 2.15 also incorporated some of the features in version 3.0 or OA. I will not discuss Enable V2.15 in this article.

Enable 3.0 was renamed Enable OA when it was released. OA, or Office Automation, further refined the integrated software package. It added over one hundred enhancements over version 2.0. Enable OA was rewritten in part to "C". This was in preparation to a complete rewrite to "C" and a UNIX version. The MS-DOS/UNIX version, version 4.0 will be covered in a later part of this series.

The program has been modified so you do not need a special version to run on a Local Area Network. Enable OA is LANable but you will need a key number for each copy you plan on running on the LAN. Each copy of the program comes with a key number or you may purchase a LAN pack with just key numbers.

When Enable Software upgraded the program, they have consistently maintained backward compatibility. This means that all of the files you have created using versions 1.1, 1.15, 2.0, or 2.15 can

be used by the program. If you have files created in Enable OA, they cannot be used in earlier versions without saving them by using the correct version option. All of the modules in Enable have an option in the saving routine which permits saving in other formats.

When you purchase Enable OA, you will receive a large box with nine manuals and 24 360K 5.25 inch disks. If you plan on installing all of the disks, you must have at least seven Meg of space on your hard drive. It is possible to run Enable using a

dual floppy system, but I do not recommend it. The large blue box provides a convenient location for storing all of the manuals and is a change from the earlier light gray box. The applications manuals are now all covered with a blue binder. Gone are the color coded manuals of 2.0. Also included are four white manuals, "Getting Started", "Supplemental Documentation", "Printer Specifications", and "Perspective."

The "Getting Started" manual is the first thing to pull from the box. This provides

The image shows two screenshots of the Enable/OA installation process. The top screenshot is the 'Initial Screen of Installation Menu'. It features a title bar 'ENABLE/OA INSTALLATION PROCEDURE' and two main options: 'Complete Installation' and 'Partial Installation'. Below these is a section for 'Partial Installation Procedure' which states: 'This procedure will install Video drivers, Printer drivers, or Tutorials on a hard or floppy disk system or create a demonstration copy of Enable for a friend.' At the bottom, it provides navigation instructions: 'Use [Left], [Right] to select, then press [Enter]. Press [Esc] to EXIT the Installation Procedure.'

The bottom screenshot is the 'First Screen Printer Installation Menu'. It shows a date and time at the top: 'Wed. Apr. 25, 1998 8:55:16 pm'. The title is 'ENABLE/OA PRINTER DRIVER INSTALLATION PROCEDURE'. It has a 'Main Menu' section with the title 'PRINTER DRIVER INSTALLATION PROCEDURE' and a list of instructions: '1 - Select each Printer Driver you wish to install.', '2 - Display a list of every Printer Driver you have selected.', '3 - Install a library of every Printer Driver you have selected.', and 'ESC - Exit Printer Driver Installation Procedure.' At the bottom, it says: 'Press [1] [2] [3] to continue, or ESC to exit. Help is available by pressing [F1]'

the basic instructions for installing the program. The first thing you will have to check is the FILES and BUFFERS statements in your CONFIG.SYS program. Enable would like these set to FILES=40 and BUFFERS=20. Enable OA does support EMS (versions 3.2 or 4.0) so if you have the memory, the EMS driver should be included in the CONFIG.SYS statements.

To install the program, insert the Install Disk #1 into your floppy drive and type INSTALL. The Install Program will prompt you for where you want the system installed (default C:\en300), if you want the tutorial installed, if you want Perspective installed, if you want Hyper Text installed, and the type of system you are installing it on. You have the option to make a complete installation or just install the printer and video drivers. Note that the partial installation procedure permits you to build a demonstration copy of Enable for a friend.

After selecting and installing the video drivers, Enable moves into the printer se-

Wed. Apr. 25, 1990 8:55:43 pm
 ENABLE/OA PRINTER DRIVER INSTALLATION PROCEDURE

Manufacturer	Printer Model
- Comrex	- EX-888
- Data General	- EX-1888
- Dataproducts	- FX-88
- DEC	- FX-88+
- Diablo	- FX-85 *
- Diconix	- FX-86e
- Dynax	- FX-108
- Epson *	- FX-108+
- Facit	- FX-185
- Fujitsu	- FX-286
- Genicom	- FX-286e
- Hermes	- FX-858

Select Manufacturer:
 Press [Up/Down] arrows to position highlight bar. Press [ENTER] to select.
 Press [F7] to select all Printer Models, [ALT/F7] to deselect all.
 Press [ESC] when finished. - [ALT-M] mark ALL - [ALT-U] unmark ALL

Printer Selection Screen

spective, installation stops at disk 13 and you don't install disks 11 and 12 (tutorial).

After you have completed the installation, you can move into the EN300 sub-directory, type ENABLE, and press <Enter> to start the program. Enable OA

strokes" in the word processor. With this installed, Enable will automatically save the document to a temporary file after the specified number of key strokes. This feature has saved me several times because of power failures and system problems. Make sure you save the parameters when finished.

A simple macro can be built which will save having to press <End> at the Enable open screen. I normally call this macro "0". To create the macro, from the Enable Main Menu press "ALT/F9 \ 0". This will start the keystroke memory. Press the <End> key and then "ALT/F9 End" to save it. Your BAT file to start Enable should be changed to read

```
cd en300
enable ,@0
cd ..
```

The ",@0" will tell Enable to run the macro named "0" which is the keystroke <End>. This simple procedure, when invoked, will cause Enable to run through to the Main Menu. You may also wish to add some memory for other DOS applications. I used this capability while writing a manual on dBase III+. I set aside enough memory so that I could run dBase and Enable at the same time. By using the ALT/Up Arrow, I was able to move between dBase and the Enable word processor. I was also able to use the window copy command in

Word Processing

1 ... Left Margin on Default Ruler
 78 ... Right Margin on Default Ruler

8 ... Distance between tabs on Default Ruler
 5 ... Number of Tab settings on Default Ruler

Yes No ... Should the Default Right Margin be justified

Yes No ... Should F2/Right position to space after last character

500 ... Number of keystrokes between automatic document saves

Your document will be automatically saved after this number of keystrokes have been entered.
 Enter a number from 1 to 65535.
 To disable this feature, enter 0 or leave blank.

NOTE: A good typist will enter around 488 to 588 keystrokes per minute.

Revising DEFAULT Profile Esc=Exit Menu F10=Exit Profile

Profile Save Keystrokes Menu

lections. The printer driver numbers run out to 154, but all of the numbers are not used. The printer driver selection procedure using 360K disks is the most difficult part of the installation. If you have 3.5 inch disks or 1.2M, 5.25-inch disks, it is not difficult. On the printer option menu you select "1" on the menu to specify the printer you wish to install. Enable will permit you to copy all of the drivers to a 1.2Meg floppy before starting the installation. To select the printer, use the cursor keys to move through the manufacturers listing. Within each manufacturer is another list of drivers for specific printers. Again, using the cursor keys you may select any printer available. Pressing [ESC] will return you to the printer menu. You must now install all of the drivers selected by pressing "3". If you are using the 360K floppies, you may have to change between disk 3A and 3B several times.

After installing the print drivers, the remainder of the installation consists of feeding in the correct disk. If you do not wish to install hypertext, tutorials, or Per-

starts just like the earlier versions. I recommend that you press F1 on the opening screen and select create/modify Profiles. This is normally only done one time. You can select your default printer, page format, telecommunications options, and overall system default setting. One item you should install is the "save after XXX key

Window Summary Options

To go to another window, enter the window number, or use cursor keys and press ENTER.
 Press ESC to return to current window

Type	Status	Line	State	File Path
WP #1	REMARK.WPF	REF/B	DRAFT	Inactive C:\EN300\
WP #2	PHYTRM.WPF	REF/B	DRAFT	Inactive C:\EN300\
WP #3	NOUVELL.WPF	REF/B	DRAFT	Inactive C:\EN300\
WP #4	WESTDIG.WPF	REF/B	DRAFT	Inactive C:\EN300\
WP #5	DPATIENT.WPF	REF/B	DRAFT	Inactive C:\EN300\
SYS #6				Current

Environment

Profile: DEFAULT 88x87: No
 COM1: Inactive COM2: Inactive
 LPT1: Inactive
 Base memory available: 319,485 Expanded memory available: 131,872

Window Summary Screen

HORDER .WPF	JCLUP1 .WPF	PACKLIST.WPF	TMPWP001.WPF
HRATESON.WPF	JCLUP2 .WPF	PAYROLL .WPF	TPACK .WPF
HRCUST .WPF	MANADD .WPF	PL3 .WPF	UP .WPF
HRORDER .WPF	MANEDIT .WPF	QTEST .WPF	WKHOURS .WPF

PASSWORD

Enter new password: **ABCDE**

Entering new Password

Enable to capture screen displays for the manual. See the "Getting Started" manual for the correct parameter.

Before we move into the new parts of the word processor, lets look at the new functions available under the MCM option on the Main Menu. The MCM, or Master Control Module, provides several new options. One of these is the Display Window Summary. If this option is selected, Enable will display all open windows and provide a summary of the Environment. The environment consists of the name of the PROFILE in use, and the COM and LPT ports in use or available. Enable will check the system and display all available output ports. Just above the Status Line the system memory will be displayed. This includes both base and expanded memory, if available.

Another function available through the MCM window is the extended file system. From the Main Menu select (M)CM, (F)iles, (A)ll. This will display all files that are in the EN300 subdirectory. This first screen is similar to that seen in earlier versions of Enable. One of the options on this screen that is new to Enable is (P)assword. Using this function, you can assign a password to the file that is highlighted above. The file to be protected must be saved in Enable OA format first if it was created in an

earlier version. If you assign a password to a file, when you try to open it, Enable will prompt for the password. If you give the wrong password it will with INCORRECT PASSWORD SUPPLIED and prompt for the password again. You must insert the correct password before the file will be

extended files capability. Pressing "X" will display all of the files again, but the format has been changed. As you move through the list of files using the cursor keys, the file the cursor is pointing to will be display on the top line of the screen. The file name and when it was created, both date and time will be displayed.

As you move through the listing of all files, note that Enable's pointer is a small arrow. The files are sorted alphabetically as a default. The file name is followed by the extension and file size. After the size are the DOS file attributes. The attributes shown are (A)rchive bit, (D)irectory, (V)olumn, (S)ystem file, (H)idden file, and (R)ead only file. You may reset the hidden and read only attributes. As an example, you may change the attribute to "H" for a file you do not wish to be displayed using the DOS DIRectory command. To change a file attribute, you must use the F7 key to mark the files. Then you press the (A)tttribute, (S)et file attribute, and then the desired attribute.

