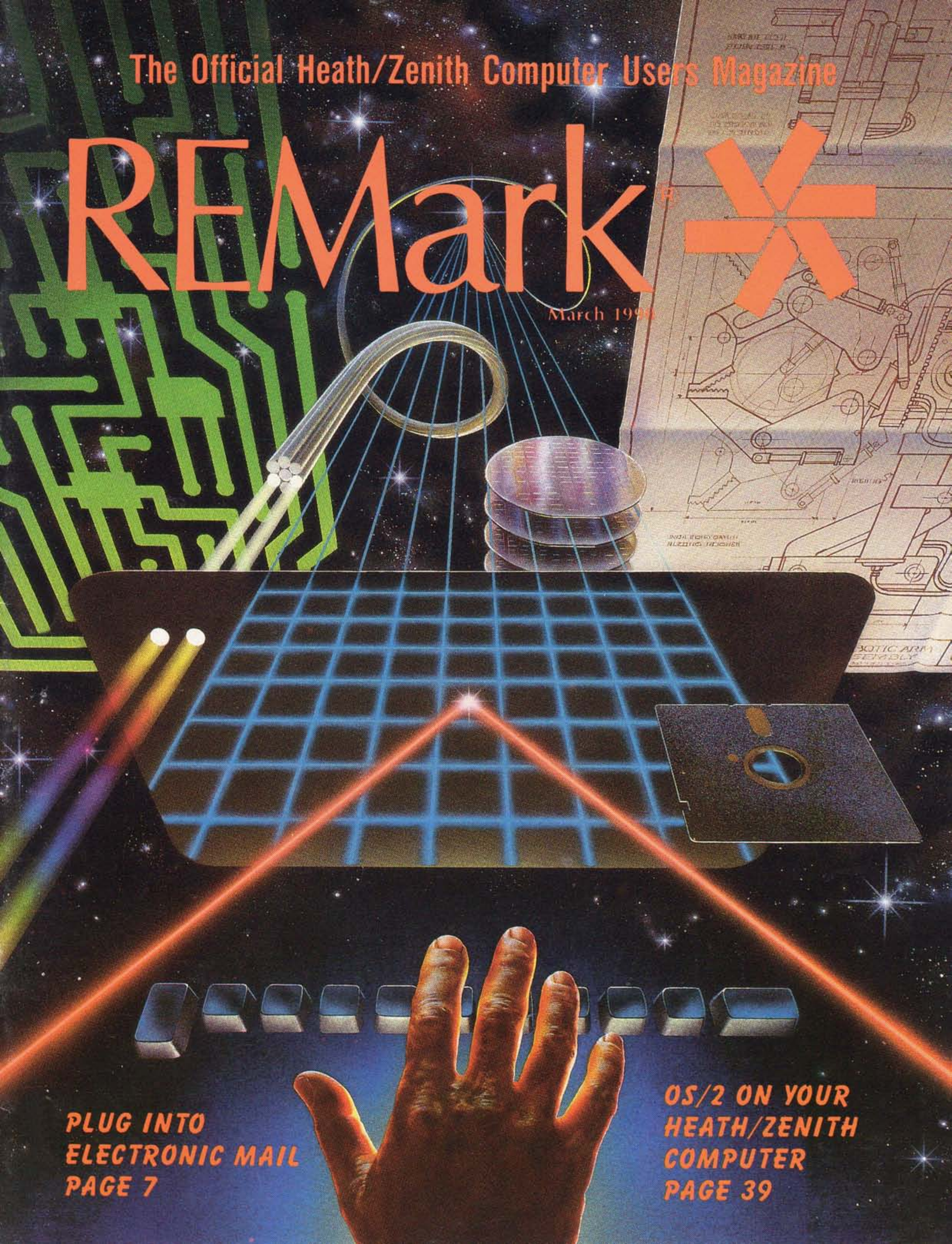
The Official Heath/Zenith Computer Users Magazine

REMark®

March 199



PLUG INTO ELECTRONIC MAIL PAGE 7

OS/2 ON YOUR HEATH/ZENITH COMPUTER PAGE 39


"How Can You Take Advantage of Me"
". . . If you don't call? I have everything you could possibly want! My software selection continues to grow, and remains my most popular feature. I'm fast, but, if you don't have the time to download, these software disks can now be purchased for a small copying charge! My message base has also become quite popular. Through it, HUGgies are exchanging more information than ever before. Finally, there's my legendary Bargain Centre. It alone, will make you come back for more! Did you know how inexpensive I am? Why pay $14 per hour connect time to someone else when your phone company charges less than $12 per hour (less on weekends) from anywhere within the continental U.S.! So, go ahead and take advantage of me. Just set your modem for 300, 1200, or 2400 baud (8N1) and call (616) 982-3956. You needn't type anything, I'll know you're there!"

The Official Heath Computer Users Magazine

REMark®

Volume 11, Issue 3 • March 1990

Resources

PC Compatibles

H/Z-100 and PC Compatibles

H/Z-100 Only

Don't Miss A Single Issue!
Let us know 3-4* weeks
before you move!

PC Compatibles

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

Managing Editor .... Jim Buszkiewicz
(616) 982-3837

Software Engineer............ Pat Swayne
(616) 982-3463

Production Coordinator .... Lori Lerch
(616) 982-3794

Secretary .... Margaret Bacon
(616) 982-3463

HUG Bulletin Board .... (616) 982-3956
(Modem Only)

HUG Parts Ordering .... (616) 982-3463

Hardware Questions ...... (616) 982-3309

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

Printer.................. Imperial Printing
St. Joseph, MI

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

*U.S. Funds

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

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

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
REMARK VOL 3 ISSUES 24-35	885-4003	N/A	1982	20.00
REMARK VOL 4 ISSUES 36-47	885-4004	N/A	1983	20.00
REMARK VOL 5 ISSUES 48-59	885-4005	N/A	1984	25.00
REMARK VOL 6 ISSUES 60-71	885-4006	N/A	1985	25.00
REMARK VOL 7 ISSUES 72-83	885-4007	N/A	1986	25.00
SEA BATTLE	885-1211-[37]	CPM	GAME	20.00
UTILITIES BY PS	885-1226-[37]	CPM	UTILITY	20.00
UTILITIES	885-1237-[37]	CPM	UTILITY	20.00

Price List

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

H/Z-100 (Not PC) Only

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
ACCOUNTING SYSTEM	885-8048-37	MSDOS	BUSINESS	20.00
CALC	885-8043-37	MSDOS	UTILITY	20.00
CARDCAT	885-3021-37	MSDOS	BUSINESS	20.00
CHEAPCALC	885-3006-37	MSDOS	SPREADSHEET	20.00
CHECKBOOK MANAGER	885-3013-37	MSDOS	BUSINESS	20.00
CP/EMULATOR	885-3007-37	MSDOS	CPM EMULATOR	20.00
DBZ	885-8034-37	MSDOS	DBMS	25.00
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

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
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 5 ISSUES 48-59	885-4005	N/A	1984	25.00
REMARK VOL 6 ISSUES 60-71	885-4006	N/A	1985	25.00
REMARK VOL 7 ISSUES 72-83	885-4007	N/A	1986	25.00
REMARK VOL 8 ISSUES 84-95	885-4008	N/A	1987	25.00
SCREEN DUMP	885-3043	MSDOS	UTILITY	30.00
UTILITIES II	885-3014	MSDOS	UTILITY	20.00
Z100 WORDSTAR CONNECTION	885-3047	MSDOS	UTILITY	20.00

PC Compatibles

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
ACCOUNTING SYSTEM	885-8049	MSDOS	BUSINESS	20.00
CARDCAT	885-6006	MSDOS	CATALOGING SYSTEM	20.00
CHEAPCALC	885-6004	MSDOS	SPREADSHEET	20.00
CP/EMULATOR II & ZEMULATOR	885-6002	MSDOS	CPM & Z100 EMULATORS	20.00
DUNGEONS & DRAGONS	885-6007	MSDOS	GAME	20.00
EZPLOT II	885-6013	MSDOS	PRINTER PLOTTING UTILITY	25.00
GRADE	885-8037	MSDOS	GRADE BOOK	20.00
HAM HELP	885-6010	MSDOS	AMATEUR RADIO	20.00
KEYMAP	885-6001	MSDOS	UTILITY	20.00
LAPTOP UTILITIES	885-6014	MSDOS	UTILITY	20.00
PS's PC UTILITIES	885-6011	MSDOS	UTILITIES	20.00
POWERING UP	885-4604	N/A	GUIDE TO USING PCS	12.00
SCREEN SAVER PLUS	885-6009	MSDOS	UTILITIES	20.00
SKYVIEWS	885-6005	MSDOS	ASTRONOMY UTILITY	20.00
TCSPELL	885-8044	MSDOS	SPELLING CHECKER	20.00
ULTRA RTTY	885-6012	MSDOS	AMATEUR RADIO	20.00

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

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".

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.

Reader Service #130

Troubleshooting the H/Z-151

Sanford Shapiro, M.D.
654 Gravilla Street
La Jolla, CA 92037

Hard Drive and Monitor ROMs

I thought the hard drive in my Z-151 computer was finished. I wasn't surprised. It was a 20 meg. Seagate ST-225, and it had given me four years of faithful and reliable service. I was more surprised to find that one turn of a single screw restored the hard drive to completely normal functioning.

I took a long time to discover hard drives. My first computer used a cassette tape player for I/O. I eventually bought a 140 K floppy disk drive and I was in heaven. Prices for floppy disk drives were high in those days, something over $400. In time, I bought a second 140 K floppy drive and I was drunk with power. When I bought my Zenith 151 computer, it came with two 360 K floppy disk drives. Who could ever need more, I wondered?

Gradually, programs became more elaborate and more complex. Disk shuffling became a nuisance. Four years ago, when hard drive prices fell to around $500, I took the plunge and bought a 20 meg. Seagate ST-225. It was a new ball game. There was no more shuffling of floppy disks and all programs were immediately available. I was hooked.

A few months ago, my hard drive started acting strangely and I had a panic attack. After four years of daily use with never a hint of difficulty, the drive began to have trouble booting up in the morning when cold. It would spin, make clicking noises and then show a message: "Disk Read Error." If I waited a few seconds and then rebooted, the drive would boot up normally. A thorough disk scrubbing with Spinrite, a hard drive utility program, showed no problems or bad sectors. Doing another Prep and Format did not help. I posted a note on the Zenith Forum on Compuserve, and I was advised to do the Prep the first thing in the morning when the drive was still cold. That did help for a few days and then the problem recurred.

I called Seagate Technical Support. I was told that the stepper motor might be showing some age and that the drive may need a little extra time to get up to speed when cold. I asked if this was the beginning of the end. Technical Support said that the drive could continue to work fine for a long time, but he told me to be very conscientious about doing backups. That advice made me even more anxious, and I immediately began shopping for a new drive. Besides, I thought, this could be an opportunity to upgrade to a higher capacity, faster drive.

As a last resort, I presented the problem to the members of our San Diego HUG. Al Brengle, a hardware specialist, thought that if Spinrite gave the drive a clean bill of health, he would still have confidence in it. Another HUG member said he had heard that some Seagate drives had the circuit boards mounted too tightly to the drives causing something to bind. Loosening a screw would cure the problem.

That, in fact, did the trick! Although I had already decided to get a new drive, I could install this drive in my other Z-151 computer. I have a second Z-151 that I bought from our National HUG Bargain Center. I removed the drive from the first computer and turned it over. Three screws secured the circuit board to the drive. Along the back edge were two screws, one in each corner. Along the front edge was one screw right in the middle. I loosened the one screw in the middle exactly one half turn and installed the drive in the computer. Once installed, it booted right up and has continued to boot and run without difficulty for months now.

The two computers have different controllers, and that may or may not be significant. When I removed the drive, I left the Western Digital controller in place. I had purchased a new SuperRom Bios chip from Western Digital for $15, and the controller would now support a 40 meg. drive. For the second computer I installed a Xebec controller that I had won in a HUG raffle.

Our local Heath/Zenith store donated the controller for our raffle. I had been eager to try a genuine Zenith controller because my Western Digital controller would not recognize the Zenith "autoboot" function. Instead of booting directly from the hard drive, the controller first looked to drive A:, and when it didn't find a disk there it would then boot from the hard drive. Much to my surprise, the new controller did the same. I called Zenith technical support and was told that my controller was configured for the IBM XT. I needed a special Zenith ROM but no one knew which one. I tried several different ROMs without success, gave up and filed the controller board away. Now I took out the controller board and would use it.

I installed the board and the hard drive in the computer. I ran the Prep and Format utilities and restored my files. A Spinrite analysis again gave the drive a clean bill of health. There was not even one bad sector and I was in business. The drive worked like a charm. I am still conscientious about doing backups, but the drive acts like it will live forever, hot or cold. It is hard to believe that loosening one screw a half turn could have that much of an effect. Nevertheless, it has worked for others and it is like a new lease on life for me. I also learned to go to my Heath Users' Group first, and not as a last resort.

40 Meg. Drive

The drive that I decided to buy is the 40 meg. Seagate ST-251-1. It is also a half height drive and has a 28 msec. access time that I thought would be a good companion for the 80386 chip now in my Z-151. It is performing, according to the Norton Utilities, three times faster than the ST-225. Although the drive was easy to install and get running, it provided a few unexpected surprises.

The drive came bundled with "Disk Manager" software, but I did not use that program. I wanted to see what I could do using the Zenith utilities (MS-DOS 3.3+). I ran Prep using the /Q switch with the number of heads and cylinders provided by the manufacturer. I then ran Format and found I had a C: drive with 40+ meg. of space. After restoring my data and running some programs, I began to feel that something was amiss. In some ways the drive seemed faster than the old one, but in some ways it seemed slower. I finally determined that my disk caching program was not working. I had been using ICache, a subset of Super PC-Kwick provided by Intel with the Inboard 386/PC. The cache runs in the 256 K of extended memory provided by the Inboard. I posted a note on the Intel BBS and was told that ICache would not work on hard drives over 32 meg. in size.

I solved the problem by switching to PC-Cache. This program is also a subset of Super PC-Kwick and is provided with the PC Tools utilities. PC-Cache loads and runs the same as ICache except that it works fine with the 40 meg. drive. There is no significant difference in speed or in memory usage between the two programs.

The next surprise was when I went to use my favorite disk optimizing program, VOPT. It refused to work. A call to Golden Bow Systems revealed that I needed to spend $18 for an upgrade. I sent in my money and received the new VOPT release which works fine. Erasing and rewriting files leads to file fragmentation. In time, the hard drive performance will deteriorate. A disk optimizing program will reorient files on the hard drive so that they are contiguous. A disk optimizing program will help maintain peak hard drive performance.

The last program to let me down was a directory manager program from PC Computing magazine called "DM." This program lists all directories and subdirectories in a tree-like manner and makes them accessible to the companion program "DR." "DR" lists all the files in a directory and allows for file copying, deleting, moving, changing attributes and other functions. The program is provided by the magazine on a disk called "DIR-MAGIC." I wrote to PC Computing, but they never answered me.

Everything else was now working fine. Not being content to let well enough alone, I wondered what it would be like to partition the drive into two 20 meg. logical partitions. I had never used the Zenith Part utility and was curious to try it. It was a nightmare. If I tried to add a DOS partition, I was told that there was no room. If I deleted the DOS partition, I found myself with an inaccessible drive.

Actually, the process was simple and straightforward. I just couldn't figure it out on my own using the Zenith MS-DOS manual. I finally wrote to Bill Adney, the well known author of computer books and popular columnist for REMark Magazine. Bill wrote back immediately, and he carefully outlined the exact procedure to follow. The steps are (after backing up all data): 1) Delete primary DOS partition, 2) Add two new partitions (both can be primary), 3) Select default boot partition, 4) Make changes and REBOOT system, 5) Format the two new partitions. That did the trick and left me with a C: drive and a D: drive.

I don't know yet if there is any advantage to having two 20 meg. logical drives instead of one 40 meg. drive, but I am pleased to have had a chance to learn something new. I am also grateful to Bill Adney for his prompt and expert support.

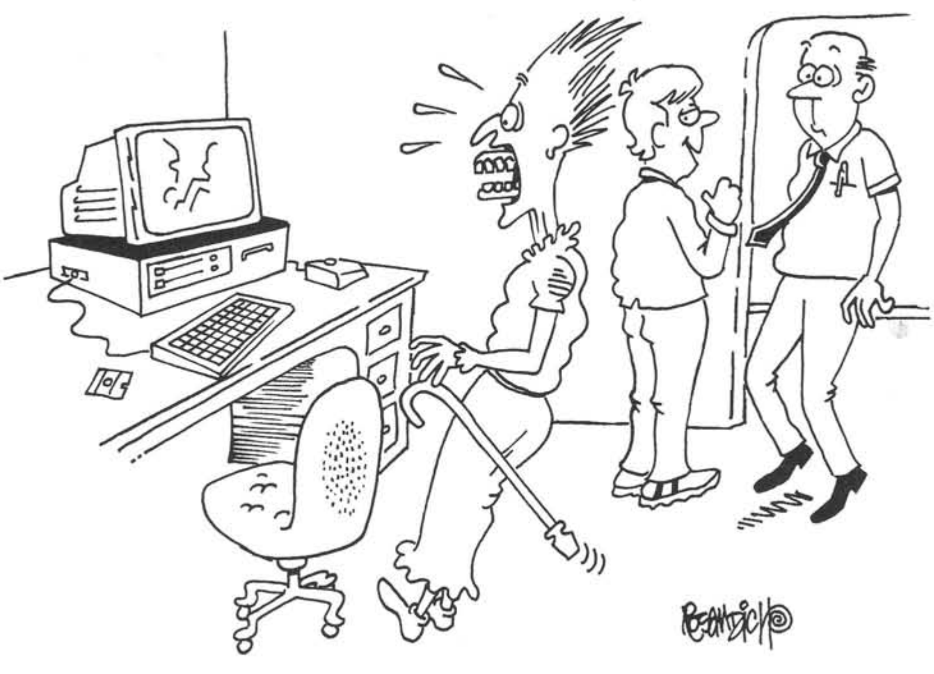
Zenith Monitor ROMs

One advantage of owning a Zenith computer is the ability to upgrade the monitor ROMs and thus ward off obsolescence. Recently, I had an experience where I had to downgrade a set of monitor ROMs.

There are two ways of identifying the version of a monitor ROM. One way is to type <Ctl-Alt-Ins> and read the ROM version on the screen. Another way is to look at the last two digits of the part number on the chips (two chips in the Z-151, part no. 444-229-xx and part no. 444-260-xx). When I took delivery of my Z-151 five years ago, the part number on the ROMs ended in "-04." (I don't remember the version number.)

About one year later I heard about upgrading the ROMs. I called Zenith technical support and asked what the advantage was of buying the new chips. I was told that it would make my computer more IBM compatible. I remembered that I had tried using a disk caching program (I

Continued on Page 28



"GRANDMA FOUND YOUR X-RATED FLOPPY!"

The Benefits of Expensive Timeshare Systems . . .

With None of the Expense!

Tom Jorgenson
First Capitol Computer
#16 Algana
St. Peters, MO 63376

As a computer user's experience increases, he or she becomes more and more aware of sources of additional information available to further his capabilities. Many users wisely join users' groups (such as the Heath Users' Group) around the country in order to exchange experiences with others. Still more purchase various educational materials or attend training classes.

Eventually many users consider paying for access time on a large timeshare system, such as the Compuserve Information System, GEnie, Source, or BIX (the BYTE Information Exchange). These systems provide a wide range of valuable services, but they do not come at any small expense. Per hour costs can easily run as high as $25 per hour, running an individual's bill well over $300 per month just keeping up with new message additions.

In this article I will cover an alternative that probably only the more experienced users are presently aware of, and one that is completely free in most areas of the country (and almost free everywhere else). The method under discussion is the electronic mail system pioneered by the International Fidonet Association (IFNA), and available throughout the country on free dial-up Bulletin Board Systems (BBS's).

What is Electronic Mail?

Electronic Mail (sometimes called EMAIL) is a very general term describing the transmission of a message through electronic means. It can be thought of simply as a way of sending a "letter" by computer rather than through the Post Office.

Unlike the Post Office, however, the sender has the ability to send a private message, a message to a group of users, or a public message which can be broadcast automatically to as many as several thousand locations around the world (and potentially viewed by a much larger number of individuals)!

Electronic mail requires a means of tying one computer system to another. After all, it would be pointless to send a disk through the mail with a word processing document (the message) on it, when the Post Office can perform such a task much more efficiently on common letterhead. The advantage of electronic mail is its ability to move from place to place at a much faster pace. Indeed, it is not unusual on BBS systems to send a message to an individual from one coast to another, and receive a reply the very next day!

The means utilized to transfer messages between computers is a device called a Modem (MOdulator/DEModulator). A modem converts digital information into sound which can be carried (in most cases) by standard telephone lines. A discussion of modems in any detail is beyond the scope of this article, and your local computer dealer will be happy to acquaint you with details. Suffice it to say that both ends of the connection use these nifty devices and that costs on your end should not run over about $175 for everything you need (a one-time investment).

Once you have a modem, you're ready to send files and information between yourself and another user with similar equipment. This other user might be another individual (in order to exchange public domain files, for example), a for-pay timeshare system, or a public bulletin board system.

Bulletin Board Systems

Bulletin Board Systems are computers with modems running special software to handle the callers, and to provide a comfortable interface for them to the remote system. This special software answers the phone, identifies the caller, handles messages, protects the system from electronic vandals, and (in the case of Fidonet systems) exchanges information with other similarly equipped bulletin board systems.

