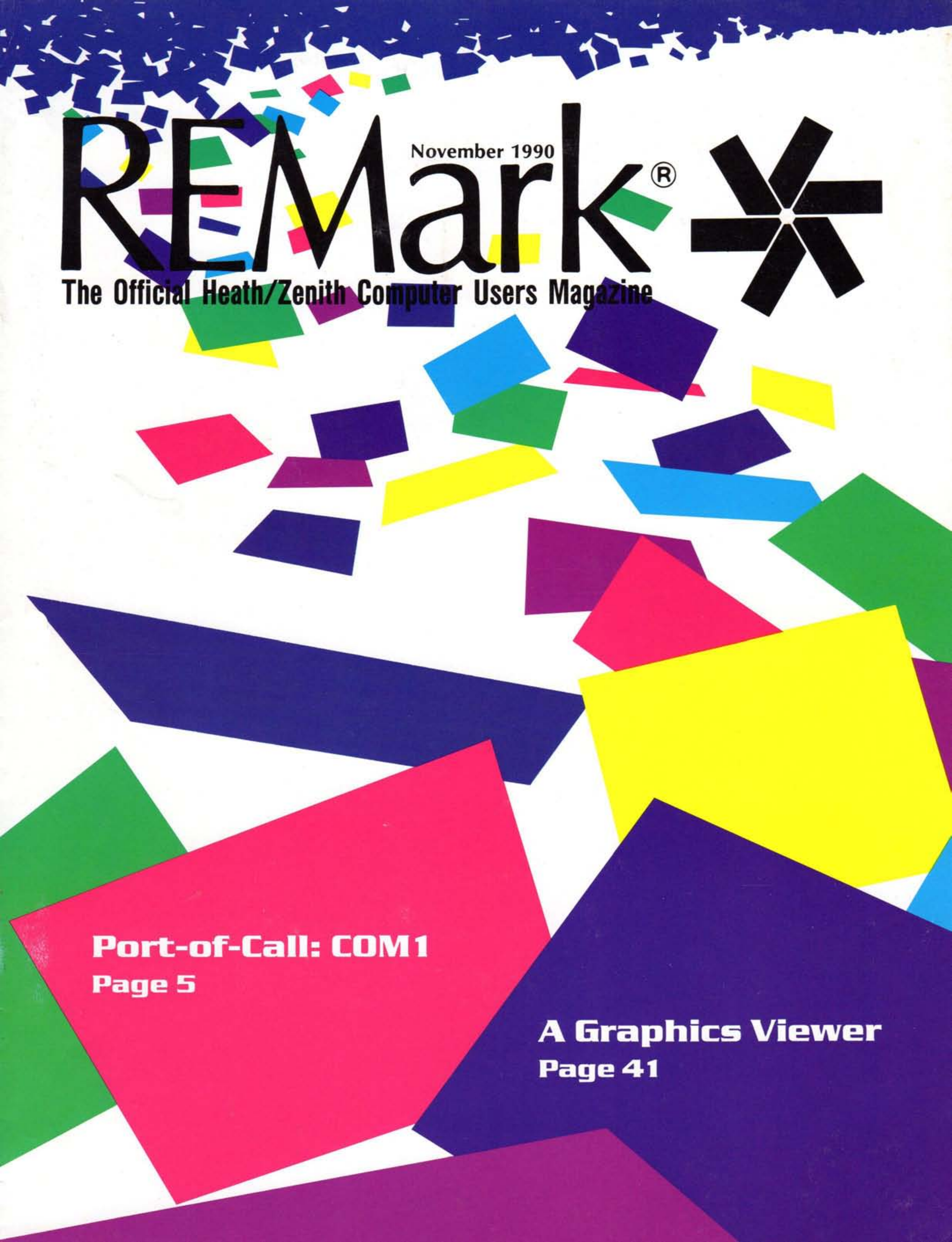
REMark®

November 1990

The Official Heath/Zenith Computer Users Magazine



Port-of-Call: COM1
Page 5

A Graphics Viewer
Page 41

HADES II

Now Version 2.04 is . . .

HOTTER 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
- ✦ A GigaByte MS-DOS 4.0 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 current HADES owners can upgrade their distribution disk for only $15. Call HUG today at: (616) 982-3463, and order 885-3040.

REMark

November 1990

The Official Heath/Zenith Computer Users Magazine

Reader Service No.		Page No.
104	*FBE Research Co., Inc.*	*33*
203	*Hilgraeve, Inc.*	*16*
114	*Micronics Technology*	*40*
117	*Payload*	*34*
149	*W.S. Electronics*	*15*

PC Compatibles

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

PC Compatibles

H/Z-100 & PC Compatibles

H/Z-100 Only

General

Resources

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)

HUG Software Orders
(616) 982-3463

Hardware Questions
(616) 982-3309

Printer
Imperial Printing
St. Joseph, MI

Contributing Editor
William M. Adney

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

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

* U.S. Funds

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

Send Payment to: Heath/Zenith Users' Group
P.O. Box 217
Benton Harbor, MI 49022-0217
(616) 982-3463

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

Articles submitted by users and published in REMark, which describe hardware modifications, are not supported by Heath/Zenith Computers & Electronics Centers or Heath Technical Consultation.

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

HUG

H8 - H/Z-89/90

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

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

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
ADVENTURE	885-1222-[37]	CPM	GAME	10.00
BASIC-E	885-1215-[37]	CPM	LANGUAGE	20.00
CASSINO GAMES	885-1227-[37]	CPM	GAME	20.00
CHEAPCALC	885-1233-[37]	CPM	SPREADSHEET	20.00
CHECKOFF	885-8011-[37]	CPM	CHECKBOOK SOFTWARE	25.00
COPYDOS	885-1235-37	CPM	UTILITY	20.00
DISK DUMP & EDIT UTILITY	885-1225-[37]	CPM	UTILITY	30.00
DUNGEONS & DRAGONS	885-1209-[37]	CPM	GAMES	20.00
FAST ACTION GAMES	885-1228-[37]	CPM	GAME	20.00
FUN DISK I	885-1236-[37]	CPM	GAMES	20.00
FUN DISK II	885-1248-[37]	CPM	GAMES	35.00
GAMES DISK	885-1206-[37]	CPM	GAMES	20.00
GRADE	885-8036-[37]	CPM	GRADE BOOK	20.00
HRUN	885-1223-[37]	CPM	HDOS EMULATOR	40.00
HUG FILE MANAGER & UTILITIES	885-1246-[37]	CPM	UTILITY	20.00
HUG SOFTWARE CATALOG UPDATE #1	885-4501	VARIOUS	PRODUCTS 1983 THRU 1985	9.75
KEYMAP CPM-80	885-1230-[37]	CPM	UTILITY	20.00
MBASIC PAYROLL	885-1218-[37]	CPM	BUSINESS	60.00
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

Price List

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
X-REFERENCE UTILITIES FOR MBASIC	885-1231-[37]	CPM	UTILITY	20.00
ZTERM	885-3003-[37]	CPM	COMMUNICATION	20.00
H/Z-100 (Not PC) Only				
ACCOUNTING SYSTEM	885-8048-37	MSDOS	BUSINESS	20.00
CALC	885-8043-37	MSDOS	UTILITY	20.00
CARDCAT	885-3021-37	MSDOS	BUSINESS	20.00
CHEAPCALC	885-3006-37	MSDOS	SPREADSHEET	20.00
CHECKBOOK MANAGER	885-3013-37	MSDOS	BUSINESS	20.00
CP/EMULATOR	885-3007-37	MSDOS	CPM EMULATOR	20.00
DBZ	885-8034-37	MSDOS	DBMS	25.00
DUNGEONS & DRAGONS (ZBASIC)	885-3009-37	MSDOS	GAME	20.00
ETCHDUMP	885-3005-37	MSDOS	UTILITY	20.00
EZPLOT II	885-3049-37	MSODS	PRINTER PLOTTING UTILITY	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	ENTERTAINMENT	20.00
HELPSCREEN	885-3039-37	MSDOS	UTILITY	20.00
HUG BACKGROUND PRINT SPOOLER	885-1247-37	CPM	UTILITY	20.00
KEYMAC	885-3046-37	MSDOS	UTILITY	20.00
KEYMAP	885-3010-37	MSDOS	UTILITY	20.00
KEYMAP CPM-85	885-1245-37	CPM	UTILITY	20.00
MAPLE	885-8023-37	CPM	COMMUNICATION	35.00
MATHFLASH	885-8030-37	MSDOS	EDUCATION	20.00
ORBITS	885-8041-37	MSDOS	EDUCATION	25.00
POKER PARTY	885-8042-37	MSDOS	ENTERTAINMENT	20.00
SCICALC	885-8028-37	MSDOS	UTILITY	20.00
SKYVIEWS	885-3015-37	MSDOS	ASTRONOMY UTILITY	20.00
SMALL-C COMPILER	885-3026-37	MSDOS	LANGUAGE	30.00
SPELL5	885-3035-37	MSDOS	SPELLING CHECKER	20.00
SPREADSHEET CONTEST PACKAGE	885-3018-37	MSDOS	VARIOUS SPREADSHEETS	25.00
TREE-ID	885-3036-37	MSDOS	TREE IDENTIFIER	20.00
USEFUL PROGRAMS I	885-3022-37	MSDOS	UTILITIES	30.00
UTILITIES	885-3008-37	MSDOS	UTILITY	20.00
ZPC II	885-3037-37	MSDOS	PC EMULATOR	60.00
ZPC UPGRADE DISK	885-3042-37	MSDOS	UTILITY	20.00
H/Z-100 and PC Compatibles				
ADVENTURE	885-3016	MSDOS	GAME	10.00
ASSEMBLY LANGUAGE UTILITIES	885-8046	MSDOS	UTILITY	20.00
BACKGROUND 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
HELP	885-8040	MSDOS	CAI	25.00
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 CATALOG UPDATE #1	885-4501	VARIOUS	PROD 1983 THRU 1985	9.75
HUGMCP	885-3033	MSDOS	COMMUNICATION	40.00
ICT 8080 TO 8088 TRANSLATOR	885-3024	MSDOS	UTILITY	20.00
MAGBASE	885-3050	VARIOUS	MAGAZINE DATABASE	25.00
MATT	885-8045	MSDOS	MATRIX UTILITY	20.00
MISCELLANEOUS UTILITIES	885-3025	MSDOS	UTILITIES	20.00
PS's PC & Z100 UTILITIES	885-3052	MSDOS	UTILITY	20.00
REMARK VOL 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
SCREEN DUMP	885-3043	MSDOS	UTILITY	30.00
UTILITIES II	885-3014	MSDOS	UTILITY	20.00
Z100 WORDSTAR CONNECTION	885-3047	MSDOS	UTILITY	20.00
PC Compatibles				
ACCOUNTING SYSTEM	885-8049	MSDOS	BUSINESS	20.00
CARDCAT	885-6006	MSDOS	CATALOGING SYSTEM	20.00
CHEAPCALC	885-6004	MSDOS	SPREADSHEET	20.00
CP/EMULATOR II & ZEMULATOR	885-6002	MSDOS	CPM & Z100 EMULATORS	20.00
DUNGEONS & DRAGONS	885-6007	MSDOS	GAME	20.00
EZPLOT II	885-6013	MSDOS	PRINTER PLOTTING UTILITY	25.00
GRADE	885-8037	MSDOS	GRADE BOOK	20.00
HAM HELP	885-6010	MSDOS	AMATEUR RADIO	20.00
KEYMAP	885-6001	MSDOS	UTILITY	20.00
LAPTOP UTILITIES	885-6014	MSDOS	UTILITIES	20.00
PS's PC UTILITIES	885-6011	MSDOS	UTILITIES	20.00
POWERING UP	885-4604	N/A	GUIDE TO USING PCS	12.00
SCREEN SAVER PLUS	885-6009	MSDOS	UTILITIES	20.00
SKYVIEWS	885-6005	MSDOS	ASTRONOMY UTILITY	20.00
TCSPELL	885-8044	MSDOS	SPELLING CHECKER	20.00
ULTRA RTTY	885-6012	MSDOS	AMATEUR RADIO	20.00
YAUD (YET ANOTHER UTILITIES DISK)	885-6015	MSDOS	UTILITIES	20.00

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

Now Available!

HUG 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 HUG's MSDOS software is available on 3-1/2" micro-floppies too! When ordering, just add a "-80" to the 7-digit HUG part number. For the standard 5-1/4" floppy, just add a "-37".

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

ORDERING INFORMATION

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

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

BUGGIN' HUG

QuikData Catalog

Dear HUG:

As regular advertisers in REMark, I may not need this, but I would appreciate it if you would let your readers know that our new 7/90 catalog has been prepared and mailed out. If any of your readers are not on my mailing list, they can simply write to me at the address below or call and request a free no-obligation catalog. There are sections on the 8-bit, Z-100, and PC/XT/AT computers, both hardware and software. A collection of some pretty good stuff, continuing our tradition of supporting the H/Z 8-bit machines and our dedicated commitment of full support of all H/Z systems. If they also request our liquidation list we will gladly send that also. It also has some good specials liquidated at below cost prices, and it is updated constantly. Or they can call our Bulletin Board at (414) 452-4345 and download the liquidation list.

Henry Fale
QuikData Computer Services, Inc.
2618 Penn Circle
Sheboygan, WI 53081
(414) 452-4172

Quattro Pro on the Z-100

Dear HUG:

This letter is to relate my recent experience in installing Quattro Pro on my Zenith Data Systems Z-100. My low-profile Z-100 has 768 kb RAM, the Heath HR-108 speed-up kit, and has the UCI EasyPC board (ROM version 1.65) installed for IBM compatibility. I also have the HUG ZPC emulator on the Z-100 side. The EasyPC has never failed to handle IBM software, until this experience. I followed the installation instructions with Quattro Pro in order to install the spreadsheet on one of my two hard drives (split between the Z-100 and IBM sides of the machine). But when I tried to run the program, it would display the opening layout of the spreadsheet, but lock the cursor and require a reset using the on-off switch; hitting control-reset hangs the machine as it tries to reboot, or sometimes a hard disk failure message occurs. Something in Quattro Pro actually affects the EasyPC ROM output because strange lines show up on the UCI opening menu that lets the user choose between the Z-100 and IBM sides of the machine, when rebooting using control-reset. Neither UCI nor Borland could help me, including the suggestion to use bare-bones AUTOEXEC.BAT and CONFIG.SYS files.

In frustration, I decided to see if Quattro Pro would work on the Z-100 side with the ZPC emulator. I tried it and to my pleasant surprise, it does, although somewhat sluggishly. So at least I could use it with the original Z-100 if I wanted to. I then reset the machine using control-reset, rebooted the IBM side, and lo and behold, Quattro Pro worked! I have repeated this several times. In order to use Quattro Pro on the IBM side of the machine, I have to first boot the Z-100 side, then boot the IBM side using control-reset. If I boot the IBM side first, Quattro Pro always locks the machine. I don't know the whys or wherefores of all this, but this might help other Z-100 users if they experience the same problem. Quattro Pro is the first program I've tried for an IBM-XT that has not worked fine on my machine with the EasyPC hardware, including Lotus 2.2, SYSTAT and WordStar 5.5.

Sincerely yours,
Wayne C. Huber, Ph.D., P.E.
University of Florida
424 A.P. Black Hall
Gainesville, FL 32611-2013

DOS 3.2 on the Z-159?

Dear HUG:

I am having some annoying problems with my Z-159-13 that some of your readers or local experts might cast some light on. It's equipped with 640K of memory, an MS mouse, a 20 Meg. disk set up as C and a 30 Meg. disk set up as D. The first problem involves the installation of MS-DOS Version 3.1 which I purchased from Zenith Data Systems several months ago. The machine came with Version 3.1 and has performed very well for me, although I quickly filled up the 20 Meg. hard disk. When I got the opportunity to upgrade to Version 3.2, I jumped at the chance because of its improved capabilities, (I seem to remember reading in REMark that it would support installation of different hard disks, and 3.5" drives).

I have tried several times to install the new OS, but the machine simply will not boot from the disk after the installation. I consistently get the error message "I/O Error" at each attempt. I can boot up from the distribution floppy by using the built-in monitor program. After doing so, I also can access the hard disk and both read and write to it. I have followed the setup program very carefully. It seems to operate normally, detects all the proper ports, etc., and gives no error messages. I have even tried to reformat the hard disk using the "format /s" command, but the result is the same. I called ZDS shortly after I had the problem and was told that I must use the local repair service here in the LA area. I tracked down the number and was told to bring the machine in, but I would have to pay by the hour for troubleshooting. I have reinstalled DOS 3.1 now and the machine works fine. As a check on my copies of DOS 3.2, I temporarily installed them on an IBM AT at work with no problem at all. This tells me that there must be some incompatibility between my Z-159-13 and ZDS' DOS 3.2. Has anyone else had these problems?

The Z-159 memory card is equipped with sockets for two additional banks of memory, but the manual mentions the need for special chips that must be installed before the additional memory can be accessed. Where are these available?

Any help you or the other HUG members can provide would be appreciated.

Conrad C. Chatburn
2522 South 4th Avenue
Arcadia, CA 91008

Port-of-Call: COM1

Laura White
759 Polfus
Benton Harbor, MI 49022

I'm glad you've taken some time for "Shooting the Bull." Hopefully, you'll be joining me monthly to catch up on some of the more prominent issues on the COM1 bulletin board.

The bulletin board (formerly HUG), as you may know, is an electronic bulletin board which can be accessed using a computer and a modem. Through the COM1 BBS, Huggies can shop for Heath/ZDS products, both old and new, download files, exchange information, or just "shoot the bull" using the message system. There's constantly an influx of useful information on the system and this article is designed to publish it for everyone.

For those of you who do have modems, this monthly piece is designed to give you a hard copy of important issues, by topic, for use as a reference. Those of you who don't have access to the bulletin board, this will give you a chance to benefit from fellow huggies' experience and knowledge. Each month, I will re-cap the issues that seem to be plaguing several users.

This month's problem turned out to be a relatively easy fix, however, the amount of advice our user received was unbelievable. He was installing a 3.5 inch high density drive, into his Z-248 when he ran into some trouble.

His drive was functioning as a 720k reading, writing, and formatting. However, at 1.4M, our user was running into a brick wall. Whenever he tried to format a high density disk, using MS DOS or PC Tools, he continually received the command, "Invalid Media or Track 0 Bad disk unuseable." After reading his manuals, and trying several "experiments", our frustrated user dialed up the bulletin board looking for help.

The advice he received covered hardware as well as software problems. Initially, he was asked to confirm that he had used SETUP to tell the hardware that the drive was a 1.4M drive and not a 720K. After ruling out the simple problems, our "pro-bono consultants" began looking at the more complex. When trouble-shooting, especially when installing a new piece of hardware, sometimes, it seems to be easier to rule out hardware compatibility problems first, and then move on to potential software/firmware problems—and that's exactly where our frustrated user's advice began.

One huggie called in and informed him that Zenith Data Systems chose to use Pin 34 to detect a media change, and also the number of holes in a disk (an extra hole tells the system that it's working with a 1.4M disk). Another caller expanded on this same subject by telling him to check if the computer believed it was connected to a high density drive by using DSKSETUP.COM to see what DOS thinks is set up for the drive. If the drive is set up as 720K, change it to 1.4M. If 1.4M is not given as an option, the machine may need a newer ROM BIOS in order to be compatible with a high density drive.

Continuing to rule out potential hardware problems, it was recommended that he check the drive, cable and connector by testing them in another machine. If this wasn't possible, he was told to try an entirely different cable. (Some Huggies seem to have at least two complete systems and a full inventory of spare parts!) Before investing in another drive, or getting involved with software problems, it was highly recommended to check the drive-select switch found on the side of the high density drive itself. For all computers with a twisted cable, the switch should be set to the second position (1 if it goes from 0 to 3, or 2 if the switch goes from 1 to 4). If the drive uses jumpers instead of a switch, the same applies. Also, when connecting the ribbon cable to a drive or drive controller, pin #1 is always at the colored edge.

Equipped with this information he called in several days later and informed everyone that he had a new drive and was planning to start from scratch. At that point, he was given advice on potential software problems. One huggie wrote in and noted that the caller was trying to format using PCTOOLS. Someone informed him that there were some glitches with PCTOOLS version 5.5, and that once the consultant received version 6.0, the glitches went away.

Our friend was also informed that there might be a problem with the syntax of his FORMAT command. If the parameters aren't specified correctly, the system gives users the message "Invalid media or Track 0 Bad-disk may be unuseable." Thinking this might be the problem, the format command x:/N:9/T:80 (where "x" equals the floppy drive letter) was passed on in hopes that this would take care of it. When formatting low density disks in ZDS high density drives, you have to tell the system to use 80 tracks with 9 sectors per track. The consultant expanded a bit and told our friend that the T:80 usually wasn't needed because both types of disks use 80 tracks.

He tried all of this advice as well as a few other pieces and found nothing was solving his predicament. Finally, someone called in and offered him one last piece of advice; add to the CONFIG.SYS the following information: DRIVPARM=/D:0 (or 1) /F:7. According to the MS-DOS manual, this tells the system that drive 0 or 1 is a 1.44 meg drive.

It turns out that, while troubleshooting, he failed to follow the most important rule. It appears that one time when he was running some tests, trying to pin-point the problem, he changed more than one variable. The problem turned out to be just what everyone was assuming wasn't the problem—at some point he changed the SETUP to 720K and didn't return it to the 1.4M after completing some particular tests.

For those of you who have spent hours and hours troubleshooting, you know just how frustrating it can be. Fortunately for our friend, his problem was finally solved and it turned out to be an "easy-fix". This happens to be the case in so many instances and unfortunately, users are quick to think its something complicated and purchase a new part to replace whatever seems to be broken.

For example, someone called in a didn't know what to do about a sticky key

Continued on Page 30

The other cats get to sing along!

That's because HEPCAT runs **with** your other programs, not **over** them. HEPCAT (HUG Engineer's and Programmer's CAlculation Tool) is a powerful pop-up calculator for all Heath/Zenith MS-DOS and Z-DOS based computers. Unlike other pop-up calculators, HEPCAT does not stop the currently running program while it is popped up. That means that you can do calculations while your computer is busy with something else. For example:

- While Lotus (tm) is loading a huge spreadsheet, you can check your kid's math homework.
- While Dbase (tm) is sorting a large database, you can add up some grocery prices.
- While your computer is busy compiling one program, you can work on number base conversions needed for another program.

HEPCAT is safe to pop-up during just about any running program — even during disk activity. And HEPCAT has other features the other guys can't touch.

HEPCAT gets along with everyone . . .

HEPCAT supports more video configurations than any other pop-up, and always pops up in the current video mode, rather than forcing the screen into a text mode as other pop-ups do. It also works properly with more programs than any other pop-up. You can pop up HEPCAT over Microsoft Windows (tm) and many other programs that other pop-ups can't work with, and even over some other pop-ups.

HEPCAT works harder . . .

HEPCAT provides a multi-function floating point calculator and a programmer's binary calculator that work together to do more than the basic four (+, -, *, /). The floating point calculator 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. It also includes the following conversions: degrees-radians, radians-degrees, Celsius-Fahrenheit, Fahrenheit-Celsius, centimeters-inches, inches-centimeters, meters-feet, feet-meters, kilometers-miles, miles-kilometers, grams-ounces, ounces-grams, kilograms-pounds, pounds-kilograms, milliliters-fluid ounces, fluid ounces-milliliters, liters-quarts, quarts-liters. 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. The HEPCAT window uses less screen space, too. It shows you more real information than other pop-up calculator displays, but it doesn't waste space by showing you a keypad layout. You already know what your keypad looks like! HEPCAT is easier to learn, too, with commands that make sense.

If you are tired of pop-ups that can only sing solo, give HEPCAT a try. HEPCAT is available from HUG as part no. 885-3045-37 for $35.00. It works on any Z-100 PC, Z-200 PC, or Z-100 (not PC) system and any version of MS-DOS or Z-DOS.

Powering Up Volume 2

How to Use the DOS Environment

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

There are many reasons that you may need to know more about the DOS "environment." Perhaps the best one is to know how to cope with the "Out of environment space" error message, which users are seeing more frequently than ever. This error message is occurring more frequently because more application programs are taking advantage of the DOS environment space. More programs are also using a "Go to DOS" feature within the program to temporarily transfer control to the DOS Command Interpreter (COMMAND.COM). From a user perspective, it is easy to recognize that you are using this feature when you must use the EXIT command to return to your original application program.

This article includes information on how the SET, PATH, and PROMPT commands work with the environment space. As you will see, the size of the environment space can be increased by using the COMMAND command (i.e., COMMAND.COM) in a special way. And this article also includes the use of the new CALL command that is available in all versions of DOS 4.0, including ZDS MS-DOS 4.0. Let's begin by reviewing some of the basic information that is also found in the original Powering Up book in Chapter 14 (Other Useful DOS Commands).

The DOS Environment Space

Technically, the DOS Environment Space is just a reserved area of memory used by the Command Interpreter COMMAND.COM to store information that is used by DOS and can be used by application programs, if they are programmed to use it. For DOS, the environment space is used to store the exact location (i.e., path) of COMMAND.COM so that the transient portion can be reloaded when necessary. The environment space also stores the default PATH value which is a null path (i.e., "No path"), although it can be changed with the PATH command. And if you have used the PROMPT command, that value is also stored in the environment space. In general, the SET command can be used to display or change the contents of the environment space. To see what an environment space actually contains, all you need to do is enter the SET command by itself, and Figure 1 shows the contents of the environment space on my Z-386/16 system.

```
COMSPEC=C:\COMMAND.COM
PROMPT=$P $Q$Q$G
PATH=C:\PF6;C:\;C:\DOS;C:\WS6;C:\BATCH;C:\UTIL;C:\HUG
```

Figure 1
Environment Space Example

The COMSPEC= (COMMAND.COM Specification) parameter is always present and is established when DOS is booted. It is used to tell DOS exactly where to find the Command Interpreter COMMAND.COM when the transient (i.e., temporary) portion must be reloaded. In this example, DOS will expect to find it in the root directory on drive C. For a floppy disk system, you would see a COMSPEC=A:\COMMAND.COM listing, which means that DOS will always expect to find the Command Interpreter in the root directory on drive A. If the Command Interpreter is NOT found, then DOS will display a message like Insert disk with COMMAND.COM in drive A and strike any key when ready. This message most often occurs in a floppy disk system because you have changed the disk in drive A to run a batch file, and the new disk does not contain COMMAND.COM. You can fix that problem permanently by simply copying COMMAND.COM to any floppy disk that contains a batch file if there is enough disk space. If there is not, then you will need to keep your bootable system disk handy and insert it whenever required so that DOS can reload the Command Interpreter.

The second line in Figure 1 shows the PROMPT command I use, and it is a line in my AUTOEXEC.BAT file that is just PROMPT $P QQ$G that displays my preferred form of the DOS command prompt which looks like:

```
C:\DOS ==>
```

The last line in Figure 1 shows the PATH command I use, and it is also a line in my AUTOEXEC.BAT file which is entered as shown. If you remember that DOS searches the PATH from left to right in order to find a command that is entered, you can see that I use the \PF6 path (for ProFinder included with WordStar 6.0) most often, which is why it appears first. I use commands from the \BATCH, \UTIL, and \HUG subdirectories less frequently, perhaps once a day, which is why they are near the end of the search path.

As you can see, each of the lines in the DOS environment space is really simple and generally relates to commands you already know. But there is a way to add information to the environment space if you need to.

Adding to the Environment Space

More and more of the current application programs are taking advantage of the environment space for one reason or another. Some programs even require a specification be added to the environment space. For example, I use the Mace Utilities and it requires an addition to the environment space using the SET command as documented in the manual. To keep things simple when I use my system, I have defined a batch file called MACEIT.BAT as shown in Figure 2.

```
SET MACE=C:\MACE
CD C:\MACE
C:
```

Figure 2
MACEIT.BAT Example

The first line of Figure 2 is the SET command which tells the Mace Utilities that the programs are in the \MACE subdirectory on drive C. The next two lines simply are used to *automatically* change to the subdirectory that contains the Mace Utilities on my system. Although I can obviously enter each of these commands on the DOS command line, it is much faster for me to use a batch file and I do not have to remember exactly what must be done each time I want to used one of the utility programs. To illustrate what this SET command does, Figure 3 shows what happened to the environment space on my system after I run this batch file.

```
COMSPEC=C:\COMMAND.COM
PROMPT=$P $Q$Q$G
PATH=C:\PF6;C:\;C:\DOS;C:\WS6;C:\BATCH;
     C:\UTIL;C:\HUG
MACE=C:\MACE
```

Figure 3
Modified Environment Space Example

Compare Figure 1 with Figure 3, and note that the fourth line MACE=C:\MACE was added to the environment space as specified by the SET command in Figure 2. The information displayed in Figure 3 was again obtained by entering the SET command by itself, which displays the current contents of the environment space. Since I have mentioned the SET command several times, let's review a little more about it.

The SET Command

SET is an internal DOS command. It can display the current environment space contents as shown in Figures 1 and 3, it can insert a string value as shown in Figure 2 or it can delete a string value from the environment space. A specific string value may be required by an application program, such as the SET command shown in Figure 2. The syntax for the SET command is shown in Figure 4.

SET	(Display environment values)
SET name=string	(Define string value)
SET name=	(Delete string value)

Figure 4
SET Command Syntax

As previously mentioned, the SET command by itself will display the current contents of the environment space, such as shown in Figures 1 and 3.

The second command line is used to set a specified name equal to a specified text string value. In Figure 2, the name MACE was set to the string value of C:\MACE, which is just the drive and directory that contains the Mace Utilities on my system. This command form can also be used to replace an existing text string with a new string value. If I had a newer version of the Mace Utilities that I wanted to use, I could enter a SET command of SET MACE=C:\MACENEW, which would replace the old string value (i.e., C:\MACE) with C:\MACENEW. To be sure that the replacement was done correctly, I could use the SET command by itself to display the contents of the environment space as usual.

The third command line is used to completely delete the string value for a specific name from the environment. For example, I could delete the MACE=C:\MACE (shown in Figure 3) line from the environment space by entering the SET MACE= command. If I again entered the SET command by itself, I would see that this line had been deleted, and the contents of the environment space would be as originally shown in Figure 1.

Aside from the command syntax, there are three important points that you need to know to use the SET command correctly. First, any application software (e.g., the Mace Utilities) which uses this feature must be especially designed and programmed to access the environment space. If the application is not programmed to use the environment space, then you will not accomplish much of anything by trying to force a program to accept a value you have entered.

Second, if you need to use a SET command, the documentation for that application software should tell you precisely what to do and the exact syntax required to do it. Be sure to read the manual and follow the instructions exactly or the application may not work correctly or at all. In addition to the Mace Utilities, some programming compilers allow you to use the SET command to define a location for temporary files that are generated during a compile. You may find it is easier to set up a batch file like I did in Figure 2 for the Mace Utilities or you may want to include all required SET commands in the AUTOEXEC.BAT file.

And third, the SET command has one idiosyncrasy that you should know about. Like all DOS commands, the SET command can be entered in uppercase or lowercase letters. Regardless of which case is entered, the name value is ALWAYS converted to uppercase because it is stored in the environment space that way. However, the string value is NOT converted to uppercase. That is, the string value is stored EXACTLY as entered. Whether or not the lowercase entry causes a problem depends on how your specific application uses the string, but you should remember that the string value is stored exactly as you typed it. Again, be sure to read the manual and follow the instructions exactly or the application may not work correctly or at all. If the example in the manual shows the entire SET command in uppercase, then I recommend you enter the entire command in uppercase.

At this point, we have covered the practical aspects of how to use the environment space with the SET command. Additional information on using the PATH command is available on page 43 of the original (Volume 1) Powering Up book. And additional information on using the PROMPT command is included in the last chapter of this series.