If the file is a database file (extension

Enter password. [R] erase, [RET] accept, [ESC] exit. T.WPF

Incorrect password supplied. Press any key to continue.

Password Prompt in Word Processing

available. If you forget the password, there is no way to recover the file. The file is encrypted so you will not be able to read it by using the DOS TYPE command.

Note the "X" function that is available on the bottom of the screen. This is the

.DBF), the attributes will not be displayed. Instead, Enable will display the number of records in the file. If you see an "*" displayed, this means that the file is in use.

Another function available in the extended directory is the sort command. The default sort is alphabetic. You may sort by date/time, extension, file name, or file size. I have written a complete menu driven business package for a tool and die company. When I modify files, I use the sort by date/time and mark files to copy for a transfer. This way I can quickly identify new files which must be moved to the customer's site. The sorting can be made in normal or reverse order.

After sorting the files, I move through the list, marking the files to be copied by pressing the F7 key. Once I have all of the files marked, I press (C)opy and tell Enable where I want the files copied to. Pressing the <Enter> key will copy all of the files to the indicated location. Enable tracks the total files marked, space available on the "copied to" location. If you have marked

ACCTPAY .SIF		1/28/90	16:14:30	1 File	4,896 bytes marked
<DIR>	..		[.D.S.]		
\$[B]	.MCH	512	[A.....]	ADD .MNU	3,159 [A.....]
\$[E]	.MCH	512	[A.....]	ADDMENU .WPF	11,654 [A.....]
\$[G]	.MCH	512	[A.....]	APCHECK .WPF	2,986 [A.....]
\$[I]	.DBM	512	[A.....]	APCHK .NDX	5,632 [A.....]
\$[M]	.WPM	512	[A.....]	APCNBR .SIF	512 [A.....]
\$[P]	.MCH	512	[A.....]	APCNBR .DBF	1,824 [A.....]
\$[T]	.MCH	512	[A.....]	APCX .NDX	7,168 [A.....]
\$[X]	.MCH	1,824	[A.....]	APE .MNU	1,153 [A.....]
\$[Z]	.MCH	1,824	[A.....]	APJCL .NDX	1,824 [A.....]
> ACCTPAY .SIF		4,896	[A.....]	APPRT .MNU	2,806 [A.....]
ACCTPAY .DBF		6,144	[A.....]	APSIM .NDX	2,848 [A.....]
ACCTRECV .SIF		3,872	[A.....]	ARCX .NDX	1,824 [A.....]

Set Attributes Pattern: [ADUSHR]
Selected: [.....]

A = Archive (back-up) bit H = Hidden file R = Read-only file
S = System file (No changes permitted to volume name (U) or directory (D).)

Toggle selected attributes by letter, [←] to SET ATTRIBUTES, [Esc] to Exit
#2 C:\MQT*. * 258 Files 1,236,277 bytes used

Attribute Setting Window

```

Viewing file C:\MQT\BALSHT.SIF                               ENABLE DEF file
000000 04 CD AB 14 00 00 33 30-30 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000010 00 00 00 00 F0 00 0A 00-42 41 4C 53 40 54 00 02 00 00 00 00 00 00 00 00
000020 03 00 F0 00 1C 00 42 61-6C 61 6E 63 65 20 53 68 00 00 00 00 00 00 00 00
000030 65 65 74 20 69 6E 70 75-74 20 66 6F 72 6D 00 01 00 00 00 00 00 00 00 00
000040 03 00 F6 00 4D 01 00 00-01 CD AB 4B 01 30 00 00 00 00 00 00 00 00 00 00 00
000050 00 00 00 00 00 50 00 00-00 00 00 00 00 00 00 00 00 00 00 00 00 00
000060 00 00 00 A0 00 00 06 6E-58 40 06 04 02 02 02 01 00 00 00 00 00 00 00
000070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 40 00 00 00 00 00 00 00
000080 00 00 00 00 00 00 01 01-02 06 00 00 00 00 0F 01 00 00 00 00 00 00
000090 4C 00 00 00 00 00 01 6C-00 00 00 00 01 6C 00 00 00 00 00 00 00 00
0000A0 00 00 00 00 20 2A 20-2A 20 2A 20 2A 00 00 00 00 00 00 00 00 00 00
0000B0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000C0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000D0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000E0 00 00 00 00 00 00 00 00-00 00 00 1C 66 6F 6F 00 00 00 00 00 00 00
0000F0 74 6E 6F 74 65 20 63 6F-6E 74 69 6E 75 65 73 20 00 00 00 00 00 00

Use: Up Arrow to scroll back 1 line, PgUp to scroll back 1 page.
     Dn Arrow to scroll forward 1 line, PgDn to scroll forward 1 page.
     [Esc] to exit View mode.         [Home] to the beginning of the file.
     [F9] toggle binary/data view    [End] move to the end of the file.

#2 C:\MQT\*. *                250 Files 1,236,277 bytes used
View of Partial Binary File

```

many files, Enable will check available space. If there is not enough space it will continue down through the marked list using small files. It will put as many files as can fit in the new location. Enable will then prompt for a new disk and will resume the copy process.

You may also use this procedure to destroy several files at one time.

Another feature is the text scan. If you are looking for a word or phrase and you do not know what file it is located in, you may mark suspect files. Enable will then check the marked files for occurrences of the specified word or phrase. When Enable finds the specified phrase, it will mark the file by highlighting it.

Another of the available functions is

View file/directory. This function will permit you to look at any file that is in the directory. If the file is ASCII or Enable Word Processing, it will display the text on the screen. If it is a binary file, the HEX values will be displayed. The display will be very similar to the DEBUG Dump command.

This has been an overview of the functions available. Most of these functions can be performed using the various PC tools available on the market. Enable has just put them together in one place.

This completes the first of a short series on the new versions of Enable. I will keep this series to about six total articles. The plan is to have one each on word processing, spreadsheet graphics, and database. After completing these articles on version 3.0, I will write two articles on the next version of Enable, 4.0. This version is unique in that it will run on MS-DOS, in a window environment, OS/2, XENIX or UNIX. You make the choice when you install it. Many new features have been added. ✧

Continued from Page 12
your system (including the AUTOEXEC.BAT and CONFIG.SYS files) and make recommendations (and changes) to "optimize" memory usage. All of the manuals include the necessary details for really fine tuning the software, but you can still get excellent results if you only use the program's suggestions.

Powering Down
I hope you will find these Quarterdeck programs as useful and cost effective as I have, but let me mention one caution. Since I did not test every possible combination of all software, be SURE you test your specific system carefully before you begin using it for production work. Even though I did not find any problems in my testing, there is sure to be at least one program that does not work because it violates one or more programming standards.

For help in solving specific computer problems, be sure to include the exact model number of your system (from the back of the unit or series from the Owner's Manual), the ROM version you are using (use CTRL-ALT-INS to find it, except for the eaZy PC), 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., modem, bus mouse or video card), the brand and model of the CRT monitor you have, and the brand and

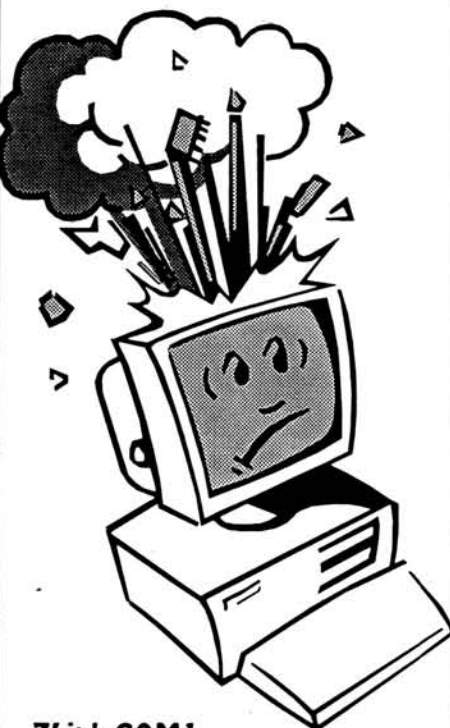
model of the printer with the type of interface (i.e., serial or parallel) you are using. Also be sure to include a listing of the contents of the AUTOEXEC.BAT and CONFIG.SYS files unless you have thoroughly checked them out for potential problems (e.g., TSR conflicts). If the problem involves any application 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 ZDS or Heath 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 Discussed

- Software**
Powering Up (885-4604) \$12.00
Zenith Users' Group
P. O. Box 217
Benton Harbor, MI 49023-0217
(616) 982-3463 (ZUG Software only)
- Manifest \$60.00
GRAM 80.00
QEMM386 100.00
DESQview 130.00
DESQview386 220.00
- Quarterdeck Office Systems
150 Pico Boulevard
Santa Monica, CA 90405
(213) 392-9701 ✧

Hardware Problems?



**ZLink-COM1
Bulletin Board is
Your Best Source for
Technical Information!
Modem: (616) 982-3956**

Programming the Function Keys in MS-DOS

Pietro A. Carboni
8 Stone Avenue
Greenwich, CT 06830

The DOS Dilemma

Many times I've heard PC users refer to DOS commands as being 'cryptic' and awkward to use. Commonly used commands have options that users don't know about because of confusion concerning what optional items, switches, redirection, and piping characters to use. Maintaining proper syntax when entering a rarely used command may be frustrating. Of course you can write batch files for most tasks, but then you encounter the problem of file names. Making a file name short - for quick retrieval - won't describe its contents and long - for descriptive reasons - could be hard to remember - resulting in no time savings at all. Alternatives like a Graphical User Interfaces (GUI) - used by Windows - have been invented to make running a computer easier, but nothing comes free: a monetary cost results from greater memory requirements, a higher resolution graphics adaptor, and matching monitor. There may also be the cost of reduced speed in execution of tasks. So what can we do with DOS to make it user friendly? What do you do with a cipher? Decode it! DOS has many features that can personalize the interface between you and the computer - making things easier and faster.

The DOS Solution

I'm going to cover one of the most powerful things you can do with DOS: program the function keys on your keyboard to carry out whatever commands you desire. With one convenient keystroke you will be able to execute DOS commands, utilities, batch files, or start up an applications program. Left to their own, the F1-F6 keys serve to edit the command line - but their usefulness is limited. The loss of these functions will be offset by the new power of these keys.

The technique of assigning commands to function keys doesn't require any special program. All you need is the ANSI.SYS

device driver from your MS-DOS distribution disks and a batch file to assign the commands to the keys. When installed by the CONFIG.SYS file, the ANSI.SYS driver employs codes known as escape sequences to 'load' the desired commands into each function key. Every computer owner should have this driver installed; it not only allows you to program the function keys, but is used by some programs to change graphic functions. If you ever ran into a program that refused to display correctly on your machine - particularly shareware game programs - it might be because you didn't have ANSI.SYS installed!