Bulletin board systems can take many forms, but most include three areas: (1) a local message area where users can leave and read messages for that BBS alone, (2) a download area where public domain files can be transferred to and from the BBS, and (3) an echomail area, which is similar to the local message area, except where the messages are sent and received (echoed) to and from other systems around the world.

The local message area is intended for the storage and retrieval of messages for users of the specific BBS being used. You might, for example, have a question for the operator of the BBS (the system operator, or "sysop"). Or, you might want to ask other callers a question about a file you've downloaded from that particular system. On most systems you can even leave a "private" message to another user or the sysop which only that person and the sysop can read (sysops have the legal responsibility for controlling their BBS's, so in practice, private messages aren't truly private from the sysop). This area alone will typically provide access to 300-500 other computer users in your local area on an average BBS.

Download areas are a means of obtaining public domain (free) and shareware (almost free) software legally and comfortably. You might, for example, determine that you need a spreadsheet program in the morning at 9:00AM, download one at 9:15AM, and be using it by 9:30AM. All this without leaving your comfortable chair! The volume of software available in this fashion exceeds belief. There are actually more than 2,000 megabytes of public domain software programs available in this fashion, and the number is still increasing rapidly.

Echomail

Most of us are at least somewhat familiar with the concept of a timeshare system. A timeshare system can be thought

(Or How to Get Plugged Into Public Electronic Mail Systems)

of as a central computer system at a remote location that you can call through your modem and operate from your keyboard. Many of the larger systems allow callers far distanced from each other to communicate with each other by sending messages in one form or another.

Bulletin board systems aren't large systems at all. Indeed, most of them are simply extra (generally older) computers that were no longer regularly used by their owners. Most BBS operators are knowledgeable computer users who have taken the time to install an extra phone line and special software on these computers so that outsiders are able to access them. But don't let the size of the computers fool you! A network of small computers easily exceeds the capabilities of a single timeshare system, due to the fact that the information is processed concurrently in many different places at once!

Some years ago a small group of people got together to produce software to tie these systems together into a network that now stretches throughout the globe with over 3,500 connecting points (BBS's). Any system on this network can communicate with any other system very simply. Indeed, the software being used makes the process completely automatic. Today this group of people have become the International Fidonet Association (based in St. Louis, Missouri), and a number of BBS software packages are available both commercially (such as Fidonet itself) and in the public domain (such as the Fidonet-compatible Opus package). Note that IFNA is a not-for-profit group formed to promote BBS usage, and that membership is not required to use any of the features discussed in this article.

The BBS software allows areas to be assigned for specific topics (one such area, the Heath Echo, is for Heathkit and Zenith Data Systems users specifically). Any BBS carrying this echo area will see any message posted in this area on any other BBS in the entire 3,500+ system network! The number of available echos exceeds 200 topics and the number of potential users probably exceeds 50,000, so any given BBS can only carry a small percentage of the total available (those most requested by his callers), but most sysops will be happy to carry any echo area for which there are a reasonable number of interested callers.

In appearance, echo areas are actually very similar to the commercial timeshare systems described above. Messages are labeled with the sender's name, date, message number, and message topic, and typically consists of one or two paragraphs of information (although an occasional message can cover several pages). If you enter a message on one of these areas, it will automatically appear on every other BBS system which carries that echo (typically overnight). You might well send a message to a buddy across country and receive a reply the very next day. A few systems utilizing a technique referred to as "crash mail" are actually even capable of delivering messages to destination addresses with an average time of 15 minutes! Western Union take note!

With over 3,500 potential systems involved, you might well wonder how this bit of magic is accomplished. Well, the basic concepts are actually much simpler in principle than you might think.

In the United States, for instance, there are 4 computer systems spread from coast to coast that are in regular connection with the others called the "backbone". When you enter a message on an echo area, the BBS system will convert that message into a specially packed format called a "packet" (not to be confused with other types of packet transmissions). These packets are sent en masse to another local BBS (the "regional coordinator", or the "network coordinator" in more active areas) that merges them with message packets from other BBS's and ships them up to the backbone. After the local BBS sends its packets to the regional coordinator, it also asks for additional message packets from other BBS's. These additional messages may actually be coming from almost anywhere on the network, and will consist of any mail directly to the BBS or any of the echo areas it carries.

The regional coordinator likewise calls its closest backbone connection and similarly transfers message packets in both directions. In order to optimize the speed of these transfers (and the associated long-distance charges), each link in the network transfers only the packets intended for itself and any other BBS which is obtaining its packets from that BBS. Thus, the backbone may have 200+ echoes that it carries. A regional coordinator may carry 50 of those echoes to service 20-30 other BBS's, and a BBS may carry 5-10 of them for its own callers.

Regional coordinators and backbone operators generally have a lot of online storage and very fast modems (coordinator to backbone communications are typically via US Robotics HST modems at 19,200 baud). Many megabytes of information are transferred on a monthly basis, and the regional coordinators generally split the costs amongst the BBS's they service. Since they may well service a number of BBS systems, these costs are actually very small on a per BBS basis (our own First Capitol Computer BBS typically pays only $6.00 per month total for everyone's messages, and we send and receive several hundred messages per week!)

In most cases, the local BBS costs are funded totally by the BBS operator — or an associated computer club. Remember, we SAID these guys were publicly spirited! A small number of BBS's charge very minor fees (for example, $.10 per message sent to a specific location rather than an echo area) to help defray costs — but in no case do such fees anywhere approach the cost of commercial timeshare systems.

About the Echoes

Now that we know what echomail is, and what echo areas are basically like, let's talk about some of the more interesting topics available on them . . .

Of primary interest to most of us, as Heath users, is the Heath/Zenith Echo Area (HZ_ECHO). The Heath/Zenith echo is carried from coast-to-coast (from Clarkston University and the Capitol Heath Users' Group on the East Coast to Bourbon Street West in Southern California), as well as a number of other BBS's in between.

Callers on the H/Z echo are constantly discussing new product releases, problem solutions, the best means of performing complex tasks, and most anything else that interests them. There are a large number of very knowledgeable (and friendly) systems gurus present, and the information available is downright irreplaceable. Best of all, as we've mentioned above, it's free, and it's all just for us!

The chef in the family might find the Cooking echo (COOKING) of more interest, however. Imagine an area where 50-60 new recipes appear each week — many of them from the better restaurants. Interested in Cajun cooking? Just ask for some Cajun tips, and you might well find yourself talking with the owner of the nice little restaurant just off Bourbon Street in the French Quarter! Do you want to know the best seafood restaurant for that vacation in Florida? Why not talk to the locals who frequent them?

One thing's for sure, your menu won't be lacking when the cook in the family gets tied into the Cooking echo!

Are you looking for a great deal on some used gear? Why not try out the National For Sale Echo (that is, if you can handle reading 300 new messages per week)? Rather deal in your back yard where you can actually see the gear? Most BBS's also carry a local For Sale echo for just this purpose.

Do you wonder what the "big brains" are thinking? One of the echos (MENSA) provides access to Mensa members (the sole qualification for Mensa is an IQ score in the upper 2% of the population) — and their conversations will surprise you! These guys and gals may have gifted IQ's, but they're far from professorly types (I'd recommend you limit the kids' access to this echo — some of the discussion topics are definitely NOT for young kids!)

Are you interested in working with the handicapped? There is an echo specifically for this purpose. How about data base programming? There are echoes for each of the major software products. Networking? There are echoes not only for end-users of networking products, but also for the very product developers themselves!

If you get the idea that the information available on these systems is rather large, then you're getting the idea. Remember, I've only discussed about 2% of all the common echoes available!

How Do I Get Started?

As mentioned above, the first thing you need to do is to get yourself a modem and software so that you can connect with one of these BBS systems yourself. If you're in a populous area, I'd recommend a 1200 or 2400 baud modem. If you're in a remote location, you might want to consider a faster modem, unless you're aware of a local connection in your local calling area.

The next step is to identify one or more local BBS's tied into the Fidonet network. One way to do this is simply to ask other local computer users. Another way, however, is simply to dial such a system and use one of the echoes to ask for phone numbers of these BBS's in your area. If you don't know of a local BBS where you can do so, please feel free to call our First Capitol Computer BBS at (314) 928-9228 to do so. You might want to make the request on the H/Z Echo, so that you can check to see which of the local BBS's carry the echo at the same time.

Once you identify the closest BBS carrying the services you're most interested in, all you have to do is call that system and use its services. In most cases, you'll be able to locate a BBS within your calling area, and be able to use it at no additional charge whatsoever. If not, you'll still be ahead of the game since even long-distance charges are generally cheaper than connect time on the commercial BBS's — and new BBS's are coming online all the time.

The first time you enter (log onto) one of these BBS's, you'll have to enter some basic information about yourself (your real name, address, and phone number are the usual questions), and until the sysop has had time to review this information, your use of the system will be somewhat limited. Actually, these questions are to protect you and the other BBS users.

By insisting on verifiable user information, the sysop is able to confirm that you are who you say you are. This makes it more difficult for a troublemaker to leave messages under your name. If a particular caller creates problems on one BBS system, the sysops are able to lock him out of not only that system, but to warn other sysops as well. Thus, the BBS's have a built-in quality control system to weed out destructive users. This helps to insure that message transmissions are reliable and in reasonably good taste.

What Can I Do to Help Fidonet Grow?

As a new participant, the most important thing that you can do is to be friendly, and join the discussions actively. Don't be nervous! Echos are frequented by a wide spectrum of users, from seasoned professionals to rank amateurs. There's room for everyone. Each new user adds his own personal touch to the whole. The more callers, the better the system becomes. Additionally, more callers means more BBS's, which means more locations for free access for everyone to the network.

If you're in an area where the H/Z Echo is currently not available, but Fidonet compatible BBS's are — then you can perform a great service for the Heath/Zenith community by asking the sysop to consider carrying the H/Z Echo on his system. You might well find yourself in contact with a number of other local H/Z users in your area, and will certainly find yourself in contact with thousands of others around the country.

Bulletin Board System Etiquette

One very important consideration is your behavior when accessing these BBS's. Sysops are public-minded individuals who are fronting the cost of your access as a kindness, and as volunteers experience everywhere, they often receive abuse at the hands of the very people they serve. They deserve your cooperation and support. Although there are thousands of systems around the country, the drop-out rate is very high — and this is largely because sysops are so often abused by their callers.

If you provide your children with access to your modem, take the responsibility to see to it that they utilize it in an adult fashion. Indeed, there are special features for children and young adults on some BBS's — but immature users sometimes are more interested in trying to break a system than to work with it. Although they are unlikely to suceed in this endeavor, their efforts interfere with other callers and are a constant source of irritation for sysops.

BBS's each have their own set of rules. If you disagree with these rules, you might want to — politely — explain your feelings to the sysop via private message, but ultimately the system is his and it is his responsibility to set and enforce the rules. In every case, there are reasons for such rules, and you are unlikely to find any that are really hard to live with. Certainly, the rules are less restrictive than virtually any pay-as-you-go service, since there is no profit motive in the Fidonet system. Should the local rules prove difficult, though, rather than attacking the system or the sysop — just switch to another system. With over 3,500 of them available, this should pose no problem.

This brings us to the topic of what BBS users call "flaming". Flaming a user means to send a message to that user which is in the form of an attack or argument. Flames are not allowed on most BBS's and certainly do more to tear down the system than to build it. If you persist in personal attacks on a BBS, you might not only find yourself unable to access that BBS, but also locked out of other BBS's as well (the power of electronic mail works for sysops as well).

Once you become comfortable with your local BBS system, you might find yourself in a position to set up your own system or interested in becoming more involved in the organization itself. Local BBS sysops are your best source for assistance in establishing a new BBS (indeed, most new BBS's are advertised on existing BBS's). If you want even more involvement, then you might want to consider joining IFNA, the International Fidonet Association. IFNA is a non-profit organization formed to establish policies and standards for the overall network. It also publishes a regular newsletter which is distributed in electronic form over the network. For more information, see the references at the end of this article.

Summary

Public bulletin board systems are a great source of new information, and a great way to meet others with interests the same as your own. (No matter WHAT those interests may be!) The cost is right, virtually free after you purchase your modem and software, and they are frequented by a number of very helpful and interesting individuals.

If you're not currently tied into at least one of the thousands of public bulletin board systems in the country, you're missing a lot of fun and excitement, not to mention missing out on making a lot of new friends!

References

A. BBS's Known to Carry the National Heath/Zenith Echo. At the time of publication, the following systems were known to carry the National Heath/Zenith Users' Echo. Other systems may (indeed, certainly do) exist, and these numbers may change at any time (systems are listed in telephone number order):

(202) 797-7236 Washington, DC
or
(703) 339-7330 Northern Virginia
Capitol Heath Users' Group (CHUG) Fido
Runs on a Z-158 with Zenith modem and 120 megabytes of online storage. Operates 24 hours per day from a location in Lorton, Virginia. This BBS is an arm of CHUG, the largest independent Heath/Zenith Users' Group in the country.

(301) 598-8248
The Black Cat's Shack

System is a PC-Designs 386 with HST modem and 140 megabyte hard drive. Runs 24 hours per day from a location in Silver Springs, Maryland. Sysop's name is Joe Keenan. Of special interest to programmers.

(314) 291-8653
St. Louis HUG BBS
System is a Z-248 with 80 megabyte drive and US Robotics 2400 modem. Open to members 24 hours per day, to non-members from 11:30PM CST to 5PM CST (non prime time). Sysop is Rich Dobyns and location is Bridgeton, Missouri.

(314) 928-9228
First Capitol Computer BBS System
Runs on a First Capitol CL-XF clone with 38 megabyte drive and US Robotics HST modem. Operates 24 hours per day. Sysop's name is Terry Mueller. Of special interest to First Capitol customers, and people who just plain enjoy good conversation. (Also acts as the main distribution hub for the First Capitol Computer Disk-Based Catalog.)

(315) 268-7993
Clarkson Alumni BBS
System is a Z-100 computer with a 10 megabyte hard disk drive and Hayes 1200B modem. Sysops are Jesse Sherman and Laurel Goolden. Operates 24 hours per day from its location in Potsdam, New York.

(505) 873-0783
The Paradise Lost BBS
Uses a Z-100 computer "and other system components cast down from paradise" with 8 MHz, 768k RAM, and 60 megabyte hard disk. Runs 24 hours per day in Albuquerque, New Mexico.

(714) 447-1623
AnaHUG
Equipment varies, but includes more than 80 megabytes of hard disk space and a 2400 baud modem. The sysop's name is John Secor and the board is of special interest to PC compatible users.

(714) 638-2298
FOG (Fifty-One Group)
Runs on a Z-100 computer with four 40 megabyte drives and an HST modem. Operates 24 hours per day, and caters to amateur radio operators. The sysop is Jim Ward.

(714) 642-0561
Bourbon Street West
One of the (two) founders of the Heath/Zenith Echo, Craig Armstrong (the sysop) is a transplant from New Orleans to California. Craig is a proficient programmer and all-around maniac (What can I say? He's a personal friend, so he MUST be a maniac!) His system is a Z-100 computer with two 40 megabyte drives and HST modem. Operating hours are 24 hours per day.

(719) 591-4582
No name given
This Colorado Springs BBS is run by one of the two founders of the Heath/Zenith Echo, Joe Rock (Joe actually started a version of this echo seveal years ago, and restarted it with Craig Armstrong later on). Runs on a Z-100 computer with 8 MHz clock, 1.5 megabytes of RAM, two 45 megabyte drives, and US Robotics HST modem. Operates 24 hours per day. This system is of special interest to budding genealogists.

(904) 651-3875
Jolly Green Giant BBS
Operates on a 768k Z-100 computer with a pair of 44 megabyte drives. Runs 24 hours per day. Sysop's name is Steve Marcussen, and system runs from Shalimar, Florida. Steve is a long-time supporter of the Heath/Zenith Echo, and was kind enough to compile the list this reference was compiled from.

(904) 651-8684
The *HOT MUDDY DUCK* BBS
Runs on an AT compatible with a 40 megabyte drive. Operates 24 hours per day in Shalimar, Florida. Sysop's name is Lloyd Wood. Of special interest to amateur radio operators.

B. The International Fidonet Association (IFNA)
For more information, contact:
Int'l. Fidonet Association (IFNA)
P.O. Box 41143
St. Louis, Missouri 63141
(314) 576-4067
Dues (should you decide to join) are $25 per year.

C. Credits

Thanks to Craig Armstrong and Joe Rock for founding the National Heath/Zenith Users' Echo.

Thanks to Terry Mueller of the McDonnell-Douglas Recreational Computer Club (and our own BBS sysop) for first getting me interested in the Fidonet system.

Thanks to Steve Marcussen for compiling the list of BBS's carrying the H/Z echo, and making it available to us for publication.

And thanks to the thousands of under-praised and over-worked BBS operators around the country who spend so much time without sleep that the rest of us might enjoy the fruits of their labors. You guys are tops!

D. Copyrights and Trademarks
This article is Copyright © 1989 by Tom Jorgenson, and the publication you are currently reading it in. It may be reproduced in unmodified form, providing credit is given the author (as shown in the article) and the publication in which it appears, and we encourage you to do so to spread the word to others!

GEnie, Compuserve Information System, Source, and BIX (BYTE Information Exchange) are trademarks of their respective companies.

Fido and FidoNet are registered trademarks of Tom Jennings of Fido Software, San Francisco, California.

Heath, Zenith Data Systems, Z-100, and other Z-phrases are trademarks of Heath Company and Zenith Data Systems.

Heath Users' Group and HUG are trademarks of the International Heath Users' Group. ✱

Interfacing to the IBM PC

Lynwood H. Wilson
2160 James Canyon
Boulder, CO 80302

One of the reasons for the popularity of the IBM PC and the other computers built to the same standard is the open hardware architecture. The expansion slots built into the back of the PC allow foreign hardware to gain access to all the necessary signals to actually become part of the host computer, instead of just being an accessory like a printer. This bus (called the system bus) offers such complete access to the host that a plug-in board can even carry another microprocessor which can take over complete control of the computer and usurp the position of the computer's own CPU. Because of the power and versatility of this interface, an IBM PC may be made to perform functions that could not have been anticipated by its designers.

If you want to do something with a PC that requires new hardware, you can quite probably find someone who makes a board that will solve your problem. If your application is an uncommon one, or if you just like to hack the hardware, you might consider building your own board. In this article, I will explain how you can build simple parallel input and output ports on an expansion board. This is a good beginning for designing more complex custom boards, since it involves decoding address and control lines, sending and receiving data, and all with the correct timing. It is also a good project in its own right, since it will allow you to use the PC to get information about the real world through the input port and control things outside the machine through the output port.

We will begin with a look at the lines in the PC system bus that we will use. I will not cover all the lines, only those we need to build parallel 8 bit input and output ports.

Addressing

The 8088 processor can use 16 address lines (of the 20 available for addressing memory) to select input or output ports, and so can address 65,536 of each. The PC, however, only decodes address lines A0-A9 for addressing I/O ports and so can have only 1024 or 400H of them. (The address range is 0-3FFH.) Furthermore, only I/O port addresses in the range 200 HEX to 3FF HEX are valid on the system bus and so only addresses in this range can be used on cards plugged into the expansion slots. Because the high 6 address lines are not used in the hardware to address I/O ports, the addresses will wrap around after 3FFH. For instance, if you write a program that tries to address a port at 400H, the port at address 0 will be selected. This is because 400H is 0000 0100 0000 0000 in binary and that looks exactly like zero if you only look at the low 10 bits. Likewise, if you attempt to address a port at location 5B3H, you will get one at 1B3H if there is one there. When the 9th address line is low, which is to say the address is less than 200H, the main board cannot receive data from the accessory cards. This is not so, you can output to an address lower than 200H on a card if you don't conflict with ports on the main board. Just restrict port addresses on cards to 200H-3FFH.

IBM set aside a block of I/O port addresses to be used for prototype cards. These are addresses 300H-31FH. These addresses should be unused and available for our experiments on any PC compatible. It is possible, of course, that a manufacturer of a computer or an accessory card has used one or more of these addresses for his own purposes so watch for interference. If anything odd happens when you talk to your experimental card, try changing its address.

The Lines We Will Use

Note that an asterisk (*) after the name of a signal indicates that it is active low, that its normal state is high and it goes low when it is true.

A0 Through A9 (Address Lines)

All of the 20 address lines, A0-A19, are available on the system bus connectors, but we will only be using the first ten of them since the PC only uses ten to address I/O.

D0 Through D7 (Data Lines)

The 8088 is considered (by IBM at least) to be a 16 bit microprocessor since it has 16 bit registers and internal data paths, but it has only an 8 bit external data bus. This is it.

AEN (Address Enable)

This is a signal that comes from the DMA system to shut down the CPU so that another device can take over the machine. On the expansion cards, it is used to disable the I/O ports to prevent erroneous I/O during a DMA cycle.

IOR* (I/O Read)

This line signals that the processor is reading data from an input port, and that the address lines 0-9 contain the address of the port to be read. The signal is active low. The port in question must respond by placing its data on the data bus. In order for the processor to be able to read the data placed on the data bus by the input port, it must be stable for 30 nanoseconds before the IOR line goes high, indicating the end of the read cycle. This delay is necessary to ensure that the CPU has time to accept the data.