More on the Environment Space

All DOS versions have had an environment space. When DOS version 2.0 was released, the user could access the environment space with the SET, PATH, and PROMPT commands. In most version 2 DOS releases however, the size of the environment space was fixed, and there was no way to change its size. Unfortunately, this size limitation complicated the use of some programs because there are occasions when you may need a larger environment space if you use complex PATH and PROMPT commands, in addition to one or more SET commands.

The size limitation problem was fixed in the release of DOS version 3, but it was not very well documented and is still not very well understood. You can actually change the size of the environment space using the Command Interpreter COMMAND.COM as you will see later in this article.

To understand the space limitation in the environment, it is necessary to briefly review how COMMAND.COM works. The Command Interpreter has three different parts: the initialization portion, the permanent portion, and the transient portion. The initialization portion is only used during the boot process and is responsible for such things as the AUTOEXEC.BAT file execution. The permanent portion is permanently resident in your computer's system memory in a *low address* that is near the beginning of RAM or 0 K. As a matter of fact, this part of low memory also contains the BIOS and the System Kernel, in addition to the resident part of COMMAND.COM. The permanently resident part of COMMAND.COM is memory-resident in much the same way as the TSR (Terminate and Stay Resident) programs that were discussed in Chapter 12 of the original (Volume 1) Powering Up book.

The last part of COMMAND.COM, the transient portion, is also loaded into low memory. This portion is transient (i.e., not permanent) because an application program can actually use this memory if needed and destroy (by overwriting) the transient portion of the Command Interpreter. For that reason, DOS always attempts to reload the transient part of COMMAND.COM, and the whole purpose of the COMSPEC= line that you saw with the SET command tells DOS where to find COMMAND.COM so that it can be reloaded. The actual amount of memory required for COMMAND.COM depends on which DOS version you are using. Later DOS versions usually take slightly more memory than earlier versions because they generally have more features. In any case, let's take a more detailed look how to use the Command Interpreter.

Using the Command Interpreter

The Command Interpreter can be executed just like any other DOS command by simply entering COMMAND. Each time you enter COMMAND, you load another copy of COMMAND.COM. By performing a simple experiment, you can even determine exactly how much memory COMMAND.COM requires. First, boot up your system as usual, and run the CHKDSK command. Record the number displayed as the *free bytes* of memory. Now enter COMMAND followed by another CHKDSK command. Again record the number of *free bytes* of memory, and note that it is less than the first number. Subtract the two, and that's how much memory COMMAND.COM requires. At this point, you have loaded two copies of the Command Interpreter. If you have ZDS MS-DOS 4.0, you can use the MEM/PROGRAM or MEM/DEBUG command to show that two copies of COMMAND.COM are actually resident in memory.

Now you have loaded another copy (called a secondary copy) of COMMAND.COM into your computer's memory, but how do you get rid of it? Simple, just enter the EXIT command to remove the secondary copy of COMMAND.COM. Again, you can use the CHKDSK command to verify that the number of *free bytes* of memory is the same as the original number when you booted the system. If you have any programs that allow you to *temporarily* exit to DOS, they also load a secondary copy of COMMAND.COM, which is why you must enter the EXIT command to return to the original program. You can also load any number of copies of the Command Interpreter and remove each of them from memory with the EXIT command. If you need additional information on the EXIT command, it can be found on page 139 in the original (Volume 1) Powering Up book. Let's get back to COMMAND.COM and see how to use it to cope with the *Out of environment space* error message.

Out of Environment Space Error

The *Out of environment space* error message can occur whenever you exceed the default capacity of the memory reserved for the environment space. In most current DOS versions, including ZDS MS-DOS 4.0, the default value is 160 decimal bytes, regardless of what you see in the manual. The error message can occur whenever you make significant increases to the length of the information entered with the PATH and PROMPT commands, but it most often occurs when additional information is added with the SET command. It is easy to temporarily increase the size of the environment space by using the general syntax as shown in Figure 5.

```
COMMAND/E:nnn[/P]
```

Figure 5
COMMAND Command Syntax

This command form is used to load a secondary copy of the Command Interpreter and expand the environment space with the /E (Environment) switch. In current DOS versions (i.e., ZDS MS-DOS 4.0 and PC-DOS 4.0), the value for nnn ranges from 160 (the default) to 32768 in decimal bytes. To temporarily cope with an *Out of environment space* error message in a current DOS version, all you need to do is enter a command like COMMAND/E:256 which will load a secondary copy of COMMAND.COM and expand the environment space from 160 (the default) to 256 decimal bytes. Earlier DOS versions used a paragraph specification as discussed on page 138 of the original (Volume 1) Powering Up book.

The /P (Permanent) is shown as optional because it is used to *permanently* load the secondary copy of COMMAND.COM into memory. As used here, the word *permanently* means that it cannot be removed with the EXIT command. And although you can temporarily expand the environment space using a form of the command shown in Figure 5, there is a much better approach if you want to permanently change the environment space size each time the system is booted. That better way is to add a SHELL= command to your CONFIG.SYS file.

Using the SHELL= Command

Adding a SHELL= command to permanently increase the size of the environment space is easy, and it will prevent an occurrence of the *Out of environment space* error message. To increase the size of the environment space to 256 decimal bytes, you can use the syntax shown in Figure 6.

```
SHELL=C:\COMMAND.COM/E:256/P
```

Figure 6
SHELL= Command Example

I recommend that you add that line near the beginning (i.e., line 1 or 2) of the CONFIG.SYS file, and be sure that it precedes any DEVICE= command. As shown in Figure 6, this example assumes that COMMAND.COM is in the root directory on drive C, so you will have to change the drive letter if you use a floppy disk system. While we are on the general subject of CONFIG.SYS and hard drive systems, let's take a quick look at one of the new features of current DOS versions, specifically ZDS MS-DOS 4.0.

The SHARE and INSTALL= Commands

If you have a bootable hard drive partition that is larger than 32 megabytes and you install ZDS MS-DOS 4.0, you will see a *Warning! SHARE should be loaded for large media* message. Without getting into the technical details, there are lots of ways to fix that problem, including the obvious one of just entering the SHARE command. After considerable experimenting, I suggest using the new INSTALL= command that is available in ZDS MS-DOS 4.0. Since we have been discussing the CONFIG.SYS file, I have shown a complete sample of a CONFIG.SYS file including the addition of SHARE with the INSTALL= command as Figure 7.

```
INSTALL=C:\DOS\SHARE.EXE
SHELL=C:\COMMAND.COM/E:256/P
BUFFERS=35
FILES=40
BREAK=ON
```

Figure 7
Example CONFIG.SYS File

I used the INSTALL= command to load SHARE from my \DOS subdirectory, and you may have to change that to \BIN if you use the ZDS MS-DOS installation program. The values shown for the BUFFERS= and FILES= commands are ones that I have determined (by experiment) work best for my system and applications. The BREAK= command simply provides better control of the CTRL-BREAK (or CTRL-C) feature of DOS.

Although it is not strictly associated with the DOS environment, let's take a quick look at the new CALL command that is part of COMMAND.COM in the latest DOS versions including ZDS MS-DOS 4.0.

The CALL Command

If you refer to Figure 9 on page 138 of the original (Volume 1) Powering Up book, you will find the syntax used to load a secondary Command Interpreter which is: **COMMAND/C** [d:][path]**your-command**.

The /C (Command) switch loads a secondary copy of the Command Interpreter, executes your-command, and exits back to the previous program, which is usually the PRIMARY Command Interpreter that displays the DOS prompt. In general, the /C switch returns to the *program* that was running at the time this command line was executed, and this *program* may be a batch file in addition to the usual COM and EXE programs. The use of the COMMAND/C syntax allowed you to have one batch file be able to execute another batch file, and then return to the next line in the original batch file as discussed on page 140 of that book. In other words, you could CHAIN one batch file to another without losing your place in the original batch file.

Chaining one batch file to another using the COMMAND/C syntax is illustrated in Figure 8.

```
ECHO OFF
REM This batch file will start a word processor or
REM spreadsheet depending on the command line options.
ECHO For word processing: START W filename.typ
ECHO For spreadsheet    : START S filename.typ
IF %1==W COMMAND/C C:\WS\WORDSTAR
IF %1==S COMMAND/C C:\SC\SUPERCAL
ECHO Word Processing or Spreadsheet completed
ECHO ON
```

Figure 8
Batch File with COMMAND/C

Please note that this is the SAME example that was shown in the original Powering Up book as Figure 12. As before, this simple batch file is designed to call either WORDSTAR.BAT or SUPERCAL.BAT, depending on whether the command line contains W or S. This makes it easy to create and use specific batch files for each one of your application programs. The batch file for a specific application (e.g., WORDSTAR.BAT or SUPERCAL.BAT) may contain commands to change to appropriate subdirectory, install a keyboard macro utility, and make an automatic backup copy of the document to a floppy disk.

The whole idea of inserting COMMAND/C in each line that calls a batch file is to load a secondary copy of the Command Interpreter so that it *remembers* the place where it left the original batch file, such as the example shown in Figure 9. By inserting COMMAND/C before each *call* to a batch file, you can even continue to execute multiple levels of batch files if you are so inclined. This example batch file will execute the last two ECHO commands or any others that you care to add. Beginning with all version 4 DOS releases, including ZDS MS-DOS 4.0, there is another way to accomplish the same thing.

CALL is an internal DOS command (i.e., part of COMMAND.COM) that performs the same function as the COMMAND/C command, and the syntax is shown as Figure 9.

```
CALL [d:][path]batch-filename
```

Figure 9
CALL Command Syntax

The CALL command can only be used for files that have a BAT file type or extension. That is, the batch-filename is just the *first part* (i.e., filename) portion of the file name. For example, if you wanted to execute the SUPERCAL.BAT file using the CALL command, the correct syntax is:

```
CALL SUPERCAL
```

Of course you can precede the batch file name with a drive (d:) or path specification if required, just like you can for your-command (e.g., SUPERCAL or WORDSTAR) as illustrated in Figure 8.

Why would you use COMMAND/C instead of the CALL command? The reason is that the COMMAND/C syntax will work with any valid *command* file which has a COM, EXE or BAT file type. The CALL command ONLY works with files that have a BAT file type.

When used to call a batch file, the CALL command syntax is identical to that used when COMMAND/C form is used as shown in Figure 10.

```
ECHO OFF
REM This batch file will start a word processor or
REM spreadsheet depending on the command line options.
ECHO For word processing: START W filename.typ
ECHO For spreadsheet    : START S filename.typ
IF %1==W CALL C:\WS\WORDSTAR
IF %1==S CALL C:\SC\SUPERCAL
ECHO Word Processing or Spreadsheet completed
ECHO ON
```

Figure 10
Batch File with CALL Command

The example in Figure 10 shows that the COMMAND/C syntax has been exactly replaced by the CALL command, and that is the only change from the example shown in Figure 8. For batch files, you can use either CALL or COMMAND/C to chain them together. For COM and EXE files, remember that you must use the COMMAND/C syntax.

As before, the batch files presented here were for illustration only, and they are not good examples of how to write a batch file. For example, each *IF* statement should have been duplicated with a lower case *w* and *s* so that a user would not have to be sure to enter those values in capital letters. The whole idea was to present the technique and syntax requirement to successfully execute multiple batch files with the COMMAND/C and CALL commands.

Powering Down

Device drivers supplied with MS-DOS are sometimes puzzling to many users, and the next article will help you how to understand and use some of the standard device drivers included with ZDS MS-DOS. If you have any questions about anything in this column, be sure to include a self-addressed, stamped envelope (business size preferred) if you would like a personal reply to your question, suggestion or comment.

Products Mentioned

HUG SOFTWARE

Powering Up (885-4604)	$12.00

Heath/Zenith Users' Group
P. O. Box 217
Benton Harbor, MI 49022-0217
(616) 982-3463 (HUG Software only)

SOFTWARE

MS-DOS Version 4.0 (List price)	$149.00
(Mail order with update card only)	49.00

3.5-inch version (OS-103-MS)
5.25-inch version (OS-105-MS)
Heath/Zenith Computer Centers
Heath Company Parts Department
Hilltop Road
St. Joseph, MI 49085
(800) 253-0570 (Heath Catalog orders only)

What are ANALOG CIRCUIT SIMULATORS?

Michael A. Hardwick
Atrium Electronics
2302 5th Street NE
Salem, OR 97303-6832

Have you ever attempted to optimize the performance of a circuit using several custom coil or capacitor values? Have you built coils 'til your hands were sore, or waited weeks for oddball parts from faraway sources? Have you blown out handfulls of expensive semiconductors in the effort to track down an elusive bug in the design of a ________ (fill in the blank)? These are a few of the reasons why electronic engineers prefer to simulate circuits, rather than rely exclusively on "breadboard" testing.

Even if you're not an engineer, circuit simulation software can be the most powerful tool in your kit, because it will allow you to evaluate the performance of many more circuits than you would be able to construct and test in a given amount of time. Analog circuit simulation programs have been around for a long time. The most widely recognized simulator is SPICE (Simulation Program with Integrated Circuit Emphasis), originally developed at the University of California at Berkely in the late sixties and early seventies (ref.1). Possibly the greatest reason for the existence of circuit simulation tools is that Integrated Circuit designers HAD to have such a tool-there is no reasonable way to bench test many proposed IC designs because the parasitic (or "stray") inductance and capacitance of the breadboard setup would corrupt circuit operation.

Several low-cost simulation programs have appeared on the market recently. The greatest number of them run on PC/XT/AT boxes, and this discussion will focus mainly on those products. A few are available for Atari, Macintosh, and other popular desktop computers. They deliver varying degrees of performance and ease of operation. In this article, we'll take a close enough look at them to help you decide whether you can use one, and help you choose the best one for your needs.

How Do They Work?

A simulation program typically takes your input in the form of a list which describes the components and interconnects (a netlist), combined with a few simple commands that specify the type of analysis desired. See figures one and two for a sample schematic and a simulator input file which represents the same circuit.

The simulator solves a matrix of equations which mathematically express the circuit operation. A DC analysis would find the bias currents, DC voltages, and power dissipations of components in the simulated circuit. Solutions are worked out repeatedly for a range of frequencies (AC analysis) or a succession of points in time (Transient analysis) to gain more information about the circuit's behavior. The types of analyses performed are closely parallel to bench testing procedures using conventional test instruments such as voltmeters, signal generators, oscilloscopes, and spectrum analyzers.

Figure three is an example AC analysis report, showing just voltage magnitude in dB at one node. The output of a circuit analysis program is generally in the form of a list of numbers, which can be difficult to interpret unless you know exactly what to look for. Better programs either include or offer as an option the ability to display results graphically. Data from the output file of figure three is displayed graphically in figure four.

Graphic display of simulation results shouldn't be considered a luxury. The Conehead mass quantities of output that may be generated by these programs often cannot be interpreted efficiently without graphics. The older SPICE version 2G6, when used barefoot, offers only character graphics. Considering the culture in which it was developed (mainframe computers), that was probably an appropriate method at the time.

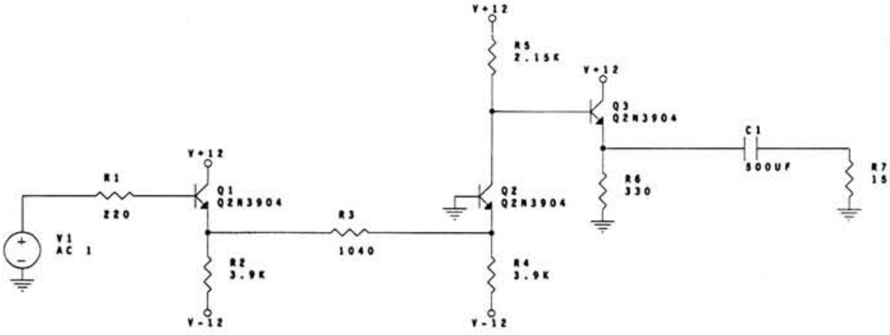
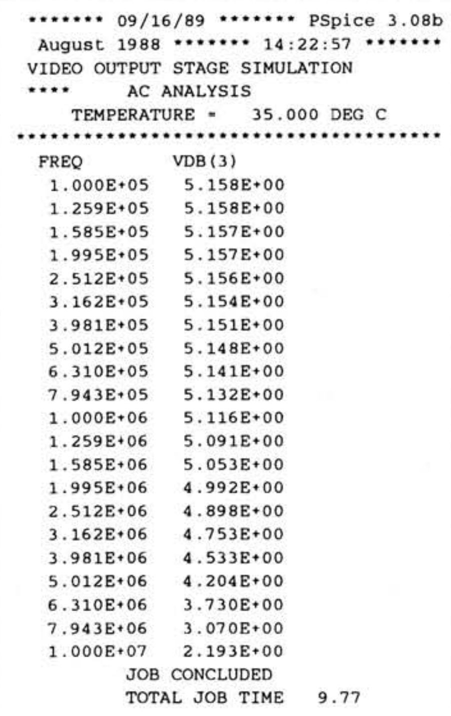


Figure 1. A sample schematic.

```
******* 09/16/89 ******* PSpice 3.08b
 August 1988 ******* 14:22:57 *******
VIDEO OUTPUT STAGE SIMULATION
****       CIRCUIT DESCRIPTION
***********************************
.TEMP 35
.AC DEC 10 100K 10MEG
.LIB QNOM.LIB
.PRINT AC VDB(3)
.PROBE
V+12   8    0         12
V-12   9    0         -12
R3     6    7         1040
R7     3    0         150
R5     8    1         2.15K
R1     5    4         220
R2     6    9         3.9K
R4     7    9         3.9K
R6     2    0         330
C1     2    3         500UF
V1     5    0         AC 1
Q1     8    4    6    Q2N3904
Q2     1    0    7    Q2N3904
Q3     8    1    2    Q2N3904
.END
```

Figure 2. A sample simulator input file.



```
 ******* 09/16/89 ******* PSpice 3.08b
  August 1988 ******* 14:22:57 *******
 VIDEO OUTPUT STAGE SIMULATION
 ****       AC ANALYSIS
      TEMPERATURE =    35.000 DEG C
***************************************
  FREQ        VDB(3)
   1.000E+05   5.158E+00
   1.259E+05   5.158E+00
   1.585E+05   5.157E+00
   1.995E+05   5.157E+00
   2.512E+05   5.156E+00
   3.162E+05   5.154E+00
   3.981E+05   5.151E+00
   5.012E+05   5.148E+00
   6.310E+05   5.141E+00
   7.943E+05   5.132E+00
   1.000E+06   5.116E+00
   1.259E+06   5.091E+00
   1.585E+06   5.053E+00
   1.995E+06   4.992E+00
   2.512E+06   4.898E+00
   3.162E+06   4.753E+00
   3.981E+06   4.533E+00
   5.012E+06   4.204E+00
   6.310E+06   3.730E+00
   7.943E+06   3.070E+00
   1.000E+07   2.193E+00
           JOB CONCLUDED
           TOTAL JOB TIME    9.77
```

Figure 3. Sample AC analysis report.

How Much Do They Cost?

The lowest priced simulator, ACAP, is $43.95, while the highest-priced ones evaluated for this article are PSpice with Probe (its graphic display companion) at $1400, and Micro-CAP III at $1495. In between lies quite a range of price/performance tradeoffs. Table one presents a matrix of the pertinent features of several simulator products. Aside from basic program cost, you should consider the added price of a graphic display program unless you already have one that will work. BV Engineering's PCPLOT is actually a general purpose graphing tool. Many simulators, including Micro-CAP III, LCAS, ECA-2, and Z/SPICE, include graphics as part of the standard program package.

Some of the better simulators REQUIRE a math chip in your computer. This would be an 8087 or 80287 in the case of XT or AT boxes, and it is not usually included with the computer as originally purchased. There is almost always a socket for the math chip on the main board (check your owner's manual). If you decide to add a math chip to take advantage of the vastly improved simulation speed it provides, be prepared to spend $100 to $400 extra. A worthwhile side benefit of installing the math chip is that some other programs will run faster, such as drafting CAD (Computer Aided Design) tools and spreadsheets.

Other items that affect cost are your display and output devices. CGA (Color/Graphics Adapter) is an adequate display for the low-cost products, and the DOS GRAPHICS.COM utility can be used for printing graphic output. Some simulators will use a "Herc" card (also known as Monochrome/Graphics adapter). EGA and VGA display adapters also support CGA, while allowing higher resolution if your software provides it. If you need higher resolution in your display, or publication-quality hardcopy, it can be had for a price! Check with the vendors on these points.

Memory requirements tend to be large for this class of software, but ACNAP will operate with 256K. ACNAP is limited to 50 nodes (a node is a connection between two or more circuit elements) and 200 components, however it has the capacity to "chain" simulations by applying the output file from one run as the input signal for the next run. The chaining capability may also allow ACNAP to process real-world signals. That has real-interesting possibilities! ACAP probably will run in a small memory machine, as it allows the user to arbitrarily set the maximum circuit size to some value below its specified limit of 40 nodes.

Circuit simulators like SPICE are capable of generating such an enormous quantity of analytic data that a floppy disk would run out of room to store the files. The program itself is no pipsqueak. Therefore, a hard disk is mandatory equipment for the larger simulation jobs. These jobs will usually be ran on an AT or 386 machine anyway, because the number-crunching speed becomes essential. In Table One, the minimum class of machine required to run the program is noted, but the software vendor should always be consulted in order to avoid disappointing results.

How Is the Input File Made Up?

Somehow, the schematic of a circuit must be converted into a netlist before the computer can simulate it. The most common netlist format is SPICE. There are basically two ways to do this: If you have general purpose schematic capture software, such as OrCAD SDT, Omation Schema, or Phase Three Capfast, you could obtain a netlist in SPICE format from a schematic file automatically. Intusoft offers SPICE_NET, a schematic capture program specifically tailored for generating SPICE input files. If you don't have access to one of these tools, it is a simple matter to sketch your circuit on paper, number the nodes, and type in a netlist using any text editor that will produce a "clean" (having no embedded formatting commands) ASCII file. The order of nodes or components in a netlist is usually inconsequential to the simulator.

A conventional SPICE style input file adds processing commands for the simulator to the basic netlist. In figure two, commands set the TEMPerature to 35 degrees C, call for AC analysis at 10 points per frequency DECade from 100KHz to 10MHz, specify a model LIBrary, call for PRINTout of AC analysis results at node 3 in DeciBels, and cause a Probe data file to be generated. Node 3 is the high side of R7, which is the output test point for this circuit.

Changes in component values or connections can be made before running a simulation by simply editing the netlist. Some simulators are integrated with a text editor or schematic editor, or provide a

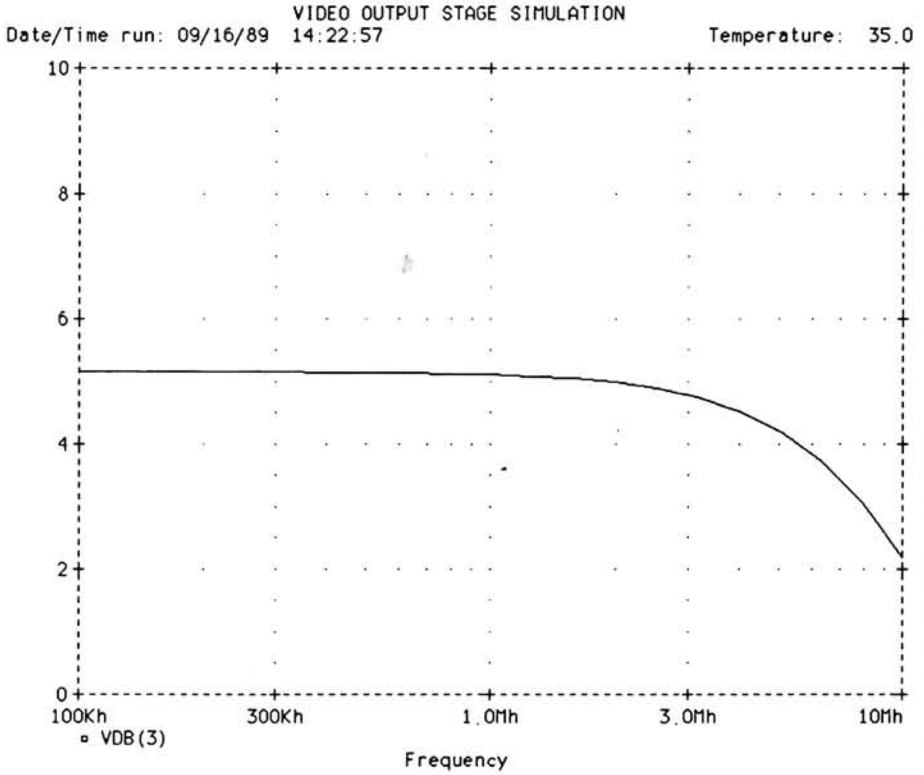


Figure 4. Graphic display of the data in figure 3.

PRODUCT VENDOR	PRICE	GRAPHICS?	MATH CHIP REQ'D?	HOST	MAX. NODES	COMPONENTS	AC	DC	TRAN	MONTE CARLO	RAM	MODEL LIBRARY?	INPUT METHOD	COMMENTS
ACAP DYNACOMP 178 PHILLIPS RD. WEBSTER, NY 14580	$43.95	NO	NO SEE TEXT	PC MAC CP/M	40		X			X		NONE	INTERNAL PREFERRED	BASIC SOURCE CODE
ACNAP BV ENGINEERING 2023 CHICAGO AVE. SUITE B13 RIVERSIDE, CA 92507	$149.95	PCPLOT $125.00	TWO VERSIONS	PC MAC CP/M	50	200	X		SPP $149.95	X	256K	EXAMPLES ONLY	INTERNAL ONLY	
DCNAP BV ENGINEERING	$124.95	PCPLOT $125.00	TWO VERSIONS	PC MAC CP/M	50	200		X			256K	EXAMPLES ONLY	INTERNAL ONLY	
ECA-2 TATUM LABS, INC. 1478 MARK TWAIN CT. ANN ARBOR, MI 48103	$675.00	YES	OPTIONAL	PC MAC	500 NODES IN 512K		X	X	X	X	256K	NONE(?)	INTERNAL AND EXTERNAL	
IS_SPICE INTUSOFT PO BOX 6607 SAN PEDRO, CA 90734	$95.00	INTU_SCOPE $250.00	REQ'D	PC			X	X	X	PRE_SPICE $200.00	640K	PRE_SPICE $200.00	SPICE_NET $295.00 AND EXTERNAL	
IS_SPICE 386 INTUSOFT	$386.00	INTU_SCOPE $250.00	287, 387, OR WEITEK WTL1167	386 ONLY	1000+		X	X	X	PRE_SPICE $200.00	2MEG EXT.	PRE_SPICE $200.00	SPICE_NET $295.00 AND EXTERNAL	MEANINGFUL SPEED INCREASE
KCAP DYNACOMP	$49.95	YES	N/A	ATARI ONLY			X					NONE	INTERNAL ONLY	
LCAS DATUM SYSTEMS PO BOX 4720 SANTA CLARA, CA 95054	$149.95	YES	OPTIONAL	PC	64	128	X				640K	NONE(?)	INTERNAL PREFERRED	EASY TO LEARN
LINCAP DYNACOMP	$49.95	CHARACTER AND GENERIC	NO(?)	PC	20						256K			
MICRO-CAP III SPECTRUM SOFTWARE 1021 S. WOLFE RD. SUNNYVALE, CA 94086	$1495.00	YES	OPTIONAL	PC	75 AC 150 TRAN 150 DC		X	X	X	X	640K	250 MODELS	INTERNAL GRAPHIC ONLY	MICRO-CAP II AVAILABLE FOR MAC. $895.00
NVSPICE 2G.6 NORTHERN VALLEY SOFTWARE 28327 ROTHRICK DR. RANCHO PALOS VERDES, CA 90274	$95.00	CHARACTER	OPTIONAL	AT			X	X	X		640K	NONE	EXTERNAL ONLY	
NVSPICE 3B.1 NORTHERN VALLEY SOFTWARE	$325.00	YES NVNUT	TWO VERSIONS	AT		>100 MOSFETS	X	X	X		640K	NONE	EXTERNAL ONLY	FASTER THAN 2G.6
PSPICE MICROSIM CORP. 20 FAIRBANKS IRVINE, CA 92718	$950.00	PROBE $450.00	REQ'D	PC			X	X	X	X	512K	2000+ MODELS	EXTERNAL ONLY	MAC II VERSION AVAILABLE
SPICE 3B.1 UC BERKELY ERL 479 CORY HALL, U OF C BERKELY, CA 94720	$150.00	YES NUTMEG	REQ'D	AT			X	X	X			NONE	EXTERNAL ONLY	SOURCE CODE ONLY
Z/SPICE ZTEC 6745 LINDLEY AVE. RESEDA, CA 91335	$300.00	YES	OPTIONAL	PC	SEVERAL HUNDRED		X	X	X		640K	Z/LIB $195.00	INTERNAL AND EXTERNAL	DOES DISTORTION ANALYSIS

Table 1. Features and prices of several circuit simulator products.

"menu driven" interface to the user. This is great for the novice, but may prevent the efficiencies of other entry methods from being realized. Before you buy, find out if the input format for a product is human readable, and whether it can be generated independently of the simulator.

OrCAD SDT and PSpice is such a popular combination that one company (ref. 13) brought out software to bind them into a system with streamlined user interaction. The trend toward sane user interfaces seems to be just gathering steam.

What Components Are Allowed?

The most basic simulators accomodate inductors, capacitors, resistors, and some element that may be used to simulate components with gain. ACAP provides a VCCS (Voltage Controlled Current Source). A signal source is always available to drive the input node of the simulated circuit. The better simulators include transistor, diode, FET, transformer, delay line, and other models. Important parameters of a component model can generally be tweaked to reflect real component performance (ref. 11). In addition, subcircuits are usually allowed. The subcircuit is a powerful tool which allows the user to create accurate (and possibly complex) models of a component or portion of a circuit, but doesn't force him to type the entire description into the netlist every time its needed.

A key issue affecting your choice of a simulator may be the size of its model library. Lack of a model library, in fact, is the biggest shortcoming of all low-cost simulation tools. ACNAP provides a few examples of transistor models, while PSpice provides thousands of assorted component models. If you expect an economy simulator to give reasonable results in complex cases, be prepared to learn a lot about modeling the components you use. The software vendors frequently offer or will recommend books on the subject (ref. 2). Your local library may also have magazines which carry articles on component modeling and circuit simulation. A good example is (reference 3).

Due to the increasing popularity of SPICE-like circuit simulators, many component manufacturers now make SPICE models available to the engineering community. As an example, Texas Instruments offers over 20 standard Op-Amp models. Sometimes these models are published in databooks (ref's 4 & 5). Intusoft sells its models unbundled (part of PreSpice) from its versions of SPICE. At least two independent vendors offer model libraries for use with SPICE derivatives.

MicroSim offers Parts (as an option for PSpice), a program which will accept published data sheet values and convert them into SPICE model parameters. Another high-end product, Micro-CAP III, includes the Parameter Estimation Program as a standard part of the $1495 package.

Non-electronic components, such as axons and cells for neural networks (ref. 9) or motors (ref. 6) and speakers, may be simulated if their characteristics can be expressed in electronic terms. This isn't too complicated, yet it opens up some extraordinary possibilities.

If your application mainly involves non-electronic system simulation, TUTSIM (ref. 10) may be the ideal product for you. TUTSIM is $595, or $40 for a fully functional (limited to 15 blocks) evaluation version. Its position in the universe is about midway between a general purpose programming language and a dedicated electronic simulator like SPICE. TUTSIM allows for user defined functional blocks (in C or FORTRAN), real-time I/O, thermodynamic property blocks, etc. The standard blocks encompass a vast range of functions. Nearly any system with components which may be described mathematically can be efficiently simulated.