For those who don't already have one, a typical CONFIG.SYS file is shown in Figure 1 - just type it in. Otherwise just add the line DEVICE=ANSI.SYS to your existing CONFIG.SYS. Full path must be specified after the '=' sign if ANSI.SYS is not in the root directory with CONFIG.SYS. This CONFIG.SYS file contains the BREAK=ON internal DOS command that allows Ctrl-C to operate under a greater number of circumstances. The FILES and BUFFERS statements speed up your computer when running certain applications - you can look up the details in your DOS manual.

```
FILES=24
BUFFERS=24
BREAK=ON
DEVICE=ANSI.SYS
```

Figure 1
Typical CONFIG.SYS

When your machine is rebooted, the new CONFIG.SYS will load the ANSI driver and you'll be set to assign commands to your function keys. We can program the function keys F1-F10, the combination of Shift+Function Key, Ctrl+Function Key, and Alt+Function Key - for a possible total of 40 commands! Unfortunately the number of keys programmed is limited to the memory space given ANSI.SYS when in-

stalled. Everything in this article was tested using the version of ANSI supplied with MS-DOS 3.21 that came bundled with my Z-159; it can program about 130 characters of data - good for 15-20 key assignments. For those who want more power in their ANSI driver, commercial and shareware replacements are available that allocate as much RAM for ANSI as desired.

Function key combinations are not the only keys that may be programmed; in fact, through the use of the ANSI reference codes, the ANSI driver recognizes most of the keys on your keyboard. The table in Figure 2 contains reference codes for each function key and key combination. Because the F11 and F12 keys were introduced as additions to the original keyboard they do not have ANSI reference code assignments and can not be programmed in the simple manner described in this article.

	Function Key +			
	[Shift]	[Ctrl]	[Alt]	
[F1]	= 59	84	94	104
[F2]	= 60	85	95	105
[F3]	= 61	86	96	106
[F4]	= 62	87	97	107
[F5]	= 63	88	98	108
[F6]	= 64	89	99	109
[F7]	= 65	90	100	110
[F8]	= 66	91	101	111
[F9]	= 67	92	102	112
[F10]	= 68	93	103	113

Figure 2
ANSI Reference Numbers for
Function Key Combinations

With ANSI installation taken care of, the batch file to update your function keys must be created. I'll call the file: DEFKEYS.BAT (DEFine KEYS). Like CONFIG.SYS this file must be written in plain ASCII format - no hidden codes al-

lowed. You can use such ASCII editors as the build in DOS EDLIN editor, the Notepad in *Sidekick*, or the nondocument mode of *WordStar*. *WordPerfect* users can implement the Text In/Out (Ctrl+F5) option to generate ASCII text output of a document.

Loading up the Keys

The familiar DOS command 'PROMPT' - aptly named because it modifies the command line prompt - is used to load the command definitions into the function keys; yet, my DOS manual doesn't mention this important additional function. Each line used to load a command in DEFKEYS will have this same basic format:

```
PROMPT=$e[0;nn;'command';13p
```

This line may look complex to the uninitiated; luckily you only type it in once and copy it for each function key assignment - changing the elements in bold face. The details of the PROMPT line are as follows: *\$e* is the escape character which informs the ANSI driver that escape sequences follow, *[0;nn* is the extended ANSI reference code that defines the function keys combination to be programmed - *nn* is one of the numbers in Figure 2, '*command*' is whatever we want the function key to issue when pressed, *13* is the ANSI carriage return code - used for commands that don't need any additional user input, and *p* signifies the end of the command definition. It is important to emphasize that commands ending in *13* are executed immediately; those that end in *p* (with no number) await additional user input and the Return key to execute. Also note that commands can be entered between double quotes, but that would cause conflict if the command contained double quotes as part of its syntax.

Now all that is needed are a suitable list of commands to load. Consider a mix of complex, rarely used, commands - commands you don't want to relearn the syntax to every time they're needed - and commonly used commands. I've included a series of DOS, utility, and batch file call ups in my example DEFKEYS. You'll want to have some of these in your version, as well as others you think important.

Figure 3 contains my example DEFKEYS; we'll take a tour through that right now. It has 18 commands programed into the F1-F10 and Shift+F1-F8 keys; they exceed my 130 character ANSI limit, but are useful for illustration. Some commands are typed in with padding spaces for reasons I'll explain later. In order to group related commands together, I've divided them into sets matching the function key layout of my enhanced AT style keyboard. REM statements indicate which series of keys are being loaded and are useful markers when updating the file.

DEFKEYS.BAT Explained

The first group of four (F1-F4) contain

commands pertaining to file listings and disk space available. Commands assigned to F1 and F2 contain the carriage control code and are executed instantly: DIR /p displays the current directory you are in one screenful at a time and DIR A: is handy when determining the contents of a stack of disks - just insert a disk into the drive and hit F2 for each one. F3 brings up the unadorned DIR command and allows you to enter whatever switches, drive, and path you wish; notice it doesn't contain the carriage control command. Rounding out the first group is the FREE utility assigned to F4. This program, found on the *PC Magazine Utilities Disk Vol. 1*, instantly reports how much storage is available on the default drive. You may also use CHKDSK to accomplish this task - except that it takes longer.

Proceeding along, the second group - encompassing F5-F8 - contains some commonly used DOS commands: COMP to compare two files, TYPE, COPY, and DELETE. Notice that none of these commands are assigned in the batch file with the carriage return code; this allows you to enter a path/filename after the command.

Next, FORM_A is a batch file designed to format, label, and check a disk when called up with F9; the PRINT command is assigned to F10. Figure 4 contains the contents of the FORM_A.BAT file. Notice that the FORMAT statement contains two switches: /v prompts for a volume label, and /q suppresses spurious prompts and information. Of course you could type the name of the batch file at the DOS prompt, but then you may encounter the file name problem - forgetting whether it was called FORMAT_A, FORM-A, FORMA, and so forth. With function key assignments all you have to remember is to hit F9.

The remaining sets of key definitions in Figure 3 constitute the Shift+Function Key combinations. Our first group, Shift+F1-F4, is used to move around the subdirectory structure; all lines have the carriage return code. Shift+F1 and F2 are of general use:

```
rem: -----FUNCTION KEYS F1-F4
PROMPT=$e[0;59;'DIR /p';13p
PROMPT=$e[0;60;'DIR a: ';13p
PROMPT=$e[0;61;'DIR 'p
PROMPT=$e[0;62;'FREE';13p
rem: -----FUNCTION KEYS P5-P8
PROMPT=$e[0;63;'COMP'p
PROMPT=$e[0;64;'TYPE'p
PROMPT=$e[0;65;'COPY'p
PROMPT=$e[0;66;'DEL'p
rem: -----FUNCTION KEYS P9-F10
PROMPT=$e[0;67;'FORM_A';13p
PROMPT=$e[0;68;'PRINT'p
rem: -----SHIFT KEYS F1-F4
PROMPT=$e[0;84;'CD\';13p
PROMPT=$e[0;85;'CD..';13p
PROMPT=$e[0;86;'CD FILES';13p
PROMPT=$e[0;87;'CD C:\LOTUS';13;'123';13p
rem: -----SHIFT KEYS P5-P8
PROMPT=$e[0;88;'FIND/i/n "test" file.ext'p
PROMPT=$e[0;89;'SEARCH'p
PROMPT=$e[0;90;'RENAME'p
PROMPT=$e[0;91;'UNERASE';13p
rem: -----SHIFT KEYS P9-F10
PROMPT=$e[0;92;'';13p
PROMPT=$e[0;93;'';13p
rem: -----SCREEN PROMPT
PROMPT $e[45;1m$ps$g$e[1C$e[m
c:1*
```

Figure 3
Sample DEFKEYS.BAT

CD\ brings you to the root directory of the default disk, CD.. moves you up one level in the subdirectory tree. The assignment of CD FILES to Shift+F3 is of use to anyone who sets up their applications software so data files are always written to a subdirectory called 'FILES.' That is, my *WordPerfect* files are found in C:\WP\FILES, my *Lotus 1-2-3* spreadsheet files in C:\LOTUS\FILES, the same for *dBASE*, and so forth. Thus when I'm in an application program directory I can quickly examine the data files subdirectory by pressing Shift+F3. The last key definition in the set - Shift+F4 - is unique in the DEFKEYS example because it uses the carriage return code twice; it is a two 'line' command. The first part - CD C:\LOTUS - brings you to the LOTUS subdirectory. The second part - 123 - actually starts up the program. The revelation that more than one line can be programmed into the keys is tempered by the fact that memory is used up quicker for multi-statement commands, and batch files are easier to edit. That is why the F9 key was programmed to call up the disk formatting file, FORM_A.BAT, instead of programmed to execute an equivalent series of statements.

Last we have the Shift+F5-F8 group that contains a variety of useful file functions. FIND is the external DOS command for

```

echo off
echo:
echo: This batch program
echo: FORMATS, LABELS, and CHECKS disks in drive A:
:start
echo:
echo: To TERMINATE this batch job at any time
echo: hit CTRL-C or BREAK
echo:
echo: INSERT disk to be formatted into DRIVE A: and
pause
FORMAT A: /v /q
CHKDSK A:
goto start
:done

```

Figure 4
FORM_A.BAT File

locating text strings in a file. Pressing Shift+F5 brings FIND/i/n "text" file.ext to the commands line; this is an example of annotating a DOS command to make its syntax available to the user. The word text inside the double quotes is replaced by the actual text the user wishes to locate, and file.ext is replaced by the file name the search is to be carried out on. Entering this information on the command line is aided by the use of the Insert and Delete key where necessary. The /i switch ignores case and the /n switch outputs the line number where the text was found.

The next two key combinations also require input. SEARCH allows the user to enter a drive, path, and filename to locate files throughout a directory structure. RENAME is the DOS command to rename a file. Lastly, Shift+F8 calls up a Norton Utilities program, UNERASE, that recovers accidentally deleted files. The remaining two key combinations (Shift+F9-F10) are entered as place holders for future expansion.

The PROMPT command at the end of the DEFKEYS listing creates your screen prompt; this is important because all the previous commands have obliterated the screen prompt. While the simple PROMPT command by itself will give the default

function keys is a template to indicate what command each key generates. Figure 5 contains a keyboard template for the DEFKEYS command assignments we just looked at. Commands are identified by short descriptions; italics indicates that a Shift+Function Key is required to execute the command.