IOW* (I/O Write)

This is another active low signal that indicates that the processor is writing to an output port. Address lines 0-9 contain the address of the port to be written to. The data bus contains the data to be written. However, this data may not be valid yet at the time that IOW goes low, so the data must be latched into the output port on the rising edge of the IOW signal.

RESET DRV (Reset Driver)

This line is high (active) during power-up. It will also go high if any power voltage gets outside of specifications during operation, or upon hardware reset (if your machine has a hardware reset). It should be used to clear (set to 0) all output ports that control things that should not be turned on at random when the machine is turned on or reset. The ports may otherwise come up with any random data. The 74LS273 is a good choice for an 8 bit latch with a clear line.

Timing

When the computer begins an I/O port write cycle, it first puts the address of the port on the address lines. Then it asserts the IOW* line. Then it puts the data to be written onto the data bus. There follows a short pause, for the output port to recognize its address and take in the data. This pause is lengthened by a wait state which the computer automatically generates during all I/O cycles. Then the IOW* line becomes false (high). At that time the port must latch in the data. Slightly after that, the data and the address are no longer valid and the next cycle begins.

An I/O port read cycle is similar except, of course, for the direction of the data flow. First the computer sends out the address of the port to be read. Shortly thereafter it asserts the IOR* line. Then there is a pause (including a wait state) to give the port time to recognize its address and put its data on the data bus. Then the computer latches in the data and the IOR* line goes false.

The critical point is the time at which the IOR* or IOW* goes false. In an output cycle, the port expects the data to be valid at that time so the port can latch it. Further, it expects that the data has been good long enough that the latch has had time to accept it. Similarly, in an input cycle the computer expects that when it makes the IOR* line false the data has been good long enough that the CPU can latch it in at that time.

There are several potential timing problems. If the address decoder is slow, its output may not be valid before the IOR* or IOW* (whichever is the case) is valid. Since the address bits may not all arrive at exactly the same time (as each other), this situation may result in the wrong address being decoded and data written to the wrong port address. (Note, however, that the data will also be written to the correct address as soon as the decoder settles down.) The worst case delay permitted in the address decode is 92 nanoseconds. This is the maximum time allowed between the time when the address on the address bus is valid, and the time when the output of the address decoder is valid.

At the end of the bus cycle it is possible for a fast address decode to beat a slow IOW* such that a write is made to an address decoded from the next cycle. In order to prevent this, the IOW* delay must be less than 200 nanoseconds. This is the delay between the time that the IOW* goes high and the time that the port latches the data.

Actually, however, the IOW* must be faster than 120 nanoseconds or the data on the data bus may no longer be valid when it is latched into the port. Also, a slow IOR* can reduce the read access time for the input port.

This may all sound pretty complex, but these constraints are really quite easy to meet. If you use LS chips and not too many layers of them, you will have no trouble. The circuits described here give plenty of margin.

Bus Loading and Driving

The drive (voltage and current) available on signals coming from the expansion bus and the drive required of signals going to the expansion bus can be fairly complex to determine. In the spirit of this article, I will only say that if you buffer the lines you are using, consisting in our examples of the data lines, the first ten address lines, and the RESET, AEN, IOW* and IOR* lines, you will have no difficulty with drive. Each line from the PC is only driving one TTL LS input, and the lines driven from our card (the data bus) come through the 74LS245 which has plenty of output.

Construction Techniques

There are, of course, many ways to build an experimental electronics project. If you are going to build a large number of the same expansion card, you can design a printed circuit board and have it etched and drilled. This process costs more in small quantity than large, but a company that does a lot of small runs can make you a few of a special board for less than you might think. In any case, we are talking here about making one experimental circuit and there are better ways. Several people sell prototype cards for the IBM suitable for point to point wiring or wire wrapping which make it all much simpler. These are the two most common ways to do this kind of prototype construction, and as always they each have their advantages and disadvantages.

Wire wrapping is easy to do, after only a little practice. It is reliable, and easy to change and modify (and fix if you or I should make a mistake). The sockets are more expensive than standard ones though, and the pins stick out far enough that a wire wrap board needs extra room in the chassis. It will normally take up two spaces.

Hard wiring is just as reliable, again after a little practice. It is compact and just a bit cheaper. On the other hand, it is slightly harder to modify and since we are building an experimental circuit to try things on, I wire wrapped mine.

The JDR Board

The board I used for the circuits described in this article came from JDR Microdevices and I found it quite satisfactory. JDR sells two such boards. One has an area laid out for buffers and address decoders with traces and places for the chips. You need only solder in the chips and a few resistors and capacitors and you have all that is necessary to connect your I/O device to the IBM. You can then concentrate on the unique part of your circuit. The rest of the board is a grid of holes with solder pads so that you can put whatever components you like almost anywhere. There are ground and power (+5) lines run all around the edges of the board, so that you are never far from them. The board includes a metal end bracket to fit the PC, with a cutout for a 37 pin D connector. It also has a pattern of holes for a D connector with right-angle PC board leads. The decoding furnished on this board is intended for I/O so if you want to build something in the line of a memory board (or memory mapped I/O), you might want to start with their other card which just has a grid of holes and pads, for you to start from scratch. Both of these cards give you access to all the signals on the bus through card edge fingers that fit the connectors in the PC and traces to bring the lines out to plated through holes and solder pads. These are equally suitable for soldered wires or wire wrap pins. And all the chips and lines are identified on both sides of the board, which is a very useful touch. Take note, however, that the polarity of the signals is not identified in the printed legends on the board. The IOR* line is simply designated IOR, for instance. Of course, you do not need to use the same board I did. There are several other good boards available, with various features.

Principles of Operation of the JDR Board

On the schematic of the JDR board, Figure 1, is shown the decoding circuitry as provided by JDR. I have omitted the bypass capacitors from the schematic. The JDR board provides for 5 bypass capacitors for the 7 TTL chips and they are quite close, in most cases, between the chips. They also provide for two 10 uF caps, one at each end of the edge connector where the 5v lines enter the board. If you build this circuit from scratch, do thou likewise. The lines shown down the left side of the schematic represent signals from the system bus. All of the other signals from the bus are also brought onto the card, and terminated in plated through holes so they can be easily used if your circuit should need them. The ones shown on this schematic are the ones used by the circuit provided by JDR. The lines down the right side of the schematic are the signals that are buffered or generated by this circuit for the user's circuits. These all terminate at plated through

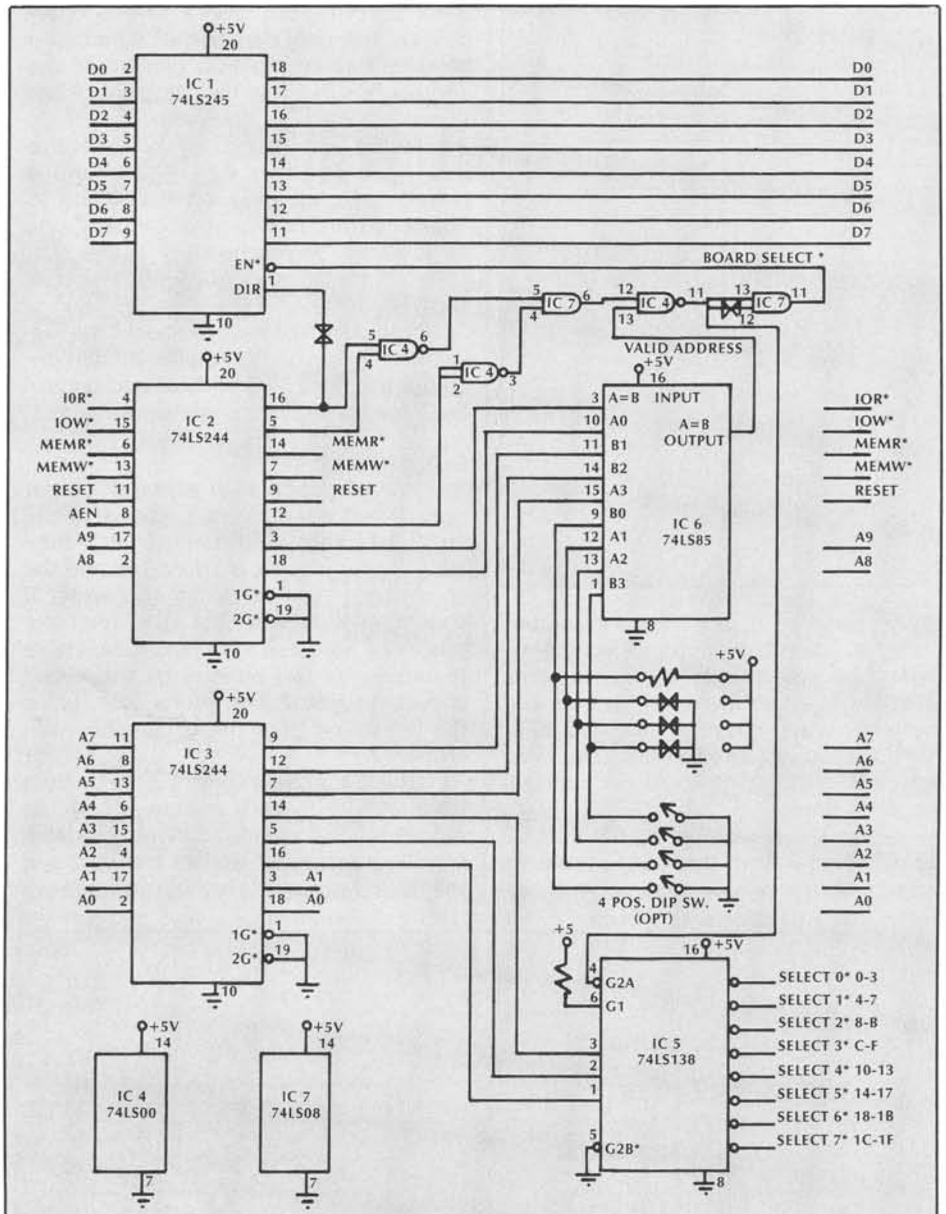


Figure 1
The JDR Board Buffers and Decoders

holes for wire wrap pins or soldered wires. All the lines are labeled, although the labels on the edge connector seem to have shrunk just a bit. They get slightly out of register with the holes at one end, but it is not a problem.

The data lines enter the card through a bi-directional bus transceiver (IC 1, a 74LS245). Its direction pin is controlled by the IOR* signal, so that the transceiver can only drive the data bus in the computer when the computer is attempting to read an input port. At other times, the transceiver brings data from the computer onto the card. The output of this transceiver (regardless of direction) is enabled by a collection of AND and NAND gates that produce a BOARD SELECT* signal.

The buffered IOR* and IOW* lines run into a NAND gate whose output will be high only when the computer is calling for an I/O operation, either read or write. This signal goes to the input of an AND gate along with AEN* (inverted from AEN by a NAND gate with its inputs tied together) and the output of that gate will be high during an I/O operation which is not a DMA operation. It is in turn sent into the input of a NAND gate along with the VALID ADDRESS and the output is BOARD SELECT*. BOARD SELECT* will be low when and only when either the IOR* or IOW* is asserted, and the AEN is not asserted, and the address lines A5-A9 show an address which is valid for this board. In other words, for our board to be connected to the data bus, the system bus must be requesting an I/O operation, either read or write, the AEN line must be indicating that there is no DMA operation in progress, and the address bus must be addressing this board. When these conditions are met, the '245 will pass 8 bits of data from our card to the system bus, or from the system bus to our card, as appropriate.

There is another AND gate in the BOARD SELECT* line which has no effect on the signal since its inputs are tied together. By cutting the trace in the bow-tie mark that runs to pin 12 of IC 7, you can further control the board selection. For example, if you were building two of these boards and you wanted each of them to respond to its own half of the address space, you could drive this pin with address line A4 on one board and use an inverter to drive the pin with A4* on the other.

ICs 2 and 3, 74LS244 octal buffers, carry signals that always originate in the computer and come to the card, so they are not bi-directional circuits and their outputs are permanently enabled by tying the G1 and G2 pins to ground. Most of the signals that pass through these buffers are used by the decoding circuits, but some are just buffered for our use.

The high five address lines, A5 through A9 are decoded by IC 6, a 74LS85 four bit digital comparator. A9, which must be high to address an expansion board, runs to the A=B input of the '85. This input is intended for use when stacking '85s so as to be able to compare words larger than 4 bits, and is used here as a sort of enable. Regardless of the other inputs, the '85 cannot find that A=B if the lower order '85 stacked with it does not find that its inputs are equal. For that reason, if A9 and, therefore, the A=B input are low, the A=B output must be low. If A9 is high, then the '85 compares the inputs A0-A3 to inputs B0-B3, and if they are the same, it asserts the A=B output. The comparison of A and B is not completely straightforward, since the A and B lines on the comparator are connected to the address lines and the lines from the address selector circuit with a fine disregard for which is which. Address bit A8 is connected to A0 on the '85, address bit A7 goes to B1, A6 to B2, and A5 to A3. The other inputs, B0, A1, A2, and B3, come from the address selector. This scramble was probably done to simplify the board layout. It doesn't matter because we are using only the A=B output, and the word on the A inputs will only equal the word on the B inputs if each bit matches its opposite number, A0=B0, A1=B1, etc.

The four bits that the address lines A5-A8 are compared to come from the address selector circuit, which can at your option be either a pull up resistor and three grounded lines or a four section dip switch and four pull up resistors. If you build the board without the switch, the four bits from the address selector will be 1000. Since we have seen that A9 must be high and we now see that A8-A5 must be 1000, the address coming from the com-

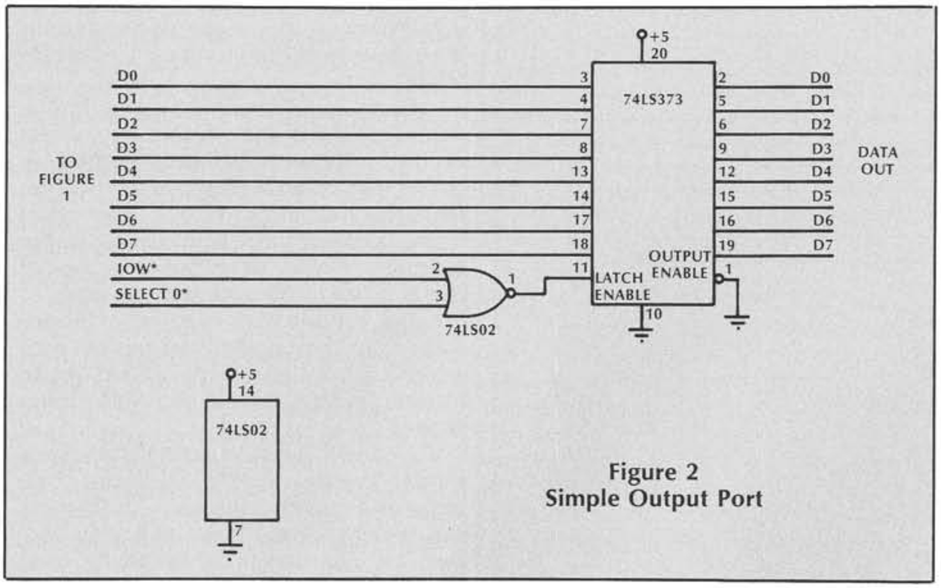


Figure 2
Simple Output Port

puter must be between 300H and 31FH (11 0000 0000 - 11 0001 1111 in binary) to assert the A=B output on the 74LS85. When this line is high, it signifies that the address on the address bus is one of the block of 32 (decimal) addresses that are valid on this board.

This block of addresses is set aside for prototype cards on all IBM PCs from the Junior to the AT. (Beware of timing problems if you use this circuit in an AT, however, due to its higher speed. See the IBM PC-AT Technical Manual for ideas.)

If you install the dip switch and extra pull up resistors, you will have to also cut the three traces shown on the schematic and on the board as little bow ties. These are on the solder side of the board, under the dip switch. Then you can set the board address anywhere from 200H to 3FFH, in blocks of 20H (32 decimal).

IC 5, the 74LS138, is enabled by the BOARD SELECT signal, and decodes address lines A2-A4. Since A0 and A1 are not decoded, the select lines from IC 5 each represent a block of four addresses. SELECT 0*, for instance, will be low when the host computer is addressing port 300H, or 301H, or 302H, or 303H. Many devices such as the 8255 programmable peripheral interface and the 8253 programmable interval timer represent 4 separate addresses, and decode the address lines A1 and A0 within themselves. For simpler ports such as 8 bit latches, it is perfectly legitimate to enable them with these SELECT* lines, even though they can then be called by four different addresses. This has the advantage of being simple and requiring fewer chips, and the disadvantage of wasting address space. The wasted address space will only be a problem if you have more than eight devices on your board, in which case the address bits 0 and 1 can be further decoded by any of several techniques. We will see some of them soon. You can, of course, build an input port and an output port that share the same address. They cannot be selected at the same time, since one is active when IOR* is true, and the other when IOW* is true, and IOR* and IOW* cannot be true during the same bus cycle. You cannot read and write at the same time.

Since IC 5 is enabled by the BOARD SELECT* signal from the gates above, its outputs therefore represent not only address decodes, but device selects. They indicate not only that one of a particular block of four addresses is present on the address bus, but also that the control bus is calling for an I/O operation.

Before we leave this circuit, notice that there are two AND gates and a NAND gate left over on ICs 4 and 7. These can be used in your circuit, if you need them. You can easily solder thin wires to the pins of the chips or their sockets.

Now that we have buffered the signals coming on and off our board and decoded the necessary address and control lines, we are ready to build some ports.

Output Port

We will begin with a simple output port. It will need to have access to the eight data lines, and it must latch their data in only when it is addressed and the computer is calling for an I/O write. It should then output that data until the process is repeated with new data. These requirements can be met by the circuit shown in Figure 2. The lines leading off to the left come from the circuit shown in Figure 1.

The 74LS373 is an 8 bit transparent latch. While its latch enable is high, its output follows the input. When the latch enable goes low, it latches the data and the outputs continue to put out whatever

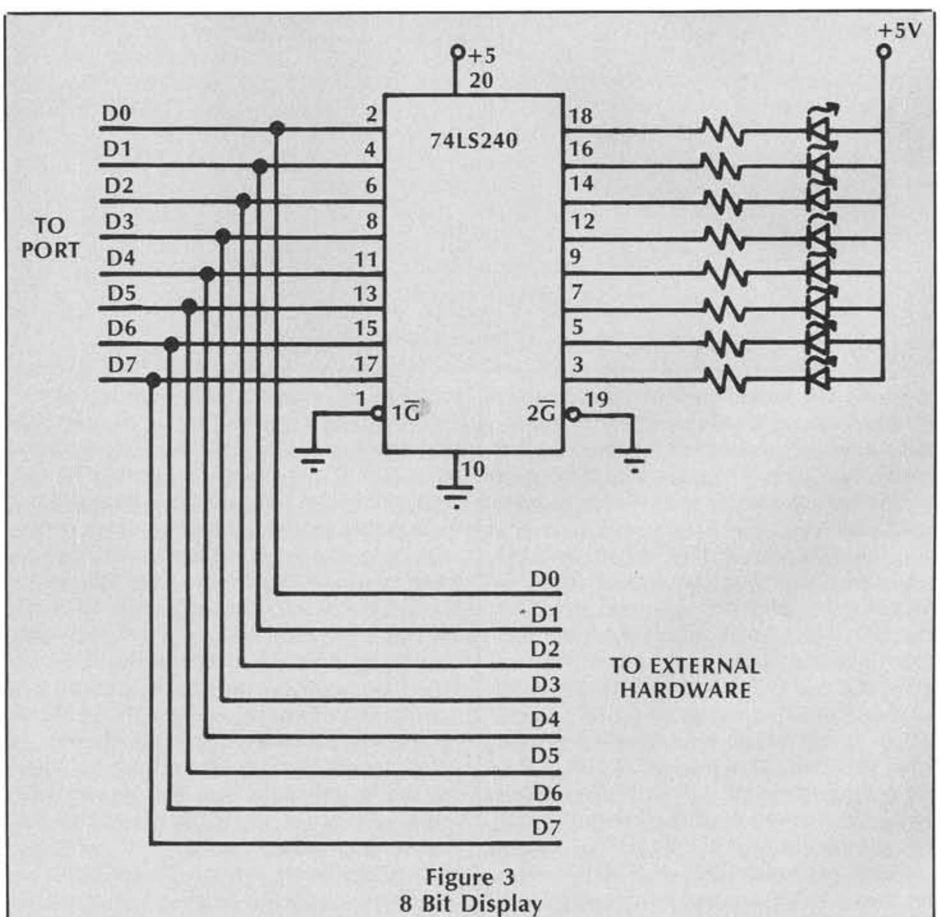


Figure 3
8 Bit Display

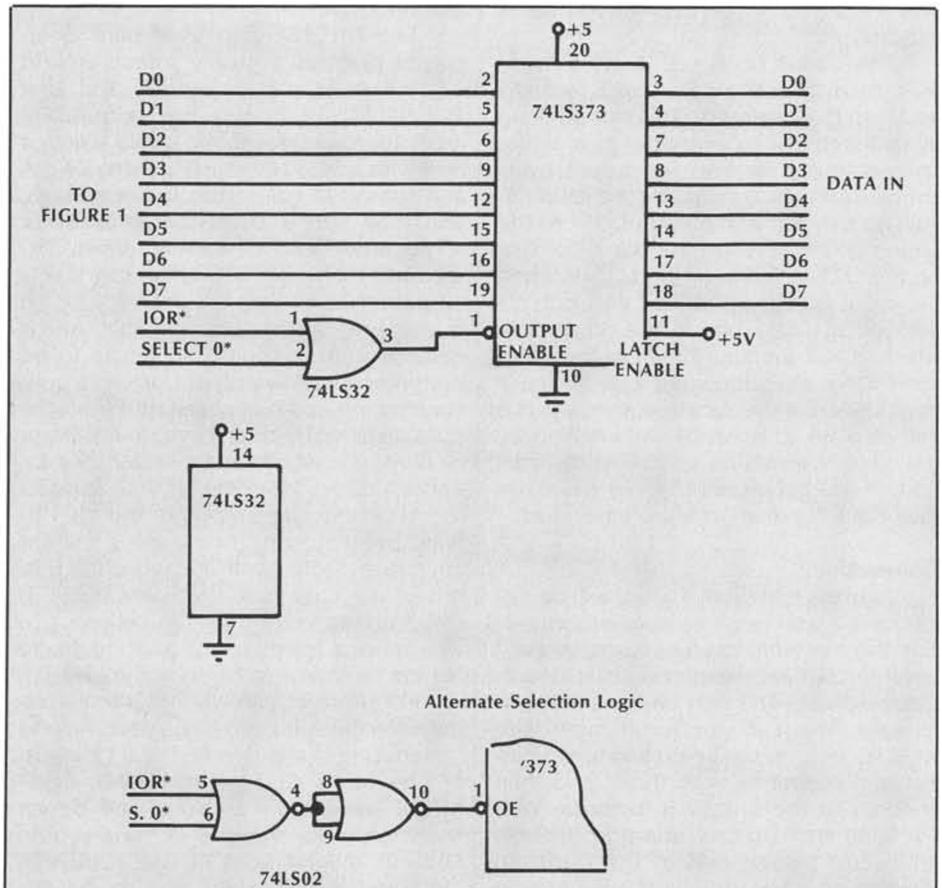


Figure 4
Simple Input Port

the inputs had on them when the latch enable went low. If the output enable is held high, the output pins on the '373 go to high impedance. When our board is addressed and IOW* goes low, IC 1 on Figure 1 will pass the data to the 74LS373 in Figure 2. If the address is in the range 300H-303H, the SELECT 0* line will go low. The combination of IOW* and SELECT 0* will cause the output of the NOR gate to go high, which will enable the '373 to accept the data. The IOW* will go high and the '373 will latch the data while both the data and the address remain valid. The output enable, pin 1, can be run out to the device that is getting the data if it needs to be able to control the output of the port.