Incidentally, TUTSIM Products publishes very informative promotional material. BV Engineering and Intusoft also combine liberal doses of information with their sales pitches. This approach to marketing is too advanced to survive in our galaxy, so take advantage without delay.

Are They Accurate?

Low-cost simulators can deliver numeric accuracy equal to the most expensive ones. In a direct comparison of ACAP with PSpice on a simple LC filter circuit, the numeric results were identical. A more pertinent question has to do with one's skill in representing a real circuit with simulated parts, and whether the model parameters accurately represent real component parameters (are all important parasitics represented?). Experience has shown that simulation can deliver highly reliable results in the hands of an expert, but most users still rely on the breadboard for final verification when possible (ref. 12). Oscillation in an emitter follower stage is an example of the type of problem that may not be predicted by simulation. If the user goes back to the simulator and upgrades his circuit model to finally erase all differences with the breadboard, he will become more expert.

The better simulators deliver more kinds of analysis, in more detail, and may present it in a more digestible form. These things can have a bearing on the reliability of a simulation.

An entertaining (sometimes) aspect of most simulators is that wildly unreasonable quantities are handled with nary a complaint. Imagine simulating an audio amplifier that delivers MEGAWATTS of power into a .001 Ohm load! Massive errors may go undetected if the user lacks experience with simulation tools, but in reality it seems to be a minor problem. The next generation of CAE (Computer Aided Engineering) software should slay this beast.

How Long Will It Take?

Execution time is a function of circuit complexity as well as machine speed. A very simple circuit can be simulated sufficiently in a few seconds or minutes. Big circuits using complex models may take hours. Circuit simulators do a lot of floating point math, so they are very compute-intensive. This means that CPU (and coprocessor) efficiency will be the major determining factor in run time, while disk access speed and other computer performance parameters have much less effect. The folks at MicroSim have benchmarked quite a variety of computer systems with PSpice, as reported in their newsletter. A math coprocessor chip will be your best investment with these programs. When choosing a simulator, find out whether it requires the math chip, and if not, whether it will use one that you install later. ACAP is distributed as BASIC interpreter source code, which slaps it with a severe speed handicap. This problem can be largely overcome by compiling it with a compatible BASIC compiler. QuickBASIC will use the math chip if it's present, and was found to compile ACAP without modification. Atrium Electronics (ref. 7) offers ACAP with a compiled version on the same disk. Having possession of the source code may allow you to customize this very economical program for some specific application.

Many circuit simulators have a facility to operate in batch mode, which will allow you to run gargantuan simulations overnight. Such a capability is needed especially in cases where many simulations must be completed in order to make a "Monte Carlo" analysis. The effects of random changes in component values are evaluated by this method (see below), in order to predict the spread in circuit performance that can be expected to occur in production.

What Kind of Output is Available?

Low-cost simulators typically leave out some kinds of analysis, and may not provide a lot of flexibility in presenting the results. The capabilities of low-cost simulation tools vary from basic AC or DC analysis to nearly the full complement of a mainframe SPICE. This could include DC Sweep, Operating Point, AC Response, Noise, DC Sensitivity, Small Signal Transfer Function, Transient Response, Distortion, Fourier, Monte Carlo, and Temperature.

The AC analysis includes magnitude and phase vs. frequency as a minimum. Most AC simulators also report group delay, which is especially handy for some kinds of filter design work. Some AC simulators will report impedance, but this can be obtained indirectly from those that do not. Transient response is the time domain response of a circuit, similar to what you would see on an oscilloscope connected to a node in a real circuit. Where this capability is provided, there will also be some means of stimulating the circuit model with waveforms you can specify. Fourier is a frequency domain view of the Transient analysis, similar to the display on a spectrum analyzer. Even PSpice leaves out the direct analysis of distortion, however.

Monte Carlo analysis got its colorful name from a form of statistical sport practiced there. It refers to a method of predicting circuit performance variation by substituting randomly chosen values within the tolerance ranges of the specified components, then running simulations with the random values until a sufficient body of data has accumulated.

ACAP is an AC simulator only, and provides no reasonable method to get graphic output. It produces a series of little reports which would have to be plotted manually. It will compute AC response for a range of frequencies, and it will report in dB if desired, but it won't take logarithmic frequency steps. A final strong point of ACAP is its ability to accept a component value tolerance and generate statistical reports on the variations of node voltages (Monte Carlo analysis).

KCAP is similar to ACAP, but adds graphics while giving up the component tolerancing feature. It runs only on Atari machines.

ACNAP is a great improvement over ACAP, but it still does only AC analysis, and it requires the companion program PCPLOT to get graphic output. The vendor offers additional programs which will do DC analysis and derive the time domain (oscilloscope-like) response of circuits. This family of programs considered together, would deliver nearly all of the features one could wish for. The total price of BV Engineering's package puts it well beyond the price of some competing packages, but if you need only part of the set, or wish to build it up incrementally, this may not be a disadvantage.

LCAS integrates the graphics, and has some other interesting capabilities such as PLL (Phase Locked Loop) analysis. It isn't extensible like ACNAP, but it's easy to learn and use.

Since SPICE is in the Public Domain, a number of small businesses have grown up around the concept of marketing SPICE with proprietary modifications and enhancements. While SPICE 2G6 (the older and most trusted version) derivatives will produce character graphics on your printer, which may be OK for some uses, their main strength is in the rich variety and quality of analyses performed.

Most SPICE vendors offer an enhanced graphing companion program for their product. The better ones will do some serious data manipulation tricks, such as multiplying a voltage trace by a current trace to obtain a power trace, or a Fourier Transform to display the frequency spectrum of a waveform. This allows the SPICE guru to get more insight from each simulation run. Berkely SPICE 3B1 (the late version) includes NUTMEG, a graphic display postprocessor. SPICE 3B1 enjoyed a dubious reception from the engineering community, but it's the only version available directly from UC Berkely on AT compatible diskettes. Remember, it's C source code only!

Summing Up

Digital circuit simulators are also available at very reasonable prices lately (that's another story), but many circuits are made up of analog AND digital components. At this time, there are no low-cost programs to simulate mixed technology circuits at a detailed level, but reasonably-priced products have appeared which are meant to simulate whole systems (ref's 8 & 10). This is done by modeling major blocks, such as a VCO (Voltage Controlled Oscillator) or phase detector, with single mathematical expressions. Discrete circuit components are not modeled in this type of simulation.

Eventually software will be marketed that simulates electronic circuitry on all levels, and at the right price. This event will likely have to wait for a new and more cost-effective generation of personal computers. Mixed analog and digital simulation software is appearing now, from at least two sources. MicroSim offers an add-on package for PSpice, and Spectrum Software claims mixed capability through "integrated switch models and macros" in Micro-CAP III. These are pricey products, but they portend much for the future. In the meantime, low-cost analog simulation tools offer strong performance and diminishing user hostility to those adventuresome enough to indulge.

References:

1 *SPICE: A Computer Program to Simulate Semiconductor Circuits*, L. Nagel, UC Berkely M520, May 1975

2 *SPICE: A Guide to Circuit Simulation and Analysis Using PSpice*, Prentice-Hall, 1988, 200 pages

3 "Simplify Circuit Analysis With Easy Op-Amp Models", G. Cotreau & B. Matthews, *Electronic Design*, Sept. 22, 1988, pp 91-96

4 "Low Power Quad 741 Op-Amps", *Raytheon Linear Data Book*, 1978, pp 1-23

5 ICL7675/ICL7676, *Intersil Component Data Catalog*, 1987, pp 5-87

6 "Bode Stability Analysis", *Hardware/Software Catalog #6*, pp 42, BV Engineering, 2023 Chicago Ave. Suite B13, Riverside, CA 92507, (714) 781-0252

7 Atrium Electronics, 2302 5th St. NE, Salem, OR 97303, (503)363-5143

8 DesignScope (for Macintosh), Brainpower, 24009 Ventura Blvd., Calabasas, CA 91302

9 *Intusoft Newsletter* July, 1989, Intusoft, PO Box 6607, San Pedro, CA 90734-6607

10 TUTSIM Products (formerly Applied i), 200 California Ave. #212, Palo Alto, CA 94306

11 "SPICE Modeling: 'Build' a Circuit In No Time Flat", Oliver J. Smith & Gene Cavanaugh, *Electronic Design*, April 13, 1989, pp 109-118

12 "Analog Simulation", Doug Conner *EDN*, November 24, 1988

13 OrSPICE from NW Silicon Specialists, Inc., 2700 NW 185th Ave., Suite 2000, Portland, OR 97229 ✱

Note: There is a student version of PSPICE that is available for download from several bulletin board systems (usually college BBS's). It has the features of the full version, but the circuit size is restricted to about 10 transistors. If enough of you show interrest in it, I will see about putting it on our COM1 (HUG) BBS.

— P. Swayne

Reader Service #149

Why stick with second-rate communications software when upgrading is this easy?

Did you know that the best communications software is not Crosstalk, Procomm or Smartcom? **It's HyperACCESS/5 from Hilgraeve.**

Seeing is believing. So we're making a very special offer to owners of Crosstalk™, Relay™, Smartcom™, Mirror™, Procomm, Telix and Qmodem. **For a limited time, you can step up to HyperACCESS/5, normally $199, for only**

$49.95

There's NO RISK. If within 60 days you're not completely satisfied, return HyperACCESS/5 for a full refund.

> **"A must-have for those interested in telecommunications..."**
> John C. Dvorak - PC Magazine, 12/12/89
>
> **"A new standard in performance of communications software has arrived, and its name is HyperACCESS/5."**
> PCResource, 8/90
>
> **"The graphics and user interface are top notch and visually impressive."**
> InfoWorld, 6/11/90
>
> **"...makes all other communications software positively obsolete...enough goodies to keep a super-power-user interested and busy for a long time."**
> REMark, 6/90

Some programs lack important features like: Zmodem, PC-to-PC power, strong script language, or terminal emulators you need. Others may have what you need, but are slow, awkward or unreliable. HyperACCESS/5 gives you everything you need, with the speed, agility, and reliability you deserve!

Feature Comparison Table		HyperACCESS/5	Crosstalk XVI	Crosstalk Mk4	Procomm Plus	Smartcom III
Compatibility	OS/2 version also available	■	□	□	□	□
	Built-in support for 70+ brands of modems	■	□	□	□	□
	Built-in support for 9600+ bps modems & ISDN	■	□	□	□	□
Ease of Use	Both menu-driven and command-driven	■	□	■	□	□
	Menus totally self-explanatory	■	□	□	□	□
	Context-Sensitive on-screen Help	■	□	□	■	■
Performance	Seconds to transfer this table, 9600 bps modems	5.1	47.3	11.7	11.6	11.5
	Fastest Xmodem, Kermit, Ymodem, Zmodem*	■	□	□	□	□
	Most reliable file transfer protocols*	■	□	□	□	□
	Transfers files up to 5 time modem's speed	■	□	□	□	□
	Zmodem protocol built right in	■	□	□	□	□
Virus Protection	Guards against downloading computer viruses	■	□	□	□	□
	Guards against copying viruses from floppies	■	□	□	□	□
	Scans your system for known viruses	■	□	□	□	□
Automation	Learns both online and on-menu actions	■	□	□	□	□
	Compiles scripts for faster execution	■	□	□	□	■
	More than 150 script commands	■	□	■	□	■
	Scripts can be assigned to any key	■	□	□	□	■
Dialing Capabilities	Dialing directory can be sorted many ways	■	□	□	□	□
	Time-delayed dialing	■	□	□	□	□
	Built-in support for voice calls	■	□	□	□	■
	Cyclical and queued dialing	■	□	□	■	□
Host Mode	Caller-to-caller and caller-to-host E-Mail	■	□	□	□	□
	Password list encrypted and lockable	■	□	■	□	□
	Separate access levels for each caller	■	□	□	□	□
	Built-in high-security dial-back access	■	□	□	□	□
	Built-in support for user-defined host services	■	□	□	□	□
Emulators	Emulates more than 10 terminals	■	□	□	■	□
	VT100 supports 132-column display	■	□	■	□	□
	IBM3101 supports block mode	■	□	■	□	□
	TV950 supports block & protect modes	■	□	■	□	□
	Virtually all key outputs definable	■	□	□	□	□
	Multiple keyboard overlays	■	□	□	□	■
Editor & Backscroll Buffer	Editor built-in, not a slow external program	■	□	■	□	■
	Supports split-screen and multi-file editing	■	□	□	□	□
	Editor handles files of unlimited size	■	□	■	□	□
	Can be configured to mimic other popular editors	■	□	□	□	□
	Backscroll buffer holds up to 250K of text	■	□	□	□	□
	Print or capture selected backscroll text	■	□	□	□	■
	Edit and retransmit selected backscroll text	■	□	□	□	□
File Management	File management available on every screen & menu	■	■	□	□	□
	Point and shoot file selection	■	□	□	□	■
	File selection across multiple drives/directories	■	□	□	□	□
	Sorts files by name, extension, date, or size	■	□	■	□	□

*Independent tests by PC Magazine and Software Digest have shown Hilgraeve's protocols to be faster and more reliable than other manufacturers', and have proven HyperProtocol, with its on-the-fly compression, to be fastest of all protocols.

Guaranteed results in 10 minutes or less

We guarantee you can install and make your first call in 10 minutes or less. HyperACCESS/5 is Hardware Aware™—it adapts itself to your PC and 70 specific modems, plus generic types—so you can place and answer calls *immediately*. And its slick Sliding Windows™ interface is more than intuitive, it's obvious!

HyperProtocol sends files faster

HyperACCESS/5's HyperProtocol has on-the-fly compression and lets you transfer files through your modem at up to 5 times the modem's speed. And now that we've put HyperProtocol in the public domain, you can get the same fast transfers when you call BBSs or friends with other comm software.

Automating communications is a breeze

HyperACCESS/5's Discerning Learning™ watches you, learning not just your keystrokes, but your intentions. Quickly, easily, and without writing scripts, you can automate every facet of your communications, even entire calls!

Upgrade to the experts' choice now!

The computer industry's top software evaluators have unanimously chosen HyperACCESS/5. Isn't it worth $49.95 to find out why?

PC MAGAZINE	TOP RATED	Software Digest RATINGS REPORT
Editor's Choice	**Best Overall**	**Best Overall**
July 1988 and May 1987	April 1990	September 1986

Attention Procomm™, Telix™ and Qmodem™ owners!

Step up to HyperACCESS/5 for DOS for only $49.95 risk-free. Proof of ownership is required.

TO ORDER:

CALL TOLL-FREE: 800-826-2760, 8am-6pm EST
OR BY FAX: 313-243-0645. See info below.
OR BY MAIL: See info below.

Hilgraeve Inc.

To order by mail or fax, please send your name, company, address, phone number, credit card number, expiration date, and signature (or send certified check or money order) to Hilgraeve Inc., HyperACCESS/5 Upgrade, 111 Conant Ave., Suite A, Monroe, MI 48161. Price is $49.95 plus $6.00 shipping. Offer good in US and Canada only. Limited time offer. Limit of one per customer.

Reader Service #203

What are the Options?

Part 1

David W. Lind
Rural Route 1 Box 3114
Bar Harbor, Maine 04609

Have you ever tried to run a program on a computer with which you weren't familiar and failed? Ever worked with a computer which was advertised as "IBM Compatible" and found that it wasn't? Unfortunately, with the proliferation of IBM clones and IBM computers with hidden optional equipment or design peculiarities, these problems are all too common. Many software designers are incorporating code into their programs which checks the hardware and operating system software to ensure the program will run properly. But, most of the software on the market does not perform these checks. At best, the small print on the package may give some indication as to what equipment is needed to run the latter software. If the intent is to use the software on unfamiliar equipment or if the capabilities of the equipment are not obvious, even this information is of little use.

I've often had the opportunity to transfer programs to computers of unknown pedigree. Therefore, I decided to develop a program which determines the capabilities of IBM PC type equipment. In the course of this work, I learned a great deal about the differences between computers and the esoteric software required to determine these differences. I hope I can save others a great deal of time and money by sharing this experience.

This article is the first of three which discusses the major features of IBM compatible computers and develops an assembly language program which determines these features. Parts of this code can be incorporated into other programs to make them more portable from one machine to another. The code can also be used to check the operation of computers with known capabilities.

Before beginning this journey into chaos, I have a few words of caution. The use of assembly language for this task is necessary because other programming languages generally are not capable of all the functions one needs. Be careful transcribing the program. Many errors will not be detected by the assembler and some errors can destroy data. Each line of code has a comment following a semicolon. These comments identify the function of the code. In addition there are blocks of comments which describe the purpose of blocks of code. These block comments begin with "COMMENT *" and end with "*". None of these comments need to be transcribed for the code to work. Debugging the program could be easier with them, but transcription is easier without them.

Some of the operating system subroutine calls, commonly known as "interrupts," are not intended for use by applications programmers. If one errs with some of these systems programming interrupts, disks can be damaged. Double check the input data for the interrupts, particularly when they deal with disks. Unfortunately, there is incomplete and contradictory data on many interrupts in many references. The code was executed on several computers, all Zenith models. Theoretically, the code will work on all IBM compatible computers and even some which are not. In the latter case, "WILD INTERRUPT" messages and incorrect information may result. So, if the program indicates that the computer is "probably not IBM compatible," use the subsequent data with caution. But, there will always be the maverick machine, advertised as IBM compatible, which will not accept the interrupts as coded. In that case, one should contact the manufacturer of the machine to determine the proper inputs or interrupts.

A good place to begin the discussion of differences between IBM compatible computers is with the microprocessor because one would expect that microprocessor design would heavily influence the way the computer operates and also determine its capabilities. IBM compatible personal computers are based upon the Intel Corporation's 80XXX series of microprocessors. The original IBM PC computer was based upon the Intel 8088 microprocessor which internally processed data in 16 binary digit (bit) words. But, the 8088 processor interfaced with 8-bit memory chips. This configuration was a result of Intel Corporation's decision to permit the use of existing 8-bit devices with their new chip and to take advantage of the wide availability of inexpensive 8-bit memory chips. Intel intended the 16-bit microprocessor to interface with 16-bit memory chips which occurred when such memory chips became commercially available. The Intel 8086 microprocessor is faster than the 8088 and interfaces with 16-bit memory busses, but is otherwise the same as the 8088. Generally, the IBM PC and IBM XT computers use the 8088 or equivalent microprocessors.

Intel subsequently developed a series of faster and more sophisticated microprocessors, the 80186 and 80286. The 80286 became the basis for the IBM AT and the basic personal system computers (IBM PS/2 models). When Intel developed a 32-bit word microprocessor, the 80386, it became the basis for the IBM PS/2 advanced personal computers. Intel deliberately designed their new microprocessors to run code written for their earlier microprocessors. Thus, a program written for the 8088 microprocessor should run on an 80386 based machine. However, the reverse is not necessarily true. Recently, the Intel Corporation marketed a more sophisticated chip designated the 80486 and is about to market even more advanced microprocessors. This series of articles does not discuss these newer microprocessors.

This wide range of related microprocessors does create problems for programmers. For example, some instructions in the earlier microprocessors must be preceded by a "wait" instruction to allow the circuitry to complete tasks before initiating new instructions. This feature is used with coprocessors, devices which perform mathematical functions more quickly and efficiently. But, the newer microprocessors and circuits are fast enough to eliminate the need for these delays. Thus, programs generated on machines based on the 80386 microprocessor using a coprocessor cannot be transferred to an 8086 based machine containing a coprocessor unless wait instructions are explicitly included. On the other hand, programs run faster on the 80286 and 80386 machines if the wait instructions are omitted. In addition, there are various hardware configurations which can cause programming problems, particularly for computers which are not strictly IBM compatible.

To help overcome some of these problems, most programmers write programs for a particular IBM model and a particular version of the operating system. Surprisingly, the ability of a microprocessor to execute applications software is more dependent upon the systems software which directs the microprocessor, than on the physical operation of the microprocessor. The IBM PC, PCjr, and XT all use the 8088 microprocessor, but their capabilities differ. There is an IBM XT-286 which uses an 80286 microprocessor like the IBM AT. It is faster than an ordinary XT, but does not have all the features of an AT. There are four basic IBM personal computer models running the Personal Computer Disk Operating System (PC-DOS) or the virtually equivalent Microsoft Disk Operating System (MS-DOS): the PC, PCjr, XT, and AT. What about the 80386 running MS-DOS? It is usually considered an AT model. When Operating System 2 (OS-2) or a similar operating system is used with an 80286 or 80386 microprocessor, the system is no longer a "PC." This article covers topics pertaining to the PC-DOS or MS-DOS based systems only and assumes some familiarity with the hexadecimal number system and assembly language programming. Of course, the program can be transcribed, assembled, and used by someone ignorant of the details. Now on to the details of identifying computer characteristics.

The first task is to determine the model of the computer. All IBM compatible computers should have the "System Model ID" stored permanently in the read only memory (ROM) of the basic input/output system (BIOS). This ID is stored at memory location FFFFEh ("15","15","15","15","14" hexadecimal) which is at the end of memory locations in ROM BIOS. Table 1 shows the codes for the current microprocessors as well as the basic IBM model or equivalent IBM compatible model. Some IBM compatible computers may use different microprocessors equivalent to or more capable than the microprocessors which IBM uses. For example, the Heath H-386 computer system uses an 80386 microprocessor. But, the BIOS identifies this machine as equivalent to an IBM AT computer when MS-DOS is used. The 80C88 microprocessor is a low-powered version of the 8088 microprocessor.

The assembly language program determines the System Model ID by initializing the data segment to the BIOS memory, transferring the System ID information to a register, reinitializing the data segment to the program area data segment, and transferring the System ID to the program data segment. The basic model information is then displayed.

Determining the System Model ID indicates the type of microprocessor present and how it is configured to interact with the hardware. Thus, if one has an "XT" model, software or hardware designed exclusively for an "AT" model will probably not work. It is equally important to determine the version of the operating system a model uses. Each new major version of the MS-DOS operating system has new interrupts and features in addition to the same interrupts and features of earlier versions. Programs which use the new interrupts and features will not run on older versions. The interrupts used in the assembler program developed in this series of articles, should work for most versions of the MS-DOS or PC-DOS.

A few more words about interrupts may be useful at this point. Interrupts may be classified as hardware, processor, user or software generated. In any of these cases, the operation being performed by the microprocessor is suspended when an interrupt occurs. Processor interrupts occur when an unusual operation is necessary, like division by zero. Hardware interrupts are generated by peripheral devices. For example, pressing a key on a computer key board generates a hardware interrupt. The interrupts which affect the compatibility of computers most are software interrupts. These interrupts are requests by programs to use a DOS or BIOS subroutine to accomplish a task which would otherwise be complex. For example, one could write a complex subroutine to access the printer port and print a character or one could use the same subroutine the DOS uses to print a character. Generally, it is best to use the DOS subroutine.

One might ask why these DOS or BIOS subroutines are called interrupts rather than simply subroutines or functions? Besides the usual propensity for technical people to be esoteric, there is a good technical reason. In the BIOS and DOS, subroutines or functions are entered by transferring control to the portion of memory containing the executable code for the requested function. This code is located outside the allocated program memory. The physical location of this code may vary from one version of DOS to the next. If a program used the actual address of the DOS or BIOS subroutine, it would probably work with only one version of DOS and BIOS. By the way, one can call DOS or BIOS subroutines in this way. An interrupt call is independent of the physical location of the subroutine.

Memory Contents	Micro Processor	Model
FFh	8088	PC
FDh	8088	PCjr
FEh	8088	XT
FCh	80286	AT, XT/286, PS/2 Models 50 and 60
FBh	8088	PC/2
F9h	80C88	Convertible
FAh	8086	PS/2 Model 30
F8h	80386	PS/2 Model 80

Table 1
System Model ID Codes

When the computer is powered, a table, called the "Interrupt Vector Table," associating numbers from 00h to FFh with memory locations is generated. The numbers are assigned to specific functions and do not change from one DOS version to the next. For example, interrupt 12h (18 in decimal) is the "Get Memory Size" function regardless of the DOS or BIOS version. However, the memory location associated with the interrupt number may change considerably. When an interrupt is called, the computer finds the address of the requested subroutine using the table. Then control is transferred to that address. So, if one calls interrupt 12h, the computer goes to the Interrupt Vector Table and finds interrupt 12h. The associated physical address might be (and is on many systems) FF841h. The computer then transfers control to memory location FF841h. Another bit of esoterica, the memory location of an interrupt subroutine entry point is called the "interrupt vector" from which we get the name of the table. Normally, this interrupt vector is expressed in the form "segment:offset." The above interrupt 12h vector would be written as F000:F841h. Another convention worth mentioning is that interrupt numbers are normally written in hexadecimal format, e.g. 12h. Be sure to include the "h" in assembly programs. If the "h" is omitted, the assembler assumes the number is in decimal format and may produce the wrong interrupt. Interrupts may have numerous sub-functions which are determined by numbers in the microprocessor registers at the time the interrupt is called. For example, one of the most common software interrupts is interrupt 21h, called "Function Request." This title is very general for good reason. INT 21h, as it is usually written, has over 100 separate functions, some of which have numerous sub-functions. Calling int 21h with 02h in the ah register will cause the ASCII (American Standard Code for Information Interchange) equivalent of the number in the dl register to be sent to the video display. If 05h were in the ah register, the output would be directed to the printer instead. Both these functions are used in the following Assembly Language program.

If one has access to a debugger like Microsoft's Code View, it is possible to trace the execution through each step of the interrupt subroutine. Very often, a single interrupt calls a rather large block of code in DOS or BIOS. That can be a convincing demonstration of the power of interrupts and a great deterrent to writing substitute code.

Which brings us to the subject of user interrupts. The systems software engineers reserved some interrupts, F1h to FFh, for users to write their own interrupts. Thus, one can insert addresses (interrupt vectors) of code at these special locations in the Interrupt Vector Table. Actually, one can change any interrupt vector, but to do so can be foolish and would certainly create compatibility problems. Any further discussion of user interrupts is beyond the scope of this article.

Every application program uses interrupts. For a program to work on a given computer, the requested interrupt must be available. The operating system version is the primary means of determining if a given interrupt function is available. In rare cases, the BIOS version is a determining factor. Although the BIOS is usually perceived as resident in ROM, some of it is in Random Access Memory (RAM). This RAM contains the Interrupt Vector Table and changes to the BIOS which override or augment the ROM BIOS. Since the BIOS changes are issued with MS-DOS/PC-DOS changes, BIOS version numbers are seldom specified to ensure application software compatibility. A knowledgeable reader may question why int 15h, function C0, "Read System-Configuration Parameters" is not used by the program below to obtain the System Model ID. The reason is that what is returned by this interrupt depends not only on the model, but also on the BIOS version. In fact, some XT and AT models will not return the parameters for certain early BIOS versions. Thus, this interrupt is best avoided for a general use program.

Frequently, programmers will write programs using a specific DOS version and then indicate that the program will run on that version of DOS or greater. In fact, the program may run on an earlier version depending upon the interrupts used. For example, INT 21h, function 02h -"Display Output" works on virtually all versions of DOS. Interrupt 21h, function 35h - "Get Interrupt Vector" requires DOS version 2 or greater. To make matters worse, some interrupts are model specific. For example, INT 13h, function 08h - "Return Disk Drive Parameters" when used for floppy disks only works on an AT or PS/2 model, but will work on virtually any computer when used for fixed disks. Thus, one needs to know the computer model, DOS version, and sometimes the BIOS version information to determine if a program will run on a computer. This information must be compared with the information provided with an application program. If program information is lacking or does not correlate with the computer information, the program may run properly. But, don't be surprised if the program fails to run one day when it issues that rare request for an interrupt that does not exist in the computer's DOS. The computer model hardware characteristics and the availability of correctly vectored interrupts are generally what determine if a computer is truly IBM compatible.

A description of which interrupts are in different versions of MS-DOS/PC-DOS is beyond the scope of these articles. The reader can find more information in three of the references listed at the end of this article: "Microsoft MS-DOS Operating System Programmer's Reference Manual", "DOS and BIOS Functions Quick Reference" or "System BIOS for IBM PC/XT/AT Computers and Compatibles." If one wants to examine the Interrupt Vector Table in an IBM compatible computer, it is normally located at address 0000:0000. Table 2 is a listing of the interrupts used in the program developed in this series of articles. The narrative will explain limitations of these interrupts.

Now that the connection between interrupts and the DOS version number is established, the means of obtaining the version number can be described. The DOS version number is obtained through int 21h, function 30h, "Get DOS Version Number." There is a problem with the use of this interrupt. For MS-DOS versions earlier than 1.28, the number 0 will be returned. Otherwise, register al will contain the major version number and register ah will contain the minor version number after the interrupt is called. Both numbers will be binary integers. For those readers not familiar with the DOS version number format, the major version number is the number before the decimal point. The minor version number follows the decimal point. Keep in mind that if the program indicates that the DOS version number is 0.0, there may not be a programming error - the DOS version may be earlier than 1.28.

No attempt is made in the following program to determine the BIOS version because the BIOS is usually updated with new DOS releases.

The EQUIP.EXE program listed determines only the model type and DOS version number. In subsequent articles, many more features will be added to include information about the computer's display, memory, disk devices, and a number of other components. The program will be expanded to about three times the size of the program by adding code to the data segment and code segment. Many features can be deleted by the reader who is familiar with Assembly Language programming.

Although the comments in the program explain it in some detail, an explanation of the design and an overview may be useful. As usual, the program is divided into three segments: the data segment, code segment, and stack segment. An extra segment is unnecessary. All the data definitions are made in the data segment to avoid confusion when code is added. The program uses five "procedures" ("PROCs") which are subroutines called by the main program. These procedures are located at the end of the code segment. Only two of these procedures are found in the program listed. The others will be added later. Except for loading the System Model ID, all calls to the DOS/BIOS subroutines are through common interrupts.

The reason for obtaining the System Model ID directly from memory was briefly explained earlier - the available interrupt does not work on all models. The program forces the assembler to make the BIOS memory the data segment. After the System Model ID is safely stored in register dl, the data segment is redefined to the program space and the System Model ID is moved to the program data segment. This process is not a recommended way to start a program and may be confusing to a novice programmer. Don't change the

Interrupt	AH Register Data	Function
11h	none	Get Equipment Status
12h	none	Get Memory Size
13h	08h	Read Disk Drive Parameters
13h	10h	Test for Drive Ready
21h	01h	Key Input with Echo
21h	02h	Display Byte
21h	05h	Print Byte
21h	09h	Display String
21h	1Ch	Get Allocation Table Information
21h	30h	Get DOS Version Number
21h	36h	Get Free Disk Space
21h	4Ch	Terminate Program

Table 2
Interrupts Used in EQUIP.EXE

order of this portion of the code.

Before any other data is processed, the Program Segment Prefix (PSP) is stored in the data segment at label "psp." The PSP contains the code and information necessary to transfer control from the program to the operating system when the program terminates. Normally, program termination is accomplished by interrupt 21h, function 4Ch. This interrupt closes files and transfers control to the PSP which terminates the program. Unfortunately, DOS version 1 does not have this interrupt. Alternative interrupts, interrupt 20h and interrupt 21h, function 00h, are available in DOS version 1. Both these interrupts require the cs register to point to the beginning of the PSP with an offset of zero (cs:00). This condition is true in .COM programs. The PSP is contained in the space defined by the "org 100h" declaration in a .COM program and the cs register contains the segment address of this code. Unfortunately, the cs register in a .EXE program points to the beginning of the code segment which does not contain the PSP.

Thus, if the program below is to be compatible with version one of DOS, it must either be converted to a .COM program or a way of terminating a .EXE program running under DOS version one must be found. The latter option was selected to allow for future expansion of the program. There are two ways to terminate a .EXE without using interrupts directly. One can make a long (far) jump to the PSP or one can call the PSP. The latter method was selected because it is less affected by the peculiarities of the DOS. The details are discussed later. For now, it is enough to know that one must save the PSP segment address.