Another good idea is to load DEFKEYS through the AUTOEXEC.BAT file; an example of this is shown in Figure 6. Notice that a PATH statement is used to indicate what directories have the external commands. The files from the DOS distribution disks are in the \BIN directory, \UTILITY has the utility files such as FREE and UNERASE, and \BATCH holds batch files that include DEFKEYS and FORM_A. Through the use of a PAUSE statement this AUTOEXEC gives the operator the power to abort installing DEFKEYS and a TSR called DOSEDIT. The shareware DOSEDIT.COM — similar in operation to the PC Magazine NDOSEDIT.COM utility — is included in the AUTOEXEC because it increases the utility of the DEFKEYS assignments by storing the last dozen command lines issued at the DOS prompt.

At the DOS prompt, when DOSEDIT is installed, each press of the keyboard up arrow brings back a previously executed

command. The down arrow moves through the commands in chronological order. This is great for repeating commands and examining previous command inputs in case of error. In addition, this allows command line editing when performing multiple operations on a file. You edit a command line under DOSEDIT with the following keys: Home moves to the beginning of the command, End jumps to the end of the command, Ctrl plus left or right arrow moves one word left or right, Insert allows characters to be typed while pushing over existing text — by default your cursor types over characters, and Delete erases the character it is on.

Using the Function Keys

With an example I will demonstrate the utility of the DEFKEYS command assignments, and the ability of DOSEDIT to enhance this utility. Function key commands are identified by the template in Figure 5. Assume you wish to locate a text file reference concerning ANSI.SYS as implemented in DOS 4.0, but you know only that the name of the file begins with the letters D-O-S. To begin the search, move to the root directory by hitting the ROOT DIR key (see Template); press SEARCH (Shift + F5) and enter 'dos*.*' as the file name.

```
C:\> SEARCH dos*.*
```

The computer generates the following list of to match ups:

```
C:\TEMP\DOS4.ANN
C:\TEXT\DOS40.TXT
```

You have two possible files. The ANN extension probably stands for ANNouncement; investigate this one first. Change directory:

```
C:\> cd temp
```

Press the up arrow to recall the SEARCH line, Home to get to the beginning of the command line, and press DIR /P (F1) — this replaces the word SEARCH with DIR /P. Thus the command is executed without having to retype the file name:

```
C:\TEMP> DIR /p dos*.*
```

This is why certain commands in DEFKEYS have padding spaces - to give them enough characters to overwrite previous commands. The DIR /p command now returns

<u>ROOT DIR</u>	<u>UP A DIR</u>	<u>FILES SUB</u>	<u>LOTUS 123</u>	<u>FIND</u>	<u>SEARCH</u>	<u>RENAME</u>	<u>UNDEL</u>	<u>FORMAT A:</u>	<u>PRINT</u>
DIR /P	DIR A:	DIR ?	FREE	COMP	TYPE	COPY	DEL		
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10

Figure 5
Keyboard Template

prompt, I opted for something colorful and informative. You can consult your DOS manual under 'PROMPT' for options to include in your screen prompt and 'ANSI' for details on escape commands for setting colors in your screen prompt.

Operations Enhancement

A practical item to have when using the

```

PATH=C:\BIN;C:\UTILITY;C:\BATCH
PROMPT $e[45;lm$g$e[1C$e[m
pause: ... CTRL-C here to SKIP loading TSR programs and DEFKEYS ...
DOSEDIT
DEFKEYS

```

Figure 6
Sample AUTOEXEC.BAT

```

----- dos4 .ann
[116] IBM PERSONAL SYSTEM/2 80286 MEMORY EXPANSION OPTION OR SOFTWARE
[120] IBM PERSONAL SYSTEM/2 80386 MEMORY EXPANSION OPTION
[122] IBM PERSONAL SYSTEM/2-8MB 80386 MEMORY EXPANSION OPTION
[165] ANSI.SYS HAS A NEW /X PARAMETER THAT ENABLES EXTENDED KEYBOARD

```

Figure 7

```

rem: -----FUNCTION KEYS F1-F10
PROMPT=$e[0;59;0;59p
PROMPT=$e[0;60;0;60p
PROMPT=$e[0;61;0;61p
PROMPT=$e[0;62;0;62p
PROMPT=$e[0;63;0;63p
PROMPT=$e[0;64;0;64p
PROMPT=$e[0;65;0;65p
PROMPT=$e[0;66;0;66p
PROMPT=$e[0;67;0;67p
PROMPT=$e[0;68;0;68p
rem: -----SHIFT KEYS F1-F10
PROMPT=$e[0;84;0;84p
PROMPT=$e[0;85;0;85p
PROMPT=$e[0;86;0;86p
PROMPT=$e[0;87;0;87p
PROMPT=$e[0;88;0;88p
PROMPT=$e[0;89;0;89p
PROMPT=$e[0;90;0;90p
PROMPT=$e[0;91;0;91p
PROMPT=$e[0;92;0;92p
PROMPT=$e[0;93;0;93p
rem: -----SCREEN PROMPT
PROMPT $e[45;1m$ps$g$e[1C$e[m
cls

```

Figure 8
UNKEYS.BAT

C:\TEMP> TYPE dos4.ann

By comparing the number of keystrokes in the above example to the number required without using DEFKEYS assignments and DOSEDIT you'll find a substantial savings in keystrokes and time. If you can acquire DOSEDIT.COM or NDOSEDIT.COM they're worth using. If you can't, leave the F3 key un-programmed in DEFKEYS; it will serve its original role in DOS of recalling the last command line executed.

Helpful Hints

Here are some final notes and reminders to consider when writing your own version of DEFKEYS.BAT. Program your most often used commands into this batch file — recall that the first three DEFKEYS assignments were variants of the oft used DIR com-

mand. Also program those powerful not-often-used commands so they'll be there when you need them — without resorting to the DOS Manual. Do check if there are any useful switches associated with the command — they can customize the command to your needs.

If your ANSI version has the RAM available you may program annotated versions of many commands. This is helpful for those commands you rarely execute and for beginners to DOS. For instance, instead of a bare RENAME command popping up when the user hits Shift+F7, an annotated version would be RENAME from.ext to.ext. Also F5 could become COMP d:path/sourcefile d:path/targetfile.

Before running DEFKEYS for the first time check to see that a p, ;, or 13 is not misplaced or forgotten in the PROMPT assignment lines; an error of this sort will immediately become apparent when you attempt to execute the faulty command.

A keyboard template is a must for convenience of use. They are easy to make and worth it if you store many commands into the function keys.

Finally, I must mention a batch program called UNKEYS.BAT, in Figure 8, that cancels the new function key assignments and returns them back to regular duty. This is required for those programs that get confused when the function keys are redefined. Just put UNKEYS in a directory accessed by the PATH statement and run it anytime the function keys inside an applications program refuse to work. Norton Utility is one program you have to 'UNKEYS' in order to use the function keys inside the Edit/display item option of the 'Explore disk' menu. Not to worry though — most programs run fine. Here is a list of programs I've tested that utilize function keys and run with DEFKEYS: *BASICA, WordPerfect, SideKick, Lotus 123, Mace, and PC Tools.* ✨

the requested file information:

DOS4 ANN 27648 7-20-88 6:07a

Since this file is large, use the FIND key (Shift + F5) to locate the reference. Because the FIND command is annotated, entering the appropriate information in the proper syntax is easy:

C:\TEMP> FIND/i/n "ansi" dos4.ann

This locates the matches in Figure 7.

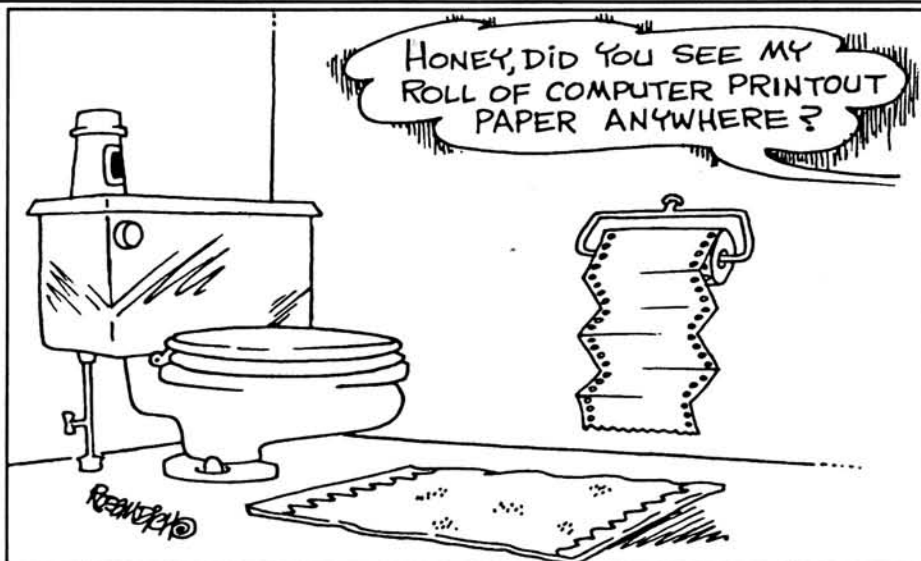
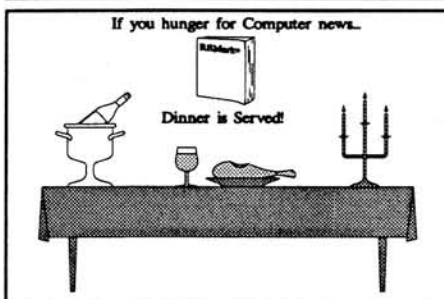
One file should be deleted to save disk space. The file in the C:\TEMP directory, having the least descriptive name, is the best candidate. Again call up the line containing the TYPE command using the arrow keys, press Home, then Del (F8), and Return to finish.

Continued from Page 34

for a laptop.