This port fills all of our requirements and is a real, useful device. To test it, we could wire its output to a socket and construct a readout of 8 LEDs and their current limiting resistors on an appropriate plug, or we could just check the output lines with a meter. Alternatively, we could provide a permanent display of LEDs driven by a 74LS240 (or a 74S240 if you need more drive), as shown in Figure 3. Note that the LEDs are turned on by low outputs from the '240, and that the '240 inverts its inputs. Thus the LEDs light when the output from the port is high. This is much easier to keep track of than the situation where a lit LED means a low output. It is as important not to design confusion into the hardware as it is with the software, although not as widely recognized.

I have found that a permanent display is extremely useful in testing and debugging, not only the port itself but also the external circuitry and the software. It's nice to know exactly what is going on.

Input Port

We can use another 74LS373 to build an input port, although the problem here is slightly different. An input port should be accessible to data from outside at all times. When the port is addressed by the computer, it must put the data on the data bus. This can be accomplished with a '373 by turning it around so that its outputs face the computer data bus. See Figure 4. The latch enable on the '373 is tied high, and the output enable is driven by IOR* and SELECT* through an OR gate so that the output is enabled only when both IOR* and SELECT* are true, which is low. This means that the input port only drives the data bus when the computer asks it to. Just as it must. The latch enable may be brought out to the device that drives the port, if it needs to be able to latch its data into the port. This could happen if the outside device only generated good data occasionally and the data had to be latched into the 74LS373 and then wait for the computer to call for it. This would be an unusual case, however.

The circuit in Figure 3 can be used as a display for the data coming into an input port. Merely connect the data lines on the left to the input port in Figure 4 and the right facing data lines to the external hardware. The indicator will display the data coming from the outside, and will change as it changes. Remember that this data gets into the computer only when the

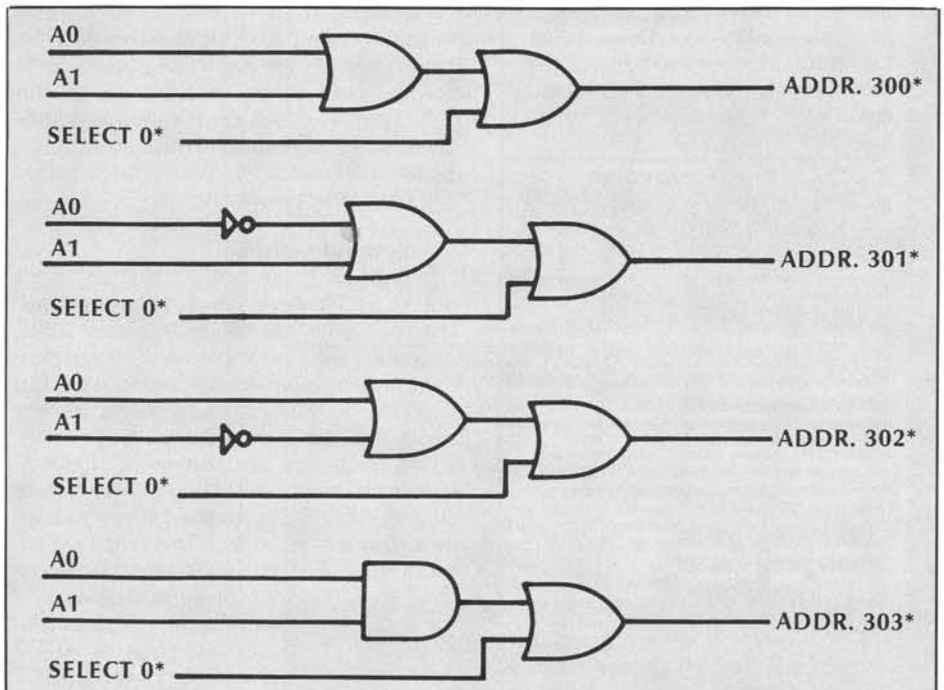


Figure 5

port is read, so not all of what you see is relevant.

This circuit on Figure 3 can also be used to display the data that is actually read into the computer. This can be done by connecting it to an output port as described in the section on output ports above, and then connecting the latch enable on the '373 in the output port to the same signal that drives the output enable on the '373 in the input port. Thus when the input port is addressed and puts its data on the data bus of the other port, which is still functionally an output port even if we are addressing it as an input port, latches in the data from the data bus and displays it. Because it is enabled by the same signal that enables the input port, it will latch (and display) the same data the CPU read from the input port.

Connections

In many cases, your board will be self contained, and need no connection outside the machine. In other cases, you will need to connect these new ports to outside hardware. This may be done in several ways. You can put some more wire wrap sockets on the board and run the input and output lines to them, and then connect to the outboard hardware with DIP jumpers. You can mount a 37 pin D connector on the end of the card and connect to it through wire wrap pins in the holes provided. The D connector offers the advantage that it is mechanically stronger, and less likely to wreak havoc on your board when the wires get pulled. The dip sockets are convenient and cheap, and if you wire them properly, you can plug an 8 section dip switch into your input port and a DIP header with 8 LEDs and their current limiting resistors into the output port for testing. Wiring the sockets like this isn't a very efficient use of the pins, however, and on balance I recommend the D connector. Fortunately, it is a 37 pin connector, so you cannot forget and plug your printer into it.

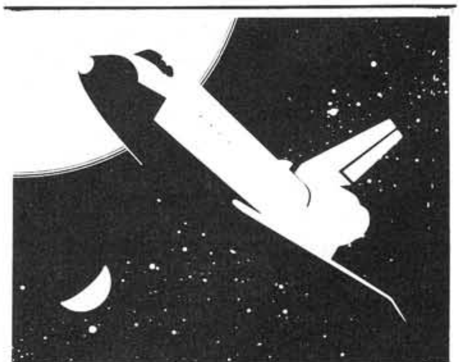
Absolute Addressing

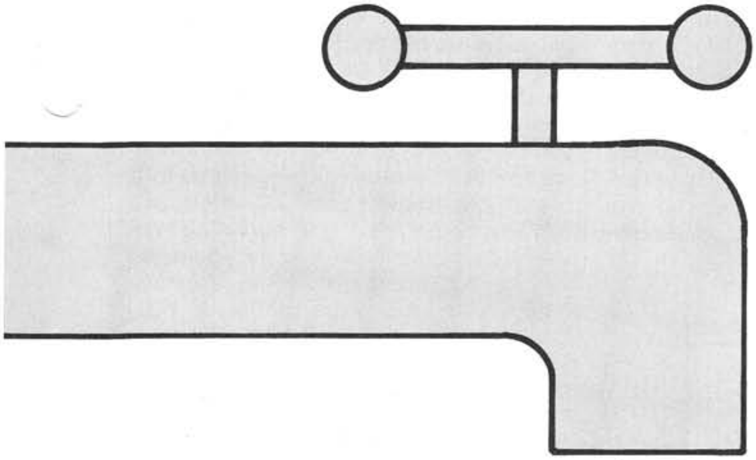
Both of these ports are enabled by the SELECT*0 line, which represents addresses 300H, 301H, 302H, and 303H. This will work fine for most of the experimental circuits we are discussing here, but if you need absolute addressing for any reason, it can be accomplished easily. Some examples are shown in Figure 5. (There are lots of other ways to do this, of course, depending on the gates you happen to have left over.) The two low address lines are decoded to select between the four addresses represented by each of the select lines from Figure 1. The output of any of these circuits, such as ADDR 300*, can be used in place of the SELECT 0* line in Figures 2 and 4. For higher addresses, use the higher numbered SELECT lines from Figure 1.

Output Drive

The 74LS373 chip used here as an output port will typically source around 2.5 mA from a high output, and sink around 24 mA from a low output. For more sourced current you could follow a '373 with a 74LS240 which will sink 24 mA and source 15 mA. If this is not enough, you may add a transistor amplifier or some other kind of current driver. Remember that most TTL chips can sink a lot more current than they can source, so if you are driving anything that needs much current, it should be set up to be turned on by a low output from your port. For example, an LED connected from the output of an LS chip to ground will not work very well, since there will not be enough drive. However, if you connect the LED from the output of the chip to +5v, (remembering of course a current limiting resistor), it will light when the output of the chip goes low. (See Figure 3) Don't use an output with an LED on it to also drive a logic circuit, as it could be held at an invalid voltage by the LED.

For those of you who wish to go further into the subject, I highly recommend "Interfacing to the IBM Personal Computer" by Lewis C. Eggebrecht. Mr. Eggebrecht was system architect and design team leader for the IBM PC, and should know if anyone does. It was written in 1983 and describes the first model PC, but the bus definition hasn't changed even though the computer has. *





Even the Kitchen Sink

Part 3

Richard J. O'Connor
848 Fenske Drive N.E.
Olympia, WA 98506

Using the Disk Caching, Security and Maintenance Tools of PC Tools Deluxe, Version 5

Introduction

After reading about all of the functions included in the PC Shell, PC Format and Desktop Manager programs of PC Tools Deluxe, you may think the kitchen sink has already been included! What else could I possibly need, you might ask?

Look at it this way: eating is fun, doing dishes is not. Working on your own computer is fun (once it has you properly trained, that is!), but the chores related to security and maintenance of your work are not. Those of us who have worked on mainframe computers in the past now appreciate the operations staff who made sure our system was secure and our files protected from accidental harm. But your personal computer is your own responsibility, unless you can afford to retain paid staff to do your file and disk maintenance for you! (If so, show THEM this article.)

Maybe the remaining programs in the PC Tools Deluxe package should be called the "Sink" Tools. For the most part, they aren't as glamorous as the ones we've covered to date — have you ever invited a friend over to watch a file backup program do its thing? But they make the job of cleaning up and protecting your work environment much easier, so that you can start off your next session with confidence that you need not fear ominous messages such as "File Not Found!"

Getting Started

By way of review, I run PC Tools on my home computer, a Z-159 with two standard floppy disk drives, a 30 MB Seagate ST238R hard disk and 1 MB of RAM. I have an Epson FX-86e printer and I use the SNAPSHOT program that comes with PFS First Publisher to capture and print the screens you'll see as Figures. One disadvantage to SNAPSHOT is that it will not capture an entire EGA screen. Be aware that many of the "full-screen" shots included as illustrations in this article are missing the last 8-10 characters on the right. There's enough there to serve the purpose of illustration, though. Please don't think that PC Tools is responsible for the partial words you might see!

The programs I'll cover in this third installment include PC-Cache (a disk-caching program), PC Secure (a file encryption and compression tool), PC Backup (a hard disk backup utility), Compress (a file and disk defragmenter), and MIRROR/REBUILD (programs that save hard disk partition and File Allocation Table information). In the three-manual documentation set supplied with Version 5.1 of PC Tools Deluxe, PC Backup is described in the "Hard Disk Backup" manual, and the other programs listed above are described along with PC Shell and PC Format in the "Data Recovery and DOS Utilities" manual (the Desktop Manager also has its own manual).

One of the first things you notice about these programs is that they use different interfaces. For example, PC-Cache uses a command-driven interface, while PC Backup uses the full-screen drop-down menu approach.

The reasoning is obvious: utilities like PC-Cache don't have many options and are designed for you to "set it and forget it." Others present you with several possible choices to make, and so the PC Shell-style screen is used for familiarity. After playing with PC Shell for awhile, I learned the tricks of maneuvering through such screens, which made learning subsequent PC Tools Deluxe programs much easier. Perhaps software designers are finally learning that consistency in interface is highly prized by users — even when we forget to write in and say thank you!

Using the PC-Cache Disk Caching Program

A "disk cache" sounds like a secret storage place to hide your blank floppies, and in our office, that would be a pretty good idea! In reality, of course, a disk cache is a reserved portion of your computer's memory where information most recently read from disk is stored. Why is this a useful tool?

Your computer processes data much faster than your disks (hard or floppy) can provide it. This is due in part to the fact that disk drives are electro-mechanical in nature, and their speed is dependent upon several internal moving parts. However, data *can* be read very quickly from your computer's random access memory (RAM), since RAM chips are electronic devices without moving parts.

Studies have found that most common personal computer applications read disk data in sequential fashion, usually requiring the next few consecutive kilobytes for each subsequent "read." Experimentation revealed that placing a relatively small amount of data read from disk into RAM (often 64K or less) eliminated many disk accesses and the delay associated with them. The same was found to be true for disk writing, but "write caches" are not as popular as "read caches." The reason is that data written to a cache reside in RAM until the cache fills; only then are they written to a disk drive. If your computer loses power before this process is completed, you will lose any information remaining in the cache.

PC-Cache is a "read cache", and all disk writes go directly to disk to avoid any accidental data loss. You can set up read caches for up to four floppy disk drives and four hard drives during installation. One very nice thing about PC-Cache is that the RAM dedicated to the cache(s) can come from conventional memory (below the magic 640K mark), expanded memory (memory above 640K that follows the LIM standard), or extended memory (memory above 1MB on an AT-style machine). My Z-159 has 384K of expanded memory, so I placed my disk cache there to save conventional memory for other applications. I chose to only cache my hard disk, since I don't read a lot of data from floppy disks during a typical session. By placing the command PC-CACHE /IA /IB in my AUTOEXEC.BAT file, I load PC-Cache and have it Ignore drives A: and B:, which effectively caches only drive C: on my machine.

Normally, you install PC-Cache at the time you first install the PC Tools package, using the installation program PCSETUP. Since PCSETUP detected the presence of expanded memory on my machine, it offered to place most of the cache in expanded memory. I entered the PC-CACHE/PARAMS command at the MS-DOS prompt and found that PCSETUP had built a cache using 6K of conventional memory and 192K of expanded memory, which was half of the 384K expanded memory I have. You can use other parameters on the PC-Cache command line to regulate the size of the cache you build, and what memory it uses. I was slightly confused, because the PCSETUP program and the manual led me to believe the default settings would consume 64K of conventional memory rather than the 6K actually used. I'm confused, but I'm not unhappy!

You can check your new timings with the PC-CACHE/MEASURES command, which totals the logical disk transfers, the actual physical disk transfers, and the number and percent of physical transfers saved by using the cache since bootup or your last PC-CACHE/MEASURES command. For example, at the end of a long session working with the second article in this series, in which I popped into MS-DOS from my word processor regularly to test new features and check settings, I logged 3,108 logical transfers, which actually required only 1,689 physical transfers, saving 1,419 transfers, or 45% (see Figure 1). A quick check just after the start of this word processing session found 64% of the first 263 logical transfers saved. These disk access savings often translate into real time saved in terms we can all understand; during the interval between the press of the <RETURN> key and your machine's response.

```
C:\>pc-cache/measures

   PC-CACHE, Version 5.1
   Unauthorized duplication prohibited.
   The PC-CACHE program is already installed.
   Measurements are as follows:
          3,108  logical transfers.
          1,689  physical transfers.
          1,419  transfers saved.
             45  percent saved.

C:\>
```

Figure 1

Using the PC Secure Program

If you've ever seen a file with the extension .ARC, it's likely you've experienced compressed files. File compression is a collection of techniques that map common characters to smaller internal representations and replace runs of repeated characters with counts in order to save disk storage space. There are several approaches to data compression, each most appropriate under certain circumstances. However, you can use data and file compression quite easily without knowing the specific techniques used.

PC Secure is a program that provides both compression and encryption of the data within disk files to increase their security. PC Secure is not a standalone compression program, such as System Enhancement Associates' ARC, Phil Katz's PKARC/PKPAK/PKZIP series, or Rahul Dhesi's ZOO. PC Secure is primarily a tool for encryption, or internal recoding, of sensitive data, which provides compression capability in the process if so desired.

Encryption of data is done following the federal DES standard, which results in a theoretically unbreakable code, resistant to decryption without the file password or Master key established by the encrypting party. If you are interested in more details about the DES block cipher itself, the manual sketches how encryption is done to DES standards in a section titled "Understanding PC Secure." If you live outside of the United States, you may find the encryption/decryption options removed from your copy of PC Secure, leaving only the file compression capabilities. Data encryption methods are considered sensitive topics, and as such, are subject to federal export regulations. The rest of us are free to explore all of the capabilties of PC Secure.

Just as you can over-protect a young child, so you can overdo file security unless you keep in mind a few simple guidelines. PC Secure gives you two ways to decipher files that you've encrypted, but their effective use requires a little discipline (oh no, not that!) on your part. First, you can decrypt a file using the file password you assigned during encryption. If you forget your file's password, you can use the second method, the Master Key approach. PC Secure will prompt you for a Master Key the first time you use it, and this key can be used as a "backdoor" entry to decrypt any file you subsequently encrypt. At the risk of belaboring the obvious, please keep in mind that IF YOU FORGET BOTH YOUR FILE'S PASSWORD AND YOUR MASTER KEY, THE INFORMATION IN THE FILE CANNOT BE RETURNED TO A USABLE STATE. The PC Tools folks use a marvelous term in their manual; the "electronic paper shredder!"

The manual steps you through the details to consider when choosing effective passwords and Master Keys. You can choose to use either a 5-32 character, case-sensitive alphanumeric key or a 16 character hexadecimal key for even greater security against random guessing by a person trying to break your code. Now, admittedly, the need for this much security may not come up around your house very often (unless you have curious young children about). However, you may have situations at work where this will come in handy. Let's look at an example of file compression, encryption and decryption so you can see how simple it is. The uses I'll leave to your imagination!

Start the PC Secure program by typing PCSECURE (all one word) at the MS-DOS prompt and pressing <RETURN>. The first time you run PC Secure, a screen appears, prompting you to enter your Backdoor Master key. Since blanks and punctuation can be a part of an alphanumeric key, you can enter a word, a phrase, or a complete sentence, if you wish. You'll be asked to re-enter the Key for verification, and then the PC Secure screen appears.

Use the mouse or keyboard keys to select Encrypt, and a list of files in the current directory appears. Once you choose the file you wish to encrypt, you're prompted to enter a file password for this specific file. Again, you'll need to enter this twice so that the program is satisifed that you'll remember what you entered! At this point, the Progress screen appears, which uses simple graphics to illustrate how far along PC Secure is in the encryption of your file. Options that you choose will directly affect the time encryption takes; you can choose to compress at the same time, to run a full 16 rounds of DES encryption (rather than the default 2 rounds), and so on.

I chose a 35K file which contained the text for article 2 of this series as an example. Encryption took about ten seconds, and the compression reduced the file to 17K. (I'm not sure if that's an accurate measure of my Hot Air Index, but no matter.) When the encryption was finished, I chose Exit, and then About from the main PC Secure menu to see the "before and after" file statistics related to this encryption run (see Figure 2). As you would expect, looking at the encrypted file reminded me of the screenful of electronic pablum I get when our two-year-old gets to the keyboard while my back is turned!

Decryption is also straightforward, and takes you through a similar menu progession in restoring your file. You are required to enter your file password twice, and after an exact match, PC Secure determines if you've entered a "bad" password or the correct one. Decryption then begins, and takes a comparable amount of time. To my relief, restoration was complete, and I could read my text file again. Note that the default operation of PC Secure encrypts a copy of the original file, deletes the original, and stores the encrypted copy under the original name. If this is a little too spooky for you, the Delete Original File option can be turned off, so that encrypted files are given the extension .SEC. You can also set up PC Secure to make all encrypted files "hidden" from MS-DOS. Should you ever forget that you did this, you'll be grateful you have PC Shell around to display them again for you!