When the program is entered, all versions of DOS should place the segment address of the PSP in registers ds and es. Since the ds register must change to establish the data segment where the PSP address is to be stored, the segment PSP address must be obtained from the es register. Note that data cannot be transferred to and from memory and segment registers directly.

The next portion of code displays a copyright notice. Throughout the program, video screen displays are made through interrupt 21h, function 09h. This subroutine requires that the string of characters be terminated with the ASCII code for "$". All versions of DOS/BIOS should have this subroutine. Terminating a string with "$" is an old convention which is awkward if "$" is a character one wishes to display. But, the interrupt does preclude the necessity of writing a PROC to display a string using another terminator.

The program displays a message asking if a printed copy of the output is required. The answer is input using interrupt 21h, function 01h, Key Input with Echo. The input is in rgister al. The program sets "prflag" to 0 if either a "Y" or "y" is input. If a printed copy is requested, a copyright notice is printed on the standard printer. The PROC "prnstr" is used for this purpose because there is no interrupt which prints a string. The prnstr PROC uses interrupt 21h, function 05h, Print Character. This interrupt should be in all versions of DOS/BIOS. The PROC prnstr is used throughout the program.

After the header is displayed and printed, the System Model ID is compared to the codes listed in Table 1. If a match is found, the appropriate message is displayed and printed. Note that in the code for some of the models, a flag called "modflag" is set to 1 or 2. This flag will be used later to ensure interrupts are compatible with the model in use.

The version number is obtained via interrupt 21h, function 30h, Get DOS Version Number. The major version number is stored in "version" and will be used later to ensure interrupts in this program are compatible with the installed version of DOS. The conversion of the version numbers from binary to binary coded decimal is accomplished with the ASCII Adjust After Multiply (aam) command which saves some code and time. The aam command converts a number less than 100 in register al to binary coded decimal numbers in registers ah and al. The name of this command is somewhat misleading. Note that if the most significant digit (in register ah) of the major version number is zero, a -16 is moved to register ah. This has the effect of blanking a leading zero.

After the various messages associated with the version number are output, the exit process is initiated. Future code will be added just before the "xit" label. The exit process includes ejecting paper (if the printer is used), scrolling the display, and terminating the program. Note that one of two methods are used to terminate the program depending upon the DOS version. Technically, the PROC "proend" should work with all DOS versions. But, it does not properly close files, which is not important for this program. The 4Ch function is the recommended way of terminating a program and should work with all recent DOS versions. It is activated for DOS versions two or above in lieu of the PROC for pedagogical reasons.

The proend PROC is defined as a FAR PROC to force the computer to store both the offset and segment addresses of the code segment on the stack when the PROC is called. NEAR or default PROCs store only the offset address. The PROC pushes the segment address of the PSP onto the stack and then pushes the number zero onto the stack. Normally, one would pop the stack before returning so the computer can pop the code segment addresses upon return. When the computer returns from this PROC, the segment address of the PSP is placed in register cs and the instruction pointer, the offset address for register cs, is set to zero. This process in effect "calls" the PSP and the program terminates.

The prnstr PROC uses the "lodsb" command. The reader should refer to the Microsoft Macro Assembler Programmer's Guide for detailed information on the use of this somewhat esoteric command.

In Part 2 of this series of articles, code will be added to determine the coprocessor installed, if any. Also memory size, printer, serial port, and internal card information will be generated. The final part will provide code to analyze disk drive type and disk capacity.

EQUIP ASSEMBLER PROGRAM LISTING

```
COMMENT *PROGRAM EQUIP.EXE
   This program identifies common equipment configurations for the Intel 8086 to
80386 microprocessors. Note that the term "installed" means that the necessary
hardware internal to the computer and input/output software is present, but not
necessarily functional. For example, an external drive may be configured, but not
physically connected at the time the program is run. The user must verify that
peripheral  equipment is connected and activated before attempting to use the
equipment identified by this program. If a printed copy of the output is desired, a
printer must be connected to the standard printer port (PRN or LPT1:) and activated
before this program is executed.
   Copyrighted 1990 by David W. Lind. This program may be copied and used by
individuals without compensation to the copyright owner provided the copyright
notices are included. Any sale or further distribution of this program requires the
written permission of the copyright owner. PC, PC/XT, PC/AT, Personal Systems 2,
PS/2, IBM, PC-DOS, VGA, CGA, EGA, and OS/2 are trademarks of the International
Business Machines Corporation. MS-DOS is a trademark of the Microsoft Corpora-
tion. Intel is a trademark of the Intel Corporation.
   No expressed or implied warranties are made for this program with respect to
merchantability or fitness for a particular purpose. The user assumes all responsibil-
ity for any damages resulting from the use of this program. *
```

Listing 1

```
data     SEGMENT                          ;Program Data Segment
outstr   DB      10 DUP (1)               ;Output string storage
discnt   DB      0                        ;Display lines counter
sysby    DB      ?                        ;System byte storage
prflag   DB      0                        ;Print flag
modflag DB       0                        ;Model flag
version DB       ?                        ;Major version number
lfcr     DB      13,10,'$'                ;Line feed/carriage return code
copyw    DB      '      EQUIPMENT LIST, Copyright 1990 by David W. Lind',13,10,'$'
quest    DB      '      Do you wish a printed copy (Y or N, default is N)? ','$'
header   DB      '      COMPUTER SYSTEM EQUIPMENT LIST:',13,10,'$'
atorps   DB      '      IBM AT (Intel 80286/80386) or compatible ',13,10,'$'
pcjr     DB      '      IBM PCjr (Intel 8088) or compatible ',13,10,'$'
xt       DB      '      IBM XT (Intel 8088/80C88) or compatible ',13,10,'$'
pc       DB      '      IBM PC (Intel 8088) or compatible ',13,10,'$'
pc2      DB      '      IBM PC/2 (Intel 8088) or compatible ',13,10,'$'
conv     DB      '      IBM Convertible (Intel 80C88) or compatible ',13,10,'$'
ps230    DB      '      IBM PS/2 Model 30 (Intel 8086) or compatible ',13,10,'$'
ps280    DB      '      IBM PS/2 Model 80 (Intel 80386) or compatible ',13,10,'$'
noibm    DB      '      This computer is probably not IBM compatible.',13,10,'$'
vermsg   DB      '      Disk Operating System version is ','$'
psp      DW      ?                        ;Segment address of PSP
data     ENDS

code     SEGMENT                          ;Program Code Segment
         ASSUME  cs:code,ds:data
start:                                    ;Program Entry Point
         COMMENT * Obtain and store System Model ID *
         mov     ax,0F000h                ;Put BIOS address in ax
         mov     ds,ax                    ;Move BIOS address to ds
         mov     bx,0FFFEh                ;Put OFFSET for System ID in bx
         mov     dl,[bx]                  ;Place System ID in dl
         mov     ax,data                  ;Put data segment address in ax
         mov     ds,ax                    ;Move address to ds register
         mov     sysby,dl         ;Put System ID in data segment

         COMMENT * Save Program,Segment Prefix (PSP) address in "psp."
                   The PSP segment address is in register es when
                  the program begins.  The PSP contains code
                   which returns control to DOS when the program
                   terminates.  For DOS versions two and above,
                   program termination is accomplished through
                   interrupt 21h, function 4Ch without the need
                   for PSP information.  For DOS version one, the
                   address of the PSP is necessary to terminate
                   the program. *
         mov     ax,es                    ;Put PSP segment address in ax
         mov     psp,ax                   ;Store PSP segment address

         COMMENT * Display copyright notice *
         mov     dx,OFFSET copyw          ;Put address offset into dx
         mov     ah,09h                   ;Put function number in ah
         int     21h                      ;Call the display subroutine

         COMMENT * Determine if a hard copy is required *
         mov     dx,OFFSET quest          ;Put address of quest in dx
```

```
         mov     ah,09h                   ;Put function number in ah
         int     21h                      ;Call string display subroutine
         mov     ah,01h                   ;Put function number in ah
         int     21h                      ;Call key input with echo

         COMMENT * Set print flag *
         mov     prflag,al                ;Put input into prflag
         sub     al,89                    ;Make al 0 if prflag is Y
         je      lblf                     ;If al is 0, go to lblf
         mov     al,prflag                ;Put prflag in al
         sub     al,121                   ;Make al 0 if prflag is y
         jnz     lblf1                    ;If not 0, go to lblf1
lblf:    nop                              ;Print copy is desired

         COMMENT * Print copyright notice *
         mov     prflag,0         ;Zero prflag
         mov     si,OFFSET lfcr           ;Put address OFFSET in si
         call    prnstr                   ;Call print string PROCedure
         mov     si,OFFSET copyw  ;Put address OFFSET in si
         call    prnstr                   ;Call print string PROCedure
         mov     si,OFFSET lfcr           ;Put address OFFSET in si
         call    prnstr                   ;Call print string PROCedure
         jmp     lblf2                    ;Go to next function
lblf1:   nop                              ;No print copy is desired
         mov     prflag,1         ;No print, set prflag to 1
lblf2:   nop                              ;End of prflag set function

         COMMENT * Display and print header *
         mov     dx,OFFSET lfcr           ;Pick up LF and CR
         mov     ah,09h                   ;Put function number in ah
         int     21h                      ;Call display subroutine
         mov     dx,OFFSET header ;Put address offset into dx
        mov      ah,09h                   ;Put function number in ah
         int     21h                      ;Call the display subroutine
         inc     discnt                   ;Increment the display line counter
         mov     si,OFFSET header ;String at location "header"
         call    prnstr                   ;Push addresses and go to PROC

         COMMENT * Examination and display and print of System Model ID *
         mov     al,sysby         ;Put System ID into al
         xor     al,0FCh          ;Compare to AT ID
         jnz     ibmjr                    ;If not AT go to ibmjr
         mov     dx,OFFSET atorps ;Put message offset in dx
         mov     ah,09h                   ;Put function number in ah
         int     21h                      ;Call display subroutine
         inc     discnt                   ;Increment the display line counter
         mov     si,OFFSET atorps ;String at location "atorps"
         call    prnstr                   ;Push addresses and go to PROCedure
         mov     modflag,2                ;Set modflag for AT
         jmp     ver0                     ;Got to next function
ibmjr:   nop                              ;Start check for PCjr
         mov     al,sysby         ;Put System ID into al
         xor     al,0FDh          ;Compare to PCjr ID
         jnz     ibmxt                    ;If not IBM PCjr go to ibmxt
         mov     dx,OFFSET pcjr           ;Put message offset in dx
         mov     ah,09h                   ;Put function number in ah
         int     21h                      ;Call display subroutine
         inc     discnt                ;Increment the display line counter
         mov     si,OFFSET pcjr        ;String at location "pcjr"
         call    prnstr                ;Push addresses and go to PROCedure
         jmp     ver0                  ;Go to next function
ibmxt:   nop                           ;Start check for IBM xt
         mov     al,sysby        ;Put System ID into al
         xor     al,0FEh         ;Compare to XT ID
         jnz     ibmpc                 ;If not IBM XT go to ibmpc
         mov     dx,OFFSET xt          ;Put message offset into dx
         mov     ah,09h                ;Put function number in ah
         int     21h                   ;Call display subroutine
         inc     discnt                ;Increment the display line counter
         mov     si,OFFSET xt          ;String at location "xt"
         call    prnstr                ;Push addresses and go to PROCedure
         mov     modflag,1             ;Set model flag for XT
         jmp     ver0                  ;Go to next function
ibmpc:   nop                           ;Start check for PC
         mov     al,sysby        ;Put System ID into al
         xor     al,0FFh         ;Compare to PC ID
         jnz     ibmpc2                ;If not PC, go to ibmpc2
         mov     dx,OFFSET pc          ;Put message offset into dx
         mov     ah,09h                ;Put function number in ah
         int     21h                   ;Call display subroutine
         inc     discnt                ;Increment the display line counter
         mov     si,OFFSET pc          ;String at location "pc"
         call    prnstr                ;Push addresses and go to PROCedure
         jmp     ver0                  ;Go to DOS analysis word routine
ibmpc2: nop                            ;Start check for IBM PC2
         mov     al,sysby        ;Put System ID into al
         xor     al,0FBh               ;Compare to PC2 ID
         jnz     ibmcon                ;If not PC2, go to ibmcon
         mov     dx,OFFSET pc2         ;Put message offset into dx
         mov     ah,09h                ;Put function number in ah
         int     21h                   ;Call display subroutine
         inc     discnt                ;Increment the display line counter
         mov     si,OFFSET pc2         ;String at location "pc2"
         call    prnstr                ;Push addresses and go to PROCedure
         jmp     ver0                  ;Go to DOS analysis routine
ibmcon: nop                            ;Start check for IBM Convertible
         mov     al,sysby        ;Put System ID into al
         xor     al,0F9h               ;Compare to Convertible ID
         jnz     ibm30                 ;If not Convertible, go to ibm30
         mov     dx,OFFSET conv        ;Put message offset into dx
         mov     ah,09h                ;Put function number in ah
         int     21h                   ;Call display subroutine
         inc     discnt                ;Increment the display line counter
         mov     si,OFFSET conv        ;String at location "conv"
         call    prnstr                ;Push addresses and go to PROCedure
         jmp     ver0                  ;Go to DOS analysis routine
ibm30:   nop                           ;Start check for PS/2 Model 30
         mov     al,sysby        ;Put System ID into al
         xor     al,0FAh               ;Compare to PS/2 Model 30 ID
         jnz     ibm80                 ;If not PS/2 Model 30, go to ibm80
         mov     dx,OFFSET ps230       ;Put message offset into dx
         mov     ah,09h                ;Put function number in ah
         int     21h                   ;Call display subroutine
         inc     discnt                ;Increment the display line counter
```

```
         mov     si,OFFSET ps230       ;String at location "ps230"
         call    prnstr                ;Push addresses and go to PROCedure
         mov     modflag,2             ;Set modflag for PS/2 Model 30
         jmp     ver0                  ;Go to DOS analysis routine
ibm80:   nop                           ;Start check for PS/2 Model 80
         mov     al,sysby        ;Put System ID into al
         xor     al,0F8h               ;Compare to PS/2 Model 80 ID
         jnz     notibm                ;If not PS/2 Model 80, go to notibm
         mov     dx,OFFSET ps280       ;Put message offset into dx
         mov     ah,09h                ;Put function number in ah
         int     21h                   ;Call display subroutine
         inc     discnt                ;Increment the display line counter
         mov     si,OFFSET ps280       ;String at location "ps280"
         call    prnstr                ;Push addresses and go to PROCedure
         mov     modflag,2             ;Set modflag for PS/2 Model 80
         jmp     ver0                  ;Go to DOS analysis routine
notibm:  nop                           ;Machine not IBM compatible
         mov     dx,OFFSET noibm       ;Put message offset into dx
         mov     ah,09h                ;Put function number in ah
         int     21h                   ;Call display subroutine
         inc     discnt                ;Increment the display line counter
         mov     si,OFFSET noibm ;String at location "noibm"
         call    prnstr                ;Push addresses and go to PROCedure

         COMMENT * Find Disk Operating System version number *
ver0:    nop                           ;DOS analysis
         mov     dx,OFFSET vermsg ;Put message OFFSET in dx
         mov     ah,09h                ;Put function number in ah
         int     21h                   ;Call display string subroutine
         mov     si,OFFSET vermsg ;Put message OFFSET in si
         call    prnstr                ;Call print string PROCedure
         mov     ah,30h                ;Put function number in ah
         int     21h                   ;Call Get DOS Version Number
         mov     outstr,ah             ;Temporarily store ah in "outstr"
         mov     version,al            ;Put major version number in memory
         aam                           ;Convert binary to BCD
         cmp     ah,0                  ;Compare ah to 0
         jnz     ver1                  ;If not zero, go to ver1
         mov     ah,-16                ;Else set ah to -16 to blank zeros
ver1:    nop                           ;Convert numbers to ASCII
         add     al,48                 ;Convert to ASCII
         add     ah,48                 ;Convert to ASCII
         mov     bl,al                 ;Store al in bl
         mov     dl,ah                 ;Put ah in dl for output
         mov     ah,02h                ;Put function number in ah
         int     21h                   ;Call display byte subroutine
         cmp     prflag,0        ;Check print flag
         jnz     ver2                  ;If not true, go to ver2
         mov     ah,05h                ;Put function number in ah
         int     21h                   ;Call print byte subroutine
ver2:    nop                           ;Process next digit
         mov     dl,bl                 ;Get next digit for output
         mov     ah,02h                ;Put function number in ah
         int     21h                   ;Call display byte subroutine
         cmp     prflag,0        ;Check print flag
         jnz     ver3                  ;If not true, go to ver3
         mov     ah,05h                ;Put function number in ah
```

```
        int     21h                     ;Call print byte subroutine
ver3:   nop                             ;Insert decimal point
        mov     dl,46                   ;Put ASCII "." in dl
        mov     ah,02h                  ;Put function number in ah
        int     21h                     ;Call display byte subroutine
        cmp     prflag,0        ;Check print flag
        jnz     ver4                    ;If not true, go to ver4
        mov     ah,05h                  ;Put function number in ah
        int     21h                     ;Call print byte subroutine
ver4:   nop                             ;Process minor version number
        mov     al,outstr               ;Put minor version number in al
        aam                             ;Convert to BCD
        add     al,48                   ;Convert to ASCII
        add     ah,48                   ;Convert to ASCII
        mov     bl,al                   ;Store al in bl
        mov     dl,ah                   ;Put ah in dl for output
        mov     ah,02h                  ;Put function number in ah
        int     21h                     ;Call display byte subroutine
        cmp     prflag,0        ;Check print flag
        jnz     ver5                    ;If not true, go to ver5
        mov     ah,05h                  ;Put function number in ah
        int     21h                     ;Call print byte subroutine
ver5:   nop                             ;Process next digit
        mov     dl,bl                   ;Get next digit for output
        mov     ah,02h                  ;Put function number in ah
        int     21h                     ;Call display byte subroutine
        cmp     prflag,0        ;Check print flag
        jnz     ver6                    ;If not true, go to ver6
        mov     ah,05h                  ;Put function number in ah
        int     21h                     ;Call print byte subroutine
ver6:   nop                             ;LF/CR
        mov     dx,OFFSET lfcr          ;Put message OFFSET in dx
        mov     ah,09h                  ;Put function number in ah
        int     21h                     ;Call display string subroutine
        inc     discnt                  ;Increment display line counter
        mov     si,OFFSET lfcr          ;Put message OFFSET in si
        call    prnstr                  ;Call prnstr PROCedure

xit:    nop                             ;End of program
        COMMENT * Eject paper if printer was used *
        mov     al,prflag               ;Put print flag in al
        and     al,1                    ;Compare to 1
        jnz     lbljs                   ;If 1, do not print
        mov     dl,12                   ;Put ASCII FF in dl
        mov     ah,05h   ;Put function number in ah
        int     21h                     ;Call print byte subroutine

        COMMENT * Scroll display *
lbljs:  nop                             ;Scroll if necessary
        cmp     discnt,23               ;Compare displayed lines to 23
        jae     skipx                   ;If equal or above, do not scroll
        mov     cl,23                   ;Move number of lines on screen to cl
        sub     cl,discnt               ;Set number of lines to scroll
lblj:   mov     dx,OFFSET lfcr          ;Put offset of LF,CR in dx
        mov     ah,09h                  ;Put function number in ah
        int     21h                     ;Call display subroutine
        dec     cl                      ;Decrement scroll counter
```

```
        jg      lblj                    ;If cl not zero, scroll again

        COMMENT * Terminate program *
skipx:  nop                             ;Start next function
        cmp     version,2               ;Version 2 or above?
        jae     skipx1                  ;If yes, use int 21h to terminate
        call    FAR PTR proend        ;Call PROC "proend" to terminate
skipx1: nop                             ;Preferred terminate routine
        mov     ah,4Ch                  ;Put function number in ah
        int     21h                     ;Call terminate subroutine

        COMMENT * This section of the code contains PROCs (PROCEDUREs) or
                subroutines called by the program above.  A PROC is a
                 "real" subroutine as opposed to a MACRO.  Program
                 control is transferred to and from a single PROC.  MACROs
                 make in-line substitutions for code which can make the
                 program very lengthy. For clarity, the data items are
                 defined in the data segment of the main program (data
                 items can be defined in the PROCedure). *

        COMMENT * The following PROCedure transfers control back
                 to DOS.  Although it should work with any
                 version of DOS, it should only be used with DOS
                 version 1.  Interrupt 21h, function 4Ch, should
                 be used with all other DOS versions unless
                 downward compatibility is required.

                 The PROC works by pushing the address of the
                 Program Segment Prefix (PSP), stored in "psp"
                 in the data segment, onto the stack.  Then zero
                 is pushed onto the stack.  The PROC is then
                 returned.  Because this PROC is defined as
                 "FAR," the instruction pointer (IP) and code
                 segment address (register cs) is popped off
                 the stack.  In a "NEAR" PROC only the IP is
                 pushed and popped off the stack.  Because the
                 stack has zero and the PSP segment address on
                 top, the program is "fooled" into executing the
                 PSP code and terminating the program.

                 WARNING:  This PROC does not properly close
                 files which were changed by the program.
                 Close all files before executing this PROC. *

proend  PROC    FAR
        push    psp                     ;Push PSP address onto stack
        mov     dx,0                    ;Put 0 in dx
        push    dx                      ;Push 0 onto stack
        ret                             ;Pop stack and return
proend  ENDP                            ;End PROCedure

        COMMENT * The following PROCedure determines if the print flag
                 is set and, if so, prints the string identified by
                 the OFFSET in si.

                 This PROC uses interrupt 21h, function 05h.  No
                 other PROCs are used.  *
```

```
prnstr   PROC    NEAR                       ;Print string Procedure
         mov     al,prflag                  ;Put print flag in al
         and     al,1                       ;Compare to 1
         jnz     prnst4                     ;If 1, do not print - go to prnst4
         mov     di,si                      ;Save message OFFSET in di
         mov     cx,80                      ;Set counter to maximum line length
         mov     bx,0                       ;Zero bx
         cld                                ;Clear direction flag
prnst1:  lodsb                              ;Get a byte from address ds:si
         cmp     al,24h                     ;Compare byte (in al) to "$"
         je      prnst2                     ;Leave loop if byte is "$"
         inc     bx                         ;Else increment bx
         loop    prnst1                     ;Decrement cx and go to prnst1 if >0
prnst2: nop                                 ;Next function
         mov     cx,bx                      ;Move length of string to cx
         mov     si,di                      ;Put si at beginning of string
         cld                                ;Clear direction flag
         mov     ah,05h                     ;Put function number in ah
prnst3:  lodsb                              ;Get a byte from address ds:si
         mov     dl,al                      ;Put byte in dl
         int     21h                        ;Call print byte subroutine
         loop    prnst3                     ;Decrement cx and go to prnst3 if >0
prnst4: nop                                 ;Next function
         ret                                ;Pop stack and return
prnstr   ENDP                               ;End of PROCedure

code     ENDS                               ;End of CODE SEGMENT

         COMMENT * Define STACK SEGMENT *
stack    SEGMENT stack                      ;Program stack segment begins
         DW      64 DUP(?)                  ;Define stack space
stack    ENDS                               ;End of STACK SEGMENT

         END     start                      ;Mark end and define start
```

References

1. Microsoft Corporation, Microsoft Macro Assembler 5.0 Programmer's Guide, 1987, Microsoft Corporation, Redmond, Washington.

2. Microsoft Corporation, Microsoft Macro Assembler 5.0 Reference, 1987, Microsoft Corporation, Redmond, Washington.

3. Microsoft Corporation, Microsoft MS-DOS Operating System Programmer's Reference, 1984, Microsoft Corporation, Bellevue, Washington.

4. Phoenix Technologies Ltd., System BIOS for IBM PC/XT/AT Computers and Compatibles, 1989, Addison-Weseley Publishing Company, Inc., Reading, Massachusetts.

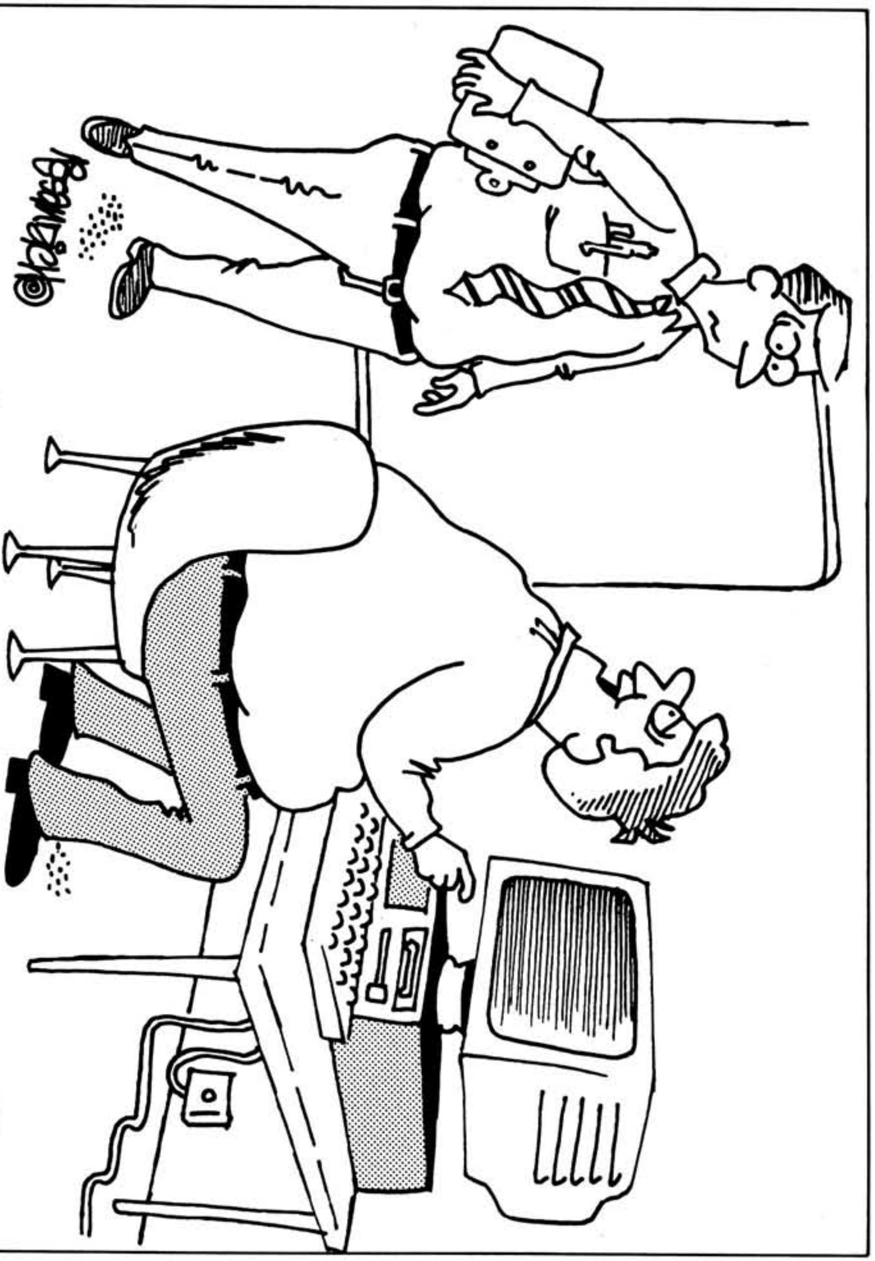
5. Que Corporation, DOS and BIOS Functions Quick Reference, 1989, Que Corporation, Carmel, Indiana.

6. Rector, Russell and George Alexy, The 8086 Book, 1980, OSBORNE/McGraw-Hill, Berkeley, California.

7. Scanlon, Leo J., 8086/88 Assembly Language Programming, 1984, Robert J. Brady Co., Bowie, Maryland.

8. Somerson, Paul, PC Magazine DOS Power Tools, 1988, Bantam Books, New York.

*



"I'M NOT GOING HOME UNTIL I'VE TRIED EVERYTHING ON THE MENU!"

The World of WP50 and Its Wonders
(4.2 and 5.1 will be considered)
Part Two

Salli Brackett
2201 Sycamore, #123
Antioch, CA 94575

We now continue the fascinating world of Wordperfect. In my first article (on page 15 of the October issue), we learned basic editing and cursor movements by creating a file called alice. If you followed the instructions in article one, you should now have alice and alice2. In this article we will use alice2 to print and then experiment with merging.

I have found that most people learn one way to print and never explore the other possibilities. Below is a description of the print menu and all its possibilities.

(At the end of this article is the Print menu for WP5.1 with explanations of the differences.)

```
Print

1 - Full Document
2 - Page
3 - Document
4 - Control Printer
5 - Type Through
6 - View Document
7 - Initialize Printer

Options

S - Select Printer        Standard Printer
B - Binding               0"
N - Number of Copies      1
G - Graphics Quality      Medium
T - Text Quality          High
```

1. **Full document** means the total document on the screen. The cursor can be anywhere in the document, and the total document will print.
2. **Page** means the page the cursor is on in the document on the screen.
3. **Document** means you print a file that is not on the screen. When you press 'D', WordPerfect asks you for a file name. After you type in the name and press ENTER, it asks you which pages you want.
   - To print total document, press ENTER.
   - To print one page, type page number (i.e., 2, ENTER).
   - To print a series of pages, type consecutive numbers with hyphen or non-consecutive numbers with comma (i.e., 2-4 or 2, 5, 7).
   - To print middle of document to end, type page # and a hyphen (i.e., 7-).
   - To print beginning of document to specific page, type a hyphen and the last page you want printed (i.e., -7).

   This is the only way you can print specific pages at one time. If you have a document on the screen and want to print specific pages. Save it. Then select this feature.
4. **Control Printer** allows you to cancel print, change order of print, stop print, or issue a go (necessary with manual feed). WordPerfect has a print queue which allows you to request several print jobs at once. (You don't need to wait till one document is printed to request another.) Below is a sample of the Control Printer Menu with 3 print jobs in the queue.

```
Print: Control Printer

Current Job

Job Number: 1                        Page Number: 1
Status: Printing                     Current Copy: 1 of 1
Message: None
Paper: Standard 85 x 66
Location: Continuous Feed
Action: None

Job List:
Job    Document             Destination   Print Options
1.     (Screen)             LPT 1
2.     (......\wp501.art)   LPT 1
3.     (.....\wp502.art     LPT 1         2 copies

Additional Jobs not shown: 0

1 Cancel Job(s); 2 Rush Job; 3 Display Jobs; 4 Go (start printer); 5 Stop;
```

5. **Type Through** makes WordPerfect a typewriter. Using this function, you can type directly to a printer. Preprinted forms would be an example of the use of this function.
6. **View Document** gives you an opportunity to see the document as it would print. It shows you the total page with headers, footers, etc. If you have columns in your document, you can see how they will space on the page. Depending on your monitor, any font changes will show as they would print, also.
7. **Initialize Printer** sends marked soft fonts to your printer. This function will be covered in more detailed in the article dealing with laser printers.
8. **Select Printer** gives you the option of selecting or editing printer file definitions which you have set up. This function will be covered in more detail in the article dealing with laser printers.
9. **Binding** is used when printing on paper with binder holes. Text will be moved over at print time so that it doesn't print over the holes. If you

change this option, it will take precedence over your left margin. BE SURE TO CHANGE IT BACK TO 0" WHEN YOU FINISH PRINTING.