The name and address of this Laudable firm:

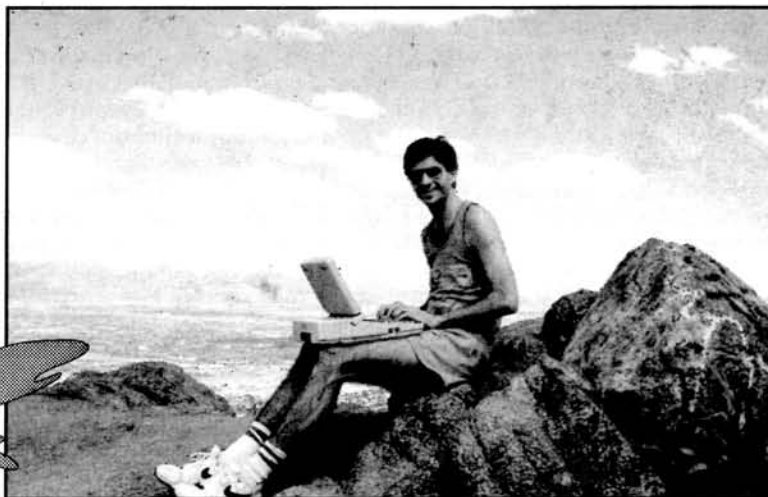
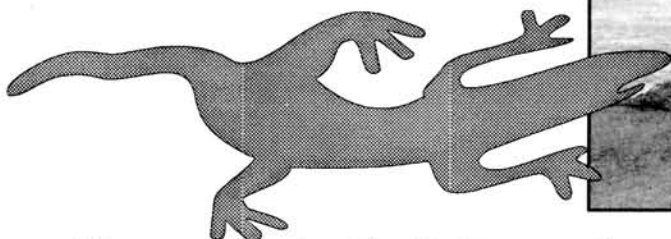
Ann Arbor Software
345 S. Division
Ann Arbor, MI 48104
(313) 769-9088 — Business Office
(313) 769-1014 — Technical Support ✨



Planes, Hotels, and Lizards

Computing on the Road

Stephen M. Kristan
4771 Hyde Park
Troy, MD 48083



How mobile can someone be with a portable computer? In today's world people seem to be more mobile than ever. Whether it's traveling across town or across country, portable computers help us take computing power wherever we go. The question in my mind is how portable are portable computers, actually; and how mobile can a person really be with one?

On a recent trip to the west coast I had a chance to find out as I traveled with a Zenith Data Systems SupersPort SX. Here is my story.

Before leaving on my trip I needed to decide what type of portable computer to take along. There are two schools of thought about portable computers. One says the ideal portable computer should be lightweight and compact (no pun intended). Machines like this are commonly called notebook computers. To minimize size and weight these machines don't offer many extra features. Notebook computers typically have less sophisticated microprocessors, CGA screens, and smaller hard drives (if any at all).

An example of a notebook computer is the Zenith Data Systems' MinisPort. This notebook computer has a 20MB hard drive, CGA screen, is lightweight (under 8 lbs.), and is small enough to fit inside a briefcase. This machine is good for word processing and communications, but not for the more power hungry applications.

The other school of thought about portable computers says the ideal computer should be as powerful as your desktop computer, just more portable. For our purposes here, I'll call these types of machines laptop computers. While still portable, laptops are usually bigger and heavier than notebook computers, but contain faster processors, larger hard drives, and

VGA screens. These computers combine portability with power.

An example of a laptop computer is the Zenith Data Systems SupersPort SX. While bigger than the MinisPort, the SX can still fit on your lap, thus the name laptop. Weighing in at 12 lbs. without the battery and almost 17 lbs. with it, the SX is heavier than the MinisPort. However, the SX has a faster processor, larger hard drive, and VGA screen, so it's much more powerful.

I have a friend who owns both a MinisPort and an SX and was willing to let me borrow either one. In determining what type of machine to take with me I analyzed my software requirements. Since I didn't relish the thought of learning new software just for this trip I wanted to run the same software on the portable that I run at home. My desktop machine has WordPerfect, Excel, Powerpoint, GB-Stat, and Prodigy, all multi-tasking under Windows 3.0. These programs could run on the SX laptop. Most would not run on the MinisPort. Therefore, I chose to take the SX with me.

Like the name implies, the SupersPort SX comes with an Intel 80386SX microprocessor. In terms of power, this chip is a hybrid between the 80286 and the standard 80386 processor. The 286 chip is a 16-bit processor, meaning it processes 16 bits or pieces of information at once. The 386 chip (now officially known as an 80386DX) is a 32-bit processor meaning, it processes 32 bits of information at once. To keep the cost down Intel gave the SX a 16-bit data path instead of the 32-bit data path of the DX. This means the SX processes data as fast as the DX, but moves data in and out of the processor more slowly. The result is a machine with throughput faster than the 286, but slower than the 386DX.

The 32-bit processing capability of the SX also gives it multi-tasking capabilities. I've always believed that a computer should work the way I do, not force me to work the way it does. For me, this means doing several things at the same time. For a computer, this is called multi-tasking; running multiple programs at the same time. A computer with a 386SX processor can work the way I do.

The SX I took with me came with a 40MB hard drive, 2MB of RAM, one high density floppy drive and an internal 2400 baud modem. The battery attaches to the end of the computer like other Zenith Data Systems' SupersPort models. The keyboard has 79 keys, but by using mode switching and multiple keys to duplicate special function keys you can get 101-keyboard compatibility. I also took a serial mouse along to use with my Windows software so I wouldn't have to rely on keyboard commands.

The night before my trip I used Laplink Plus to transfer software from my desktop machine to the SX. Laplink worked great. No laptop owner should be without this program. After connecting the special cable between both machines I ran Laplink. To transfer files from the desktop to the SX I simply highlighted the files to be transferred and issued the copy command.

My desktop software was configured for VGA so all the files I transferred ran on the SX. The SX has monochrome VGA and uses various shades of gray to display different colors. I found some of the colors were difficult to distinguish on the laptop. This was because the colors were initially configured for color VGA. I fixed this by changing the color settings in the application software for a better display in monochrome mode.

Next I packed everything into my carry-on bag. ZDS makes a nylon carrying case specifically for the SX; however, I decided not to use it. That case comes with the official Zenith Data Systems logo on it. Some unscrupulous would-be thief might be tempted to steal the case knowing that it contains a computer. My non-descript carry-on bag looks less revealing.

My second reason for using a carry-on bag is size. It is bigger than the ZDS case so I can fit my travel supplies in with the computer. These supplies include my walkman to be used while waiting in airports and a few computer magazines and books. I also threw in a few back-up diskettes should my hard drive crash. Laptop hard drives are built more ruggedly than most desktop computer drives, and I've never experienced any problems with them on the road. Even so, I always back up my data onto floppies while traveling.

I learned the hard way several years ago not to assume all hotels have modular jacks. Ever since I was forced to install modular jacks in a Cleveland hotel three years ago, I've been traveling with my own phone conversion kit. It includes a modular jack conversion kit, several small screw drivers, and several pieces of wire. It also fits into my carry-on bag. With the bag packed I was ready to go.

A few years ago the FAA became concerned about the growing number of portable computers on airlines. Apparently, bombs with electronic timing devices look similar to computers when passed under the airport security checkpoint x-ray machines. A movement was started to outlaw portable computers on planes. Instead, the FAA started stricter security checks of portable computers. This means that when passing through the airline x-ray machine, travelers will often be asked to boot their computer. This shows the security people that the computer is for real.

I mention this because travelers sometimes think of this as an annoyance. But if turning the machine on is all it takes to insure that I can carry it with me on a plane, in my mind, it's well worth the slight inconvenience. Also, all Zenith Data Systems portable computers can be run from batteries so this shouldn't be a problem. Just make sure the battery is charged when you reach the airport.

When I got to the security checkpoint at the airport I was asked to boot up the SX. The machine booted to the C: prompt and the security guard let me through. (I was glad that I had remembered to charge the battery before I left.) When I reached the gate the plane was already boarding so I went straight on board. Even with the SX inside, my carry-on bag still fit under the seat in front of me.

Once the plane was airborne, it was time for me to do some portable computing. The first thing I wanted to know was how

easy would the SX be to use on the plane. I found the SX narrow enough to fit comfortably on the fold down tray in front of me. With the battery attached the machine is deeper than the tray which caused the keyboard to be closer to my lap than I'm normally used to, yet not enough to bother me.

The SupersPort SX has a page white backlit screen that makes it easy to read at any angle with various lighting. During my flight, we went from daylight to night time and the cabin lighting changed. To compensate for this change, I adjusted the backlighting and the screen remained easy to read. In fact, the screen was so easy to read I felt like the entire cabin could peer over my shoulder and read the screen as I typed.

Since I started the trip with a fully charged battery I was interested in finding out how long it would last. I used the machine for word processing, spreadsheets and statistics for about 2 hours and the low battery indicator light never came on. Consistently throughout the trip, the battery lasted between 2 and 3 hours per charge.

My first stop was San Francisco, California. Here I had the chance to try out my communications software from my hotel room. Because San Francisco is close to Silicon Valley my hotel gets many computer people with portable computers. Because of this, the hotel has modem jacks in all rooms. (No need for me to rewire the phones at this hotel.) All I needed to do was plug in my modem.

One point to keep in mind when communicating from a hotel. Keep your phone calls short. Hotels typically charge an extra service charge for making long distance phone calls. Some hotels, such as mine, even add a service charge for local calls. I decided to use Prodigy to keep in touch with the people back home. Before I left, I had looked up the local Prodigy access number for San Francisco so my calls wouldn't be long distance. Once I had programmed the local access number into my software everything worked fine.

I spent two days in San Francisco and the SX machine never missed a beat. I normally worked on the machine in the hotel room every night and found all my software ran without any problems. The machine was light enough that I could move it and work from anywhere in the room. The first night I worked at the desk and the next night I worked lying on the bed with the SX on my lap. This SX was definitely mobile in a hotel room.

As I moved around the room I found the mouse to be more trouble than it was worth. To work properly, the mouse needs a completely flat surface. A mouse pad would have been ideal, but I had forgotten to pack one. After the first night of trying to mouse around without a pad, I decided to forgo the mouse and use the keyboard

equivalent commands instead.

On the way back from San Francisco I stopped off in Phoenix, Arizona for a vacation. To me, a vacation is just not the same without a computer. I guess I need to have a computer near me to relax. Since the SX worked well on my trip to San Francisco, I



decided to take it along to Phoenix.

The first day in Phoenix found me at the pool. Here, the SX worked well with the backlighting allowing me flexibility. I could work in direct sunlight for a while, then move into the shade. No matter where I worked the screen could be adjusted for lighting.

That night I took a brief walk around Camelback Mountain. Camelback Mountain is the most famous mountain in Phoenix. Named Camelback for its shape when viewed from a distance. The top of the mountain is the camel's hump, 1,800 feet high. After scouting out the trail up the mountain I decided to return the next day for a climb to the top. I also decided to bring the SX with me. Being mobile from a hotel room is one thing, but would the SX be mobile on top of a mountain?

For the climb up the mountain I decided to carry the SX in a small pack that could be carried on my back. This worked better than my carry-on, since it kept my hands free for the climb. Most of the climb takes place on trails, but there are some spots where the climb is over rocks and nearly vertical. Even with the battery attached, the SX still fit into the backpack.

The next day I returned to the mountain and started the climb. It begins at the base of the camel's neck. Then moves up to the head and down the other side of the neck. From there it's up the hump to the highest part of the mountain. Parts of the trail have steps to climb. Other parts have rocks to climb instead. In all, the climb to the top typically takes two to three hours, and one to two hours to climb down.