Using PC Backup

It's apparent that Central Point Software is quite proud of the Backup and Restore programs packaged with PC Tools Deluxe. PC Backup has its own 66 page manual (Version 5.1) and is packed with features. Let's walk through a typical backup session and look at just some of the possibilities.

When you execute the PC Backup program (type PCBACKUP and press <RETURN> from the MS-DOS prompt), the information on your hard and floppy disk drives is catalogued before the work screen appears (see Figure 3). This screen is very similar to the PC Shell screen in appearance, with a directory tree for the default drive displayed on the left-hand side and a list of the files in the current directory on the right-hand side. A helpful status

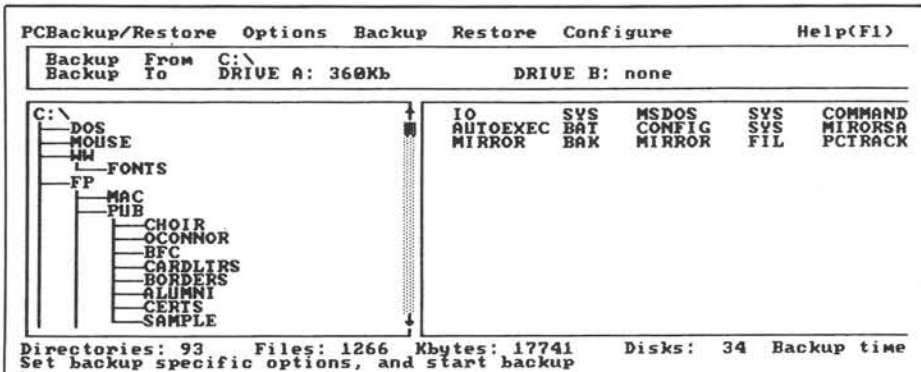


Figure 3

line at the bottom of the screen indicates the total number of directories, files, and size (in kilobytes) for the selected drive. In addition, the estimated number of diskettes and time it would take to back up these files (assuming formatted diskettes were available) is shown. These numbers will vary depending on the backup options you set, as you'll see shortly.

The menu bar at the top of the screen allows you to select Options, Configure your session, Backup and Restore files. Under Options, you can choose Quick Compression of files (which only does a small amount of data compression in order to avoid any speed penalty), a fuller Compression (which requires less diskettes but takes more time), and Verification of the backup. You can choose to display the time elapsed during backup or restore (this is quite handy in practice), to be warned of possible file overwriting, and to save or load a setup, which is a set of options specifically created and named by you. The Configure menu allows you to define the disk drive types to back up to, to reset foreground and background colors in any of the onscreen windows, and to save the current settings as the new default.

The Backup and Restore menus are slightly more complicated, for here is where you select the files or directories to work with and the source and destination drives involved. To perform a backup, you choose a "From" entry, a "To" entry, and a backup "type," before choosing the files to back up and starting the process.

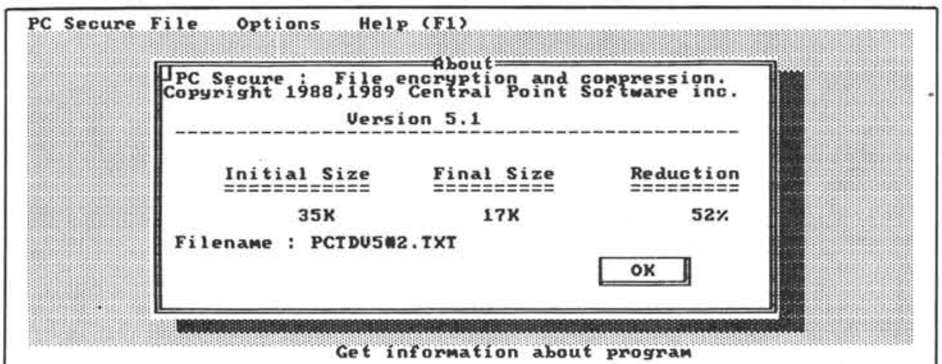


Figure 2

Backup "type" includes both Mode (Full, Full continual, Archive, and Continual) and Method (DMA or DOS compatible). The manual goes into useful detail on modes, including suggestion for setting up regular daily and weekly backups to keep your most recent work safe and secure. The DMA (Direct Memory Access) method uses the DMA controller present in many machines to speed the backup process by reading from the source drive and writing to the destination drive simultaneously. Disks created using the DMA method cannot be read by MS-DOS, and so a special program, PCBDIR.COM, is provided to read the directories created for you. If you don't have a DMA controller, PC Backup will warn you and suggest you use the DOS compatible option, which is slower but more reliable in such machines.

Buoyed with confidence now, let's entrust some files to PC Backup and see what happens! I selected a subdirectory on the hard disk of my Z-159 named COMM which had ProComm communications software files, PC-Kermit files, and a few pieces of miscellany for a total of 19 files and 339K of data. It took 30 seconds to format a blank 360K disk and 55 seconds to use PC Magazine's SWEEP utility to copy the subdirectory, including subdirectories \COMM\FILES and \COMM\PCKERMIT, for a total of 1 minute, 25 seconds. [Note: I'm still running MS-DOS Version 3.1. You more modern folks at 3.21 and above could use the XCOPY command to do this.] I then tried several combinations of PC backup modes and options with the following results:

DMA method:
Quick compress, unformatted disk 1:13
DMA method:
Full compress, unformatted disk 1:15
DMA method:
Full compress, formatted disk 1:16
DOS method:
Full compress, formatted disk 1:29

Two interesting points emerged during the testing. First, PC Backup can't find a DMA controller in my Z-159. I chose the "Do High Speed Anyway And Let Me Suffer The Possible Consequences" option,

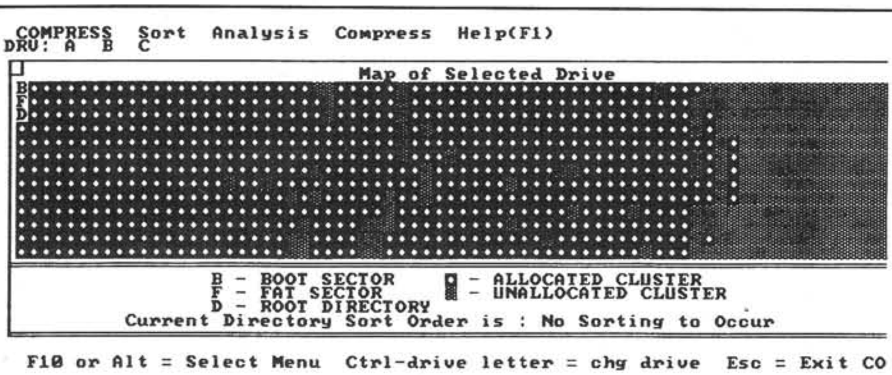


Figure 4

and it turned out to have the greatest impact on backup speed. The other is that the estimated times are only a few seconds shorter than the actual times, and the difference may be due to the fact that PC Backup seems to spend an inordinate amount of time writing a directory to the backup disk once backup is finished. It just "feels" like too long a period of time, relative to the time backup itself takes. These were all single disk trials, and perhaps that additional time wouldn't seem as long on a multi-volume backup. I definitely have one of these coming up, since PC Backup informs me that it will take 55 diskettes and 16 minutes to backup my entire hard disk. Well, maybe the next time diskettes go on sale . . .

Restoring those same files was much quicker. After selecting the Restore To and Restore From options from the Restore menu, I chose DMA method, overrode the warning message again, and found that restore took only 52 seconds. Of course, I'm brushing past a lot of capability here, as you can easily select certain files or subdirectories to restore, rather than doing a full restore. The point is that PC Backup works as I would like such a program to work; it's intuitive, powerful, configurable, and quick.

Using Compress

Up until now, I've used the term "compress" to mean "squeezing data within files to reduce storage space." The PC Tools Compress program is an application designed to improve the performance of your hard disk. In this sense, "compress" means to rearrange hard disk files so that they are contained in single contiguous areas of the disk, which decreases seek time when the files are accessed.

Hopefully, this dual use of the word Compress won't confuse you. This process is often referred to as "unfragmenting" or "defragmenting." See what Central Point *could* have named the program?

How do your files get fragmented in the first place? Because files can be deleted and extended in size on a storage device. When a file N bytes long has been deleted, MS-DOS will use that space for the first N bytes of the next file saved. Similarly, when you replace a file with a larger version, MS-DOS skips over occupied sections of the disk to find the next free space to write those extra bytes. The operating system keeps track of where the "pieces" of your files are, but excessive "scattering" of files slows down access, and thus your response time.

The best way to get a feel for the fragmentation of your hard disk is to see it yourself. When you start Compress (type COMPRESS and press <RETURN> at the MS-DOS prompt), you'll see a Disk Map that resembles Figure 4. As you can see, I'm not plagued with bad sectors yet on this hard disk, but there has been a little file fragmentation out near the end of the allocated clusters. So what do I do to clean this up?

The first step is to use the Disk analysis command on the Analysis pulldown menu. This analysis will reveal bad clusters and fragmented file chains, along with any cross-linked file chains, unattached file clusters or bad clusters within file chains. These last three are best corrected by the MS-DOS command CHKDSK/F before you proceed with Compress.

[As an aside, I wouldn't know about the Disk analysis command if I hadn't tried it on a Z-248/12 at work, where it pops up with the summary table in just a few seconds. When I run Compress on my Z-159 with Seagate ST238R 30 mb hard disk and Seagate controller, the Disk analysis command pops up the colored box for the summary table and then locks up my machine! I can run a Disk analysis on a floppy disk with no problems, but not on my hard disk. I've contacted Central Point Software about this; I'm not really certain at this point if the problem is in the software or in my hardware. All of the other options and tools in the Compress program have worked well so far.]

You can get a closer look at just which files are fragmented and to what extent by selecting the File analysis command. The screen that pops up displays the names, number of clusters, number of file areas, and percent of fragmentation for each of the files in the current directory. Finally, you can run a surface analysis to find and lock out marginal clusters that may go bad and fail to store data properly in the near future. The surface analysis done by Compress is very complete. This means that it can take several hours, and is an excellent way for your machine to spend the wee hours of the night when only hamsters and assembly language programmers are awake.

Next, you can choose a sort from the Sort menu to re-order your directories, if you wish. Directories will be sorted during compression, and you can sort them by date, name, extension or size in either ascending or descending order. Now it's time to select the Compress menu and make your final choices before compression.

There are three compression techniques to choose from; Unfragment only (which will unfragment files but leave free disk spaces scattered about your disk), Full Compression (which will unfragment files and move them to the front of the disk) and Full Compression with Clear (which also erases data left in sectors unused after file or subdirectory deletion). Once you choose a technique, you have the choice of Standard file ordering (which lets Compress decide where to put your files and runs the fastest), DOS file ordering (subdirectory files are kept together), or COM/EXE order, which places all executable files first. If you run executable files often and don't change

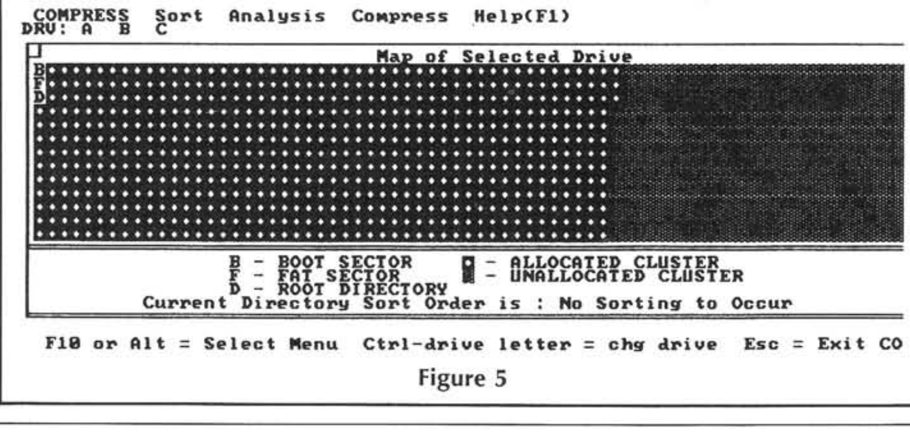


Figure 5

them much, this option may give you better hard disk performance by lowering the average amount of seek time the system needs to find those files.

Now you're ready to Compress — if needed. Choose Analyze Disk Organization at this point, and the program will let you know if compression is recommended, based on the condition of your hard disk and the options you've selected. If you decide to proceed, simply select Begin COMPRESS, choose CONTINUE in the subsequent Warning Box. The Disk Map will reappear, allowing you to watch as Compress re-organizes your hard disk. Summary statistics are collected by Compress during the re-organization, and if you select the Print Option prior to starting the compression, you can view this report either on the printer or in a disk file, at your option.

When I selected Analyze Disk Organization, it suggested that I run Compress, which was no surprise. I chose to sort my subdirectories by name in ascending order, selected Full Compression with standard file ordering, and decided to skip the printed report. When I clicked on Begin Compress, the program responded with one more bright red WARNING box to explain what I was about to do, in case I hadn't been paying attention. I clicked on Continue, turned off the monitor, and called it a night. The next morning, the screen informed me that Compress had taken 1 hour, 20 minutes and that everything worked just fine. From the appearance of the Disk Map (see Figure 5), I trust that it must have. After Compress finishes, you will be asked if you would like to run Mirror again (it's a good idea) and then suggests that you re-boot your machine. At that point, I had a tidy hard disk and neatly sorted files; not bad for a night's "work!"

Using Mirror and Rebuild

The last of the programs I'll describe in the PC Tools Deluxe package are the "safety net" programs, Mirror and Rebuild. Mirror is a program that takes a snapshot of the File Allocation Table (FAT) and the root directory of your hard disk. This information is kept in a hidden file on your disk, and can be used by the Rebuild program to rescue your files from the disaster of inadvertent ERASEing or FORMATting of your hard disk.

The manual uses a lot of boldface ink to stress that Rebuild is not to be used casually. In particular, files that have been created or changed since the last time Mirror was run will be lost forever if you Rebuild your hard disk. Rebuild creates a new FAT and root directory based on what it finds on your hard disk if you haven't been running Mirror, so it's a very good idea to get into the Mirror habit.

If you installed PC Tools Deluxe using the PCSETUP program, a line was added to your AUTOEXEC.BAT file to run Mirror on your hard disk, and also to initiate Delete Tracking for that disk. In this way, a record is kept of the most recent N file deletions you made (where N is a number you can set but defaults to 202 for a 30 mb hard disk) so that you can run Undelete to recover any such files whose space hasn't been overwritten. You now have a fresh copy of your FAT and root directory made at the start of each session, along with a backup copy of the previous MIRROR file. The manual gives examples of setting up batch jobs to run Mirror after other applications as well. The number of safety nets you need to install is entirely up to you. Mirror runs quickly and unobtrusively, so it won't get annoying if you decide to use it often in "self-defense."

Mirror has a special option invoked by typing MIRROR/PARTN and pressing <RETURN> at the MS-DOS prompt. This will store a copy of your hard disk partition table on a floppy disk file called PARTNSAV.FIL, which you then store in a safe place. Should your partition table ever get corrupted, you can run REBUILD/PARTN to restore the table and make your hard disk usable again. PCSETUP will prompt you to do this during installation, and it's a worthwhile step to take.

I haven't run Rebuild yet, and I hope I won't have to. I view the Mirror and Rebuild programs the way I view fire extinguishers: knowing where they are and how to use them is just as valuable and less burdensome than practicing with them every day.

Final Thoughts

This wraps up our three-part "guided tour" of PC Tools Deluxe, Version 5 (and 5.1!). Although there are certainly a few aspects that need improvement, I feel that this is the most comprehensive and useful toolbox I've ever seen for the MS-DOS environment. There's no question that you get a tremendous amount of value for your money. If even a small subset of the capabilities provided here can make your sessions more effective, it's worth your time to take a closer look at PC Tools Deluxe.

As I write this, Central Point has started deliveries of Version 5.5, which is a major extension of Version 5.1 containing a host of new features and a new (higher!) price. Are the new features worth the retail price of $129 or the upgrade price of $20? Have the folks at Central Point Software addressed my earlier concerns and fixed the bugs I came across?

I received my copy last week, and these are the things I'm curious about. If I find enough interesting aspects, perhaps you'll see one more article to bring you up to date on this most recent release. I know it supports IBM and Novell Token Ring networks, which may interest some of you. Until then, I'd be happy to answer any questions you have about PC Tools Deluxe. Send your questions either to me or to Buggin' HUG.

Till then, see you in the "Kitchen!"

Products Discussed

PC Tools Deluxe, Version 5, Version 5.1
Central Point Software, Inc.
15220 N.W. Greenbrier Parkway, #200
Beaverton, OR 97006

PAYLOAD * PAYLOAD * PAYLOAD * PAYLOAD

Seagate HARD DRIVES

MODEL	CAPACITY/FORMAT/SPEED/SIZE	DRIVE ONLY	XT KIT
* ST-125	21 MEG / MFM / 40 MS / 3.5"	$228.00	$278.00
* ST-138	32 MEG / MFM / 40 MS / 3.5"	$271.00	$321.00
* ST-138-1	32 MEG / MFM / 28 MS / 3.5"	$296.00	$346.00
* ST-151	42 MEG / MFM / 24 MS / 3.5"	$380.00	$430.00
* ST-138R	32 MEG / RLL / 40 MS / 3.5"	$263.00	$323.00
* ST-157R	49 MEG / RLL / 40 MS / 3.5"	$312.00	$372.00
* ST-225	21 MEG / MFM / 65 MS / 5.25"	$199.00	$249.00
* ST-251-1	42 MEG / MFM / 28 MS / 5.25"	$324.00	$374.00
* ST-4096	80 MEG / MFM / 28 MS / 5.25" FH	$548.00	$598.00
* ST-238R	32 MEG / RLL / 65 MS / 5.25"	$216.00	$266.00
* ST-277R-1	65 MEG / RLL / 28 MS / 5.25"	$366.00	$426.00
* ST4144R	122 MEG / RLL / 28 MS / 5.25" FH	$628.00	$688.00

*** 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,	$282.00
* ST-138/Z150 Kit	32 Meg, 40 MS,	$325.00
* ST-151/Z150 Kit	42 Meg, 24 MS,	$434.00

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,	$358.00
* ST-138/Z148 Kit	32 Meg, 40 MS,	$399.00
* ST-151/Z148 Kit	42 Meg, 24 MS,	$499.00

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 $99.00 External Model $128.00 Cable for External Modem $8.50

VISA · DISCOVER · MasterCard

PAYLOAD

15718 SYLVAN LAKE * HOUSTON TX 77062

FLOPPY DISK DRIVES

⇨MITSUBISHI MF501	5.25" 48 TPI DS/DD 320K/360K	$ 79.00
⇨MIITSUBISHI MF504	5.25" High Density 360K/1.2 MEG	$ 92.00
⇨ MITSUBISHI M-353	3.5" in 5.25" frame 720K	$ 94.00
⇨MITSUBISHI M-355	3.5" in 5.25" frame 1.44 MEG	$105.00

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

PAYLOAD CUSTOM ASSEMBLED COMPUTERS

⇨We assemble 8088 XT, 80286 AT or 80386 IBM compatible computers to your specifications. Please write or call for a work-up sheet showing items available and prices. We can use many existing parts and take others as trade-in from your Z-100 to reduce your costs.

VIDEO MONITORS

⇨ZCM-1490	ZENITH Color Flat Screen VGA	$ 679.00
⇨MA2565	SAMSUNG Amber TTL 720x350	$ 89.00
⇨CW4644	SAMSUNG Color RGB 640x200	$ 274.00
⇨CM4531	SAMSUNG Color EGA 640x350	$ 389.00
⇨CN4551	SAMSUNG Multi-sync VGA 800x560	$ 489.00
⇨CM1440	SEIKO VGA 1024x768 .25 dot	$ 609.00

VIDEO CARDS

⇨EGA350	PARADISE AUTOSWITCH 640x350	$ 99.00
⇨EGA480	PARADISE AUTOSWITCH 640x480	$ 109.00
⇨VGAPLUS	PARADISE AUTOSWITCH 800x600	$ 172.00
⇨VGAPLUS16	PARADISE AUTO 16 BIT 800x600	$ 214.00
⇨AUTOEGA STB SYSTEMS 640x350	$ 149.00	
⇨AUTO VGA STB SYSTEMS 800x600	$ 199.00	

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	100 ns	$ 2.65
⇨41256	256x1	120 ns	$ 2.30
⇨1Mbit	1Mx1	80 ns	$ 9.40
⇨SIMM	1Mx9	80 ns	$94.00

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 $217.00

⇨Dual Drive Unit $309.00 Bare drive and cable for internal mount $127.00

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

UCI Memory Upgrade Pal Chip Set

⇨For the Z-100's with the newer motherboard part number 181-4918 or greater. This chip set allows the installation of 256K RAM chips on the motherboard. With the addition of 27 256K 150 ns RAM chips (sold separately) a total memory of 768K is obtained. PAL Chip Set $64.00

UCI EasyWin HARD DRIVE SYSTEM

⇨Complete Hard Disk System for mounting inside your Z-100. Includes S-100 bus board, matched XT hard disk controller, EasyWin software, manual and Misc installation hardware. Order a hard disk (ST-125 or ST-138 recommended) under the SEAGATE HARD DISK DRIVE ONLY listing to complete the kit.$288.00

CDR Z-100 SPEED MODULE

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

Z-100 SERIES SOFTWARE

⇨PART NUMBER /DESCRIPTION	LIST PRICE	SALE PRICE
⇨MS-463-1 Z-Basic (16 bit)	$175.00	$10.00
⇨MS-253-1Basic-80 (8-bit)	$175.00	$10.00

⇨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 8.0% state sales tax.