10. **Number of Copies** gives you an option of printing more than one copy of your document. BE SURE TO CHANGE IT BACK TO 1 WHEN YOU FINISH PRINTING.
11. **Graphics Quality** and **Text Quality** allow some printers (particularly dot matrix) to print in different modes. If your printer has a draft mode (the fastest speed), you can use it to print something out quickly. The quality of the print is not as good. The 'Do Not Print' mode is for printers that can't print text and graphics in one pass. BE SURE TO CHANGE IT BACK TO 1 WHEN YOU FINISH PRINTING.
12. **Error messages** will come up under the **Control Printer** menu. If, after you print a document (on the screen or disk), you hear a BEEP, go to the **Control Printer** menu. There will be a **message**, a phrase under **Job Status**, and usually an **action** to take to correct it. Below are some common messages and what they mean.
    - **Waiting for a 'Go'**
      Insert standard form in Manual feed — Press Go to continue. This means that your standard form is set at manual. Therefore, you need to press Go for every page. To change your form to continuous or a Bin, go to Select Printer in the printer menu and edit your form.
    - **Trying to Print**
      Printer not accepting characters. The printer says 'I'm not ready to work.' There are several possible reasons for this. 1. Is the printer turned on? 2. Is the printer ON LINE? (There's a button on the front of your printer that says ON LINE. Make sure it is lit up.) 3. Are your cable connections at the printer or at the computer nice and tight? 4. If the answer to all above is yes and it still doesn't print, contact WordPerfect Printer Support.
    - **Document not formatted for current printer. Continue (Y/N)?**
      This means that when you saved the document the selected printer is not the one showing in your Printer Menu. Check your Select Printer Option (Shift F7.) If it is the correct printer, Press Y. If it isn't, Press N. Then press S for Select Printer and select the correct one. Now try printing your document again.
13. **Initialize.** Initializing the printer downloads soft fonts that were marked with an * in the printer file. You only need do this when you turn on your printer. The fonts are then downloaded to your printer until you turn your printer off.
14. **Down and dirty printing** means "a quick print but I can't remember the name of the file." It is done from the LIST FILES. Press F5. Cursor to **alice2**. Press 'P'. You will be asked which pages. For choice of pages, see #3.
    Use alice2 to print several different ways, experimenting with the explanations above.

Now that you are an expert at printing, let's explore the curious world of merging. I feel an explanation of how a merge works is helpful before actually creating the files necessary. (WP51 treats merge codes slightly differently. If you have WP51, check your manual. The concepts in this article will work for WP51.)

- **FIELD** is one area of information (variables).
- **RECORD** is one completed set of fields.
- Each **SET** of fields forms a RECORD.
  FOR EXAMPLE:
  Fields:

name, department, number (all three fields form a record)

OR

title, firstname, lastname, company, address, phone (all six fields form a record)

PRIMARY FILE is the form you wish the finished product to take. It can be a letter, envelope, list, etc. It uses the merge code F (^F) to delineate fields.

SECONDARY FILE is the list of variables in a specific format using the merge codes (^R and ^E). ^R delineates the end of a field; ^E delineates the end of a record.

MERGED DOCUMENT is the combination of both PRIMARY and SECONDARY files in final form. *This is the file you print.*

WHAT THE PROGRAM DOES DURING MERGE: (See Figure 1 for a visual concept of the merge process.)

The program looks at the variables (field designations using merge code F, ^F) in the PRIMARY file. It pulls the information for the corresponding variables (end of field designated by merge code R, ^R in the SECONDARY file) and places this information in the designated place (from the PRIMARY file) in the MERGED DOCUMENT. When it sees a merge code E, ^E, it creates a new page and repeats the process until all the records are used from the SECONDARY file. The result is a MERGED document with several pages of the PRIMARY document's form with different information in the variable designations. This is the document you print. YOU DON'T NEED TO SAVE IT.

TIP: I suggest naming primary files by subject, and secondary files with the same name EXCEPT the extension would be 'adr' (i.e., alice2 (letter), alice2.adr (secondary file)). That way

The Primary File would look like this:

```
SUBJECT
    DATE
TO:     ^Fname
        ^Fdepartment

FROM:   Judy Smith
        Operations

This is a test.
```

The Secondary File would look like this:

```
  ^N
  name
  department
  number
 ^N^R
 ^N^R
 ^E
---------------
 John Brown^R
 Accounting^R
 4567^R
 ^E
---------------
 Peggy Wright^R
 Personnel^R
 4238^R
```

The merged document would look like this:

```
SUBJECT                 DATE
TO:     John Brown
        Accounting

FROM:   Judy Smith
        Operations

This is a test.
                        Page 1
```

```
SUBJECT                 DATE
TO:     Peggy Wright
        Personnel

FROM:   Judy Smith
        Operations

This is a test.
                        Page 2
```

Figure 1. Visual concept of the merge process.

you can find all address files at the one time.

Press F5

Cursor to the end of the directory name

In place of the '.*' type .adr

You will have a list of all the secondary files.

First we will set up the secondary merge file. Then we can, using the same secondary file, merge it with several different primary merge files (the forms). *There may be some confusion at first. Complete the exercise and the **light** will go on.*

This example is to demonstrate setting up an address file. For the secondary file you will use merge codes N (next record), R (end field) and E (end of record). They are symbolized by ^N, ^R, ^E, respectively.

The first half of the secondary file is the description (or name) of your fields. The ^N and ^E are found by using Shift F9. (WP 4.2 uses Alt F9 for ^N and Shift F9 for ^E.) The ^R is F9.

Press Shift F9, n, ENTER.

Each name must have a begin bold and end bold around it.

Press **bold** (F6).

Type **title.**

Press the right arrow key, then ENTER. Reveal Codes (Alt F3) shows that there is a begin bold code [BOLD] and an end bold code [bold] around the word 'title.' The merge codes must be used or the file will not work.

Press **bold** (F6)

Type **firstname**

Press the right arrow key, then ENTER.

Press **bold** (F6)

type **lastname**

Press the right arrow key, then ENTER.

Press **bold** (F6)

Type **company**

Press the right arrow key, then ENTER.

Press **bold** (F6)

Type **address**

Press the right arrow key, then ENTER.

Press **bold** (F6)

Type **phone**

Press the right arrow key, then ENTER.

Press Shift F9, n, F9. (Repeat this **five** times.) Press Shift F9, e.

This is the end of the descriptive part of your secondary file. It should look like the sample below. You can copy this and 'append' to a special file and use it in any secondary file.

```
^N
title
firstname
lastname
company
address
phone
^N^R
^N^R
^N^R
^N^R
^N^R      ^E
```

Notice that there are 6 fields and 5 ^N^R - **always** put one less set of ^N^R than fields (the names in bold).

Below is the address list. The list can be as long as you like. For this example we will use two 'records.' Each 'record', ending with ^E, contains the fields typed above in bold. Start this list **immediately** underneath the ^E.

Ms. (Press F9)
Salli (Press F9)
Brackett(Press F9)
Press F9 **
2201 Sycamore, #123
Antioch, CA 94509 (Press F9)
779-0771 (Press F9)
Press Shift F9, e
Mr. (Press F9)
Joe (Press F9)
Cottontail (Press F9)
Wonderland, Inc. (Press F9)
111 Main Street
Wonder, LA 00000 (Press F9)
789-9999 (Press F9)

The last record needs no ^E.

** Even if there is no information for a given field, there *must* be a ^R for every field. Notice the second ^R after 'Brackett'. The 4th field is 'company.' Since there is no company name, just put in the merge R (^R).

You total secondary file should look like this:

```
^N
title
firstname
lastname
company
address
phone
^N^R
^N^R
^N^R
^N^R
^N^R
^E
===============================
Ms.^R
Salli ^R
Brackett^R
^R
2201 Sycamore, #123
Antioch, CA  94509^R
779-0771^R
^E
===============================
Mr.^R
Joe^R
Cottontail^R
Wonderland, Inc.^R
111 Main Street
Wonder, LA  00000^R
789-9999^R
```

MAKE SURE THERE ARE NO EXTRA BOLD CODES IN THE FIRST SECTION. MAKE SURE THAT THERE IS NO SPACE BETWEEN THE INFORMATION AND THE ^R'S IN THE SECOND SECTION.

Save this as alice2.adr.

You don't need names with the field. You can use field 1, field 2, field 3, etc. If you use this method, you don't need the upper half of the previous exercise. Just start putting in names.

The Primary File would look like this:

```
January 2, 1990

^F1^ ^F2^ ^F3^
 ^F4?^
^F5^

Dear ^F1^ ^F3^
```

The Secondary File would look like this:

```
Ms.^R
Salli^R
Brackett^R
^R
^2201 Sycamore, #123
^Antioch, CA  94509^R
^779-0071^R
^E
_____________________________
Mr.^R
Joe^R
Cottontail^R
Wonderland, Inc.^R
111 Main Street
Wonder, LA  00000^R
^789-9999^R
```

Now let's create a letter, an envelope, labels, and a mailing list. These are all PRIMARY files.

A LETTER SAMPLE: (Retrieve (Shift F10) alice2)

NOTE: For your own stationary, you would measure down from the top of the paper.

Determine what is the first line you can print on to make the letter look proper. Your top margin will be 1 line less than the first printed line. For this form assume the letter will be a full page. Make any font changes or l/r margin changes you want.

The ^F is the Merge Code F. Press Shift F9, F. Type in name of field when requested and press ENTER. The field names will be the ones you created **IN BOLD** in the secondary file. They must be EXACTLY the same.

Cursor to 'John'
Del End of Line (Ctrl End)
Press Shift F9, f
Type **title**
ENTER, SPACEBAR
Press Shift F9, f
Type **firstname**
ENTER, SPACEBAR
Press Shift F9, f
Type **lastname**
ENTER

Cursor down one
Press Home, Left Arrow
Del End of Line (Ctrl End)
Press Shift F9, f
Type **company**
ENTER

The question mark after 'company' is to eliminate a blank line if this field is empty. Cursor to the last ^ after company and insert a '?.'

Cursor down one
Press Home, Left Arrow
Delete End of Line (Ctrl End)
Press Shift F9, f
Type **address**
ENTER

Cursor to 'M' in **Mr.**
Delete End of Line (Ctrl End)
Press Shift F9, f
Type **title**
ENTER, SPACEBAR
Press Shift F9, f
Type **lastname**
ENTER
Type colon (:)

Your letter should look like the sample in Figure 2.
Save it as alice2.mrg

AN ENVELOPE SAMPLE:

Clear the screen (F7,n,n).

Make an envelope file. (You must have an envelope form already in your printer definition. To create an envelope form, check instructions on last page.)

Change Paper Size

Shift F8, p (page format), s (size), e (envelope), e (envelope)

Change margins

m (margins), 12, ENTER, 0, ENTER, ENTER, l (line format), m (margins), 40, ENTER, 2 ENTER, EXIT

Using the same method as in the LETTER SAMPLE, type as below. REMEMBER: the ^F is merge code F (Shift F9, f).

^Ftitle^ ^Ffirstname^ ^Flastname^
^Fcompany?^
^Faddress^

Save as **env.mrg**

Now you can merge the letter (alice2.mrg) and the envelope (env.mrg) with the secondary file (alice.adr). Clear Screen (F7, n, n).

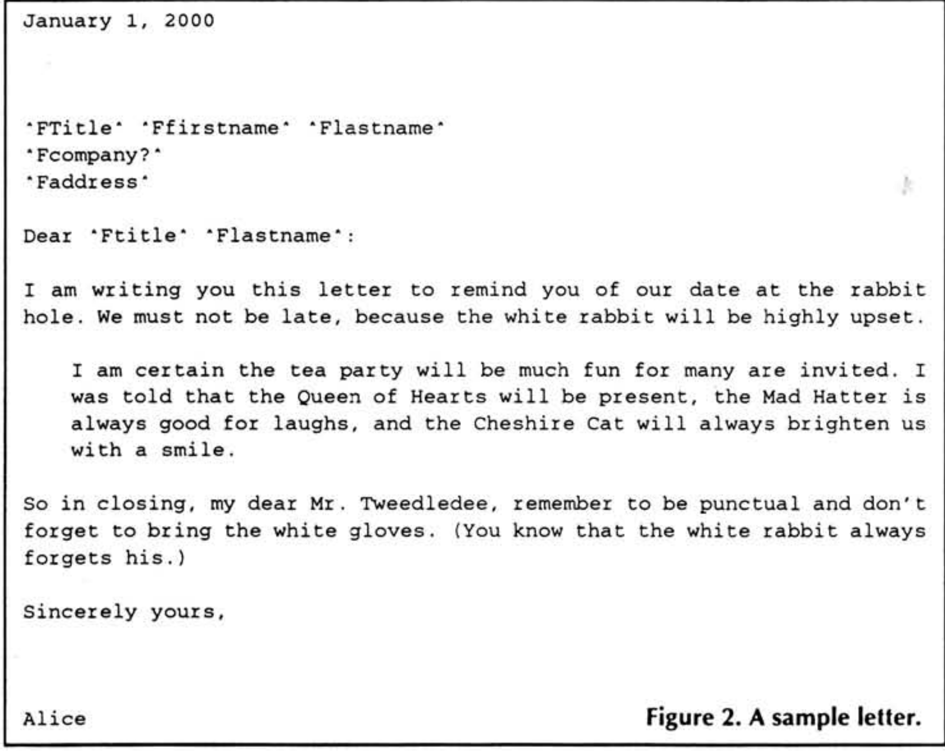
To merge:
Press Ctrl F9, m
Type **alice2.mrg**
Press ENTER
Type **alice2.adr**
Press ENTER

Go to top of document. Look at the address, then page down. You will see the different addresses with the same letter. Clear Screen (F7, n, n).

Now create the envelope merge, using Ctrl F9. The primary file is env.mrg; the secondary file is **alice2**.adr. Each page will have a different name. If you wish, print the merged document. It is not necessary to save it.

NOTE: If you see a blank line between 'Salli Brackett' and '2201...', that means you forgot the '?' after 'company' in the primary file (env.mrg). Retrieve env.mrg, edit it, save and do the merge again.

The next sample is much more complicated, but the understanding of this sample will help you to create many other files of similar nature. This is my personal favorite for that reason. You can use the same concept (with modifications) to create 2 or 3 across labels or any type of list that reads better in column format. The following example will be 2 columns across, but the concept works for 3 or 4 as well. **THE NEXT TWO EXAMPLES NEED TO BE FOLLOWED VERY CAREFULLY.**



```
January 1, 2000



^FTitle^ ^Ffirstname^ ^Flastname^
^Fcompany?^
^Faddress^

Dear ^Ftitle^ ^Flastname^:

I am writing you this letter to remind you of our date at the rabbit
hole. We must not be late, because the white rabbit will be highly upset.

    I am certain the tea party will be much fun for many are invited. I
    was told that the Queen of Hearts will be present, the Mad Hatter is
    always good for laughs, and the Cheshire Cat will always brighten us
    with a smile.

So in closing, my dear Mr. Tweedledee, remember to be punctual and don't
forget to bring the white gloves. (You know that the white rabbit always
forgets his.)

Sincerely yours,



Alice
```

Figure 2. A sample letter.

A MAILING LIST WITH TWO COLUMNS:

(WP51 treats this differently. Read the manual under merge/labels.) Clear Screen (F7, n, n).

NOTE: If you wish to change l/r margins or fonts, do so before setting column definition.

Create a column definition (ALT F7)

Press d (define), t (type), p (parallel), m (margins: 8, 45, 50, 80), F7.

Turn column on

Press c (column) (This is a toggle, an on/off switch).

Type in first column. REMEMBER: The ^F is merge code F (Shift F9, f).

^Ftitle^ ^Ffirstname^ ^Flastname^
^Fcompany?^
^Faddress^
^Fphone^

At the end of ^Fphone^, press Ctrl Enter. This gives you your second column. Look at your status line. It says ' Col 2 Doc 1 Pg 1 ...'

(To move between columns, use the GOTO key (Ctrl Home and the arrow keys).)

Go back to the first column (Ctrl Home, Left Arrow).

Copy the merge codes in column 1 to column 2.

Cursor to ^Ftitle^ (make sure you are passed the Col On code (ALT 3))
Press Block (ALT F4)
Cursor to the ^ after phone.
Press Move (Ctrl F4)
Press b (block), c (copy)
Goto 2nd column (Ctrl Home, right arrow)
Press ENTER
Press Shift F9, n (This means print next record here.)
Cursor to the end of ^Fphone^ in the second column.

Now turn off the column mode.

Press ALT F7, c (column), (this turns the column mode off)

Put in command to make merge print on same page.

^P means 'primary file name.' When merged, WordPerfect will print any file that is named in between the p's, i.e. ^Pletter^P. If there is no name, it will print the original primary file. The combination ^N^P^P tells WordPerfect to continue to print to the end of the page.

Press ENTER, ENTER (This gives you two blank lines between names.) Press Shift F9, n

Press Shift F9, p

Press Shift F9, p

As a safety measure against addresses splitting improperly at the end of a page, Block Protect both columns.

Cursor to beginning of Column 1, making sure you are passed all codes (ALT F3).

Press ALT F4, cursor to end of Column 2, Shift F8, y.

If you wish page numbers on your list, create a header with 2 extra blank lines (like the one below) and use a suppress page (so that the header doesn't print on page one).

Cursor to top of document, making sure to be at THE BEGINNING of all codes.

Press Shift F8, p (page format), h (header), a (header A), p (every page - create header)

Type

Alice's Mailing List
Page ^B (this is Control B)

Press ENTER twice

The header looks like this:

```
Alice's Mailing List
Page ^B
```

Suppress page:

Press F7, u (suppress), a (all), F7

You finished document should look like this:

```
^Ftitle^ ^Ffirstname^ ^Flastname^        ^N^Ftitle^ ^Ffirstname^ ^Flastname^
^Fcompany?^                              ^Fcompany?^
^Faddress^                               ^Faddress^


^N^P^P
```

Save as alice2.lst.

To best test this file, you need more names in the secondary file.

WONDER #5: WP50 allows you to copy an item several times but blocking only once.

Follow instructions below:

1. Retrieve (Shift F10) alice2.adr
   Cursor to end of file
   Press F9, e (must have ^E because this is not end of file)
2. Place cursor on 'm' in 'Ms.'
   Press ALT F4
   Goto end of file
   Press Del, y
3. Press F1, r (undelete key)
4. Repeat #3 until you have 32 names (pages)
5. Save and replace (F7, y, enter, y)

Now create the mailing list, using Ctrl F9. The primary file is **alice2**.lst; the secondary file is **alice2**.adr.

LABEL FORM: - This is for laser printers ONLY (The sample below is for Avery Label 5160; it has 3 labels across, 3 lines of top/bottom margin and labels start 3 columns in from the left.) The concepts learned below will work on any label but the margins, etc. will be different.

BEWARE: The problem with labels in a laser printer is the consistency of the placement. The key to this is creating a 'dummy' column and putting in advance down codes and fixed line height.

Retrieve (Shift F10) alice2.lst. Make sure the cursor is on the 'Col Def' code (Reveal Codes, ALT F3). To create the label form, you will use **alice2**.lst with a few modifications.

First you change the t/b margin and l/r margins and justification.

Press F8, p (page format)
Change top/bottom margin (press M)
Press 3
Press ENTER
Press 3
Press ENTER twice
Press l (line format)
Press m (margins)
Press 3
Press ENTER
Press 3
Press ENTER
Press j, n (justification off)
Press F7

Cursor passed the column definition code (Reveal Codes F11 or Alt F3).

Next change the column definition codes.

Press Alt F7, d (define)
Press n (number of columns)
Press 4
Press ENTER
Press m (margins)

Below are the column margins based on a 10 pitch font, press ENTER between each entry.

2, 3
4, 28
31, 58
60, 89 ENTER
Press F7 twice

Delete the first definition code. (Cursor to the first definition in Reveal Codes and press DEL.)

Now you will add column one (the dummy column).

Cursor passed the 'col on' definition (ALT F3).

Press Ctrl ENTER (You are now in the second column. Check status line.)

Return to Col 1 (Ctrl Home, Left Arrow)

Place the codes for advancing down and a fixed line height in the first column

Press Shift F8
Press o (other), a (advance), d (down), 5

(This tells the printer to advance down 5 lines. There are 6 lines in each label. You always advance down 1 line less than the total lines in the label.)

Press ENTER, ENTER

Press l (line format), h (line height), f (fixed), 1

(This makes the line height fixed at one.)

Press ENTER
Press F7

Add column 4 and copy merge codes to column 3 and 4.

Go to Col 3 (Ctrl Home, Right Arrow, Ctrl Home, Right Arrow)

Cursor to ^Fphone^

Delete line (Ctrl END)

Create 4th column (Ctrl ENTER)

Return to Col 3 (Ctrl Home, Left Arrow)

Copy merge codes to column 4 (Alt F4, Cursor to end of codes, Ctrl F4, b (block), c (copy), Ctrl Home, Right Arrow, ENTER)

Return to column 2 (Ctrl Home, Left Arrow, Ctrl Home, Left Arrow)

Cursor to ^Fphone^
Delete line (Ctrl END)
Cursor down one
Press DEL twice
Save this as alice2.lbl.

To test this merge with alice2.adr and print on plain paper. If you hold the printed paper up to the labels, you can see if they are properly set.

You now know the basics of merges. Have fun experimenting with the other codes. Explanations of these are in your manual.

Creating an Envelope Form in WP50.

Shift F7
S (select printer)
E (edit)
F (forms)
Cursor to standard
A (add)
E (envelope)
S (size)
E (envelope)
O (orientation)
L (landscape)

Press F7 until back to document screen

Here is the WP51 Print Menu:

```
Print

1 - Full Document
2 - Page
3 - Document
4 - Control Printer
5 - Multiple Pages
6 - View Document
7 - Initialize Printer

Options

S - Select Printer          Standard Printer
```

```
B - Binding                        0"
N - Number of Copies               1
U - Multiple Copies Generated By   WordPerfect
G - Graphics Quality               Medium
T - Text Quality                   High
```

1. Multiple Pages allows you to print specific pages from a document on the screen.
2. Multiple copies Generated By allows you use the Number of Copies option on this menu or chose the Copies choice on your printer.

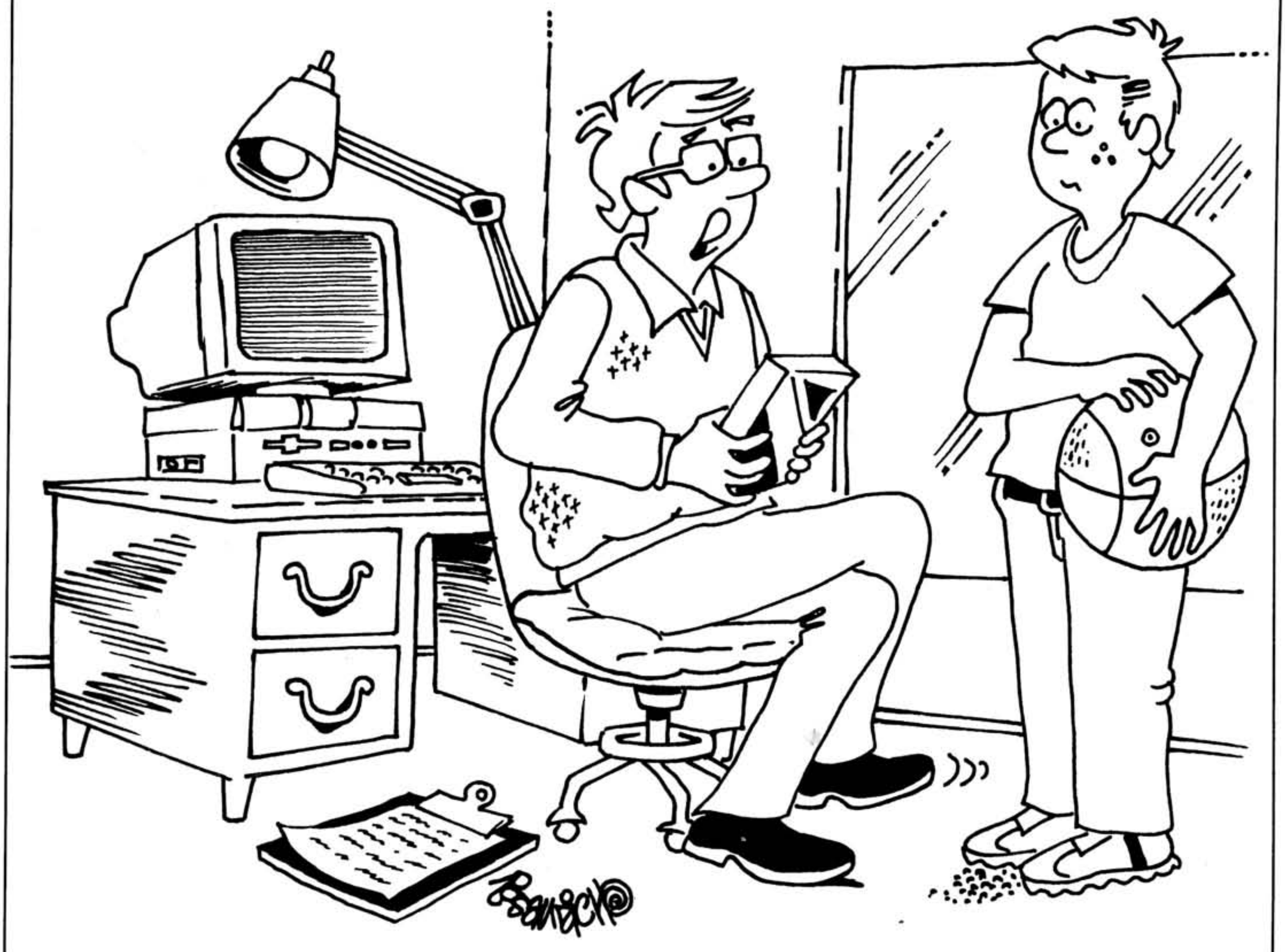
✱

Continued from Page 5

on his keyboard and wanted to know where to buy a new one. Within 24 hours, he was informed that a little WD-40 can loosen up sticky keys and save a lot of money. The bulletin board is full of useful information like this and I'll do my best to pass as much of it as possible. Next month we'll be taking a look at a problem that so many Z-386/16 owners have run into with Windows® 3.0!

Meanwhile, I would like to thank those who participated in this month's high density drive problem: Brian Hanson, Chuck Santose, Kittredge Seely, Rudy Swider, and Denis Tominski.

✱



"IT'S A VCR TUTORIAL ON HOW TO RUN A HOME COMPUTER... THE PROBLEM IS I HAVEN'T FIGURED OUT HOW TO RUN MY VCR YET!"

QEMM-386

Jeffrey F. Goldman
3155 Rhapsody Court
Colorado Springs, Colorado 80920

Quarterdeck Office Systems Expanded memory manager 386 (QEMM-386) is a control program to enable your 386 based PC to access more than 640K of RAM under MS-DOS. It takes advantage of the unused memory areas in the address space of DOS between 640K and 1024K. Before we move on to a full discussion of QEMM-386, lets take a look at how we wound up with the limitations imposed on us by MS-DOS.

When IBM designed the IBM PC, it made certain hardware and software decisions that have imposed limitations on the DOS architecture we must live with today. The limitations imposed by these decisions essentially cripple the more advanced capabilities available in the 80286, 80836 and 80486 microprocessors. Until recently, despite the ability to access 16 megabytes (80286 and above) of memory, only 640K of memory is available to programs running under MS-DOS.

Why did we end up with such a limited operating system? When IBM introduced the IBM PC in 1981, the state of the art was the Apple II, the Heath H89, and the Radio Shack TRS-80. For these systems, 48k was a standard amount of memory and 64K was the limit imposed by their microprocessor architecture. For any number of reasons (cost being the primary one), IBM chose the Intel 8088 microprocessor to be the brains of its new computer system. The 8088 chip is a hybrid - a 16/8 bit chip. This means it has a 16-bit internal processing architecture but only an 8-bit Input/Output (I/O). In addition, it has a 20 bit memory address which allows it to access 1 megabyte (1024K) of memory. Of this total address space, decisions had to be made about how much memory to reserve for programs and how much memory to reserve for system overhead (video adapters, rom bios, future expansion, etc.).

Although CPM by Digital Research was the standard for 8-bit microcomputers, IBM chose to go with a startup company, Microsoft. The name of the operating system was PC-DOS (Personal Computer Disk Operation System) and the generic version MS-DOS (Microsoft Disk Operating System). With this marriage of IBM and Microsoft, decisions were made about how to divide the address space of 1024K for the PC. This decision is summarized in Figure 1. Based on the size of memory at the time, 640K was considered more memory than anyone thought they would need for a personal computer. In fact, Zenith made some different decisions on the 1 megabyte architecture which is why Z-100s can address 768K of main memory. As programs became more powerful and feature laden (and the advent of memory resident programs such as Sidekick became popular), the 640K limit of DOS became a problem. Although newer microprocessors could access more than 640K of RAM, no DOS based program could use it. The ability to use more than 640K requires a new operating system, Operating System 2 (OS/2). The problem is most of the programs people are using are still DOS based programs. With the development of the newest processors, 80386 and 80486, there are advanced memory management functions built in to them that allow them to use some of the memory between 640K and 1024K without interfering with DOS. This is the function of Quarterdeck Systems QEMM-386.

As you can see from Figure 1, depending on how your system is configured, there are a number of areas in the 640K-1024K address that IBM and Microsoft reserved for future use that are unfilled by anything. QEMM-386 finds these unused address spaces and uses them for things such as device drivers and memory resident programs that would normally use up conventional memory. In figure 2, you can see the picture that QEMM-386 has of my memory map.

QEMM-386 has three primary functions on a 386 machine. The first is to manage extended memory. This is an important

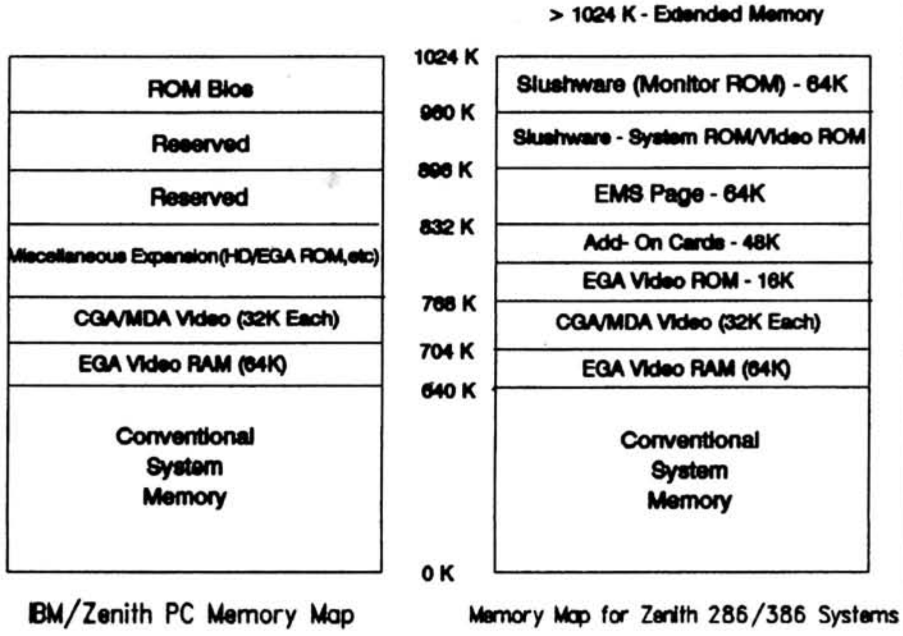


Figure 1
Memory Maps

feature because it allows you to use all of the extended memory in an 80386 machine as either extended memory, expanded memory, or a combination of the two. Why is this important? It is important because you don't have to change the dip switches on your expanded memory board every time you change your memory requirements. It will also alleviate the problem Bill Adney had with configuring his Z-515 memory board (Remark April 1990). There are constantly changing memory requirements between extended and expanded memory (Lotus 123 V.3, Microsoft Windows, and many other new programs) and QEMM-386 gives you the capability to switch the types of memory you have in your system.