New Zenith Data Systems 286 SlimsPort

Zenith Data Systems also sells the SlimsPort 286. This portable computer is for the person who wants something more powerful than the MinisPort, but doesn't need the full power of the SX. With a VGA screen and 286 processor, it can run Windows applications (albeit not in 386/enhanced mode, so there is no multi-tasking). However, in weighing in at just over 9 lbs., including battery, it's several pounds lighter than the SX.

I recently had a chance to work with a SlimsPort for two weeks. The keyboard has been improved with separate cursor arrow keys and separate PgUp, PgDn, Home, and End keys. These keys are combination keys on the SX. The 20MB hard drive worked fine but I filled it up quickly. It seems that Windows applications take up more disk space than comparable character-based applications. For my needs, with a larger hard drive available and true multi-tasking capability, I'd still pick the SupersPort SX.

Right away I felt the weight of the SX. In the backpack it felt heavier than I remembered when in my carry-on bag. After climbing 20 minutes I wished I had chosen the MinisPort instead. I started wondering if climbing Camelback with the SX was a good idea after all. When I would move sideways the backpack would swing back and forth reminding me the computer was still there. After climbing for an hour I started to get used to the weight. It almost felt like the SX was getting lighter the more I climbed. I even got to the point where I didn't notice it at all. I started to think again that perhaps this machine was mobile.

Once I had forgotten the computer was on my back I started to notice how gorgeous Camelback really is. I also started to experience mountain life first hand (pun intended). As I was climbing over rocks I placed my hands between some vegetation. Suddenly, something scampered over my hand. It was a small lizard who had

come out to view this strange human climbing a mountain with a computer.

As I continued climbing, the trail became nearly vertical. I was glad the backpack left both hands free. I don't think I could have made it to the top had I not used some sort of backpack. The weight of the laptop would force the backpack to swing sideways at times and bang into the side of a rock. After passing another lizard sun bathing on the rocks I reached the top of the mountain. The total travel time up the mountain was nearly two and one half hours.

It was time to rest and find out if the SX still worked. I picked the most comfortable looking rock to sit on and started to unpack. With the sun shining and clear blue sky, the view from the mountain top overlooking Phoenix was magnificent. The laptop appeared undamaged from the climb as I placed it on my lap. After turning the machine on, I adjusted the backlighting

for the sun and was ready to work.

As I sat on top of the mountain, I used Windows to multi-task between WordPerfect, Excel, and Powerpoint. As I typed on the computer, I realized how portable this machine was. While heavier than the MinisPort, the SX was still mobile enough to carry up to the top of Camelback. And once at the top, the SX was powerful enough to run the software that I needed.

Just then I was startled as another lizard started to scamper over my feet. When he saw me working on the laptop he stopped. As he cocked his head back and forth he stared at me for several moments. The puzzled look on his face seemed to say, "What a funny place for a computer."

The trip down the mountain took less than 90 minutes. Although banged around from a mountain climb in 90 degree weather, the laptop never missed a beat. Packed in a backpack it traveled well. The VGA backlit screen was easy to read and could be adjusted for different lighting.

In thinking back over my trip to San Francisco and climbing the camel in Phoenix, I've come to a conclusion. Portable computers make it easy to take computing power with you when you travel. There is no doubt in my mind that a light notebook computer is easier to travel with than a heavier laptop computer. Yet, sometimes computing needs can't be met with simple notebook computers. In times that call for multi-tasking and VGA graphics, it takes a more powerful laptop. On this trip, I found that I could have the more powerful laptop computer and not give up mobility. Happy Portable Computing! ✱



"WHAT ARE THESE LAPTOPS I HEAR SO MUCH ABOUT?"

Index to REMark: 1990

Jan Axelson
2209 Winnebago Street
Madison, WI 53704

REGULAR COLUMNS & FEATURES

Buggin' HUG (letters)

Current Contents search software on Z-100, X-10 problem with XT, PC-DOS 4.0, Harvard Graphics on Z-100, Restore, Backup fixes for MS-DOS 3.3+, Feb, p.5

A-Bus article correction, installing hard drives, Jun, p.4

Modem and mouse port, Sprint on Z-100, Intel Inboard 386/PC in H/Z-150, MPI printer parts, X-10 software, DEL character, Packet TNC modem controller, X-10 problem on AT, Jul, p.5

Disk Technician Advanced, Aug, p.4

Heathkit RS-232 Breakout Box, hard drives for ZW-241, Oct, p.4

QuickData catalog, Quattro Pro on Z-100, DOS 3.2 on Z-159, Nov, p.4

IMAGER support, life with a laptop, Dec, p.4

On the Leading Edge (W. Adney)

Computer model numbers, SupersPort 286, MS-DOS 3.3 Plus, Jan, p.55

Lotus 123 V3.0 problem, memory maps, memory usage, MS-DOS 3.3+ GDU & GDUTSR, Feb, p.35

Look-and-feel issue, user interfaces, Foxpro 1.0, EISA and MCA, 386SX, Mar, p.23

Expansion memory, installing a 4MB Z-515 Board in a Z-386/16, Apr, p.29

HyperACCESS/5, Voice Master Key System, Jun, p.25

OS/2, Unix, ZDS Group Sale, ZDS and QED, eaZy PC, Jul, p.33

SupersPort SX, reducing bad sectors on hard drives, SIMMs for the Z-286, Z-386/20/25/33, Sep, p.21

Help for eaZy PC owners, ZDS MS-DOS 4.0, Oct, p.43

ZDS MS-DOS 4.0, COMMAND, SHARE, SHELL, INSTALL, powering up and down, Nov, p.35

Christmas gifts, disk compatibility and DASDRVR.SYS, index tabs for manuals, general goodies, PC Tools, Mace Utilities, HUG software, educational software, Imager, Voice Master Key Sys-

tem, Dec, p.25

Powering Up, Volume 2 (W. Adney)

Using Special Zenith and MS-DOS Key Sequences, Jan, p.9 (also Jul p.6)

How Disks and DOS Work Together, Feb, p.25

What a Disk Really Contains, Mar, p.43

Abort, Retry, Pfu!—COMMAND Error Messages, Apr, p.11

Coping with Common DOS Error Messages, Jun, p.11

Connecting a Modem to Your Computer, Jul, p.23

Connecting a Printer to Your Computer, Sep, p.41

Choosing a Programming Language, Oct, p.33

How to use the DOS Environment, Nov, p.7

Understanding MS-DOS Device Drivers, Dec, p.44

REMark's Remarks (ZDS news — H. Fale)

Groupe Bull sale, Z-386 SX, SupersPort SX & 286e, TurbosPort 386e, Apr, p.5

Zenith/Bull sale, Zenith systems contracts, liquidation and auctions, ZCM-1492, MinisPort RAM upgrade, Z-386/33E (first EISA shipment), May, p.4

Groupe Bull purchase, enhanced MinisPort, MinisPort at U. of California, Windows 3.0 & Asymetrix Toolbox on ZDS systems, ZDS management changes, price reductions, Z-286 LP Plus, upgrades to MS-DOS 4.0 & Windows 3.0, Windows problems, modem on eaZy PC, Oct, p.6

Z-100 Survival Kit (P. Herman)

#10 (how to patch programs for ZPC), Jan, p.65

#11 (problems that require patching for ZPC), Mar, p.35

#12 (Q & A), May, p.33

#13 (Z-100 for new users), Jun, p.7

HARDWARE HOW-TO

Apple LaserWriter II, Using with a Heath/Zenith System, (P. Swayne), Aug, p.39

CD-ROM Setup on a ZDS Computer, (H. Ogg), Dec, p.39

Disk Drive, Upgrading an H/Z-151 with a 1.2MB or 1.44 MB, (R. Maskasky), Jun, p.34

Drive, Adding a 3.5" to an H/Z-158, (E. Piche), Sep, p.13

Electronic Mail Systems, How to Get Plugged Into, (T. Jorgenson), Mar, p.7

Failure, 10 Ways You Can Cause Computer, (R. Brenner), Jan, p.72

Floppies for the Z-100, 5.5 Meg, (G. Elwood), Sep, p.37

Floppy & Hard Drives on Z-181's, Installing External (D. Myers)

Part 1 (research), May, p.21

Part 2 (accessing the system bus), Jul, p.39

Floppy Drive in the Z-151, 6.6 Meg (also Inboard 386/PC), (S. Shapiro), May, p.43

Floppy Drives, Adding External (and Other TurbosPort 386 User Tips), (R. Siebers), Jan, p.7

Graphics Printer or Epson FX (J. Day)

Part 6 (downloading and graphics), Feb, p.28

Part 7 (programs for downloading and graphics), Jul, p.28 (listings, Oct, p.22)

Hard Disk Drive(s) in a Z-386, Installation of, (R. Mueller), Jan, p.75

Hard Drive and Monitor ROMs, Troubleshooting the H/Z-151, (S. Shapiro), Mar, p.5

Hard Drives, Taking the Mystery Out of, (D. Cool), May, p.28

How Components Fail, (R. Brenner), Oct, p.41

Interfacing to the IBM PC, (L. Wilson), Mar, p.11

LaserJet Printers, Programming (A. Neibauer)

Part 1 (basic concepts), Jun, p.31

Part 2 (PCL commands), Sep, p.29

MIDI, Add to your SupersPort, (T. Perdue), Aug, p.9 (foil patterns, Oct, p.25)

Modem, A Data Transfer Solution, (R. Brenner), Dec, p.14

NEC Multisync Monitor, Using on a VGA

Equipped Z-150, (R. Maskasky), Aug, p.45
Novell Netware on ZDS Computers: An Educational Example, (H. Ogg), Aug, p.16
Reports and Custom Printer Drivers, Programming, (A. Neibauer), Apr, p.35
Upgrading Old Computers and Other Challenges, (S. Shapiro), Dec, p.11

HARDWARE REVIEWS

HS-2862 SupersPort 286 VGA, (P. Swayne), Feb, p.7
Octoport, Heath, An Intelligent Switch-box, and More, (T. Perdue), Sep, p.5
Postscript, Implementing, (A. Neibauer), Sep, p.39
Printers for Electronic Publishing at Home, Affordable, (R. Siebers), Jun, p.17