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

PAYLOAD * PAYLOAD * PAYLOAD * PAYLOAD

LOOK-AND-FEEL ISSUE, USER INTERFACES, FOXPRO 1.0, EISA AND MCA, 386SX

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

Because you can find all kinds of computer-related news in other publications, I generally stay away from reporting general kinds of news, except when it may have some impact on users of Heath/Zenith computers. Of course, any items that are specifically for Heath/Zenith computers are fair game, but I occasionally find out about general news items that I think you will find of interest.

Although many magazines typically report on the previous year in January, that usually omits something that was important which happened at the end of the year. For that reason, I usually take a look at some of the issues from the previous year in a summary that was actually *written* at the end of the year. And a couple of interesting things happened in late 1989 that may have a profound impact on software that many of us use. Perhaps the most profound impact will be the old look-and-feel issue that I originally mentioned in my March 1989 column.

The Look-and-Feel Issue

It seems that everyone is getting into the look-and-feel lawsuit business. The whole look-and-feel issue is basically a claim by a company that another company (a competitor) cannot have what appears to be the same user interface — including the screen display and command keys. From everything I've seen there have been no allegations that any program code was copied, which would be the basis for a suit on copyright infringement; rather it is the look-and-feel of the user interface. For example, Apple sued Digital Research over the use of the trashcan icon in the GEM Desktop. That was settled out of court, and DR changed the GEM Desktop. Apple also sued Microsoft because of the similarity between Windows and the Macintosh Finder program. Lotus takes a dim view of any competitor who attempts to display a user interface like 1-2-3. And the developers of the Crosstalk communications program went after a competitor who used a Crosstalk-like interface, not to mention the fact that Ashton-Tate got more than a little disturbed because they claimed that some of the FoxBase+ (Fox Software) screen displays were identical to those used in dBase. Ashton-Tate also claimed that the dBase language itself is also protected by copyright which, according to the modest research I have done, is contrary to currently accepted legal practices, customs, traditions, and laws. All of this is history which I mentioned last year. But there are a couple of news items about what is going on with Apple and Fox Software.

You may remember that Apple's position was that Microsoft Windows, and presumably the OS/2 Presentation Manager as well, has a look and feel like the Mac's Finder program, despite the fact that Microsoft had a license from Apple for the use of the interface in Windows. I still think that is really stretching a point because it is really difficult for me to see how the look and feel of the Finder program is similar to Windows, except for the fact that both are called Graphical User Interfaces, or GUIs (pronounced Gooeys) for short. Perhaps the fact that these programs are called GUIs explains why this problem has become something of a sticky wicket. But I made an observation last year that apparently has turned out to be fact.

Despite what many people believe, Apple did not actually "invent" the graphical user interface (later called the Finder program) used in the Lisa and Macintosh computers. The model upon which the Mac's Finder program was based was originally developed at the Xerox Palo Alto Research Center (called PARC, as in "park") for the Xerox Star system in the 1970s. One only needs to take a look at the Xerox Star computer system (I have seen it) to see the striking similarities between it and Finder. And I have mentioned before that it seemed strange that Apple could sue anyone because they did not originally develop the interface.

The real news is that Xerox filed a $150 million dollar lawsuit against Apple during the week of December 11, 1989, in the U.S. District Court in San Francisco. This lawsuit alleges that Apple unlawfully "borrowed" from the original Xerox Star interface. As part of this lawsuit, Xerox has asked the court to cancel any copyrights on the Lisa/Macintosh Finder software which has any resemblance to the Xerox Star system. Xerox has also asked the court to declare that it is the sole developer and owner of the GUI. And Xerox is asking for $100 million in royalty and license fees plus $50 million in damages. How very interesting!

Aside from the Apple-Microsoft suit, Apple also sued Hewlett Packard because of the similarity of HP's New Wave GUI that is based on Windows. To place this in the proper time perspective, both of these suits were filed about two years ago (early 1988 as I recall). Apple already lost a major battle in July 1989 because of a ruling that all but two of the visual displays in Windows are covered by the existing Apple/Microsoft license agreement. The two remaining points of contention involve overlapping windows and movable icons (pictures).

I've always thought that the Apple contention that they invented the GUI was ridiculous. Now that Xerox has finally entered the "picture", this whole situation goes from the ridiculous to the totally absurd. If Xerox wins, then Apple would not have the right to license the GUI in the first place, so I suppose that Apple would have to drop the suit against both Microsoft and Hewlett Packard for starters. Moreover, it seems to me that both Microsoft and Hewlett Packard would have a legitimate claim against Apple for starting a lawsuit without having legal ownership of the interface. No doubt the lawyers in both companies have "considered" that kind of move. And as you might expect, Apple's response to the Xerox suit is that the allegations are "completely unfounded" and that their works (i.e., Finder) are "wholly original." Although Apple has promised to contest the suit, I would really be disappointed in our legal system if Xerox did not win. Even though I'm not a lawyer, it seems to me that there is a significant amount of evidence which suggests that the Finder interface program — originally released on the Lisa in 1983 — is a remarkably striking imitation of the Star interface released by Xerox in June 1981.

One of the other interesting points about this lawsuit is that Xerox is alleging that Apple "misrepresented" the originality of the GUI during the copyright registration by failing to acknowledge that the GUI was based on the earlier Star system. In other words, Xerox is apparently NOT claiming that there were copyright infringements, probably because the Star system had been available for about five years by the time the Xerox copyright was finally registered. I can't help but wonder if part of the problem is the length of time that it takes to get final "approval" of copyrights and patents, especially for the fast-changing computer industry.

This whole look-and-feel issue has already had an effect on the computer industry. I won't speculate as to what might happen if Xerox wins this suit because you can guess at that as well as I can. Actually, it really does not matter much who wins this battle because the results will probably affect all computer users, regardless of what system they use. As consumers, we will get stuck paying for this battle as usual, and that's what bothers me the most. There are also some undercurrents related to this whole issue that may help decide the outcome.

Where Interfaces Came From

For all of us who use a PC compatible computer, it is interesting to explore the question of why the basic interface is different. There is really not much mystery about it because there are two basic ways to design an interface. What it looks like is a whole different discussion.

The first way is to use some kind of a menu system. These have been around for years and years, and they are typically found on a mainframe computer application. Their level of sophistication varies from the most simple to the most complex, such as Finder or Windows. I personally like the ProFinder software that is supplied with WordStar because I have found it the best for my needs. In particular, ProFinder provides the capability for me to add a 39- character file description for every file, so I don't have to remember exactly what the file contains. That particular feature is probably the most important to me because I have so many files on my hard drive. ProFinder includes a menu that can be easily customized to anyone's applications, using any commands that you can enter from a command line in a batch file kind of approach. ProFinder also works just fine as a DOS shell and has its own built-in functions that make it easy to perform just about any kind of file manipulation.

I have never cared much for Windows because I thought it was quite limited especially because I could never "attach" my own description of the contents of a file. And to complicate matters, Windows does not even include a TYPE or "view" command that allows us to quickly view the contents of a file. Besides, there has been an incredible lack of support for Windows-based software, and even Microsoft is just getting around to releasing a version of Word that can take advantage of Windows' features. In any case, a basic menu system can be created with a batch file, almost any decent DOS shell or it can be as sophisticated as Windows, Finder or GEM.

The second type of interface is command-line driven, like the basic PC compatible and virtually all of the older 8-bit systems, such as those which used the CP/M operating system (e.g., the H-8 and H-89). Actually, this interface has been used by data processing people on mainframes for years, which explains why virtually all computers prior to Apple's Lisa used it. Of course, it takes a while to learn the commands, but once you understand what is going on, it really is easy. That whole thing probably makes it quite clear why most of today's computers use the command-line as the basic user interface. And the fact that IBM chose the command-line interface for the original PC is probably a result of what most existing computer users were familiar with at the time. I suspect an additional part of the problem is that most of the early PCs were based on the 8088 MPU and had very limited display capabilities (i.e., CGA). The point is that Windows is very slow on these older systems, especially if only floppy disks are used, and their displays aren't that great either.

Even today, a lot of the Windows-based software is slow with a fast system, such as a Z-386, and a hard drive. For example, I have tried Microsoft's Windows-based Excel spreadsheet on my Z-386/16, and Borland's Quattro spreadsheet is noticeably faster on my system. I have also tried the Ami word processor, and it is considerably slower than the non-Windows-based WordStar 5.5 and Word 5.0 that I use most often. To be fair to Windows, I have also used Word 4.0 on a Mac II (yes, I know how to use a Macintosh), and it is even slower than Ami and some other Windows-based software (including Excel). The bottom line is that I am not impressed with these GUIs because the amount of program code (called "overhead") required to run them really slows down my system. And that is the price one has to pay for the "convenience" of using a GUI. I, for one, am not willing to pay it, at least until it does not slow me down and impact my work.

Historically, that's where the basic types of interfaces originated. The command-line interface is just a carry-over from mainframe computers. Menu-based systems have evolved from the simple DOS shells to the GUI in attempt to find an interface that makes a computer easier to use.

Fox Software

As I mentioned earlier, Ashton-Tate went after Fox Software because of the alleged similarities of some FoxBASE+ displays to dBASE, not to mention the alleged problem with the dBASE language itself. If you have been keeping track of what is going on, you probably know that Ashton-Tate has been steadily losing ground to a number of other data base programs, especially FoxBASE+, probably because many users are finding that competing software works better (and has more features) than dBASE and has fewer bugs. But once in a while I see something that is so spectacular that it deserves a few comments, and that spectacular new product is called FoxPro.

If you do any kind of heavy-duty data base work with dBASE-compatible software, you really should take a close look at Fox Software's new FoxPro. FoxPro 1.0 essentially has all of the same features as FoxBase+ (and the latest dBase versions), but it is a quantum leap forward in data base software and makes dBase look obsolete. FoxPro has all of the latest and greatest programming commands, including mouse support, and it has incredibly fast displays. And based on a quick count, FoxPro has well over 100 language enhancements that make programming a data base application MUCH easier, not to mention its advanced debugging capabilities. And of course, you can still use dBase files directly with FoxPro, even though the program automatically converts dBase files for use with FoxPro. Although FoxPro is really intended for serious data base applications that need, and can afford, its features, there is no question that FoxPro is something that all dBase programmers should see. If you want to see why I think FoxPro is such a dynamite program, call Fox software at (419) 874-0162 and ask for the free demonstration disk. After you have seen that demonstration, I think you will agree that it makes dBase look kind of sad.

One note of caution before you run out and spend all kinds of money for this new FoxPro software. The lead programmer of a company that has worked with FoxPro 1.0 for a few months tells me there are a number of bugs in the software. That is not too surprising because this is version 1.0, and version 1.0 of *ANY* software usually contains more bugs than you might expect. Still, Fox Software has an excellent reputation for fixing bugs, so I believe you can generally expect much better support than you will ever get from Ashton-Tate on a dBase product.

The 80286 Chip is Dead

Don't believe it. The 80286 Micro Processor Unit (MPU) is a long way from being dead. High-performance 286 chips have a significant price advantage over the 386 MPUs, and it is likely to remain that way for some time. And unless you are using your computer for MPU-intensive applications (e.g, CAD — Computer Aided Design), a 286 computer — the Z-241, Z-248 or Z-286 — will work fine. Even then, you can see some dramatic performance improvements by adding a numeric co-processor (i.e., 80287) to a 286 system, although how much improvement depends on the software. Software that can use a numeric co-processor, such as most spreadsheets and CAD software, can benefit by adding it.

There is little reason to upgrade to an 80386-based system, unless you plan to use 386-specific software, such as Windows/386 or Paradox/386. For most of the common applications, like Word Processing, I don't think it makes sense to spend the money for a newer system because most performance improvements that you can SEE are quite limited. In cases like this, I prefer to upgrade my systems based on things I can see, rather than some exotic benchmarks. Benchmarks are based on some factors that are usually irrelevant to the way I use my system, which is mostly used for word processing. Sure, I use spreadsheets (Quattro) and graphics (GEM) software, but I don't use them often enough to make it worthwhile to spend a lot of money to buy a new computer for those less-frequently used applications. My spreadsheets are fairly small, and they run just fine in 640 K, so I don't need a lot of memory for them either.

One of the biggest problems I see is with an individual who decides to buy a newer and faster system for data base processing. The important point to understand about data bases is that they are I/O-intensive, meaning that they do a LOT of disk reads (Input) and writes (Output). Sure, they use the computer's MPU, but most of the time required to process large data bases is spent for disk I/O, which means that a newer and faster computer will not speed up processing that much. I know several people who are extremely disappointed after spending a lot of money for 386 systems, only to find that the time for data base processing has only been reduced by 10% or so. For processing large data bases, it is much more cost effective to buy a faster hard drive, especially if you upgrade to a 25 ms unit from a 65 ms unit. In this example, you might reasonably expect to see the time reduced by 50% or more because of the faster hard drive.

There is one other advantage to the 286 systems, especially for Heath and Zenith computers. These systems all use the ISA (Industry Standard Architecture) bus, which is the one used on the original IBM PC. The real advantage is in the cost of hardware because you can use standard 16-bit cards (e.g., memory and video) in these systems, assuming the cards are designed to work at the clock speed — most are now.

Technically, the 286 uses a bus that provides a data width of 16 bits and uses a 24-bit memory address, so that it can access memory up to 16 megabytes. In contrast, 8088-based systems use a 20-bit memory address, which is where the one megabyte limitation comes from. And although the 286 can perform multitasking (e.g., the OS/2 operating system), it is not particularly efficient in doing so because the chip cannot switch easily between the real and protected mode. Without getting into all of the gory details as to what these modes are, suffice it to say that the real mode (sometimes called the DOS compatibility mode) is essentially the same as what you are used to in DOS, and protected mode is used for OS/2 multitasking.

These technical distinctions are important when you take a look at the 80386 and 386 SX MPUs.

Missing the Bus

The 80386 MPU uses a bus with a data width of 32 bits. Note that this is TWICE as wide as the ISA bus (16 bits). Because the original ISA bus standard did not specify how to double the size of the bus, many manufacturers, including Zenith and Compaq, came up with a "proprietary" bus that provided the required 32-bit data width when they originally released 386-based computers. Zenith's Z-386/16 is one example of that, and the proprietary bus structure is the reason that you must use Zenith memory boards in order to have the system work at the 32-bit data width with zero wait states. Of course, you can add 16-bit memory boards — assuming they will work at that clock speed — but memory access time will increase significantly and cause system performance to suffer.

There are many ways to increase the width of a bus to 32 bits. Originally, Zenith developed one technique, and Compaq used another. And other manufacturers used other techniques. IBM developed the Micro Channel Architecture (MCA) and took significant legal steps to protect it so it could not be "cloned" by other computer manufacturers. As a result, a few of these computer manufacturers, notably Zenith and Compaq, formed a group to standardize a new 32-bit bus called EISA — Extended Industry Standard Architecture. Which bus is the "standard" is a matter of controversy, but I have doubts that the MCA is as popular as IBM would have you believe.

Manufacturing MCA-compatible boards is tricky because the card edge connectors are much closer together than on ISA or EISA boards. And despite IBM's claims that the MCA bus is much faster and has more capabilities than EISA, I have a number of reasons for being skeptical of those claims. To be candid, I find it quite difficult to believe that the engineers from these "other" computer companies aren't at least as smart as IBM's engineers. If you really believe that this is possible, I have some bridges for sale in San Francisco. Given that the EISA consortium includes nine different computer companies (sometimes called the "Group of Nine"), it is certainly believable that this group has designed a bus which is at least as good as MCA. Although I have not seen any of the Zenith EISA computers yet, I expect that they will have some features, including speed, that will be more than a little competitive with MCA. At this point, I expect that my next Zenith computer will have an EISA bus, which should give you an idea that I believe it will be better than MCA. I have worked with some of IBM's MCA systems, and I am not impressed.

Comparing the 80386 and 386SX

To get back to the subject of MPUs, there is a lot of talk about the 386SX chip, but few people understand what that really means. And even though Zenith has not introduced any systems with the 386SX chip (at least not as of when I wrote this), I think you will find the differences between the 386 and 386SX chip interesting.

Internally, the 386SX chip is nearly identical to the 80386, which means that it can run 386-specific software. Like the 386 MPU, the SX supports internal processing at the full 32-bit data width. The SX also supports the powerful memory management capabilities that provide the support necessary for multitasking, but that is about where the similarities end.

The standard 386 uses a 32-bit memory address, which means that it can talk to memory up to four *GIGA*bytes (a gigabyte is 1024 megabytes); however, virtually all software (including current versions of OS/2) is written to use no more than the maximum of 16 megabytes that is available on the 80286 MPU. At the present time, there is no reason to buy a standard 386 system based on its memory addressing capability.

On the other hand, the SX uses a 24-bit memory address (like the 80286) which means that it is limited to 16 MB of memory. Like the 286, the SX also uses a 16-bit bus, which supports all of the standard ISA add-on boards. From a user perspective, the real advantage of the 386SX chip lies in its cost — it is much cheaper than a full-blown 80386.

Unless you really need 386-specific software, the 386SX chip does not provide any significant advantages over the 286 systems. In fact, you can even find 286 systems that are considerably faster than 386SX-based systems, so that is not a good reason either. And since current software is limited to using 16 MB of memory, the standard 386 is not used to its full capability, so that does not provide any advantage. The fact that a 386SX chip can use the ISA bus is a significant advantage because it does keep hardware costs down.

If you already have an 80286 system, I think it is a waste of money to consider a 386SX system because you will not see much, if any, performance improvement. I think that the only reason for getting a 386SX system is when cost is a prime consideration and you have a requirement to run 386-specific software. Otherwise, it makes more sense to save your money until you can afford to buy a full-blown 80386 (or newer) computer.

Computer Viruses

Now that the furor over the "Friday the 13th" virus has quieted down, I think it is time to take a look at the general subject again. Some people will tell you that the virus scare is simply media hype, and that the computer virus problem is more fiction than fact. They are flat wrong. Like Santa Claus, computer viruses do exist, and they are real. Some incredibly naive people will tell you that the virus problem is NOT a problem, but you will find that they have never had to deal with the situation. Those of us who have to deal with computer security issues find that a virus problem can occur more frequently than you might expect. But now that we can look at this general problem, especially the Friday the 13th virus, in a more reasonable and rational frame of mind, let's see what this is all about.

Depending on who you talk to, the "Friday the 13th" virus had a variety of names, probably because there were several variations of problems. The latest one — which was a time bomb using Friday, October 13, 1989 — as the trigger, was also called the "Columbus Day virus" or the "Israeli virus." These viruses caused problems by overwriting the boot sector, deleting files, and other types of general destruction. At my last count, there were over 30 distinct and identifiable virus programs (with names) that could infect PC compatible and Macintosh computers.

Some experts will tell you that the problem is greatly exaggerated, but agree that viruses do exist. On the other hand, you can find people that will give credit to the media for preventing problems by making users aware of the nature of the viruses, including the trigger dates, if applicable. As it turns out, the "Friday the 13th" virus did not cause too many problems in the United States, probably because it was so widely publicized in the media. Computer users were extremely cautious last October, and the number of *reported* problems was minimal. Unfortunately, there is considerable evidence that many companies simply do not report virus problems because of embarrassment or fear that they will lose customer confidence. So, the available statistics probably do not reflect the real depth of the problem, and nobody can accurately estimate it.

Like a biological virus, a computer virus is most easily fought if you can avoid it. A lot of the virus-infected programs are transmitted through bulletin boards, and it really does not matter how conscientious the operator is. As new virus detection techniques and protection programs are invented, the mischief-makers develop new viruses that are immune. And so the cycle goes.

Unfortunately, there is a considerable amount of bad information available about the general virus problem. Perhaps the worst of the lot is to call any destructive program code a virus, but that is simply not true. Actually, a computer virus can be benign (i.e., non-destructive), but that is more often the exception than the rule. Some of the benign computer virus programs simply display some kind of "hello, I am here" message when they are triggered and let it go at that. If you have ever seen that kind of unexpected message, you should carefully consider where you have been getting software — that could have been a destructive virus.

When one looks at the computer virus problem, it is important to have an understanding of what the term means. According to my Webster's New Collegiate Dictionary, a VIRUS is "the causative agent of an infectious disease." Obviously, that definition can apply to computer hardware as well as a biological system, although the word "disease" is defined in terms of its harmful effect on an animal or plant. The most important point about the definition of a virus is that it is the *causative agent* of the disease and may not be harmful by itself. For a computer virus, this means the virus is simply the "carrier" (or agent) for the disease, not the disease itself.

A computer virus program really contains two parts. The first part is the actual virus "program" that somehow attaches itself to one or more files or disk areas in the infected computer system. Some viruses always attack system files, — COMMAND.COM, IBMBIO.COM or IBMDOS.COM — some attack certain disk areas (e.g., the boot sector), and some attack any COM or EXE file. In nearly all cases, the virus attack amounts to simply attaching some kind of destructive program code to the target file(s). The virus itself is really just a parasitic program that attaches itself to other files and programs.