The second function of QEMM-386 is to map memory resident programs and device drivers into the unused memory segments between 640K and 1024K (1 megabyte). This saves a considerable amount of conventional memory - the memory which all of the DOS programs need. This is important because many programs require at least 512K of free RAM in order to run. With a couple of memory resident programs loaded you can easily overload your conventional memory confronting you with the prospect of unloading some of your memory resident programs or giving up your favorite word processor or spreadsheet. In addition, it may speed up or increase the capacity of your programs because most DOS programs run better with more RAM because they are able to load more of themselves or their data files into faster RAM rather than having to access the disk. The last function of QEMM-386 is not really of concern to Zenith owners - ROM shadowing. Zenith machines already have this in hardware - referred to in the manuals as slushware. This feature of Zenith machines did cause me some problems in installation.

Now that we've gone through the basics of memory in Zenith PC systems, lets see what QEMM-386 will do for us. My machine is a Zenith 386/16 running MS-DOS 3.3 Plus with a 71 Mbyte Miniscribe hard disk, 2 Z-505 Memory Boards(2 megabytes of RAM total), a Prism Imaging Systems Elite VGA card and a Samsung CN-1455N Multisync Monitor. My config.sys and autoexec.bat files with QEMM-386 are as follows:

Config.sys with QEMM-386

```
FILES=20
BUFFERS=1
device=c:\util\QEMM.SYS X=C000-C7FF
        X=E000-E7FF EXTMEM=678 RAM
DEVICE=c:\util\loadhi.sys C:\BIN\ANSI.SYS
device=c:\util\loadhi.sys
        c:\bin\vdisk.sys 38 128 16/E
device=c:\util\loadhi.sys
        c:\bin\burndev.sys 5000 V-
```

Autoexec.bat with QEMM-386

```
echo off
prompt $e[37;44m
path=e:\;c:\util;c:\util\pct;c:\bin;c:\bat;c:\win
loadhi egadmp h >nul
loadhi buffers 20
loadhi ced -fc:\util\ced.cfg
loadhi mouse
pc-cache/sizext=640/ia/ib/max=16k
menu
```

I will now explain each entry of the config.sys and autoexec.bat files. Loadhi.sys and loadhi.com are the Quarterdeck programs to load device drivers and memory resident programs into high memory. Files tells DOS how many different files a DOS program can have open at once and many programs require up to 20 (default is 8). Buffers are a small DOS based disk cache (512 bytes per buffer). I have it set to one because I use the PC-Tools disk cache. Ansi.sys is the standard Ansi device driver that some programs require for graphics displays and also allows me to use prompt to display screen colors. Vdisk.sys is the DOS supplied ram drive software and burndev.sys is a screen blanker.

In the autoexec.bat file, egadmp is a memory resident graphics screen dump program for EGA/VGA displays. CED (Command Editor) is a memory resident program that retains DOS commands in a buffer and defines expanded commands under DOS (DOS based macros). Mouse is the standard mouse driver for the Logitech mouse. Buffers is a Quarterdeck Systems supplied program that allows you to add DOS buffers outside of the config.sys file. All of the above memory resident programs and device drivers can be loaded into high memory.

The net gain in memory by using QEMM-386 was 32K. This figure was calculated by removing all loadhi.sys and loadhi.com statements from the config.sys and autoexec.bat files and rebooting without QEMM-386 loaded. Before using QEMM-386, I had 546,672 bytes after loading all my device drivers, memory resident programs, and DOS. After using QEMM-386, I had 578,400 bytes free.

As far as the installation and customization features, the manual was very short - 14 pages. When I initially loaded up QEMM-386, my machine locked up and I could not get it to run any of my programs. Resolving this problem required a special entry for the QEMM.SYS device driver (X=C000-C7FF, X=E000-E7FF). Because of the way Zenith's ROM BIOS is configured, you have to exclude those memory areas where there is a conflict. To find this out, I had occasion to call Quarterdeck's Technical Support. I would rate their support good. When I got through the technicians were excellent, but it is not toll-free and sometimes I was put on hold for as long as 20 minutes. In addition, technical support is only provided free for the first 90 days.

The other startup parameter in the QEMM.SYS line designates how much of the total extended memory remains as extended memory. QEMM-386 automatically configures all other extended memory as expanded memory compatible with the LIM Expanded Memory specification version 4.0.

There are a number of other things QEMM can do if you have conflicts with other programs. You can specify which areas of memory to include or exclude, which area of memory to use for the expanded memory page frame (if there is a conflict with the standard one), and finally whether to map the high memory. Also, if you have added a 386 add-in board (ie. Intels' Inboard 386) it can sort your memory by speed and use the fastest memory for your conventional memory. Finally, if you really need to maximize your memory usage and are using a character based program, QEMM can map the video ROM space into conventional memory and gain up to an additional 64K of RAM under DOS.

In Figure 2, you can see what QEMM views as the memory map of my machine. It includes two programs to do this - QEMM.COM and LOADHI.COM. All of the memory addresses are listed in hexadecimal arithmetic. QEMM.COM shows where QEMM-386 thinks various DOS functions are in memory and LOADHI.COM shows where programs and device drivers have been moved into the high ram.

The bottom line is if you have a 386 based machine you should have QEMM-386. In addition, they now have a program for 8088/80286 machines with expanded memory called QRAM. According to the company, it will do many of the same things for those machines as QEMM-386 does for a 386. Although QEMM-386 has a retail price of $59.95, I was able to obtain it through mail order for $38 ($35 plus $3 S/H).

Contact
Quarterdeck Office Systems
150 Pico Boulevard
Santa Monica, CA 90405
Phone: (213) 392-9851
Fax: (213) 399-3802

```
Output of QEMM.COM                         Output of LOADHI.COM

Area          Size  Status         Area        Size  Status
0000 - 9FFF   640K  Conventional   B000-B0F0   3K    Used
A000 - AFFF   64K   Video          B0F1-B0F9   0K    Used (egadmp)
B000 - B7FF   32K   High Ram       B0FA-B102   0K    Used (ced)
B800 - BFFF   32K   Video          B103-B3B7   10K   Used (mouse)
C000 - C7FF   32K   Excluded       B3B8-B7FE   17K   Available
C800 - CFFF   32K   High Ram       C800-C88E   2K    Used (egadmp)
D000 - DFFF   64K   Page Frame     C88F-CC95   16K   Used (ced)
E000 - E7FF   32K   Excluded       CC96-CFFE   13K   Available
E800 - EFFF   32K   High Ram       E800-E863   1K    Used
F000 - FFF    64K   ROM            E864-EADD   9K    Used
                                   EADE-EFFF   20K   Available
```

Figure 2

*

Classified Ads

DESKJET PRINT CARTRIDGES REFILLED Black ink, $8.00. Send print cartridge and check to Altomare PC Solutions, 27 Cove, Alameda, CA 94501.

FOR SALE: Z-100 Two 360k drives, C.ITOH color monitor, best offer, (303) 674-0522.

"DO YOU MIND IF I USE IT FOR A CHANGE?"

SEAGATE ST-251-1 42 MEG HARD DRIVES ON SALE

Seagate ⇨HARD DRIVES

MODEL	CAPACITY/FORMAT/SPEED/SIZE	DRIVE ONLY	XT KIT
* ST-125	21 MEG / MFM / 40 MS / 3.5"	$229.00	$277.00
* ST-138	32 MEG / MFM / 40 MS / 3.5"	$277.00	$325.00
* ST-138-1	32 MEG / MFM / 28 MS / 3.5"	$307.00	$355.00
* ST-151	42 MEG / MFM / 24 MS / 3.5"	$353.00	$401.00
* ST-157R	49 MEG / RLL / 40 MS / 3.5"	$286.00	$339.00
* ST-225	21 MEG / MFM / 65 MS / 5.25"	$199.00	$247.00
*** ST-250R**	**42 MEG / RLL / 70 MS / 5.25"**	**$248.00**	**$288.00**
*** ST-251-1**	**42 MEG / MFM / 28 MS / 5.25"**	**$289.00**	**$337.00**
* ST-4096	80 MEG / MFM / 28 MS / 5.25" FH	$582.00	$631.00
* ST-238R	32 MEG / RLL / 65 MS / 5.25"	$218.00	$271.00
* ST-277R-1	65 MEG / RLL / 28 MS / 5.25"	$348.00	$401.00
* ST4144R	122 MEG / RLL / 28 MS / 5.25" FH	$623.00	$671.00

* IDE, SCSI, ESDI AND OTHER SEAGATE MODELS AVAILABLE. PLEASE CALL.

*** ZENITH PC COMPUTER UPGRADES ***

SmartWatch from FBE RESEARCH

⇨Installs in ROM Socket on the CPU Board in Zenith computer series Z-100/138/148/150/160 and most all other XT computers. This clock/calendar contains a ten year battery and keeps your computer informed of both date and time at each boot-up. Instructions and software included. $35.00

Z-150/160 MEMORY UPGRADE

⇨This kit includes a replacement memory decoder PAL chip for the standard Z-150/160 memory card (not for the Z-157/58). With this PAL and the 18 pieces of 256K RAM chips (included), you will expand the memory on the card to 640K or 704K. ZP640+/18 Kit.....$59.00. PAL chip only ZP640+....$18.00

Z-150 SERIES HARD DISK DRIVE KIT

⇨These kits include high speed Seagate drives with autopark heads. Each kit includes all cables, hardware and instructions to mount the hard drive under your two floppy drives in your Z-150 series computer.

* ST-125/Z150 Kit	21 Meg, 40 MS,	$281.00	
* ST-138/Z150 Kit	32 Meg, 40 MS,	$329.00	
*** ST-251/Z150 Kit**	**42 Meg, 28 MS,**	**$341.00**	**SALE PRICED**

Z-148 HARD DISK DRIVE KIT

⇨Includes the hard disk drive and a Z-148 compatable controller together with the Z-148 Expansion Card described below. All required cables, hardware and instructions are included for you to replace one floppy with a Seagate Hard Drive in your Z-148. Add only $30.00 the the following price if your would like us to include a SmartWatch.

* ST-125/Z148 Kit	21 Meg, 40 MS,	$352.00	
* ST-138/Z148 Kit	32 Meg, 40 MS,	$399.00	
*** ST-251/Z148 Kit**	**42 Meg, 28 MS,**	**$408.00**	**SALE PRICED**

Z-148 EXPANSION CARD

⇨Adds one full length and one half length IBM expansion slot to your Z-148 for hard drive controller, video card, modem, etc. ZEX-148......$79.00

Z-150 VIDEO ELIMINATOR

⇨For the Z-150 or Z-160 only. Not required for the Z-157/158/159 computers. A small piggyback board which replaces the scratch pad memory on your current video card. This allows the removal of the original Zenith video card and replacment with an EGA, VGA or any other 8 bit video card. Order VCE-150 .... $54.00

2400 BAUD MODEMS

⇨Fully Hayes compatible 2400/1200/300 Baud with Software. Internal $84.00 External Model $128.00 Cable for External Modem $8.50

PAYLOAD

15718 SYLVAN LAKE * HOUSTON TX 77062

⇨**MICROSOFT WINDOWS SOFTWARE** version 1.04 for PC (not Z-100)
⇨**FREE** with any order from this ad. Just ask for it and add $5.00 for shipping and
⇨handling. Includes 5 disks and 300+ page manual. Offer good for limited time.

WESTERN DIGITAL HARD DISK CONTROLLER CARDS

⇨WD XT GEN2+ 8 BIT, MFM, DUAL HARD DRIVES, XT COMPUTERS	$48.00
⇨WD1004-27X 8 BIT, RLL, DUAL HARD DRIVES, XT COMPUTERS	$53.00
⇨WD1006V-MM2 16 BIT, DUAL HARD, DUAL FLOPPY, 1:1 AT	$99.00
⇨WD1006V-SR2 16 BIT, RLL, DUAL HARD, DUAL FLOPPY, 1:1 AT	$109.00

⇨HARD DRIVE CABLE SETS, XT.. $3.50, AT .. $4.50, AT + Floppy $6.50

PRINTERS

⇨STAR NX-1000 II 9 PIN, 180 CPS, 45 CPS NLQ,	$179.00
⇨SEIKOSHA SL-90 24 PIN, 240 CPS, 80 CPS NLQ, PAPER PARKING	$314.00
⇨PANASONIC 1180 9PIN, 192 CPS	$204.00
⇨PANASONIC 1124 24 PIN, 192 CPS	4335.00
⇨PANASONIC 1624 24 PIN, 192 CPS, WIDE CARRAGE	$422.00
⇨PANASONIC 1695 9 PIN, 288 CPS, WIDE CARRAGE	$458.00
⇨PRINTER CABLE	$12.00

FLOPPY DISK DRIVES

⇨MITSUBISHI MF501	5.25" 48 TPI DS/DD 320K/360K	$ 68.00
⇨MIITSUBISHI MF504	5.25" High Density 360K/1.2 MEG	$ 81.00
⇨MITSUBISHI M-353	3.5" in 5.25" frame 720K	$ 84.00
⇨MITSUBISHI M-355	3.5" in 5.25" frame 1.44 MEG	$ 94.00
⇨TOSHIBA ND352	3.5" with 5.25" frame 720K	$ 74.00
⇨TOSHIBA ND356	3.5" with 5.25" frame 1.44 MEG	$ 79.00

⇨M-355 and ND356 run on AT compatible or special controller only.

PAYLOAD CUSTOM ASSEMBLED COMPUTERS

⇨We assemble 8088 XT, 80286 AT, 80386SX or 80386 IBM compatible computers to your specifications. Please write or call for a work-up sheet showing items available and prices.

VIDEO MONITORS

⇨ZCM-1492	ZENITH Color Flat Screen VGA	$ 679.00
⇨MA2565	SAMSUNG Amber TTL 720x350	$ 89.00
⇨CW4656	SAMSUNG Color RGB 640x200	$ 223.00
⇨CM4592	SAMSUNG Color EGA 640x350	$ 339.00
⇨CJ4681	SAMSUNG VGA 720x400	$ 360.00
⇨CVB4581	SAMSUNG Multi-sync VGA 1024x768	$ 429.00
⇨CM1440	SEIKO VGA 1024x768 .25 dot	$ 569.00

VIDEO CARDS

⇨BASIC VGA	PARADISE AUTOSWITCH 640x480	$ 99.00
⇨VGAPLUS	PARADISE AUTOSWITCH 800x600	$ 146.00
⇨VGAPLUS16	PARADISE AUTO 16 BIT 800x600	$ 177.00
⇨VGA 1024-256k	PARADISE AUTO 1024x768	$ 194.00
⇨VGA 1024-512k	PARADISE AUTO 1024x768	$ 247.00

MEMORY CHIPS, ETC.

⇨Memory chips are once again at reasonable prices. The market prices have been changing daily, therefore we are only able to list estimated prices. Please call for the current price before placing your order. We buy in large quantities and work on the smallest of margins in order to bring you great values.

⇨41256 256x1 80 ns.....$2.95	SIP 1Mx9 80 ns $68.00
⇨41256 256x1 100 ns $2.65	SIM 1Mx9 80 ns $69.00
⇨41256 256x1 120 ns $2.30	SIM 256x9 80 ns $24.00
⇨1Mbit 1Mx1 80 ns $7.40	V-20 replaces 8088 CPU $14.00

Z-100 SERIES COMPUTER UPGRADES

High Density 1.2 Meg Drives

⇨External floppy drive set-up complete with drive, power supply, case and cables. Ready to connect to your 8" floppy controller. Single Drive Unit $207.00

⇨Dual Drive Unit $299.00 Bare drive and cable for internal mount $115.00

ZMF100A by FBE Research

⇨A modification package which allows 256K chips to be used on the old-style motherboard (part number85-2653-1) to reach 768K. Simple assembly with no soldering or trace cutting. Compatible with Easy PC and Gemini Emulator. Order 27 256x1 RAM chips to complete this kit. ZMF100A...............................$60.00

⇨Please Mail, Phone or FAX your order. All hardware carries the manufacturers warranty plus the PAYLOAD 90 day guarantee. No surcharge on credit card orders. COD shipments on request. Add $5.00 to all prepaid orders for handling and shipping in the Continental USA, we pay the balance. Actual shipping costs for foreign, overseas and net billing orders. We accept purchase orders from schools, government and approved accounts. Mail or phone your order for prompt friendly service. Texas residents please add 8.25% state sales tax.

PHONES: ** ORDERS AND INFO **(713) 486-0687** ** FAX: **(713) 486-8994** **

PAYLOAD * PAYLOAD * PAYLOAD * PAYLOAD

Reader Service #117

On the Leading Edge

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

ZDS MS-DOS 4.0, COMMAND Command, SHARE Command, SHELL= Command, INSTALL= Command, Powering Up and Down

I have been working with ZDS MS-DOS 4.0 for several months now, and I have discovered some interesting things about how it works. There are some things I should have guessed sooner, but there was a very good reason that I did not. And that reason was that I did not have a bootable partition larger than 32 megabytes. I recently changed the bootable partition size from about 30 MB to nearly 50 MB when I installed ZDS MS-DOS 4.0. As most of you know by now, my drive C contains all of the software I use, and drive D is used strictly for data to minimize the backup time. And of course, I ran a backup on ALL partitions before I installed the new released of MS-DOS.

Installing ZDS MS-DOS 4.0

I had decided to delay changing the partition size on my 80 MB Seagate ST-4096 hard drive until I had a good reason to do so, and the installation of ZDS MSDOS 4.0 was when I decided to do it. Because I originally made the drive C partition so small (30 MB), I was continually juggling application programs in order to use them. My drive D was about 50 MB because I thought I would have more data than programs, but that has turned out not to be the case. Today's software takes so much disk space that I finally decided to reverse the partition size. I used the Imager card and software to back up both of my C and D partitions to VCR tape before I booted the system with the ZDS MS-DOS 4.0 distribution disk.

I booted the system with the ZDS MS-DOS 4.0 distribution disk and aborted the install because I wanted to change the partition size with the PART command before I ran the FORMAT command on both partitions. As I mentioned in the previous column, the new PART command is easy to use, and it is much better than the one in version 3.3 Plus. As usual, I had to delete both of the existing partitions in order to change the partition sizes because PART does not have a "change" feature. The lack of a change feature is really a technical discussion because IBM's FDISK command does not have it either. What it amounts to is a technical problem that does not allow "overlapping" partitions. That is, you cannot have one partition ending cylinder in the middle of another partition, and it is much easier to avoid a potential problem by requiring that an existing partition (or partitions) be deleted to change the size. By the way, that is a common technique which is used in programs that do not have a "change" feature. As another example, there is no way that DOS will allow you to rename a file from one drive to another. What you have to do is COPY a file from the old drive to the new drive (and directory), then use the DEL command to delete the file from the old drive.

Although I could have used the INSTALL program after I changed the partition sizes, I still do not like the fact that the program forces me to use the \BIN subdirectory, so I did everything manually. I ran the FORMAT C:/S and the FORMAT D: commands to reformat both partitions. Then I checked to make sure that partition C was bootable by using CTRL-ALT-INS to get to the ROM Monitor and entering the BW0 command to boot from the hard drive. Remember that this was the first time I had defined a bootable partition larger than 32 megabytes.

I was surprised to see the "Warning! SHARE should be loaded for large media" message when I booted the system. When in doubt, I read the manual, and on page 1.127 (SHARE command), it mentions that SHARE is automatically loaded if you have a hard drive partition larger than 32 MB and SHARE.EXE is located in the same directory as your COMMAND.COM file. Hmmm... In order to avoid problems, I simply entered the SHARE command on the DOS command line because that was suggested by the manual and the error message.

Then I created the \DOS subdirectory and copied all of the programs from the distribution disks into it. After that, I restored all of the other directories to both drives using the Imager tape I had created earlier. Now I had the system up and running, so it was time to do a little research on permanently fixing the error message about SHARE.

Reading the Manual

As most of you probably guessed from my last column, I spent considerable time looking through the manuals for ZDS MS-DOS 4.0 to see what was new and what was not. In fact, I always spend some time looking through new software manuals to help me get acquainted with the general contents, organization, and hints on how things work. I generally have three things on hand when I do that: a Highlighter to mark important information so I can find it later, paper clips to use as "flags" for important details on various pages, and the smallest size of Post-it Note Pads to use as "tabs" for the various sections.

The trick of using the Post-it Note Pads as tabs is especially important in a large manual (e.g., Word Perfect 5.1) or a "combination" manual like the ZDS User's Guide and Command Reference. I seldom refer to the User's Guide, so I have Post-it Notes showing the COMMAND REF and the INDEX for the Command Reference. To make things easier to find, I have also used the Post-it Notes to indicate where the APPLY, DSKSCAN, and NLSFUNC commands begin. I have even been known to use red ink to indicate section beginnings and blue ink to indicate some kind of division within a section.

Using paper clips to flag important information that I might want later is another timesaver I use. As I went through the manuals, I just added paper clips to the pages containing "new" items so I could find them when I wrote last month's column. As I mentioned last time, I did not even try to discuss everything new that I found, but I did talk about all of the new things that I thought would be important to most HUG members. For example, I did not mention the four new DEBUG subcommands (XA, XD, XM, and XS) that allow you to perform various functions involving expanded memory, but those pages have paper-clip flags because they are new. Perhaps you will find that these simple and inexpensive tricks will work for you and help you save some time when you are looking for something that you KNOW you have read in a manual. In any case, back to the ZDS MS-DOS manual.

I also noticed (and flagged) the new REM and INSTALL= commands that can be used in the CONFIG.SYS file. As I noted in my last column, the INSTALL= command can only be used with specific EXE support files: FASTOPEN, KEYB, NLSFUNC, and SHARE. The fact that INSTALL= could be used with the SHARE command caught my eye, but I did not make use of it until later.

At this point, the real question I had was how to "automatically" enter the SHARE command so it would do exactly what it should do (i.e., avoid that tacky error message) and use the least amount of memory. In order to figure this out, I had to do some experimenting.

Experimenting with the System

Many people are surprised that I experiment with my production system as much as I do, but I take a lot of precautions before I do it. Perhaps the most important thing is that I take a complete hard drive backup (both partitions) before I begin. I never know what might happen, and good backups have saved a lot of work more than once. By the way, I take backups before I install either new hardware or new software, just in case.

Now I could experiment with various configurations to determine which one was the best. And I used the new MEM command to help me decide which approach was the best for my system, and the results surprised me.

Since I inevitably receive letters about "this does not work in my system", I should define the basic setup of my system. I am using a Z-386/16 with ROM version 2.6E, and ZDS MS-DOS version 4.0. The VER command says that I have MS-DOS version 4.01 and BIOS version 4.00.02. Although I believe the general results apply to all current DOS versions (including PC-DOS 4.0), I have not tested every other possibility, so be advised.

During my testing, I discovered an error in the ZDS MS-DOS 4.0 Command Reference manual on page 1.27 (COMMAND). For the COMMAND command, the manual states that the /E (Environment) switch has a default value of 128 bytes, which is wrong. The default value is actually 160 bytes. If you enter a command like COMMAND/E:159, you will get a "Parameter value not in range" error message. You can also prove that the default value is 160 bytes by using the MEM command that shows that the COMMAND Environment is A0 hex bytes long which translates to 160 decimal bytes, assuming you have not used the /E switch to change the size of the environment space of course. Apparently Microsoft does not know the default size of the environment space in this version either...

I discovered a second error on page B.17 (SHELL) in the ZDS MS-DOS 4.0 User's Reference. During my testing, I deleted COMMAND.COM from the root directory on drive C and added the following command line to my CONFIG.SYS file: SHELL=C:\DOS\COMMAND.COM/E:256/P. After rebooting the system, I ran the MEM command, and after completing, the "Cannot load COMMAND, system halted" error message appeared. That caused a hard freeze, and I had to use the hardware reset switch to reboot the system. The other alternative was to power off, then power on, but I added a hardware reset switch because I do not like to do that. In any case, I rebooted the system, and used the SET command to check the COMSPEC parameter. What I saw was COMSPEC=C:\COMMAND.COM. I found that particularly puzzling because I had deleted COMMAND.COM from that drive and directory as stated above.

Page B.17 clearly states that "MS-DOS sets the COMSPEC environment variable equal to the d:, path, and the filename specified on the SHELL command line. This setting overrides the default value for the COMSPEC (the drive and path name of the command processor initially used to start MS-DOS)." After considering that something might be unique to the name COMMAND.COM, I renamed that file (in the \DOS subdirectory), and changed the command line in my CONFIG.SYS file to: SHELL=C:\DOS\MYSHELL.COM/E:256/P. I rebooted the system and tried the MEM command again. Upon completing execution, I got another "system halted" message which required a hardware reset. The SET command STILL had the COMSPEC specification wrong: COMSPEC=C:\COMMAND.COM instead of C:\DOS\MYSHELL.COM as specified in the SHELL= command.

Aside from the error in the manual, I spent considerable time fooling around with the SHELL= command. Interestingly enough, I also discovered that I did not always get the "system halted" error message. Regardless of how I rebooted the system with either a "soft boot" (CTRL-ALT-DEL) or a "hard boot" (with the hardware reset switch), I would get a system halt after running a program like MEM or CHKDSK about every fifth time. When I get those kind of inconsistent results, I do not trust the whole idea and I look for another approach. Apparently, Microsoft does not want anyone to use a Command Interpreter other than COMMAND.COM.

One interesting point about the MEM command: it always reported the "correct" file name (either COMMAND or MYSHELL) depending on what the SHELL= command line specified.

By now you are probably wondering why I got into all of this in the first place. Well, it seemed to me that if I followed the general suggestions in the manual, I could eliminate one copy of COMMAND.COM (in the root directory) on my hard drive. Remember that my original objective was to eliminate the "Warning! SHARE should be loaded for large media" message that was displayed when the system was booted. As I mentioned earlier, page 1.127 in the manual says that SHARE automatically loaded when a hard drive partition is larger than 32 MB and SHARE.EXE is located in the same directory as the COMMAND.COM file. That is the reason I used the SHELL=C:\DOS\COMMAND.COM/E:256/P in my CONFIG.SYS in the first place. And since the manual clearly stated that the SHELL= command overrides the default COMSPEC value, I did not expect any problem when I deleted COMMAND.COM from the root directory...at least not until I discovered that the COMSPEC value was NOT changed by the SHELL= command.

One thing about the SHELL= command that did work was the fact that SHARE was definitely loaded when I specified the \DOS subdirectory for COMMAND.COM (or MYSHELL.COM). I verified that by using the MEM command, and the name of the Command Interpreter did not matter. I also noticed that SHARE was always loaded BELOW the Command Interpreter when I used the SHELL= command for the Command Interpreter. In my system, SHARE was loaded at 127B0 hex and the Command Interpreter (either COMMAND.COM or MYSHELL.COM) was loaded at

14060 hex. I also copied SHARE.EXE back to the root directory and deleted the SHELL= command from CONFIG.SYS, just to see what happened when I rebooted. I observed that the default in both cases was to load SHARE first (then COMMAND.COM), so I decided to keep that "standard" to hopefully avoid any problems in the future.

Since I now knew what did not work, it was time to find something that would. And I like to follow the KISS principle: Keep It Simple, Stupid. I also remember to SMILE: Simplicity Makes It Lots Easier.

If you noticed that I used the /E (Environment) switch with the SHELL= command, you probably recognized that I need an environment space larger than the default 160 bytes. In fact, I have used a 256-byte environment space for several years because I occasionally need that extra space for various SET commands that I use with some C compilers. Even though that takes a little bit more memory, the "cost" is so small that I do it on a permanent basis, and the SHELL=C:\COMMAND.COM/E:256/P command has been in my CONFIG.SYS file for some time. In other words, I still had the requirement for the larger environment space that I wanted to include in the CONFIG.SYS file.

The simple way to "activate" the SHARE command was to simply use the new 4.0 INSTALL= command which is "INSTALL=C:\DOS\SHARE.EXE" for my hard drive setup. If you recall last month's column, I mentioned that SHARE.EXE is one of the files that works with the new INSTALL= command. Since you may be wondering what my CONFIG.SYS file looks like at this point, it is shown as Figure 1.

```
INSTALL=C:\DOS\SHARE.EXE
SHELL=C:\COMMAND.COM/E:256/P
BUFFERS=35
FILES=40
BREAK=ON
STACKS=16,128
```

Figure 1
CONFIG.SYS File Contents

Note that I used the INSTALL= command for SHARE to load it first because that was the observed standard as I noted earlier in this article. The SHELL= command is used to expand the environment space with the /E switch, and /P keeps that turkey working right, regardless of what the manual says. If you want to see something interesting, I would encourage you to try the SHELL= command shown in Figure 1 WITHOUT the /P switch. The values shown for the BUFFERS= and FILES= commands are ones that I have determined (by experiment) work best for my system and applications. The BREAK= command simply provides better access to the CTRL-BREAK (or CTRL-C) feature of DOS. And I use a STACKS= command because I have had an occasional problem with an "Internal stack failure, system halted" error message in previous DOS versions. The value specified is exactly twice the default of 8,128; and I have not had any recurring problems with that since I changed it.

Many people still ask me how I figure all of this out. Well, in this case I spent a couple of days fooling around with all of this stuff, and that is typical for the information that I include in this column. I would encourage you to experiment with your system because that is the way you can learn something about it. If you do decide to experiment, then you should follow two important safety rules. First, be sure you have a good backup of all disks that you use for experimenting, especially all partitions on a hard disk. I also recommend that you make special backups of all partitions on a hard disk whenever you change ANY hardware, or add or upgrade any software, including DOS and whatever application programs you use. For floppy disk systems, you can use the DISKCOPY command to make backups. For hard drive systems, you can use the DOS BACKUP command or any special utility backup program of your choosing.

The second safety rule applies only to systems with hard drives because the first rule already covered it: be sure that you have a working copy of the bootable FLOPPY disk that contains your current DOS version. That disk should at least contain EDLIN or some other editor that can edit ASCII files like CONFIG.SYS and AUTOEXEC.BAT. When you are experimenting with your system, especially with CONFIG.SYS, it is not difficult to make a mistake such that the hard drive simply will not boot. I manage to work myself into that particular corner every month or so for one reason or another. As I was testing various things for this article, I had to use my "special" bootable floppy twice because of various problems with commands I had entered in the CONFIG.SYS file. Both cases involved simple typographical errors in a CONFIG.SYS command line, but that was enough to keep my hard drive from booting the system. I had to use the floppy to boot the system, and I edited the CONFIG.SYS file to fix the problem using EDLIN from the floppy disk. It really does not matter how much you know about computers or your software because you can easily make the same kind of mistakes that I did.

During the time I was experimenting with these configurations, a friend asked me some questions about turning a computer on and off.

Powering Up and Down

There has always been a lot of controversy as whether one should leave a microcomputer running all the time or power it off. And although I do not expect my comments to end the controversy, I have some observations that began in 1967 when I was working in the data processing organization at Purdue University while I was studying for my degree in Electrical Engineering. At that time, I was working with an IBM 7090/7094 and a CDC 6500. Today, I have a broad background in computer technology ranging from microcomputers to minicomputers (e.g., DEC VAX and others) to mainframes (e.g., IBM 3090 and Cray II). In order to understand where perhaps part of the controversy began, I think it is important to mention some of the special requirements for mainframes.

Mainframes have all kinds of special requirements, ranging from special power to chilled water and air conditioning. An IBM 3090, for example, has a requirement for 400 Hz power that is typically supplied by special generators. Big mainframes like this also require chilled water to cool the inside of the computer. Of course that means special plumbing, not to mention the need for cooling towers that are typically located on the top of a building. Special kinds of air conditioning are required because both the air temperature and the humidity must be regulated. Humidity regulation is quite important because condensation may be a problem when cool air strikes a warm electronic or mechanical component. Special kinds of air conditioners and air handlers circulate cool air both above and below the raised floor in a computer room.