SOFTWARE — HOW TO USE APPLICATIONS SOFTWARE & OPERATING SYSTEMS

Analog Circuit Simulators, What are, (M. Hardwick), Nov, p.11
Backup Utilities, DOS File, (F. Starr), Apr, p.46
Batch File and Menuing Programs, Enhanced, (H. Ogg), Oct, p.27
dBASE III (D. Cool)
 Part 2 (EDIT, BROWSE, APPEND, delete, recall), Feb, p.31
 Part 3 (custom data entry screens, simple programming), May, p.45
 Part 4 (programming), Jun, p.21
 Part 5 (integrated PROJECTS update program), Jul, p.7
 Part 6 (analyzing the project update program), Sep, p.18
 Part 7 (procedure files), Oct, p.10
 Part 9 (making your programs user friendly), Dec, p.35
dBASE, A Versatile Mailing List Program in, (P. Swayne), Aug, p.25
DOS and UNIX (T. Bing)
 Part 2 (features of MKS toolkit), May, p.17
 Part 3 (stalking the great 'awk'), Feb, p.44
DOS Shells, Getting Started with, (A. Neibauer), Dec, p.21
Electronic Mail Systems, How to Get Plugged Into, (T. Jorgenson), Mar, p.7
EMM.SYS, How to Use, (W. Adney), Jan, p.49
Enable, Mouse Capability for, (E. Zimmerman, Jr.), Jan, p.13
GDU (ZDS' General Disk Utilities), Getting Started with (J. Axelson)
 Part 1 (sorting and undeleting), May, p.37
 Part 2 (examining and editing disks, more undeletes), Jul, p.43
Hard Disk Easy, Making Your, (M. Haverstock), Feb, p.18
Menu System, The Best Hard Disk, (P. Walberg), Feb, p.21
MS-DOS File Recovery, (G. Horst), Aug, p.41

MS-DOS, Exploring the Extras in Zenith Data Systems', (J. Axelson), Apr, p.15
New Software, Perils of, (E. Demaree), May, p.25
NUM LOCK Off Revisited, (B. Hall), Jan, p.69
PC Tools Deluxe (R. O'Connor)
 Part 3 (using disk caching, security and maintenance tools), Mar, p.17
 Part 4 (final look at the newest version...5.5), Apr, p.39
TSR Utilities (Managing MS-DOS Resident Programs), (M. Wilson), Apr, p.19
UNIX on an 8088, (J. Bazhaw), Aug, p.21
Upgrading Old Computers and Other Challenges, (S. Shapiro), Dec, p.11
WordPerfect 5.0 and Its Wonders, (S. Brackett)
 Part 1 (basic editing), Sep, p.15
 Part 2 (printing and merging), Nov, p.25
WordPerfect Macros (E. Wiggins)
 Part 1 (introduction), Aug, p.7
 Part 2 (examples, editing), Sep, p.27

SOFTWARE — PROGRAMMING

Apple LaserWriter II, Using with a Heath/Zenith System, (P. Swayne), Aug, p.39
Assembly Language, (P. Swayne)
 Part 2 (more fundamentals), Jan, p.61
 Part 3 (flags, and the instruction set (part 1)), Feb, p.41
 Part 4 (the instruction set, part 2), Mar, p.29
 Part 5 (instruction set, part 3), May, p.9
 Part 6 (instruction set, part 4), Jul, p.20
 Part 7 (instruction set, part 5 (sample program, introduction to I/O)), Aug, p.34
 Part 8 (more ways to say hello), Sep, p.7
BASIC Programs, Command Line Input for Compiled, (K. Granzow), Jun, p.29
BASIC, Single-Byte Integers in, (K. Granzow), Aug, p.31
C++, Introduction to, (L. Wilson), Dec, p.17
Color PAINT.ASM for the H/Z100, Part 2, (S. Vagts), Feb, p.10
Flags: A Tutorial, (G. Hoellerich), Nov, p.44
Fractals on the H/Z-100 (PC or not PC) in C, (R. Rasch), May, p.11
Graph for a Year, (T. Custin), Mar, p.31
Graphics Printer or Epson FX, (J. Day)
 Part 6 (downloading and graphics), Feb, p.28
 Part 7 (programs for downloading and graphics), Jul, p.28 (listings, Oct p.22)
Graphics Viewer for the Z-100, (G. Elder), Nov, p.41
LaserJet Printers, Programming, (A. Neibauer)
 Part 1 (basic concepts), Jun, p.31

 Part 2 (PCL commands), Sep, p.29
Options, What are the, Part 1 (determining computer capabilities), (D. Lind), Nov, p.17
Postscript, An Introduction to, (A. Neibauer), Aug, p.13
Reports and Custom Printer Drivers, Programming, (A. Neibauer), Apr, p.35
Searching for Binaries, (A. Grattan), Sep, p.9
Trip Planner in C, (R. Rasch), Apr, p.23
VGA Graphics Modes in GW-BASIC V3.2, Using, (P. Swayne), Apr, p.27
VGA, Programming with, (M. Mangerson, G. McDonald)
 Part 1 (theory), Jun, p.39
 Part 2 (examples), Jul, p.13

SOFTWARE REVIEWS

Autosketch, Power Drawing for Less, (P. Swayne), May, p.13
Batch File and Menuing Programs, Enhanced, (H. Ogg), Oct, p.27
Calendar Creator Plus, Managing Your Schedule with, (T. Bing), Sep, p.33
COMMAND.COM, A Replacement for (4DOS), (M. Wolfson), Oct, p.17
Counterpoint Menu System, (E. Zimmerman Jr.), Dec, p.31
Desktop Publishing, Cheap Tricks: Inexpensive Utilities for, (H. Ogg), Jun, p.43
Diagnostics for the Heath/Zenith Personal Computer, (R. Brenner), Feb, p.47
Disk Technician Advanced, (R. Brenner), Jun, p.19 (letter, Aug p.4)
First Choice (small business accounting), (D. Pitney), Apr, p.43
Novell Netware on ZDS Computers: An Educational Example, (H. Ogg), Aug, p.16
OS/2 on the Zenith Computer, (H. Ogg), Mar, p.39
Postscript, Implementing, (A. Neibauer), Sep, p.39
QEMM-386 (expanded memory manager), (J. Goldman), Nov, p.31
Unix on a Shoestring (Exploring MKS Toolkit), (B. Rylander), Oct, p.38
Windows: Do you do them?, (J. Bazhaw), Nov, p.47
WordPerfect Executive, (E. Wiggins), Aug, p.37

MISCELLANEOUS

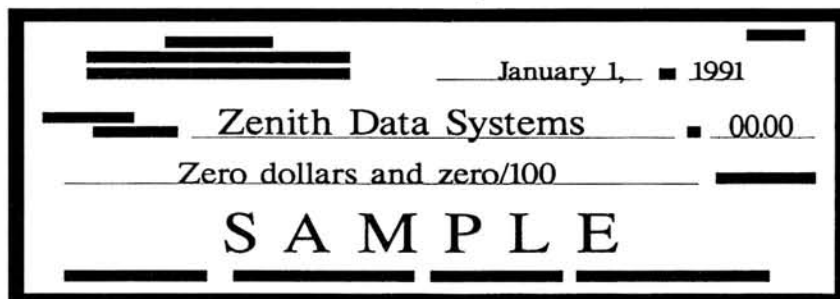
BBS, A Look at the New HUG, (P. Swayne), Jun, p.15
Cartoon Contest Winners, May, p.41
CIS—GO ZENITH problems (BBS's), (H. Fale), Aug, p.5
Computer Operator's Guide to Insanity, (S. Brackett), May, p.7
Heath Users' Group, Welcome to the, Jan, p.4
Heath/Zenith Related Products (HDOS 3.02), (J. Buszkiewicz), Jul, p.47
HUG Local Clubs, Jan, p.16
HUG New Products (YAUD (Yet Another Utilities Disk), HEPCAT Update), Apr, p.4

HUG Software Listing, Jan, p.18
 HUGPBBS Gets a Facelift, (J. Buszkiewicz),
 Jun, p.5
 Index to REMark: 1988-1989, (J. Axelson),
 Dec, p.5
 Port of Call: COM1 (BBS news) (installing
 a 3.5" drive on a Z-248), (L. White), Nov,
 p.5
 I made this index using PC-File Version

5.0, then converted it to a text file for pub-
 lication in REMark.
 If you prefer an electronic index, you can
 download REMINDEX.EXE from the HUG
 BBS. REMINDEX.EXE is a self-extracting file
 that contains REMark indexes for 1988-
 1990 in the form of a PC-File-compatible
 database. Or, send \$6 to Jan Axelson,
 2209 Winnebago Street, Madison, WI

53704, I'll send you the files on a 5-1/4"
 360K MS-DOS floppy.
 Using REMINDEX and PC-File, you can
 quickly search for occurrences of any topic
 or combination of topics you wish.
Note: To use the disk-based index, you
 must have PC-File Version 5.0 or a compat-
 ible database. ✱

All Checks must be made out to
 Zenith Data Systems

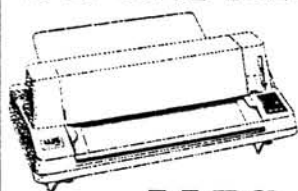


W S Electronics

(513) 376-4348 **** Since 1975 **** (513) 427-0287

1106 State Route 380, Xenia, Ohio 45385

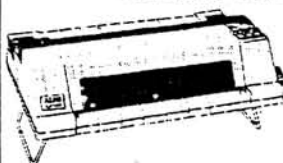
YOU AND YOUR BIG IDEAS.



- * 300 cps draft speed
- * Wide carriage
- * 24 pin printing
- * Front panel controls

ALPS Allegro 500XT™

THE ALPS ASP1600 PRINTER. COSTS VERY LITTLE, JAMS NOT AT ALL.



- * Auto tear bar prevents paper waste. With the flick of a button, your fanfold paper advances to the perforation, then automatically returns to top-of-form position.
- * Rugged 9-pin head delivers crisp output at 192 cps in draft mode, 38 cps in letter quality.
- * Compact 9-lb. body makes for easy portability.
- * Printer stand is built-in.
- * Prints labels easily.
- * Full Epson FX-85 compatibility.

ALPS
 AMERICA
 Built by popular demand.