The second part of a virus is the destructive code itself, and it can usually be identified in a specific category. The TIME BOMB, which uses time or some kind of timing as a trigger, is the most common. The "Friday the 13th" virus is one example of a time bomb. Another time bomb blows up after it has reproduced itself exactly four times (the Lehigh virus).

A second category of destructive code is called the worm. When released (or triggered), the WORM reproduces itself so that it physically grows and typically "gobbles up" something, like memory or disk space. The problem that occurred with the Internet System in October 1987 was called a virus, but it was really a worm that reproduced itself in computer memory until there was no space left.

Another kind of destructive program is called the Trojan horse, named after the Greek Trojan Horse. Unlike the two categories of destructive code described above, the Trojan Horse is a known and identifiable program that may perform a useful task and seem harmless (such as a game or disk utility program to recover a deleted file), but causes some kind of destruction when it is run, like reformatting a disk or scrambling the File Allocation Tables.

Although I mentioned the time bomb and worm as examples of the kind of code that may be carried by a computer virus, both can also be found in stand-alone programs, such as games and utilities. The time bomb's distinguishing characteristic is that it has a trigger based on time (e.g., date), and the worm is self-replicating. Either, or both, may be carried by a virus or they may be in a stand-alone program, just like a Trojan horse.

From what I have seen and read, the virus which carries the time bomb is the most popular today. And the technical distinction between the different kinds of computer virus infections probably is of little concern to most users because it is becoming quite common to refer to any of these destructive programs as a virus. The technical distinctions among these different kinds of destructive programs still exist, but popular usage of the term "virus" includes just about any kind of "unauthorized," destructive program.

The FlipFast Guides

As mentioned last month, I still have a number of Zenith-specific FlipFast books (MS-DOS and GW-BASIC) available as a set of one each for $9.00 (plus $3.00 shipping). I wrote the 544-page MS-DOS book includes DOS information for versions up to 3.20, and a picture of the cover is shown as Photo 1 below.

The MS-DOS Guide also includes every DOS command and error message with practical explanations about how to fix whatever is wrong. A considerable number of useful examples is included with each command with explanations as to what each does. The last half of the book is dedicated to all kinds of technical information (for programmers) about DOS, such as interrupts (INT), function calls, disk formats, etc. MS-DOS commands for both Z-100 (up to the last version, 3.10) and PC compatible systems are included in the book.

I also did some technical editing on the GW-BASIC Guide (written by Bill Barden) to be sure it was compatible with Zenith versions up to version 2. A picture of the cover is shown as Photo 2 below.

This 408-page book includes all kinds of handy information about how to use GW-BASIC commands, statements, operators, in addition to how to write better and faster BASIC programs.

Since you may be wondering how good a bargain this really is, the MS-DOS Guide originally sold for $24.95 and the GW-BASIC Guide originally sold for $21.95, so this two-book deal is less than the original cost of a single book. To keep shipping costs low, these books are shipped by the Postal Service's book rate, so be sure to allow PLENTY of time for shipping — at least a month from the day you send in your order. Several people have written to me noting that the delivery time can be as long as six weeks, depending on what kind of service you have in your area.

Sets cannot be broken into two of the same book, and quantity discounts are available for more than five sets. Be sure to tell me about how many you need for a specific quote because most of the savings is in shipping costs. Because I still have a number of these books left, I am extending this offer until April 30, 1990, and I will not mention this next month. If I am out of books when I receive your order, I will return your check or money order to you. Ordering information is listed at the end of this article.

Powering Down

For next month, I have some new hardware and software that I think will be of considerable interest to you, especially if you are using OS/2.

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

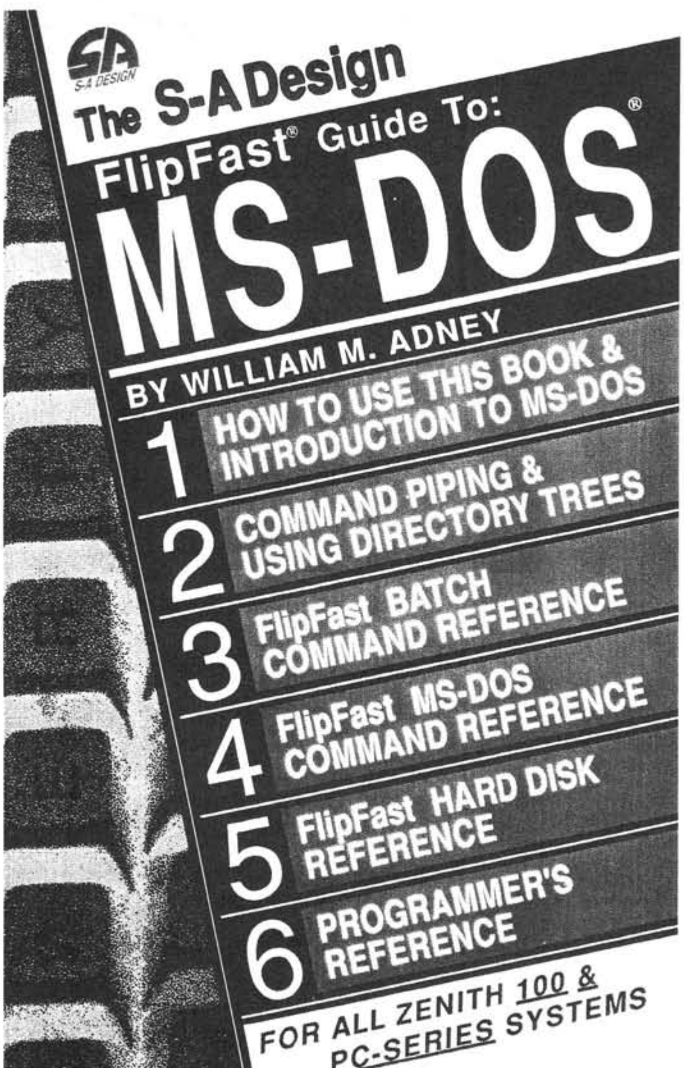


Photo 1
FlipFast Guide to MS-DOS

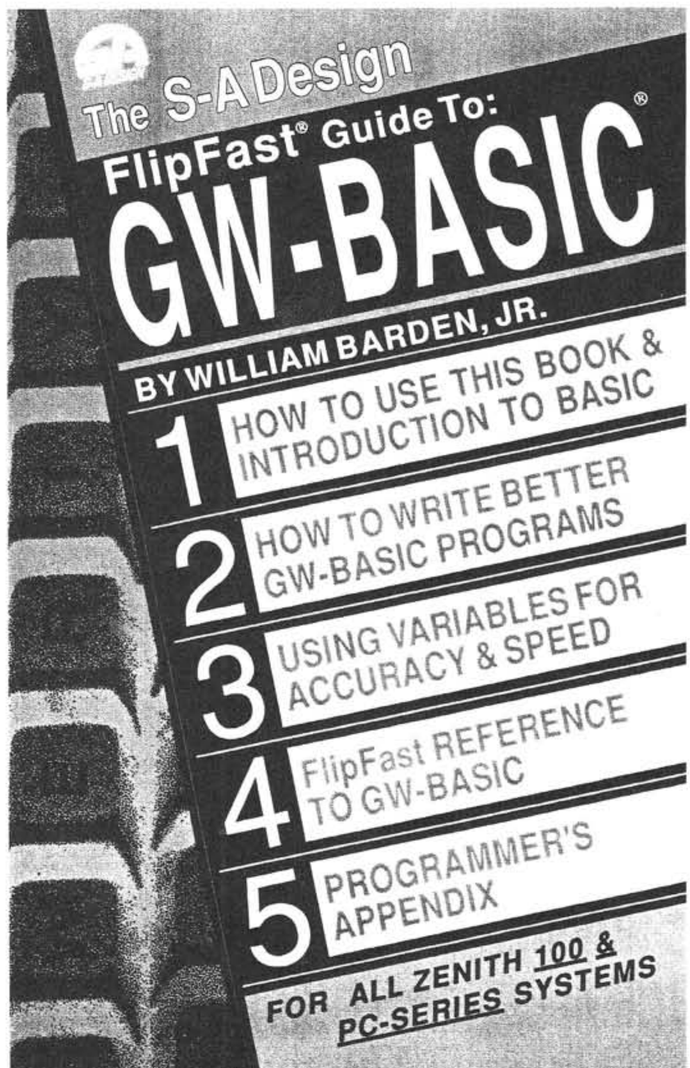


Photo 2
FlipFast Guide to GW-BASIC

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 Heath/Zenith systems in general, be sure to include a self-addressed, stamped envelope (business size preferred) if you would like a personal reply to your question, suggestion, comment or request.

Products Discussed

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

Software
Flipfast Guides $12.00
(includes shipping)
FlipFast Guide to MS-DOS
FlipFast Guide to GW-BASIC
William M. Adney
P.O. Box 531655
Grand Prairie, TX 75053-1655

MS-DOS 3.3 Plus (OS-51-3)
List Price $149.00
Mail Order w/Update Card Only 49.00
Heath/Zenith Computer Centers
Hilltop Road
St. Joseph, MI 49085
(800) 253-0570
(Heath Catalog orders only)

FoxBase+ $395.00
FoxPro 1.0 795.00
FoxPro Demonstration Disk FREE
Fox Software
(419) 874-0162

*

Continued from Page 6

think it was called "Lightning") and that it hadn't worked. I ordered and installed the new ROM chips, V.2.3B, part number "-09", and the program did work. I didn't think any more about upgrades until a year ago when I installed the Intel Inboard 386/PC. The computer would not boot up and instead gave an error message: "Timer Interrupt Failure." I called Zenith and ordered the latest version of the ROM chips. I received and installed V.3.0B, part number "-16." That cured the problem. The computer readily booted up with the Intel 80386 CPU.

The next chapter in this story began when I bought the second Z-151 computer. The computer was a refurbished model sold by the national HUG Bargain Center through their BBS. The price was low enough that it was worth getting just for the spare parts.

The monitor ROMs in the new Zenith showed a part number ending in "-15," and version number 2.9D. My children quickly adopted this computer for their games. I transferred the game software and was surprised to find that one program would not run. It was a game called "Memory House" that I bought from IBM from their "Personally Developed Software" catalog. This program runs under BASIC and it is a neat matching game for both pictures and shapes. It has various levels of difficulty and the kids love it.

When I first got this program I found that it would not run under GW-BASIC. I tried Compaq BASIC and with that the program ran fine. When I transferred the program to the new Zenith it would not run. I got out the original disk, reinstalled the program in the old Zenith and, much to my surprise, it wouldn't run there either. Now I was stumped.

After giving myself some time to calm down, I tried to think of what was different. It occurred to me that I might not have run that program since I installed the Inboard. If so, then the only difference was in the monitor ROMs. I removed V.2.9D (part number "-15") and reinstalled V.2.3B (part number "-09"). That solved the problem. Memory House came up and ran like a charm. It has been working fine ever since.

I don't know what part number "-15" does that part number "-09" doesn't do, but I am going to hold on to those chips. One day I am certain to find out.

Products Mentioned

Adney & Associates
William M. Adney, Writer
Computer Systems Consulting
P.O. Box 531655
Grand Prairie, TX 75053-1655
(214) 642-0467

PC Tools
Central Point Software
9700 S.W. Capitol Highway
Portland, OR 97219
(503) 244-5782

Seagate Technology
920 Disc Drive
Scotts Valley, CA 95066
(800) 468-3472

SpinRite
Gibson Research Corp.
Laguna Hills, CA
(714) 830-2200

Heath Users' Group
P.O. Box 217
Benton Harbor, MI 49022-0217
(616) 982-3463
(616) 982-3956 (BBS)

Intel Inboard 386/PC
Intel Corporation
5200 NE Elam Parkway
Hillsboro, OR 97124
(800) 538-3373

VOPT
Golden Bow Systems
2665 Ariane Drive, Suite 207
San Diego, CA 92117
(619) 483-0901

Western Digital Corp.
2445 McCabe Way
Irvine, CA 92714
(800) 847-6171
(800) 777-4787 (Tech. Support)

Zenith Data Systems
St. Joseph, MI 49085
(616) 982-3571 (Parts Department)
(616) 982-3309 (Tech. Support)

*

ASSEMBLY LANGUAGE
PART 4

PAT SWAYNE
HUG SOFTWARE ENGINEER

THE INSTRUCTION SET (PART 2)

This is the fourth part in my series on Assembly Language. In the last installment, I began a discussion of the processor's instruction set. I will continue that discussion in this installment, starting with arithmetic instructions.

Arithmetic Instructions

The arithmetic instructions include the obvious (add, subtract, multiply, and divide), and a group of instructions that are used to manipulate numbers before or after arithmetic operations. The instructions that have two arguments in their operands all can use the direct, index, or register addressing modes in both arguments, and also the immediate mode in the source argument. However, both arguments cannot use the direct mode or index mode at the same time. Here are the arithmetic instructions.

ADD — Adds the number in the source to the number in the destination, and places the result in the destination. The carry flag is set if there is a carry out of the addition. ADD also affects the Aux Carry, Overflow, Parity, Sign, and Zero flags, depending on the result.

ADC — (Add with carry) Adds the number in the source and the value of the carry flag (0 or 1) to the destination, and places the result in the destination. This instruction is used in adding numbers larger than 16 bits. ADC affects the same flags as ADD.

SUB — Subtracts the number in the source from the number in the destination, and places the result in the destination. The carry flag is set if the source number is larger than the original destination. SUB affects the same flags as ADD.

SBB — (Subtract with borrow) Subtracts the number in the source and the value of the carry flag (0 or 1) from the destination, and places the result in the destination. This instruction is used in subtracting numbers larger than 16 bits.

INC — Increments (adds 1 to) the number in the operand, which can be a register or direct memory location. INC affects the same flags as ADD except for the carry flag, which is not affected.

DEC — Decrements (subtracts 1 from) the number in the operand, which can be a register or direct memory location. DEC affects the same flags as INC.

NEG — (Negate) Changes the sign of the number in the operand. It does this by calculating the 2's complement of the number. Or, another way of looking at it is that it complements all of the bits in the number and then adds 1 to the result.

DAA — (Decimal Adjust for Addition) This instruction corrects the result when two packed decimal numbers are added together (and the result is in AL). I will have to explain packed decimal numbers in order to explain this instruction. A packed decimal number (also called a binary coded decimal) is a way of expressing a one or two digit decimal number in binary form. The ones digit of the number is placed in binary form in the lower half (or nibble) of an 8-bit register. The tens digit is placed in the upper half. For example, the decimal number 12 would be expressed as 00010010 in binary, or 12 in hexadecimal. As you can see, when a packed decimal number is expressed in hexadecimal form, it "looks like" the decimal number that it represents.

The DAA instruction works this way. If the low nibble of the number in AL is greater than 9, or if the Auxiliary Carry flag is set, then 6 is added to the number, and the Auxiliary Carry flag is set. If the upper nibble of AL is greater than 9, or if the Carry flag is set, then 60 (hex) is added to the number, and the Carry flag is set. To illustrate how DAA works, suppose you had 14 (hex) in the AL register and 16 (hex) in the BL register, and you added BL to AL. The result (in AL) would be 1A (hex), but if the original two numbers were actually packed decimal numbers, then that would not be the correct answer. If you process the result with the DAA instruction, you can see that the lower nibble is greater than 9, so 6 will be added to the number 1A (hex) plus 6 (hex) equals 20 (hex), which, when considered as a packed decimal number, is the correct answer to 14 (decimal) plus 16 (decimal).

DAS — (Decimal Adjust for Subtraction) This instruction corrects the result when one packed decimal number is subtracted from another (and the result is in AL). If the low nibble of the number in AL is greater than 9 or the Aux Carry flag is set, then 6 is subtracted from the number and the Aux Carry flag is set. If the upper nibble of AL is greater than 9, or if the Carry flag is set, then 60 (hex) is subtracted from the number, and the Carry flag is set.

AAA — (ASCII Adjust for Addition) There are four "ASCII Adjust" instructions, but none of them are for working directly with ASCII numbers. Instead, they work with "unpacked decimal numbers". An unpacked decimal number exists when each place (ones, tens, etc.) of a decimal number occupies an entire 8-bit register. In other words, an 8-bit register can only contain the values 0 through 9 if it contains a valid unpacked decimal number. The ASCII value of a particular number is the value that will cause that particular number to appear on your screen, or be printed on your printer. As it turns out, the ASCII value of a number is the actual value plus 30 (hex). So if you work with unpacked numbers, you can convert them to ASCII just by adding 30 (hex) to each digit.

If you are just starting out in Assembly Language, it is probably not important for you to learn the ASCII adjust instructions right now, but you should at least be aware of their existence. At least two of them are more useful than you might suspect, as you will see later.

The AAA instruction converts a number in the AL register into a valid unpacked decimal digit. If the low nibble is greater than 9, or if the Aux Carry flag is set, AAA adds 6 to the number, adds 1 to the AH register, sets the Aux Carry flag and the Carry flag, and zeros the high nibble of AL. If the low nibble of the number is 9 or less and Aux Carry is not set, the high nibble is zeroed, and the Carry and Aux Carry flags are zeroed. If the AAA instruction is executed immediately after the addition of two unpacked decimal numbers, and if the next higher place digit is in the AH register, then the instruction will add any carry out of the addition to the next higher place.

AAS — (ASCII Adjust for Subtraction) This instruction corrects the result when one unpacked decimal number is subtracted from another (and the result is in AL). If the low nibble of the number in AL is greater than 9 or the Aux Carry flag is set, AAS subtracts 6 from the number, subtracts 1 from the AH register, sets the Carry and Aux Carry flags, and zeros the high nibble of AL. If the low nibble of the number is 9 or less and Aux Carry is not set, the high nibble is zeroed and the Carry and Aux Carry flags are zeroed.

MUL — Multiplies two unsigned 8 or 16-bit integers. The multiplicand must always be in the AL or AX register. The multiplier for MUL (and the other multiply and divide instructions) can be specified with the register, direct, or index addesssing modes. If the result is the same size as the multiplicand (8 or 16 bits), then the result will be in the AL or AX register, and the Carry flag and Overflow flag will be clear. If the result is larger than the size of the multiplicand, then the overflow will be in the AH register if the overflow is out of 8 bits, or in the DX register if the overflow is out of 16 bits, and the Carry flag and Overflow flag will be set. The Aux Carry, Parity, Sign, and Zero flags are left in an undefined condition by this instruction.

IMUL — Multiplies two signed 8 or 16-bit numbers. It works like MUL except that when the result does not overflow, the upper half of the double accumulator (the AH or DX register) contains an extension of the sign of the result. In other words, if the result is negative and there is no overflow, the AH or DX register will contain FF or FFFF hexadecimal. If the result is positive and there is no overflow, the AH or DX register will contain zero.

AAM — (ASCII Adjust for Multiply) This instruction corrects the result when two unpacked decimal numbers are multiplied. The result is a valid two-digit unpacked decimal number in AH and AL. AAM divides AL by 10 (decimal) and puts the quotient in AH and the remainder in AL.

One handy use for the AAM instruction is to convert binary numbers to ASCII decimal numbers, ready for display on the screen or printing. Any binary number whose decimal value is 0 through 99 can be converted. Here is an example of a conversion routine.

```
AAM                ;CONVERT NUMBER TO UNPACKED DECIMAL
ADD     AX,3030H   ;CONVERT UNPACKED DECIMAL TO ASCII
XCHG    AL,AH      ;CORRECT ORDER OF DIGITS
MOV     BUFFER,AX  ;STORE RESULT IN A BUFFER
```

The XCHG instruction is required because the AAM instruction puts the ones place of the converted number in the AL register, and the tens place in the AH register. When you store the number in a buffer for printing or display, you will most likely want the tens place to come first in the buffer, and since the low byte of a 16-bit number is stored first by the CPU, the bytes must be exchanged before you store the number.

The AAM instruction actually consists of two bytes. The first byte is D4 (hex), and the second byte is 0A (hex), or 10 decimal. Some members of the 8086 processor family will allow you to change the second byte to get a new version of the AAM instruction, which works in a radix other than decimal. For example, if you changed the second byte to 10 (hex), or 16 decimal, you would have an instruction that could be used to convert binary numbers to ASCII hexadecimal. Most assemblers do not have a mnemonic for altered versions of AAM, so you must "build" it yourself, using the DB (define byte) directive, as in

```
DB      0D4H,10H   ;AAM 10H INSTRUCTION
```

If you have an assembler that allows it, you can put the new radix in the operand field, as in

```
AAM     10H        ;CONVERT TO UNPACKED HEX
```

DIV — Divides the 16-bit number in AX by an 8-bit operand, or the 32-bit number in DX, AX by a 16-bit operand. If the operand is 8 bits, the result is returned in AL, and the remainder in AH. If the operand is 16 bits, the result is returned in AX, and the remainder in DX. If the result is too large (as when division by zero is attempted), the processor generates a "type 0" interrupt. When I get to the INT instruction, I will explain interrupt types, etc. MS-DOS usually processes this interrupt. It causes a "Divide overflow" message to be printed, and it terminates your program and returns to the command prompt. You may have seen this happen from time to time. The result flags are left in an undefined state by the DIV instruction.