The most important point about mainframes is that powering them up is not a trivial task like it is with a microcomputer. For a mainframe, all of the support utilities (i.e., power, air conditioning, and chilled water) must be operational BEFORE the mainframe is actually powered up. It is not a simple matter of flipping one or two switches. The generators must be started to supply 400 Hz power, the air conditioning and air handlers must be powered up to keep the air cool, and the chilled water pumps must be started to circulate the water. And I have not even mentioned a lot of other important things that must be done, such as powering up the hard drives and controllers for the system. When a computer room is completely powered down, it is a very time consuming process to get everything up and running correctly. We generally call this process a COLD START, although I have heard some people refer to it as a FROZEN START because it is so difficult to do. As a result, most data processing managers of large mainframe systems do not like to power down the computer room for any reason. Because even a very short power interruption (even a few seconds) can cause all kinds of problems, many mainframe systems have an Uninterruptible Power Supply (UPS) that includes special batteries and can cost more than a million dollars. A very short power blip can cause a mainframe system to have problems for days or weeks, until all of the problems are fixed. Problems I have personally seen include head crashes on hard drives and replacement of printed circuit boards whose components were ruined by a power surge when the power was applied. That's why you will usually see computer room personnel running around flipping switches (to off) if power is lost in the computer room.

I hope this explanation provides some insight as to why traditional wisdom says that mainframe computer systems should never be powered off. With that as background, let's take a look at the general issue.

To get away from some of the emotional reactions to the computer controversy, I will begin by using an automobile as an example. Almost any mechanic you talk to will agree that short, frequent trips will shorten the life of various automobile components considerably. When you start your car first thing in the morning, that is probably the very worst time. Oil is cold and it is not circulating, so the mechanical friction (and wear) is at its highest because the oil has all drained into the pan. Really cold mornings are murder on engines, which may explain some of the popularity of some engine treatments that contain teflon and claim to reduce wear, especially on start-up. Short trips are hard on the engine because the oil does not get hot enough to "boil away" some of the corrosive chemicals that always get into it, not to mention the fact that the engine may never reach its designed operating temperature for maximum efficiency and fuel economy. And I expect that everyone knows that frequent starts and stops are hard on a battery because it may never be charged long enough to recover from each start, which drains a considerable amount of power. Frequent power cycling (on and off) is bad for a car, and I doubt that any knowledgeable person would disagree with that.

To move more into the realm of electronics, what about televisions? Does your TV have an "instant-on" feature? Even though my 17-year-old Zenith TV is nearly all solid state, the picture tube is always on, and it takes only about five seconds to display a picture when the power switch is turned on. Regardless of what the original design intentions were, it turns out that the "instant-on" feature has apparently helped me keep using the original picture tube for 17 years, although it will soon be time to replace the set for other reasons. I believe that the fact that the electronic components, especially the picture tube, probably have had a much longer life because of that feature.

What about Microcomputers?

What about microcomputer components? Before I begin, I would like to qualify this by stating that I am only talking about individual microcomputers, not something like a file server used on a Local Area Network (LAN). LANs can be tricky and difficult to start, and I think a good case can be made for leaving it running all the time, especially if it has a UPS to cope with power interruptions. That is NOT a recommendation, because there are some potential hazards that apply to any microcomputer, file server or not, as you will see. If you are responsible for a LAN, I'll leave it to you to decide what is the best for your particular system. In other words, there is no "right" answer that applies to every system.

Let's keep things relatively simple by breaking down a basic microcomputer system into its three components: power supply, printed circuit boards (including the motherboard), and disk drives. Peripherals, such as printers and external modems, also have a power supply and printed circuit boards too, and they may have special components to perform their function, such as a CRT or a print head. Most of these comments apply to those peripherals too.

The power supply. Like a car, it usually has the most problems during powering up. That's when a voltage surge is most likely to cause a problem. In other words, the voltage on internal components will jump from zero volts (power off) to as much as 120 volts or so, even though the final output will be on the order of five to twelve volts of direct current (DC). That's kind of like starting a car in the morning, popping the clutch, and expecting to be at 55 miles per hour in MUCH less than a second. A power supply will frequently fail during power up for much the same reason that a light bulb will usually burn out when you flip the switch. Perhaps you have seen a light bulb go "nova" (get extremely bright) when you flip the switch, and a component in a power supply will usually fail in the same way, even though you don't see it. On the other hand, a power supply is designed to operate reliably in the heat generated during extended use.

Printed circuit boards. Assuming that a power supply is correctly designed, the voltage surge that occurs during powering up should never reach the printed circuit boards at a level high enough to cause a problem. Changing heat levels seem to cause the most problems, especially with socketed RAM and ROM chips and even SIMMs. The metal pins on these components will expand slightly when they are heated and contract when the system is powered down. Repeated heating (expansion) and cooling (contraction) can cause these chips to work their way out of the sockets. This problem is generally called CHIP CREEP because of the way these chips can actually move in their sockets, which can cause wide variety of problems, depending on which chip becomes loose first. Many of you may recall that I have mentioned in other articles that a CHIP MASSAGE is one of the first troubleshooting things you should do when your system seems to develop a sudden hardware problem and no new hardware has recently been installed. That is, carefully remove each of the printed circuit boards, place them on a firm surface, and press each socketed chip back into its socket. You will probably be surprised at how much chip movement you will find when you try this. Repeated heating and cooling can also affect how well a connector makes contact, so it is a good idea to check them as long as you have disassembled the system anyway. By the way, a chip or connector massage can also help fix a problem with corrosion on the metal pins.

Now for disk drives. Floppy drives are not much of a concern because they are cheap and easy to replace. They also do not have much vulnerability to voltage or heat problems, at least not to the extent that hard drives do. And floppy drives don't have the vulnerability of tremendous amounts of data that hard drives have, especially for users who do not back up that data on a regular schedule. Moreover, hard drives have special considerations because of heat and potential mechanical problems.

The most frequent mechanical hard drive problem I have seen is simply that the bearings wear out. That is usually an easy problem to diagnose because the drive will squeal or screech when the computer is powered up. If you completely disconnect the hard drive from the power supply and the controller, the fact that the computer is now "quiet" (after powering it up of course) is the dead giveaway that you have found the problem. The bearings are probably shot. And although it is true you can find hard drives that have survived for a long time with bearing noises, I personally would not trust one with my data, even though I take regular backups. I mentioned the bearing problem only because it is generally associated with the length of time that a hard drive is powered on, not the number of on-and-off cycles.

When a hard drive is powered up, the most significant "surge" occurs in the drive motor because it starts at zero RPM and has to get the platters spinning at 3,600 RPM for most hard drives. From a pure mechanical perspective, the drive motor probably is the most vulnerable during power-up, but I have only seen one hard drive motor failure in the last eight years.

Repeated heating and cooling cycles can affect a hard drive's reliability because of the expansion and contraction of mechanical components, such as the platters and heads. This is the primary reason that I like the ZDS PREP program that performs low-level hard drive formatting. PREP performs a multi-pass test (4 in the latest version) that helps ensure the hard drive is up to operating temperature. Even so, I like to run PREP twice on a brand new hard drive: the first time is to generally check it out, and the second time is "for real." When I installed an 80 MB Seagate ST-4096 MFM hard drive in my Z-386/16, it only took PREP about 40 minutes to complete. Because I know that PREP really exercises a hard drive (watch the drive light!), there was little doubt in my mind that the unit was up to "normal" operating temperature after the first run. To make sure that the low-level format was as good as it could possibly be, I reran PREP again to ensure that the low-level format was written at that drive's normal operating temperature. While that may be overkill, I have never had any problem with that drive. Long-time HUG members may recall that I have recommended running the old ZDS DETECT program (which checks for bad sectors) at the end of a day or session to allow plenty of time for the hard drive to reach operating temperature. The same thing is true for the DSKSCAN program in ZDS MS-DOS 4.0, which is one of the new programs that I mentioned last month. You may occasionally have problems with a hard drive when you turn the system on, especially if the low-level format and the high-level FORMAT were done when the hard drive was at operating temperature. Perhaps this kind of problem helps explain the popularity of Steve Gibson's SpinRite program that "renews" a hard drive's low-level format.

The electronics on a hard drive are generally not much of a problem, regardless of which kind you have: MFM, RLL, ESDI, SCSI or IDE. Most of a hard drive's electronic components are not socketed, so chip creep is not a very common problem.

Now that you know some of the considerations, what is the real answer? Should you leave your computer running all the time? Should you power it off, and if so, how often?

I checked with several people at both Heath and Zenith Data Systems, and I really could not get an "official" answer to the question. In most cases, each individual had two answers. Microcomputers at work were generally left on all the time, although the monitors were usually powered off to prevent screen burn. All of the people I checked with also had micros at home (no surprise), and every one of them turned the system off at the end of a session, which amounted to once a day. Please keep in mind that this is NOT an official statement from any of these people; rather, it is what they actually do. In general, I have a suspicion that they power off their home systems for the same reason I do: to save electricity, even though it does not cost much to run a personal computer on a daily basis.

What I Do

I try to schedule my work so that I only power up my system once a day, and I do not leave it running unattended for a long time. My personal opinion is that it is best to minimize the power cycling, but I am not a fanatic about it. Sometimes I have to power up the system again, especially if someone calls and I have to check something, but that only happens once or twice a month at most.

For my purposes, I think it is better to power off at the end of a day if for no other reason than to conserve electricity. I have done that for a number of years, and I have had some of the problems I mentioned earlier.

I have had to replace two power supplies: one in my Z-100 and one in my Z-248. In both cases, I did not consider that abnormal because both systems had a LOT of use (i.e., power-on time) before the power supplies died. And in both cases, the power supply failed when I powered up the system one morning. Lest you think that is too out of step, I suggest you ask someone about the really puny power supply in the original IBM PC. Those computers regularly ate those 65-watt power supplies, and you can still find lots of ads for replacements, even though you will rarely find anyone who even sells a 65-watt unit today.

All of the hard drive failures I've had have occurred on powering up, but I did not consider that to be unusual either. The last failure was on a 40 MB CDC drive that had lots of constant and extended use almost every day for about two and a half years. That's the one I replaced with my current 80 MB Seagate drive about a year and a half ago (January 1989). Power supply and hard drive failure discussions also need to include one other point to be complete.

Virtually all of today's equipment has some kind of MTBF (Mean Time Between Failure) rating. For example, I can remember when a good hard drive had a 5,000 or 10,000 hour MTBF. The pamphlet that came with my Seagate drive says that the ST-4096 has a service life of five years and an MTBF of 25,000 (power-on) hours. Some of the RLL drives in the pamphlet are listed as having an MTBF of 35,000 hours. In terms of this general discussion, there is no mention anywhere in the Seagate information I have on the effects of power cycling or how it will affect the MTBF or service life. And regardless of what the documentation says, it is important to remember that the MTBF is really an estimate of equipment life, not a guarantee.

A friend of mine told me that he had recently read a suggestion that a hard drive should be "warmed up" to operating temperature (for about 15 minutes or so) before it is used. That presumably helps avoid "losing" the low-level format information because of the heat expansion situation I mentioned earlier. My only comment about that idea is that I have never "warmed up" any of my hard drives before use, and I have never experienced any problem with them because of it. I do agree that is it better to run diagnostic programs (e.g., Mace Utilities or DSKSCAN) after a hard drive is warmed up, but that's only because a media glitch is more likely to show up when a drive is at operating temperature (hot) rather than when it's cold. My friend tells me that the idea of warming up a hard drive was discussed using the analogy of warming up a car's engine for a few minutes before use, but my mechanic does not recommend that for my car. He says it is better to start the engine and begin driving it at low speeds to warm it up rather than letting the engine "carbon-up" at idle. Driving at low speeds places a load on the engine, which warms it up faster, brings it to operating temperature sooner, and thus increases engine efficiency and improves overall fuel economy. Regardless of the analogy, I am not personally convinced that warming up a hard drive before general use is of any great importance one way or the other.

At this point, I hope you recognize that I have NOT suggested or recommended which is the best approach. I have not said that it is better to power off a system once a day. I have said that I do power-off my own systems every day, and I have not observed any particular problems as a direct result.

I have kind of neglected a discussion about the potential hazards of leaving a system powered on continuously, so let's see what kind of bad news can be found for that side.

Continuous Running

While leaving a system powered on continuously might sound like a way to avoid a few of the problems I mentioned, there are at least two potential problems. I have personally experienced the first one: the area I live in does not have very reliable power. I have had as many as six power blips in one afternoon. A "power blip" is a short power interruption that is less than five seconds, but is sufficiently long to knock out my fluorescent desk lamp. If you leave your system powered on continuously and live or work in an area that has brownouts or blackouts, then you might have more frequent power cycling than you think. Fortunately, that problem is relatively easy to solve, but the solution is not cheap. Even though I do not leave my system running continuously, I bought an Uninterruptible Power Supply (UPS) for my production system, primarily so I would not lose any work when these power blips occur. As a side benefit, my

UPS also provides surge protection, as well as EMI and RFI filtering.

The second problem is large voltage surges, such as lightning strikes that can occur during violent thunderstorms. We get some rather violent thunderstorms here in Texas, and I power off my system during that time, despite the fact that my UPS has a built-in surge protector. Even a surge protector may not help much if a lightning strike hits very close, but why chance it? Large voltage surges can also occur after brownouts or blackouts when power is fully restored. If you leave a system on continuously, the real point is that the system is ALWAYS vulnerable to large voltage surges. More importantly, some surge protectors may not work as well as you think, which means that you should get a good one if you leave your system on all the time.

The problem of large voltage surges brings one other thing up. I will not have an internal modem card in my desktop computers for a lot of reasons. The main one is that I have a friend who had a voltage surge on his TELEPHONE LINE that completely fried his external modem, even though it was powered off. We know that it came in on the telephone line because his computer and monitor were plugged into the same wall socket (with a power strip), and neither of them was affected. We had a violent thunderstorm that day, and we think that a nearby lightning strike was the culprit. As to what would have happened if he had had an internal modem, well, your guess is as good as mine, but I really idon't want to think about it.

As I mentioned at the beginning of this discussion, there is a lot of controversy about the subject. You will probably find that one microcomputer manufacturer recommends a daily power-off and another recommends leaving the system on continuously. And whichever side you choose, you will be able to find PLENTY of stories that support that side. Unfortunately, those stories do not really represent a serious study of the situation, which leaves the entire decision up to you. The only recommendation I can really make is to review the facts and some of my experiences mentioned here, and see what's best for you.

Powering Down

For next month, I will continue my custom from the last several years and suggest gift ideas that are useful for any computer user. Be sure and watch for Santa Bill's column in the next issue.

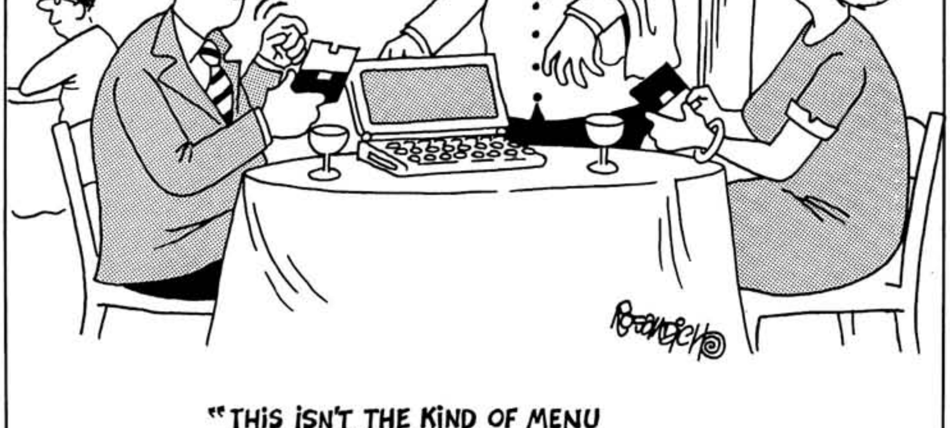
For help in solving specific computer problems, be sure to include the exact model number of your system (from the back of the unit or the model series from the Owner's Manual), the ROM version you are using (use CTRL-ALT-INS to find it), the DOS version you are using (including both version and BIOS numbers from the VER command), and a list of ALL hardware add-ons (including brand and model number) installed in your computer. The list of hardware add-ons should specifically include memory capacity (either added to an existing board or on any add-on boards), all other internal add-on boards (e.g., modems, bus mouse or video cards), the brand and model of the CRT monitor you have, and the brand and model of the printer with the type of interface (i.e., serial or 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

HUG Software
Powering Up (885-4604) $12.00
Heath/Zenith Users' Group
P.O. Box 217
Benton Harbor, MI 49022-0217
(616) 982-3463 (HUG Software only)

Software
MS-DOS Version 4.0 $149.00
(List price)
(Mail order with update card only) 49.00
3.5-inch version (OS-103-MS)
5.25-inch version (OS-105-MS)
Heath/Zenith Computer Centers
Heath Company Parts Department
Hilltop Road
St. Joseph, MI 49085
(800) 253-0570 (Heath Catalog orders only) *



"THIS ISN'T THE KIND OF MENU WE HAD IN MIND."

A Graphics Viewer for the Z-100

Gregory D. Elder
Qtrs 4301-A
USAFA, CO 80840

One of the best features of the Z-100 is its graphics capability. It supports 8 colors, has a resolution of 640 x 225 pixels (640 x 448 pixels when in interlaced mode), and is always in graphics mode (you do not have to switch between text mode and graphics mode, such as with IBM-PC systems). Ever since Heath/Zenith discontinued support of the Z-100 (how unfortunate for us Z-100 owners), new software for the Z-100 has been few and far between. Whenever I see a nice piece of software for the IBM-PC, I either have to do without or hope that someone will put out a version for the Z-100 (which 99 percent of the time does not happen). In some cases, I will try to write a Z-100 version of a program myself.

There are many graphics viewers available for IBM-PC's. These programs simply read a file stored in a particular graphics image format and display the image on the computer screen. One popular graphics format is GIF(tm) which stands for "Graphics Interchange Format." (GIF was developed by CompuServe, Incorporated). Having seen several GIF viewers for the IBM-PC but none for the Z-100, I decided to write my own Z-100 GIF program. My task was simplified by the fact that some of the IBM-PC programs have been placed in the public domain, complete with source code. Writing a program for the Z-100 then became a matter of modifying an existing program for the IBM-PC to work on the Z-100.

This article will explain the GIF standard and the resulting program I developed to display GIF files on the Z-100. Information about the GIF standard was taken from the document *Graphics Interchange Format: A Standard Defining a Mechanism for the Storage and Transmission of Raster-Based Graphics Information,* dated June 15, 1987 by CompuServe, Incorporated. My GIF program, GIFZ100, is based upon two public domain programs for the IBM-PC—GIFSLOW and GIFEGA both written by Mr. Jim Griebel.

The GIF Standard

As previously mentioned, GIF was developed by CompuServe as a standard for storing color graphic images. You can view GIF files on any computer system as long as you have a program which can interpret the GIF standard. GIF viewers exist for IBM-PC's, Amiga's, Atari ST's, and other computer systems. Therefore, by using the GIF standard, you can exchange color graphic pictures among a variety of dissimilar machines.

Like any computer-based file, a GIF file consists of a series of bytes arranged in a specific order. A program to display GIF images needs to extract the various pieces of information contained within the GIF files. Figure 1 illustrates the general format of a typical GIF file.

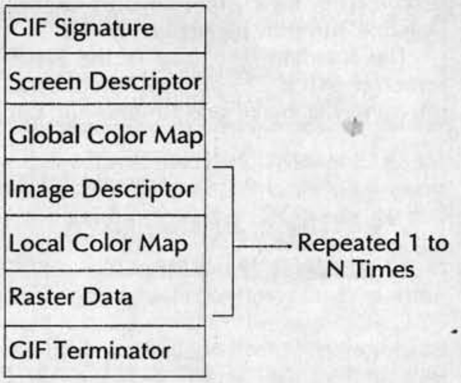


Figure 1. GIF File Format.

As shown in Figure 1, the first piece of information in a GIF file is the GIF Signature. This consists of 6 bytes which identify the file as containing a GIF image. The GIF Signature will simply be the characters 'GIF87a'. The '87a' indicates that the file conforms to the GIF standard as defined in 1987. If the standard GIF format is revised, then that part of the signature will change.

The second part of a GIF file is the Screen Descriptor which consists of 7 bytes of data. The Screen Descriptor provides some basic parameters for the image contained in the file. The first two bytes of the Screen Descriptor give the screen width of the image in pixels. The next two bytes give the screen height of the image. Keep in mind that depending on where an image was created, its size may be larger than the physical screen of the computer on which you intend to display the image. GIF viewers must therefore be designed with this in mind. You can either scroll large images or truncate them when displayed. (The Z-100 GIF viewer can use interlace mode for large images or squash images to fit on the screen.) Byte 5 in the Screen Descriptor indicates if a global color map follows and the maximum number of colors used in the image. The number of colors used will range from 2 (for black and white images) to a maximum of 256 colors. Byte 6 in the descriptor provides the background color of the image. Finally, byte 7 in the descriptor is not used.

Following the Screen Descriptor block may be a Global Color Map (this is optional). If the GIF file contains more than one image, then there may be one color map for each individual image. Each entry in the color map consists of three byte values. These byte values represent an intensity for one of each of the colors red, green, and blue. The intensity values can range from a low of 0 (no color) to a high of 255 (maximum color intensity). For example, the color black is the absence of all color so the values for red, green, and blue would be (0, 0, 0). On the other hand, since white is the presence of all colors, the values for red, green, and blue would be (255, 255, 255). All other colors are combinations of varying intensities of red, green, and blue. Therefore, bright yellow would be represented as (255, 255, 0). The structure of the Color Map is shown in Figure 2.

	Byte #
Red Intensity	1 Red Value for color 0
Green Intensity	2 Green Value for color 0
Blue Intensity	3 Blue Value for color 0
Red Intensity	4 Red Value for color 1
Green Intensity	5 Green Value for color 1
Blue Intensity	6 Blue Value for color 1
.	
.	n Blue Value for color n*

* - n represents the maximum number of colors in the color map

Figure 2. Color Map Structure.

The next portion of the GIF file is the Image Descriptor. The first byte in the Image Descriptor is a comma. Commas are used as image separators. If a GIF file has more than one image in it, each image will be separated by a comma. The next 8 bytes in the Image Descriptor provide information concerning where on the screen the image should start (origin of the top, left corner for the image), and information about the size of the image (width and height in pixels). The last byte of the Image Descriptor indicates whether a local color map exists for the image and whether the image is formatted in sequential order or interlaced order.

Local color maps in GIF files are optional. If one is present, it will have the same format as the Global Color Map. If a GIF file contained multiple images, it would be possible for each image to have its own local color map.

The next part of a GIF file is the raster data, that is, the data which actually defines the image. The data consists of a series of color index values into the global color map or local color map (if one exists). Each color index value represents one pixel of the image. These pixel color index values are stored sequentially left to right for a row of the image. Finally, the raster data is stored in a compressed format to reduce the size of the GIF file. With the raster data and color maps, a program simply has to put pixels on the computer screen using the appropriate colors.

The final part in a GIF image file will be the GIF Terminator. The Terminator is represented by a semicolon. This simply informs the program reading the GIF file that the end of the file has been reached and no images remain to be processed.

Using GIFZ100

The previous few paragraphs may seem a bit too technical for some people but I really simplified in many cases the description of GIF files. For a thorough description, you must read the GIF document provided by CompuServe which I mentioned at the beginning of the article.

With the GIF image file format known, a program to display GIF images simply reads through GIF files extracting information about image size, color, placement, etc. Then, the individual pixels of the image are plotted on the computer screen. This, in essence, is what the GIFZ100 program does. I will not go into detail explaining the inner workings of the program. (The source code is available if you want to see how the actual software.) Instead, I will explain the running of the program from a user's perspective.

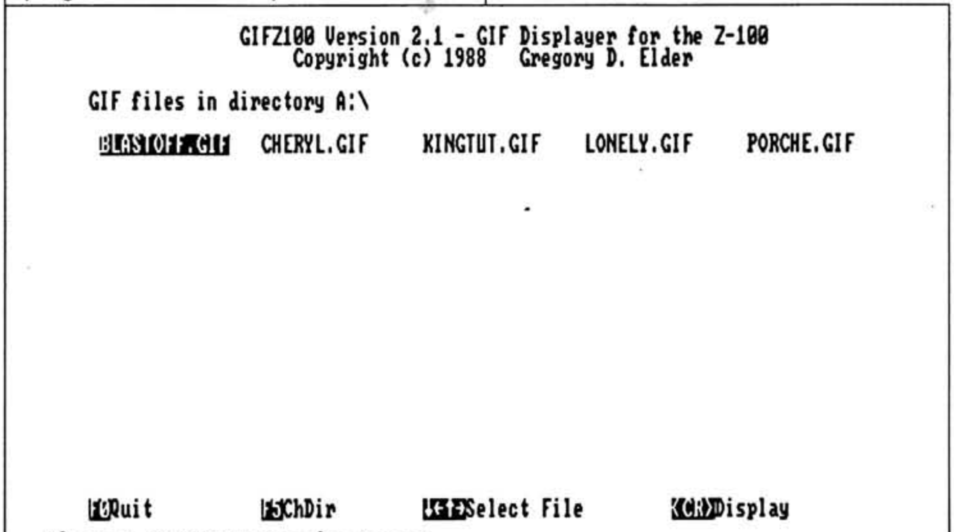
In order to run GIFZ100, you must have a Z-100 with MS-DOS version 2.1 or higher. You will also need a color monitor (I use a ZVM-135 monitor.) After booting your Z-100, place the disk containing GIFZ100 into one of your disk drives (if using a floppy disk system). If you are using a hard disk, change to the directory which has GIFZ100.

To begin the program, just type GIFZ100 at the DOS prompt, followed by pressing the RETURN key. At this point, your screen will appear as shown in Figure 3. As you can see by the figure, GIFZ100 displays all of the GIF files available in the current directory. At the bottom of the screen is a menu of GIFZ100 functions. The F0 function key is used to quit the program. The F5 key allows you to change directories. This is useful if you have more than one directory which contains GIF files. The arrow keys are used to move between the GIF files displayed at the top of the screen. As you move between files with the arrow keys, the current GIF file will be highlighted in red. Finally, pressing the RETURN key will display the image contained in the current GIF file, i.e., the filename currently highlighted in red.

The standard resolution of the Z-100 screen is 640 x 225 pixels. The GIFZ100 program will make adjustments for GIF images which are larger or smaller than this size. For example, if the GIF image has a width of 320 pixels, GIFZ100 will double every horizontal pixel so that the width is 640 when shown on the Z-100. The GIF image shown in Photo 1 has a width of 320 pixels but has been doubled to 640 for display on the Z-100. On the other hand, if an image is taller than the Z-100's standard screen height of 225 pixels, GIFZ100 will provide the user with some options. If the Z-100 has enough video RAM for interlace mode, which provides a screen height of 448 pixels, then the user can select to use interlace mode. The user also has the option of having GIFZ100 squeeze and squash the image. This means that if the image is too tall or wide, GIFZ100 will compress the image to make all of it fit onto the screen. The final option for images that are too tall or wide is to truncate the image when displayed. For example, if an image were 720 pixels wide, only the first 640 pixels would be shown on the screen, leaving 80 pixels which the user cannot view. (An enhancement for GIFZ100 would be to provide for scrolling of large images.) Photo 2 is another example of a GIF image displayed on a Z-100 with the GIFZ100 program.

As mentioned earlier in the article, GIF images can have up to a maximum of 256 colors. The Z-100, however, can only show 8 different colors. So how does GIFZ100 handle images which have more than 8 colors in them? Remember, the color map found in GIF files indicates the intensities of red, green, and blue used for each color in the image. The intensities can range from 0 to 255. When GIFZ100 processes the color map, it turns on the appropriate color (red, green, or blue) only when the intensity is greater than 127. In this manner, it maps all colors to the 8 provided by the Z-100. Images which have more than 8 colors in them won't look perfect on the Z-100, but it still gives you a good representation of the picture.

There are numerous GIF files available



```
              GIFZ100 Version 2.1 - GIF Displayer for the Z-100
                    Copyright (c) 1988   Gregory D. Elder

  GIF files in directory A:\

   BLASTOFF.GIF   CHERYL.GIF    KINGTUT.GIF   LONELY.GIF    PORCHE.GIF















   F0Quit          F5ChDir         ↓←↑→Select File       <CR>Display
```

Figure 3. GIFZ100 Opening Screen.

on various computer bulletin board systems, CompuServe, and from computer user groups. In addition, GIF images exist for anyone's taste in graphics. Pictures range from such things as the Space Shuttle blasting into space, to Cheryl Teigs, to racing cars, to King Tut's golden mask. With the availability of GIFZ100, us Z-100 owners no longer have to feel left out, or envious of our friends with IBM-PCs who have GIF viewers. Hopefully, this program will add a bit nore life to those old Z-100s.

GIFZ100 was written in Turbo Pascal. I made use of Keith Greer's great TurboZ package to handle the graphics aspect of the program. TurboZ provides a set of Turbo Units for the Z-100 which allows you to run Turbo Pascal Versions 4 and 5 programs on the Z-100 in native mode. I've placed GIFZ100 into the public domain. If you would like a copy of the program, including source code, send $5.00 to me. I'll also provide a diskette containing sample GIF images. You can mail to me at the address at the beginning of this article.

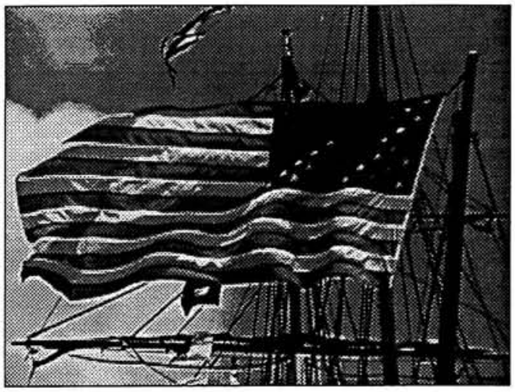
Products Mentioned

Turbo Pascal
Borland International, Inc.
4585 Scotts Valley Dr.
Scotts Valley, CA 95066

TurboZ
Keith Greer
1405 Bills Dr.
Beavercreek, OH 45385

*

"MUST YOU READ AT THE TABLE ?!"



Flags: A tutorial

Gil Hoellerich
2617 Country Way
Fayetteville, AR 72703

Efficient assembly programmers respect flags and know what they represent. So, with tongue-in-cheek, one could say that they are patriotic. However, the flags in assembly language are **not** cloth such as our **stars and stripes** and they **do not** represent our heritage as a country. Instead, these flags represent **conditions** generated by the commands which have been executed.

This article will pose and, hopefully, answer the five questions about flags most often asked by a **novice** asssembly-language programmer. Familiarity with registers and knowledge of some of the mnemonic commands is assumed. So this article is not intended as a first introduction to assembly programming.

Where are the flags? What are they?

The flags are in the **Flag Register** in the 8088 "chip", which is a 16-bit register just as the general registers (AX,BX,CX,DX). Unlike the general registers, the programmer doesn't look at the contents of the flag register as one byte (8 bits) or one word (16 bits); instead, each bit is considered individually as a potential flag. Nine are presently used and the remaining seven bits are undefined and unused.