SPECIAL

Z-315	EMS Kit for Z-159	\$ 10.00
Z-417	H D Controller Z-248	\$100.00
Z-317	H D Controller Z-150	\$ 75.00
ZCM-1400-1	Swivel Base for 1490	\$ 5.00
CB-4364-39	Diagnostics for 184	\$ 5.00
ENABLE 2.0	Word Processing	\$ 99.00

Quantities limited to stock on hand

ATTENTION:

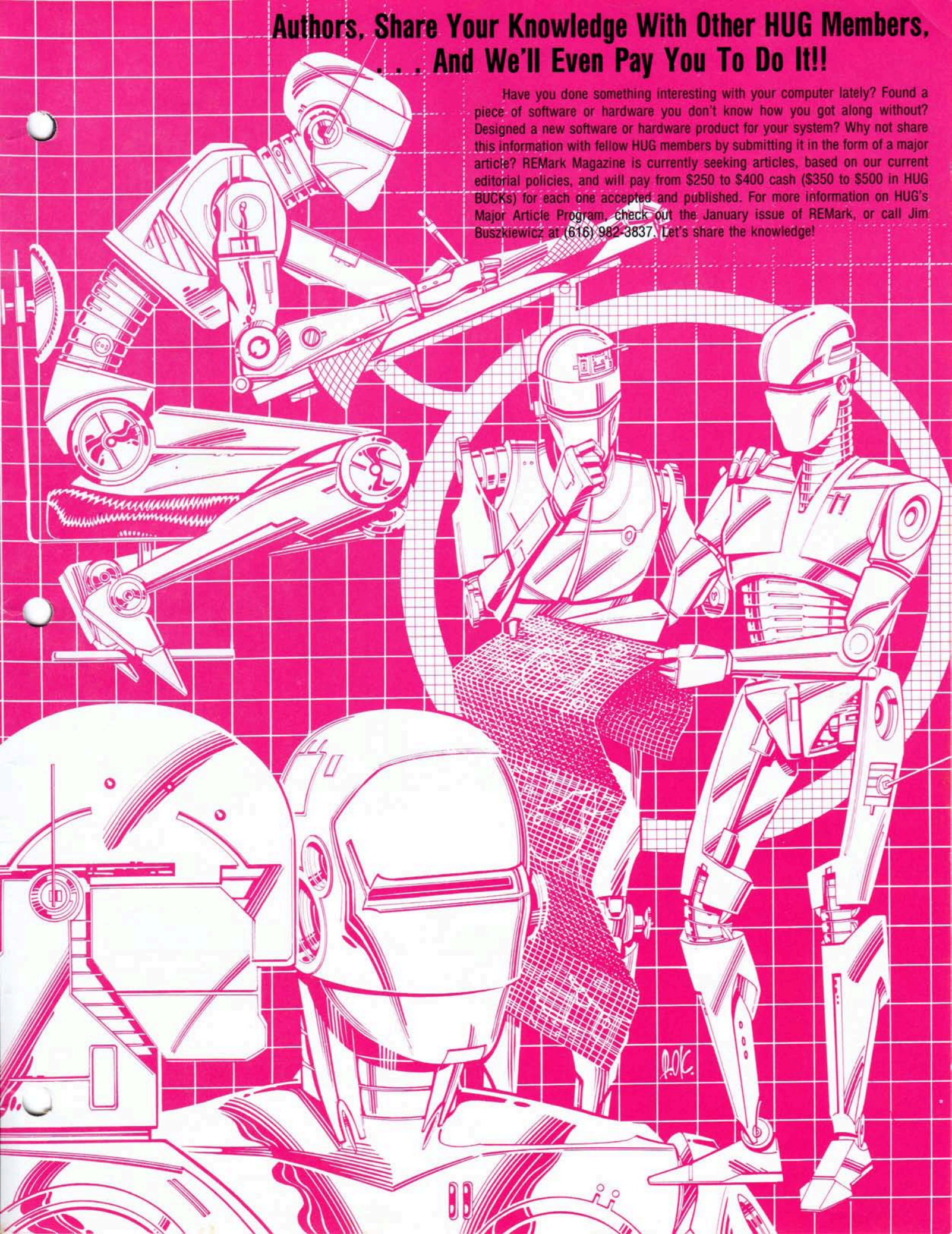
Federal Government Offices
 We stock ALL ALPS Printer Models and we stock
 ALPS PARTS and RIBBONS for all ALPS models
 including your P2000's and ASP-1000's

****GOVERNMENT DISCOUNTS OFFERED****

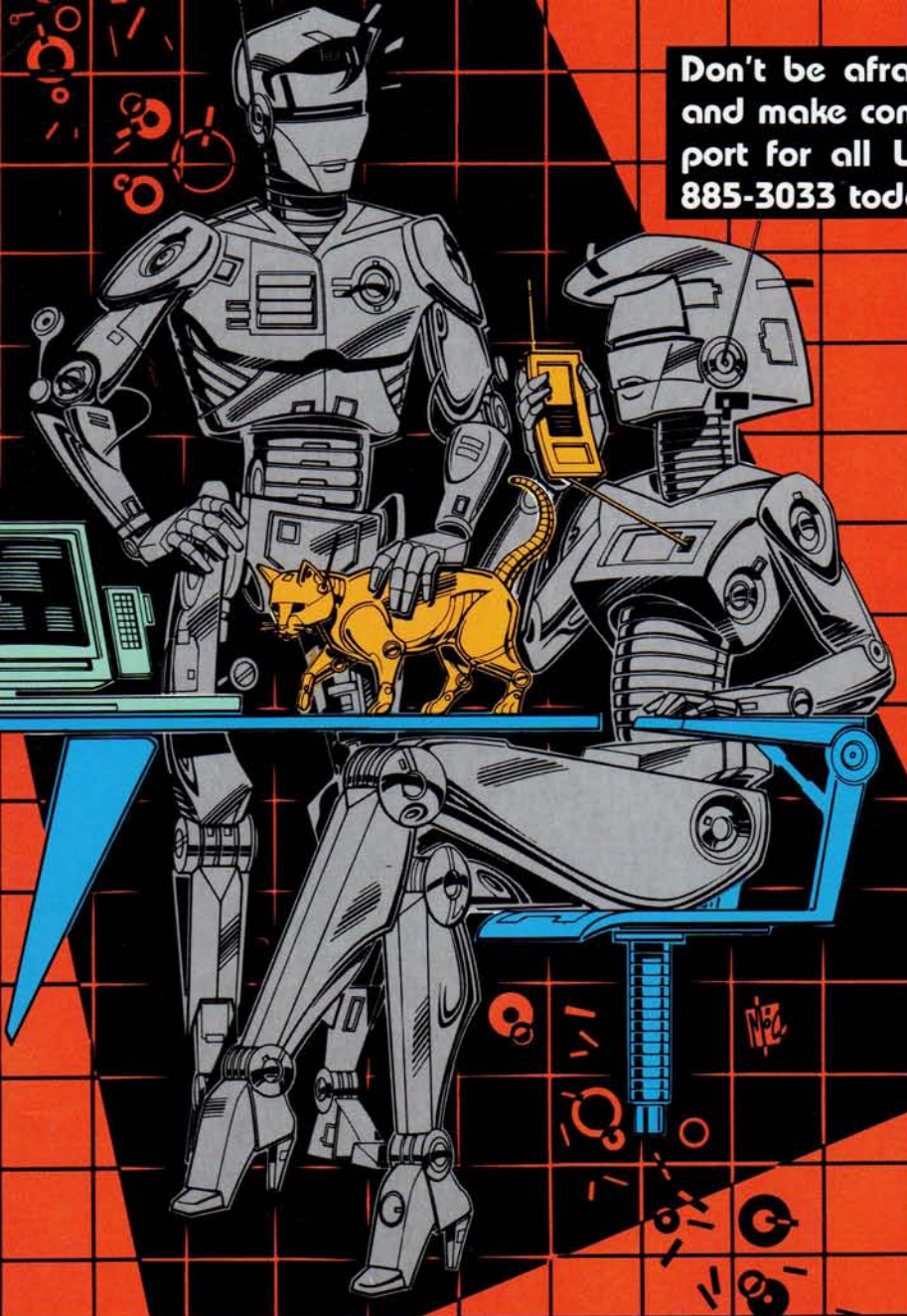
We are looking for good dealers.
 ALPS Authorized Distributor and Service Center

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

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



Don't be afraid to communicate! Get HUGMCP and make contact the easy way. Now with support for all Laptops, order HUG Part number 885-3033 today.



```
HUGMCP Commands
F1 -- Prints This List, Your Storage Buffer Size, And How Many Bytes Are Presently In The Storage Buffer.
F2 -- Allows Sending A Defined Message, Or Character Sequence. These Messages Are Entered Using The (F5) Setup Command.
F3 -- Toggles The Storage Buffer On and Off. When The Buffer Is On, The Char's On The 25th Line Will Be High-Lighted.
F4 -- Allows Saving Data To Disk From The Storage Buffer, Or Directly From The Modem By Way Of XMODEM Protocol.
F5 -- Allows Sending Data From Disk, Using Either XON-XOFF, Which Optionally Can Be Ignored, Or XMODEM Protocol.
F6 -- Enters The Setup Mode So This Software Can Be Configured.
F7 -- Clears Out Any Data That May Be In The Storage Buffer.
F8 -- Send Data In Storage Buffer To Printer.
F9 -- Exits Back To MS-DOS.

Storage Buffer = 524288 Bytes
Storage Buffer Usage = 0 Bytes

Select Message (A-O), (F1) To List, Anything Else To Abort --) _
F1-Hlp F2-Msg F3-Defr F4-Sav F5-Set F6-Cfg F7-Clr F8-Print F9-Exit COM
```

```
Send Configuration Data #1
This function allows the baud rate to be changed. Depending upon which menu screen you're in, normally it'll allow you to set it either 300, 1200, or 2400 baud. Some connections do a half, and it will allow faster baud rates.
This function allows you to change the word length. Normally you'll have 8 bits, but it's adjustable by Modem driver's software, and it is also necessary for XMODEM Protocol to work properly.
This function allows the changes of the word parity. Normally the parity should be set to 0 data bits. This value is acceptable to most modems, but it is necessary for XMODEM Protocol to work properly.
This selection allows you to enter messages which can be automatically sent with the F10 key. Up to 16. To eliminate messages can be used. Selection 0 is special. It should contain the computer's number as determined. Selection 0 is also special. This selection can only be used to test when this program is first executed by selecting the proper option during setup.

Type (SPACE BAR) For New Help, Anything Else To Continue:
F1-Hlp F2-Msg F3-Defr F4-Sav F5-Set F6-Cfg F7-Clr F8-Print F9-Exit COM
```

```
HUGMCP Configuration Menu:
0 -- Modify Baud Rate
1 -- Modify Parity Type
2 -- Modify Word Length
3 -- Modify Or Add Baud-Messages
4 -- Miscellaneous Functions
5 -- Change Screen Color Assignments
6 -- Display Current Configuration
7 -- Make Changes Permanent

Select 0-C, (F1) For Help, Anything Else To Quit --) _

Baud Rate: 19200
Parity: NONE
Word Length: 8
Duplex: FULL
Response In Keyboard Disable: NO
Storage Buffer Data Parity Bit: SET TO ZERO
Send Modem Initialization Text: NO
Write Character: NORMAL
Modem Port Set to: COM1

F1-Hlp F2-Msg F3-Defr F4-Sav F5-Set F6-Cfg F7-Clr F8-Print F9-Exit COM
```

ZENITH
data systems



Groupe Bull

BULK RATE
U.S. Postage
PAID
Zenith Users' Group

POSTMASTER: If undeliverable, please do not return.

\$2.50
P/N 885-2136