The name of the nine flags in the Flag Register and their usual abbreviations are:

Overflow	OF	Interrupt	IF
Aux Carry	AF	Trap	TF
Parity	PF	Sign	SF
Carry	CF	Zero	ZF
Direction	DF		

Each flag in the Flags Register is assigned to a specific bit, as shown below (i.e., carry to bit 0 and overflow to bit 11):

```
15|    |    |    |    | O | D | I | T | S | Z |    | A |    | P |    | C |0
                        v   i   n   r   i   e        u        a        a
                        e   r   t   a   g   r        x        r        r
                        r   e   e   p   n   o                 i        r
                        f   c   r                    C        t        y
                        l   t   r                    a        y
                        o   i   u                    r
                        w   o   p                    r
                            n   t                    y
```

As is the case with all registers, each bit can have the value of either 0 or 1. In the flag register, however, when the bit has the value of **0** we say that it is **clear** and when it has the value of **1** we say that it is **set**.

To aid in understanding the answers to the next two questions, we will use the debug program on the .com file generated from Listing 1. (At this point, knowledge of the MASM, LINK, EXE2BIN and debug programs by the reader is assumed. A brief review is given below Listing 1 for proceeding from Listing 1 to a .com file for those readers requiring such guidance.)

I suggest that you follow the step-by-step action within debug on your computer as you read the following discussion. (The method used here is an expanded version of the one used by Kathe Spracklen in Z-80 and 8080 Assembly Language Programming.)

The display of the f.com program before the first command is executed:

```
AX=0000  BX=0000  CX=001E  DX=0000  SP=FFFE  BP=0000  SI=0000  DI=0000
DS=12E9  ES=12E9  SS=12E9  CS=12E9  IP=0100   NV UP EI PL NZ NA PO NC
12E9:0100  B008          MOV      AL,08
```

(Don't be alarmed if the DS, ES, SS, and CS register contents fail to match the contents shown above!)

How does one know if a flag is set or clear?

In the debug display shown above, the first line and the left portion of the second line shows the contents of the registers in hexadecimal. For example, CL (right byte of the CX register) contains 1Eh, which would be 30 decimal. The center of the last line shows the **next command** to be executed. In the display above, this is MOV AL,08, which will place 08 in the right byte of the AX.

At the right end of the second line of the display above, are eight unique sets of two letter combinations beginning with NV and ending with NC. Each set represents the condition (clear or set) of a specific flag, as shown below. For example, if the Overflow flag is clear, the letters NV will appear in the left-most position; if it is set, the letters OV will instead appear in that position; if the carry flag is set, the last position will be CY.

```
0 (clear)     NV  UP  DI     PL  NZ     NA  PO  NC
FLAGS:        O___D___I______S___Z______A___P___C__
1 (set)       OV  DN  EI     NG  ZR     AC  PE  CY
```

Notice that all flags are clear before the execution of the first command.

How is a flag changed from clear to set or visa versa?

There are two general methods to changing the condition of a flag: first, as a result of a change in the contents of a register; second, in response to a command which changes only the flag. Included in the first general method are the arithmetic operations on a register. To illustrate that we will explore the effect of addition on the flags. See Figure 1.

· Next let's look at the compare command.

```
CMP     BL,4
```

Since the values in these registers are equal, when 4 is **subtracted** from the contents of BL the result is zero. Hence, the zero flag is set from NZ to ZR.

```
CMP     BL,AL
```

Since the value in the BL register is greater than the value in AL register, the zero flag is not set.

```
CMP     AL,BL
```

Since the value in BL is still greater than AL, zero flag is not set. However, because

Figure 1. Although debug displays the register contents in hexadecimal, seeing the contents in binary can help in the understanding of the setting of some flags. So, we will also show, at the left of each command line, the contents in binary **before the command is executed**, and at the right of each command line, the contents in binary **after the command is executed**. An explanation of each action will be described in the appropriate note.

before command			COMMAND	after command			NOTE
flags	reg	contents		flags	reg	contents	
NV UP DI PL NZ NA PO NC	AL	00h 00000000b	**MOV AL,8**	NV UP DI PL NZ NA PO NC	AL	08h 00001000b	**A**
MV UP DI PL NZ NA PO NC	BL	00h 00000000b	**MOV BL,4**	NV UP DI PL NZ NA PO NC	BL	04h 00000100b	**A**
MV UP DI PL NZ NA PO NC	AL	08h 00001000b	**ADD AL,BL**	NV UP DI PL NZ NA PE NC	AL	0Ch 00001100b	**B**
MV UP DI PL NZ NA PE NC	AL	0Ch 00001100b	**ADD AL,5**	NV UP DI PL NZ AC PE NC	AL	11h 00010001b	**C**
MV UP DI PL NZ AC PE NC	AL	11h 00010001b	**ADD AL,0F0h**	NV UP DI PL NZ NA PO CY	AL	01h 100000001b	**D**
MV UP DI PL NZ NA PE CY	AH	00h 00000000b	**ADC AH,00**	NV UP DI PL NZ NA PO NC	AL	01h 00000001b	**E**
MV UP DI PL NZ NA PO NC	AX	101h 0000000010000001b	**ADD AX,0e100h**	NV UP DI NG NZ NA PO NC	AX	E101h 1110000100000001b	**F**
MV UP DI NG NZ NA PO NC	AX	E201h 1110001000000001b	**ADD AX,0e200h**	NV UP DI NG NZ NA PO CY	AX	C301h 11100001100000001b	**G**
MV UP DI NG NZ NA PO CY	AX	C401h 1100010000000001b	**ADD AX,8000h**	OV UP DI NG NZ NA PO CY	AX	4301h 10100001100000001b	**H**

Notes to Figure 1.

A. No change in flag settings after these commands.

B. After this command, the parity flag changes from PO to PE because the value in AX has changed from 1000 to 1100. 1000 has an odd number of 1's (one); 1100 has an even number of 1's (two). There will be subsequent changes in this flag.

C. After this command the aux carry flag is set from NA to AC since a 1 is carried from the right nibble to the left nibble.

D. After this command, the aux carry flag is cleared from NA to NC since there is no carry from the right to the left nibble. However, the carry is set from NC to CY because the sum produces more binary digits than the AL register can hold (1 = Fh = 10h).

E. After this command, the carry flag is cleared from CY to NC since carry is added to AH.

F. Note that the sign flag is changed from PL to NG although we have added a positive number. Since a 1 appeared in the leftmost bit, the chip assumes that it is a negative number. It can't distinguish between signed and unsigned numbers.

G. The carry flag is again changed from NC to CY.

H. The overflow flag is set from NV to OV because the msb (most significant bit) produces a carry without receiving one. Similar results would have been produced if the msb had received a carry without producing one. Also, note that the sign flag has changed from NG to PL since the msb is 0.

the result of subtracting BL from AL is negative the sign flag is set from PL to NG.

How do we use the conditions of the flags?

Conditions of the flags are used in several ways. Most often the flags are used when issuing the conditional jumps. So before using a conditional jump, the programmer must know which flags are used to determine if the conditional jump is executed. These are shown below in tabular form. For some, only one flag is examined, i.e., JC jump when CY is present; for others, the flags are checked for boolean combinations* of the conditions (set or clear). i.e., JG will be executed if either (but not both!) NV or PL is present; if neither or both NV and PL are present, then if ZR is present, JG will be done. See Figure 2.

The interrupt and trap flags will not be discussed because of the infrequent use of these flags by beginners. The direction flag has not been discussed yet because it is not used with arithmetic operations, with compare or with conditional jumps. The direction flag is used with the LODSx and STOSx commands.

LODSx and STOSx use the flag when determining the next memory location to access. If the direction flag is clear (UP), the command will go to the next higher numbered memory location; if the flag is set (DN), the command will go to the next lower numbered memory location. The programmer can control the direction flag by the use of the set (STD) and clear (CLD) commands.

One other flag which the programmer can control is the carry flag. This flag can be set by the STC command and clear by the CLC command. In addition, the carry flag may be **complemented** by the command CMC; this means that if the carry flag is clear (0), then it will be set (1), or if the carry flag is set, then it will be cleared.

Only the addition operation has been covered here. Other arithmetic operations also affect the flags. Subtraction will affect the same flag as the CMP command since the CMP command subtracts. An attempt to borrow with subtraction will set the carry flag. Multiplication and division will also affect the flags.

Hopefully, using the principles covered here, the novice programmer can now use a reference such as intel's 86/88 Programmer's Reference to determine which flags will be affected. For example, in this reference the INC command (which adds one to the content of the register) shows the flags affected as AF, OF, PF, SF, ZF; the reader should now be able to show that each of these flags will be affected after a certain number of such commands. A general description will indicate the effect of the flags on the operation; to determine the detail steps of the affect, one must be able to understand the pseudocode and the symbols used and will not be covered here.

Listing 1

```
CODE     SEGMENT
         ORG     0100H
         ASSUME  CS:CODE
BEGIN:   MOV     AL,8
         MOV     BL,4
         ADD     AL,BL
         ADD     AL,5
         ADD     AL,0F0H
         ADC     AH,00
         ADD     AX,0E100H
         ADD     AX,0E200H
         ADD     AX,8000H
         CMP     BL,4
         CMP     AL,BL
         CMP     BL,AL
         RET
CODE     ENDS
         END     BEGIN
```

To converting Listing 1 to a .com file, perform these steps.

Figure 2

CONDITIONAL JUMP	OF	SF	ZF	PF	CF
JA JNBE			NZ	and*	NC
JAE JNB					NC
JB JNAE					CY
JBE JNA			ZR	or*	CY
JC					CY
JZ JE			ZR		
JG JNLE	(NV xor*	PL) or*	ZR		
JGE JNL	NV xor*	PL			
JL JNGE	OV xor*	NG			
JLE JNG	(OV xor*	NG) or*	ZR		
JNC					NC
JNE JNZ		PL			
JNO	NV				
JNS		PL			
JNP JPO				PO	
JP JPE				PE	
JO	OV				
JS		NG			

1) Enter Listing 1 into flags.asm file, using an editor or word-processor in the non-document mode.
2) Use MASM to assemble the source file flags.asm into an object file flags.obj.
3) Link flags.obj to flags.exe.
4) Perform exe2bin on flags.exe to convert to f.com.
5) Enter f.com into debug program.
6) Begin with the r command; this should produce the display shown above.
7) Step through using the p command. As you are stepping through the file observe the register contents and flags notation.

*

"ROGER GOT A LITTLE FRUSTRATED DOING THE TAXES SO HE TOOK A SMALL BITE OUT OF HIS APPLE."

Windows:

"Do you do them?"

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

Anyone who hasn't seen a puff piece or review of Microsoft s release 3.0 of WindowsTM must have just returned from deep space or has no business messing about with computers. InfoWorld and Government Computer News both had huge supplements on Windows 3.0 and I suppose PC Week did too, but they haven't renewed my free subscription. And they were just the tip of the iceberg; PC Magazine has added a column for Windows.

My upgrade copy of Windows 3.0 arrived about a week after the big event was announced. At that time my 20MHz 80286 machine was configured with the usual 640K, plus 384K shadow RAM, and 3M of additional memory configured as expanded (not extended). All memory is on the mother board. (If you are wondering what happened to the H/Z-150, it is now on the kid's table, across the room.) In addition, I was running QRAMTM by Quarterdeck to stuff device drivers, etc., into the area above DOS and keep the lower 640 as free as possible. After reading over the installation instructions for Windows, I removed QRAM to avoid any possible conflicts and changed the expanded memory to extended. Installation was swift and there I was with 3,712 kilobytes of memory available for programs.

The graph is a snapshot of the memory usage at the time this was written. The advantages of protected mode operation are quite apparent. DOS has its 640K (actually 640K + 56K from QRAM adding in some shadow memory); Windows uses 400K; the Window applications running take 472K; and 2080K are left over. The Window applications running are: Word for Windows; Windows Program Manager, Notepad and Calculator; plus some utilities from hDC, including the Memory Viewer that created the graph. Although not evident in the graph, SmartDrive is running a 2M disk cache. SmartDrive is a disk caching program furnished with Windows. My systems run much faster with this cache in use.

With the previous versions of Windows I found it difficult to run two major applications. Even running one took a bit of patience. Now I can run Word for Windows plus Corel Draw!, and show almost 1.8M of free memory, notice no degradation in performance, and switch between them with a mouse click (or Alt-Tab). This is a considerable boost in performance over Windows 2. Adding Excel to the screen still leaves 1.6M free.

Your current Window applications, those not designed for version 3.0, may or may not run in protected mode. By the time this appears in print all vendors should have their upgraded versions available.

The current version of PageMaker (3.0) bombs in protected mode. The upgrade to 3.01 is only $25; it took several weeks to arrive. Corel got their upgrade (1.2) to me quickly but charged $99 for it! The upgrade for Crosstalk for Windows was free and slower still in arriving.

A major improvement in my use of Windows is support for Xon-Xoff on the serial port driver. Part of the time I operate with a direct connection to a Unix machine. At 9600 baud I found Windows 2.11 not to work properly and had to disable flow control. Sometimes that led to some strange screens. The terminal program included with 3.0 is much more useable then previous versions; I used it for several weeks while waiting for the Crosstalk upgrade to arrive.

A new program I have added is GrammatikTM for Windows. I had been using Grammatik IV, a DOS program, with Word 5. Since I was converting to WinWord it made sense to use the Windows version of Grammatik. The manual makes no mention of whether it works with files that contain graphics. My first attempt with such a file produced some spectacular results when I loaded the corrected file back into WinWord. Error messages of "Unrecoverable disk error" and others related to the graphic required me to abandon the file and revert to the backup. I have not yet determined if it was cockpit error or a program problem.

I read one review of the update to WinWord 1.1 that stated the "new look" in the icons and buttons is not available with an EGA display. Not so. Page 12 of the installation guide says to add the following line to your win.ini file to turn on the improved display for the EGA display adapter: NewLook=1.

After the initial installation of Windows was over I found that Windows 3.0 and QRAM will co-exist. There is an upgrade to a Windows 3.0 compatable version, but I have yet to identify a problem with the older version. QRAM allows me to increase the memory available to standard DOS applications both running normally and from within Windows. Standard applications now have some breathing room to run under protected mode if you do not want to exit Windows to run them. DOS programs run full screen and not in a window in what Microsoft calls standard mode. I have been successful running the following DOS programs from within Windows:

Word 5.0
Paradox 3.0
Reflex 2.0
Brief 3.0
Multiplan 4.2

New Print Shop
Autosketch

Softcraft s Font Solution Pack did not run even under DOS. It failed, thinking Windows is not present, because it is looking for some files that are no longer used by Windows. After writing to Softcraft about the problem I received a beta version (2.1b) of WYSIfonts that corrects the problem. There is still one bug in the printer selection menu but the default, LaserJet II, is just what I needed.

Initally, the only TSR program I had running, besides QRAM was FontSpaceTM (by Isogon). It compressed my 12.3M of PCL fonts down to 3.2M (74%). It decompresses them on the fly when the printer needs them and there is no slowdown I can see, which is what they claim.

With all this going on, running chkdsk from within Windows shows just over 518K of RAM available to DOS. Be sure you don't run chkdsk with the "fix" switch (/f) set as it will scramble your disk drive.

My wife uses the computer so infrequently, she has decided to stick with Word 5.0 and avoid the learning curve for WinWord. To accommodate her needs, I initially set up Windows to run the Program Manager with the Word icon highlighted. A punch on the return key started Word for her. The only new command she had to put up with was to exit Windows (Alt-F4).

Besides my system I have Windows running on three other platforms; two are 80286 CPUs and one is an 80386SX. One 80286 is a 10MHz system with only 640K. Now the manual for Windows says for real mode operation you need only an 8086 or 8088 and 640K. My advice is to forget real mode with only 640K for any CPU. The fixed disk will never stop and you will drink lots of coffee waiting for something to happen. The only event that does not require swapping to disk is moving the mouse cursor over the screen. Adding an AST Sixpak 286 board with 512K of extended memory made this machine useable with Windows.

The other AT class machine is a 12MHz 80286, first set up with 512K of extended and then later with 2.5M of extended. Performance with only 512K was acceptable. Adding the other 2M helped but not as much as it could have since the memory had to go on an expansion card and not the mother board. This slows down memory access to the buss speed but it is still faster than disk swapping.

The 16 MHz 80386SX running Windows is slightly slower than my 20 MHz 80286. I tried the 386 enhanced mode with this machine. This mode is supposed to allow multiple DOS applications to run in separate windows. However, things run a little slower. The real problem was running just command.com the screen update was very slow; a half-second or so would pass before the keypress would show up in the window. Another turn-off to extended mode is the need for setting aside 1M or 2M of disk space for a hidden swap file. Extended mode might be practical on a 33MHz 08386 but no one has volunteered a machine for me to experiment with.

Printing is still slow with Windows. At least it is easier to disable the Print Manager in 3.0 than it was the Spooler in 2.11. With the older version of Windows I had edited the win.ini file to turn the Spooler off. Then I discovered that Crosstalk would not print without the Spooler. There was no convenient way to change the status of the Spooler without editing win.ini and restarting Windows. Now there is a check box to turn the Print Manager (the Spooler replacement) on or off at any time.

The brain-damaged MS-DOS Executive has been replaced with a very workable Program Manager and File Manager. The default installation runs Program Manger as the shell program. I have been using hDC s Windows Express (new version) as the shell.

With previous versions of Windows, doing something simple to your machine, like upgrading the video card required completely reinstalling the program. Now it is possible to make a change in the video, keyboard or mouse from the Windows Setup program. But I still keep several revisions of win.ini around, just in case.

Windows 3.0 has breathed new life into the 80286 machine, I feel. For example, preparing the screen shot of the graph was a breeze. I punched PrtSc to capture it to Clipboard, then fired up Paintbrush with a couple of mouse clicks. The Clipboard contents were pasted into Paintbrush for editing. After editing it was copied back into Clipboard and then pasted into the Word document.

Oh, I also discovered a side benefit of working with programs that use a mouse. I dumped a full glass of liquid into my keyboard doing this paper. Word started beeping as keys shorted out so I unplugged the keyboard. Fortunately I had already named the document so I could do a file save and exit with the mouse. Otherwise I would have lost most of my work. Yes, the keyboard survived; a quick wash and dry and it is as good as new.

With things working so well, I got bolder and decided to try the new Adobe PostScript cartridge for my LaserJet II. At only $250 it was too good a bargain to pass up. One advantage it has over the Pacific Data cartridge is it requires only 1.5M of printer memory, whereas the Pacific Data needs 2.5M of printer memory. With a lower initial cost, true PostScript capability and the savings on printer memory I could see no way to go wrong.

Besides the cartridge Adobe includes a program, Printer Control PanelTM by LaserTools and all the font outlines in the cartridge also on disk, plus a program to generate screen fonts and bitmaps for PCL use.

Printer Control Panel can be configured to switch automatically between the native PCL mode and PostScript mode. The LaserJet control panel also can be used to switch between modes. When the printer is turned on, with the cartridge installed, it starts out in PostScript mode. It also ejects a test page at turn on but this can easily be disabled.

The only problem I have had to date was trying to print the train engine furnished with CorelDraw! I had previously printed it in PCL format and was anxious to see how PostScript compared. After giving the print command, CorelDraw! ground away, the printer indicated data was being accepted, but all for naught. After several minutes everything came to a stop but no output, not even a blank sheet. I was afraid I was running out of printer memory. However, a call to Adobe provided the secret to printing the file.

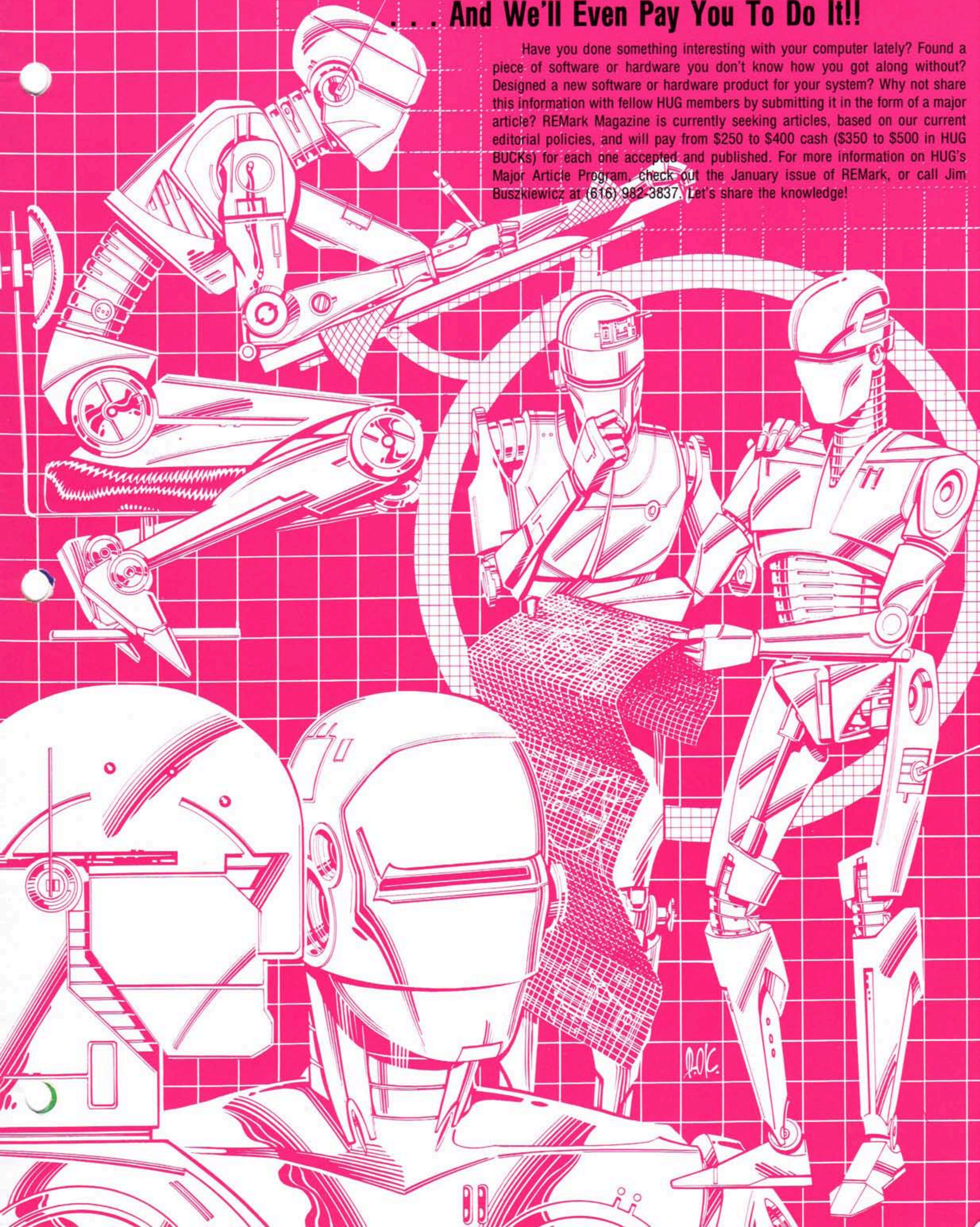
Because the graphic is complicated it prints very slowly and the system is timing out. Adobe technical support suggested the following steps to print this graphic:

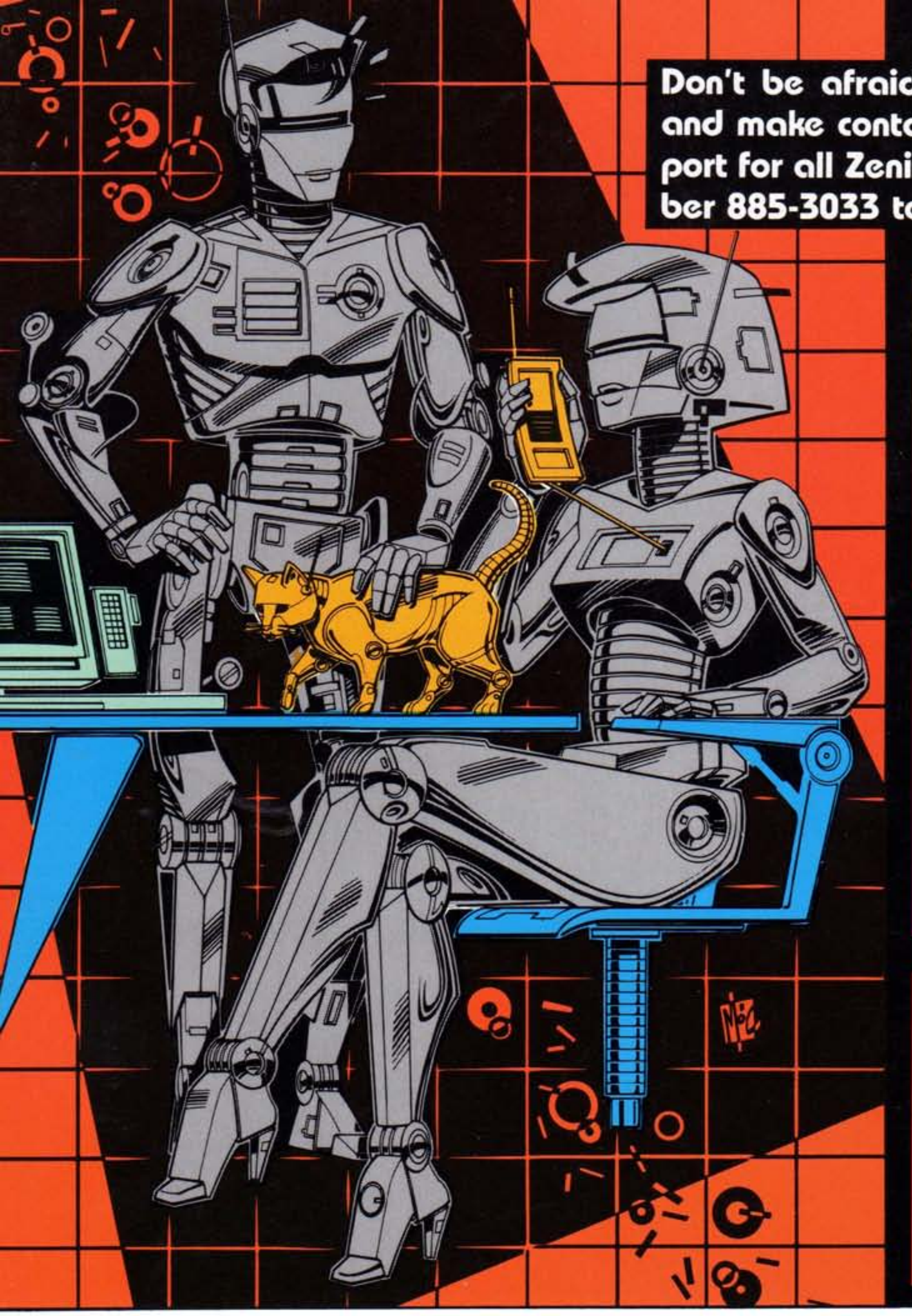
- Turn off Printer Control Panel
- Add the following mode command to your autoexec.bat file: MODE LPT1,,P
- With Windows Control Panel set the printer port to LPT1.OS2

The demand for Windows applications appears to have really taken off. Several vendors have had to run to keep up with the demand; I observed some applications back ordered recently. I switch back and forth between command line interfaces (Unix and DOS) and the Windows graphical interface daily. I find advantages and disadvantages in both camps. Overall, I favor the Windows interface.

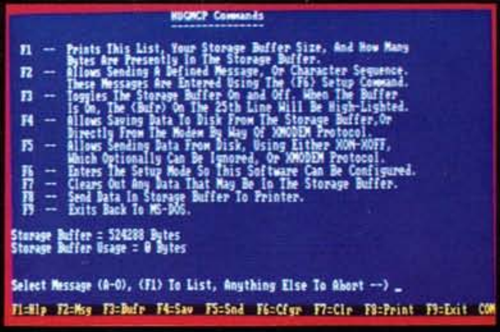
Authors, Share Your Knowledge With Other HUG Members, . . . And We'll Even Pay You To Do It!!

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!

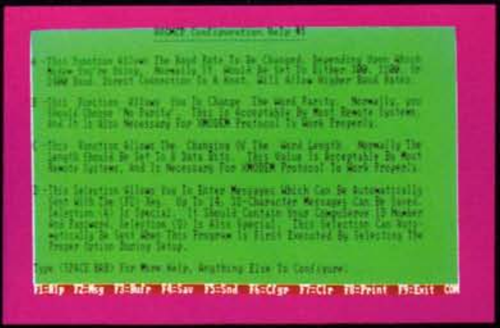




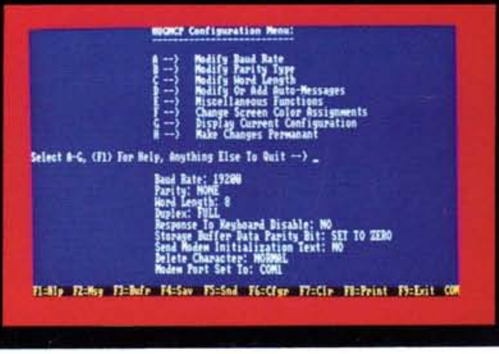
Don't be afraid to communicate! Get HUGMCP and make contact the easy way. Now with support for all Zenith Laptops, order HUG Part number 885-3033 today.



```
                         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 (F6) Setup Command.
F3  --  Toggles The Storage Buffer On and Off. When The Buffer
        Is On, The (Bufr) 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=Bufr  F4=Sav  F5=Snd  F6=Cfgr  F7=Clr  F8=Print  F9=Exit  COM
```



```
                  HUGMCP Configuration Help #1
                  ----------------------------

A -This Function Allows The Baud Rate To Be Changed. Depending Upon Which
   Modem You're Using, Normally It Would Be Set To Either 300, 1200, Or
   2400 Baud. Direct Connection To A Host Will Allow Higher Baud Rates.

B -This Function Allows You To Change The Word Parity. Normally, you
   Should Choose 'No Parity'. This Is Acceptable By Most Remote Systems
   And It Is Also Necessary For XMODEM Protocol To Work Properly.

C -This Function Allows The Changing Of The Word Length. Normally The
   Length Should Be Set To 8 Data Bits. This Value Is Acceptable By Most
   Remote Systems, And Is Necessary For XMODEM Protocol To Work Properly.

D -This Selection Allows You To Enter Messages Which Can Be Automatically
   Sent With The (F2) Key. Up To 14, 32-Character Messages Can Be Saved.
   Selection (A) Is Special. It Should Contain Your CompuServe ID Number
   And Password. Selection (O) Is Also Special. This Selection Can Auto-
   matically Be Sent When This Program Is First Executed By Selecting The
   Proper Option During Setup.

Type (SPACE BAR) For More Help, Anything Else To Configure.
F1=Hlp  F2=Msg  F3=Bufr  F4=Sav  F5=Snd  F6=Cfgr  F7=Clr  F8=Print  F9=Exit  COM
```



```
                  HUGMCP Configuration Menu:
                  --------------------------

                  A --)   Modify Baud Rate
                  B --)   Modify Parity Type
                  C --)   Modify Word Length
                  D --)   Modify Or Add Auto-Messages
                  E --)   Miscellaneous Functions
                  F --)   Change Screen Color Assignments
                  G --)   Display Current Configuration
                  H --)   Make Changes Permanant

Select A-G, (F1) For Help, Anything Else To Quit --) _

                  Baud Rate: 19200
                  Parity: NONE
                  Word Length: 8
                  Duplex: FULL
                  Response To Keyboard Disable: NO
                  Storage Buffer Data Parity Bit: SET TO ZERO
                  Send Modem Initialization Text: NO
                  Delete Character: NORMAL
                  Modem Port Set To: COM1
F1=Hlp  F2=Msg  F3=Bufr  F4=Sav  F5=Snd  F6=Cfgr  F7=Clr  F8=Print  F9=Exit  COM
```

Heath
Users'
Group

P.O. Box 217
Benton Harbor, MI 49022-0217

$2.50

P/N 885-2130