IDIV — Divides the signed 16-bit number in AX by an 8-bit operand, or the 32-bit number in DX, AX by a 16-bit operand. This instruction works like DIV except that all numbers are treated as signed numbers. The remainder will have the sign of the dividend (the number in the accumulator). For division by a byte, the quotient must be in the range -127 to +127 decimal. For division by a word (16 bits), the quotient must be in the range -32,767 to +32,767. If the quotient is outside the specified range, the processor generates a type 0 interrupt.

AAD — (ASCII Adjust for Division) This instruction is used to modify the dividend before division is done so that the result of dividing two unpacked decimal numbers will be an unpacked decimal number. AAD multiplies the number in AH by 10, adds that result to the number in AL, and places the final result in AL. Just as the AAM instruction can be used for converting binary numbers to ASCII decimal, the AAD instruction can be used to convert ASCII decimal numbers to binary. If you have a two digit number stored in a buffer, you can convert it with code like this.

```
MOV  AX,BUFFER ;GET THE NUMBER
SUB  AX,3030H  ;CONVERT TO UNPACKED DECIMAL
XCHG AL,AH     ;SWAP TENS AND ONES PLACES
AAD            ;CONVERT TO BINARY
```

Like the AAM instruction, the AAD instruction can be modified to work in other radixes. The two bytes in the instruction are D5 (hex) and 0A (hex), and you can change the second byte to change the radix. You should be aware, however, that if you use an altered AAD instruction in a program, it may not run on some computers. All 8086 family processors made by Intel, or their direct clones, will execute the instruction correctly. However, it is known that the V series processors made by NEC, often used in "speed up" kits for 8088-based computers, always assume the radix to be 10 no matter what you do to the instruction.

CBW — (Convert Byte to Word) This instruction converts a signed 8-bit number in AL to a 16-bit number in AX. If the number is positive, AH is cleared to zero. If the number is negative, AH, is set to FF (hex). CBW is useful in preparing a signed 8-bit number for division using the IDIV instruction.

CWD — (Convert Word to Double word) This instruction converts a signed 16-bit number in AX to a 32-bit number in DX, AX. If the number is positive, DX is cleared to zero. If the number is negative, DX is set to FFFF (hex). CBW is used for preparing a signed 16-bit number for division with IDIV.

This concludes the discussion of the arithmetic instructions. Next time, I will preset the logic and compare instructions. *

Graph for a Year

Thomas G. Custin
1660 S. Ivy Trail
Baldwinsville, NY 13027

Did you ever want to plot a chart showing values of your daily weight for several months or even a year? That's what I wanted to do, but found that available "graphics" software packages covered at most 30 days. That's when I decided to develop my own graphics program that would span a year. There are other applications besides plotting your weight. Figure 1, for example, shows a daily plot of the Dow Jones Industrial Stock averages for 9 months.

The program, which is divided into three parts, is written in BASICA for the H/Z-100; however, hints have been included to adapt to PC compatibles. The first part generates the chart raster and adds axis labels. The second describes how data is stored in a sequential data file, and finally, the third section provides the plotting equation that serves to plot the data on the graph. Each of the three parts will be described together with the accompanying BASICA statements.

The Chart Raster

The starting point is generation of the X-Y axes in suitable format to fit the computer screen. BASICA statements that generate the chart raster are listed in Table 1. The first five statements clear the screen, call up the data file to be used to plot the graph, and request values for F and N. This is the source of the data points used to plot your curve, and scale factors that we will talk about later. Next, number 60 "LINE" statement locates the Y-axis with enough space to provide 10 columns to the left. These will be used to label the Y-axis. Similarly, the X-axis has been located to allow two rows of characters below the X-axis. This is followed by statements 80 to 160 that provide a set of tick mark lines for reticles to easily read the curve.

To some extent the computer screen limits usable scales, since lines of text used for labels can only be positioned at 24 discrete lines on the H/Z-100 computer screen. Allowing for labels and a title at the bottom of the graph, and identification at every other line limits the field to 10 divisions of the ordinate scale.

The interval between Y-axis reticles was chosen at exactly two lines of text. As explained above, the table has been written for the H/Z-100 which has 225 pixels in the vertical frame. The H/Z-100 has 25 lines of text on the screen, or 9-pixels per character height. Since the reticle markers on the Y-axis have been chosen at a two character line separation, there will be 18 pixels between reticles along the Y-axis. Labels on the tick marks will customize the graph to your application. The basic chart applies to any application since it is an optimum arrangement to fit the computer screen. Your choice of labeling will show divisions to provide convenient interpolation of the information displayed. Finally, as will be explained, the plotting equation will normalize the data points to fit your chart.

Selection of the scale on the graph is key to ease in assimilating significance to the curve. If I used a scale of 0 to 200 pounds for my weight chart, a 20 pound loss would only span one set of tick marks. A better graph would use a scale of 150 to 200 pounds. That provides a curve with greater spread and gives me a better feeling for my accomplishments in weight loss. As will be explained later, the value for "N" and the plotting equation will adjust to fit the scale.

Table 2 is a continuation of the program that provides the axis labels. I designed my weight chart to cover a range of 150 to 200 pounds. Statements 170 to 200 place identifying numbers along the Y-axis tick marks. Statements 210 to 250

```
10 CLS
20 PRINT "ENTER NAME OF DATA FILE":INPUT J$
30 PRINT "ENTER VALUE FOR FULL SCALE":INPUT F
40 PRINT "ENTER SCALE FACTOR":INPUT N
50 CLS
60 LINE (80,10)-(80,206)
70 LINE (80,206)-(600,206)
80 LINE (80,188)-(85,188):LINE (80,170)-(85,170):LINE (80,152)-(85,152)
90 LINE (80,134)-(85,134):LINE (80,116)-(85,116):LINE (80,98)-(85,98)
100 LINE (80,80)-(85,80):LINE (80,62)-(85,62):LINE (80,44)-(85,44)
110 LINE (80,26)-(85,26)
120 LINE (120,203)-(120,206):LINE (160,203)-(160,206):LINE (200,203)-(200,206)
130 LINE (240,203)-(240,206):LINE (280,203)-(280,206):LINE (320,203)-(320,206)
140 LINE (360,203)-(360,206):LINE (400,203)-(400,206):LINE (440,203)-(440,206)
150 LINE (480,203)-(480,206):LINE (520,203)-(520,206):LINE (560,203)-(560,206)
160 LINE (600,203)-(600,206)
```

Table 1
Chart Raster Listing

```
170 LOCATE 3,8:PRINT "200":LOCATE 5,8:PRINT "195":LOCATE 7,8:PRINT "190"
180 LOCATE 9,8:PRINT "185":LOCATE 11,8:PRINT "180":LOCATE 13,8:PRINT "175"
190 LOCATE 15,8:PRINT "170":LOCATE 17,8:PRINT "165":LOCATE 19,8:PRINT "160"
200 LOCATE 21,8:PRINT "155":LOCATE 23,8:PRINT "150"
210 LOCATE 4,4:PRINT "W":LOCATE 5,4:PRINT "E":LOCATE 6,4:PRINT "I":LOCATE 7,4
220 PRINT "G":LOCATE 8,4:PRINT "H":LOCATE 9,4:PRINT "T"
230 LOCATE 11,4:PRINT "I":LOCATE 12,4:PRINT "N"
240 LOCATE 14,4:PRINT "P":LOCATE 15,4:PRINT "O":LOCATE 16,4:PRINT "U"
250 LOCATE 17,4:PRINT "N":LOCATE 18,4:PRINT "D":LOCATE 19,4:PRINT "S"
260 LOCATE 24,11
270 PRINT "   4    8    12   16   20   24   28   32   36   40   44   48   52"
280 LOCATE 25,12
290 PRINT "                WEEKS FROM START OF DIET"
```

Table 2
Axis Labels

serve to vertically deploy the scale identification, "WEIGHT IN POUNDS" parallel to the Y-axis. The four remaining statements in Table 2 provide the X-axis identification in weeks based on a one year chart.

The statements in Table 2 are where you can customize your chart to your specific application. Location of the Y-axis allows ample room to change the scale. The statement LOCATE 3,8 in 170 positions the top number of the Y-axis in the third row at the 8th column. This allows space for 3 digits. If you wish to have a scale with, for example, 5 digits, the number 8 would have to be reduced to 5. This would place the Y-axis label right up to the numbers; however, statements 210 to 250 that identify the scale can be moved from column 4 to 2 and still fit nicely on the screen.

The X-axis is two lines up from the bottom of the screen to allow for identification characters for the abscissa scale and its identification. The remaining four statements in Table 2 provide the number scale in weeks and the abscissa label, but note statement 280. It calls for a location at line 25! We'll talk more about this later.

If you have a PC compatible, you will find that it has only 200 vertical pixels per frame. It might first seem impossible to fit the raster on its screen, but it also has a 25 line screen, only using 8 pixels per character line. I can't say that I've tried it, but I believe it should work if you figure 16 pixels between reticles along the Y-axis. For that case, the bottom line should fall at 183.

Sequential Data File

Data used for plotting the curve are stored in a sequential data file. A file of this type is accessed in the same sequential order in which they were written, starting at the beginning of the data block and proceeding in order until an end-of-data marker flag is encountered or the required number of items has been read.

There are several ways to generate a sequential data file, but the most convenient is use of a spreadsheet. A spreadsheet can serve an added function of smoothing. In some cases, the point-to-point fluctuations may be larger than desired for the final curve. Use of a spreadsheet allows a convenient way to average data points. Daily variations of 2 or 3 pounds spoil appearance of the weight plot and fail to add useful information. I have found that averages of 3 points generally works out to be about the best. This was done for the Dow Jones plot in Figure 1. Averaging is quite simple with the spreadsheet. A second column is added where each data point is the last entry plus the two previous entries divided by 3. This then becomes the data file. It is important to remember that the last data entry must always be followed by the "end-of-data" marker flag. As the program is written, statement 320 calls out a flag of 9.000. That number always has to appear after your last data entry.

In lieu of a spreadsheet, you can maintain a sequential data file using a word processor. In either case, it is important to be certain that the data file is a single column with the newest entry at the bottom followed by the "end-of-data" marker. Statement 320 listed in Table 3 is the one that calls up the "end-of-data" value of 9.000 to terminate the loop. The "end-of-data" marker can be any character chosen to be highly improbable to appear in the data file, but statement 320 must be corrected accordingly.

Whether generated by a word processor or spreadsheet, your data file identification has to be accessible to BASICA and entered in response to the first "INPUT" statement of line 20.

```
300 Q=80
310 OPEN "I", #1, J$
320 INPUT #1, R
330 IF R=9!, THEN 380
340 Q=Q+1.429
350 R1= 197-((180*N)*(R/F)-((1*N)-1)
                                    *180)
360 CIRCLE (Q,R1),1
370 GOTO 320
380 CLOSE #1
```

Table 3
Plotting Statements

Plotting Equation

The nine plotting statements that serve to plot points on the chart are listed in Table 3. The first statement, number 300, sets the starting point at the Y-axis. This is followed by statement 310 that opens the sequential data file, and 320 that inputs each of the "R" data points. Statement 330 closes the loop when the "end-of-data" marker is reached. Statement 340 serves to adjust the scale so that data points for the 28 days in four weeks on the abscissa scale are spread over the 40 pixel intervals between reticle markers.

Statement 350 computes the location for each data point "R" from file. The value of F is the full scale value of the field, or the highest data point. The ratio of R/F serves to normalize the data to fit the chart. The value of N, as previously noted is the ratio of the full scale value to the amount covered by the abscissa. For the case of my weight chart, the full scale value (200) divided by abscissa coverage (50) provides a value of 4 for N.

The constant 197, in equation 350, which represents the bottom of the scale was obtained with two maneuvers. First, if you will note, subtracting 18 pixels from the 225th pixel (bottom pixel on the H/Z-100 screen) would provide 207. The value 197 is ten pixels above. One of these has been added because the line for the X-axis cannot occupy the same pixels as the label characters directly below. For this reason, the entire raster has been moved exactly one pixel above character line locations. The remaining 9 pixels result from a bit of skullduggery in statement 280. Calling out line 25 is an illegal function call. There are only supposed to be 24 usable lines per screen. The 25th line is reserved for special flag lines. One example is the function key callout displayed by BASICA in response to CTRL-T. It is also true that BASICA displays an "OK" to appear on the 24th line without destroying the function key flag line. Unfortunately, this also deletes the top 9 pixels from the screen. Fortunately, BASICA doesn't object to the illegal function call as long as the function key flag does not appear on the screen. For this reason, it is important to remember that THIS PROGRAM WILL NOT RUN IF THE FUNCTION KEY FLAG LINE IS DISPLAYED. BASICA will tell you there is an illegal function call in statement 280.

If you run the program just to statement 290 and watch carefully, you can see BASICA slide the entire display, except the title line, up one line to fit in the "OK". This leaves the title sitting at line 25 with the "OK" at line 24 and numbers on the abscissa at line 23. Since the screen shift occurs at statement 290, all of the subsequent displays are effectively moved down one line. In order to plot the data points at the correct position relative to the raster, it becomes necessary to subtract 9 pixels. These 9, plus one pixel to

```
390 PSET (80,179):PSET (120,179):PSET (160,179):PSET (200,179):PSET (240,179)
400 PSET (280,179):PSET (320,179):PSET (360,179):PSET (400,179):PSET (440,179)
410 PSET (480,179):PSET (520,179):PSET (560,179):PSET (600,179)
420 PSET (80,161):PSET (120,161):PSET (160,161):PSET (200,161):PSET (240,161)
430 PSET (280,161):PSET (320,161):PSET (360,161):PSET (400,161):PSET (440,161)
440 PSET (480,161):PSET (520,161):PSET (560,161):PSET (600,161)
450 PSET (80,143):PSET (120,143):PSET (160,143):PSET (200,143):PSET (240,143)
460 PSET (280,143):PSET (320,143):PSET (360,143):PSET (400,143):PSET (440,143)
470 PSET (480,143):PSET (520,143):PSET (560,143):PSET (600,143)
480 PSET (80,125):PSET (120,125):PSET (160,125):PSET (200,125):PSET (240,125)
490 PSET (280,125):PSET (320,125):PSET (360,125):PSET (400,125):PSET (440,125)
500 PSET (480,125):PSET (520,125):PSET (560,125):PSET (600,125)
510 PSET (80,107):PSET (120,107):PSET (160,107):PSET (200,107):PSET (240,107)
520 PSET (280,107):PSET (320,107):PSET (360,107):PSET (400,107):PSET (440,107)
530 PSET (480,107):PSET (520,107):PSET (560,107):PSET (600,107)
540 PSET (80,89):PSET (120,89):PSET (160,89):PSET (200,89):PSET (240,89)
550 PSET (280,89):PSET (320,89):PSET (360,89):PSET (400,89):PSET (440,89)
560 PSET (480,89):PSET (520,89):PSET (560,89):PSET (600,89)
570 PSET (80,71):PSET (120,71):PSET (160,71):PSET (200,71):PSET (240,71)
580 PSET (280,71):PSET (320,71):PSET (360,71):PSET (400,71):PSET (440,71)
590 PSET (480,71):PSET (520,71):PSET (560,71):PSET (600,71)
600 PSET (80,53):PSET (120,53):PSET (160,53):PSET (200,53):PSET (240,53)
610 PSET (280,53):PSET (320,53):PSET (360,53):PSET (400,53):PSET (440,53)
620 PSET (480,53):PSET (520,53):PSET (560,53):PSET (600,53)
630 PSET (80,35):PSET (120,35):PSET (160,35):PSET (200,35):PSET (240,35)
640 PSET (280,35):PSET (320,35):PSET (360,35):PSET (400,35):PSET (440,35)
650 PSET (480,35):PSET (520,35):PSET (560,35):PSET (600,35)
660 PSET (80,17):PSET (120,17):PSET (160,17):PSET (200,17):PSET (240,17)
670 PSET (280,17):PSET (320,17):PSET (360,17):PSET (400,17):PSET (440,17)
680 PSET (480,17):PSET (520,17):PSET (560,17):PSET (600,17)
690 END
```

Table 4
Grid Listing

protect the X-axis from the abscissa numbering requires that the ordinate zero reference start at pixel 197 instead of 207. The remainder of the equation in statement 350 serves to normalize data points to fit a field that covers to a full scale of "F" expanded by a ratio "N". Other scales and variations are readily obtained by selecting values for F and N and changing statements 170 to 290 to customize the graph to fit your application.

Jazz It Up

While the 38 statements of Tables 1 through 3 provide all of the necessities to plot a curve using over 500 data points, Table 4 adds a grid listing that improves readability. These statements place a single dot at the intersections of the X and Y-axis reticles. Putting the three tables together with my WGT file produced the graph shown in Figure 2. ✱

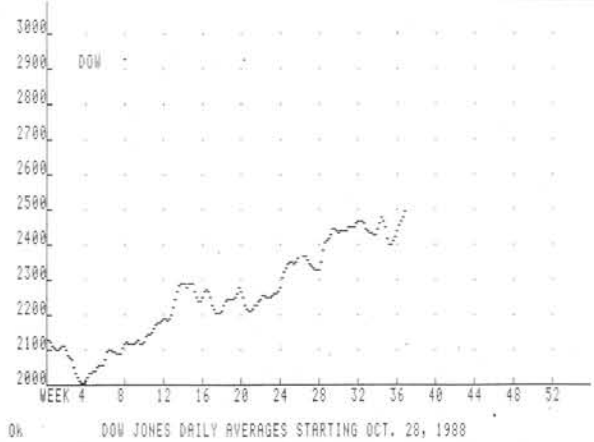


Figure 1
Dow Jones Stock Averages

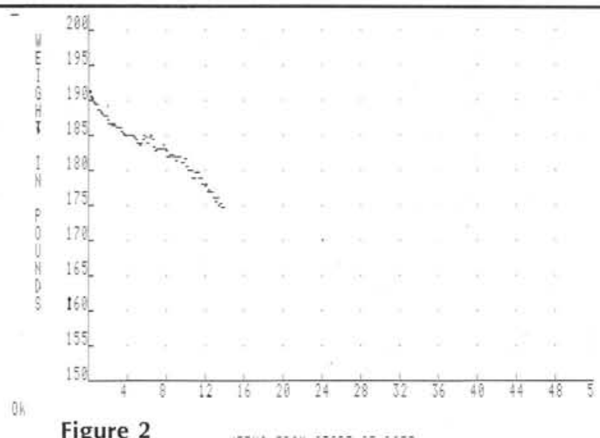
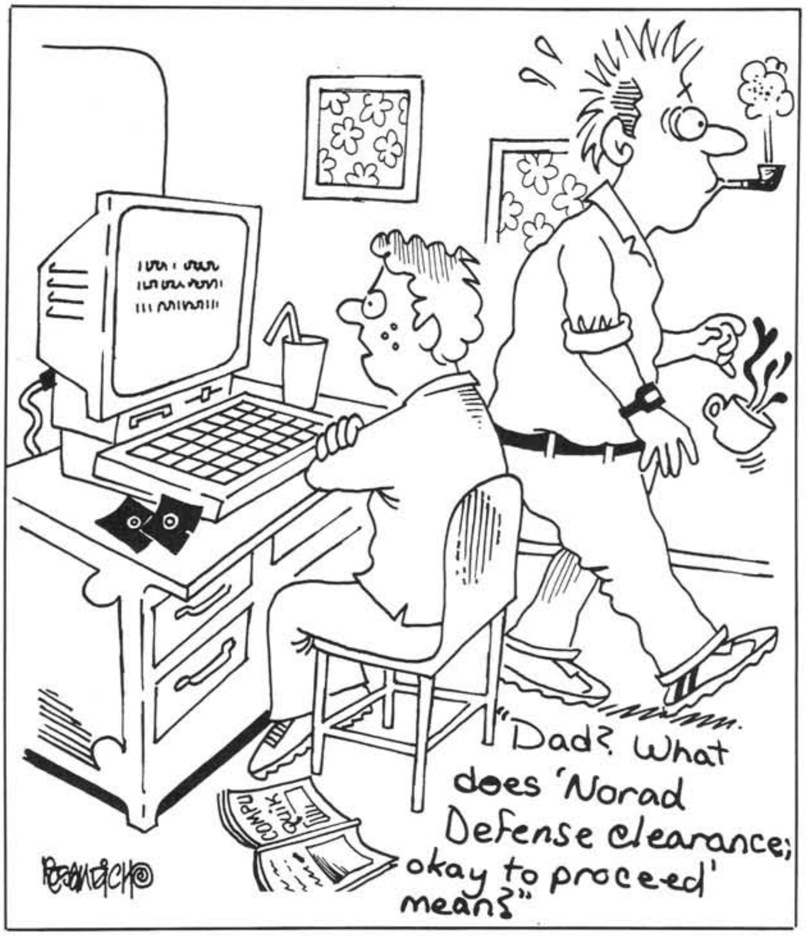


Figure 2
My Weight Chart

Z-100 Survival Kit #11



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

Patching Programs to Run Under ZPC

The last two installments of Z-100 Survival Kit have dealt with the subject of patching programs for use under HUG's ZPC emulator program. In the last issue we covered some of the preliminaries, like how to use DEBUG. This month's column will conclude the discussion with specific details of what types of problems we encounter in PC programs that require patching to run under ZPC.

Surveying the Scope of the Problem

It seems to me, the most logical way to go about this project is to start from the top and work down. By that, I mean to say that we first need to understand the major reasons why patches are required, and then consider some of the strategies for making the patches. Then, as space permits, we can go into some examples of actual patches that might be applied to a program. I'm sure that a comprehensive treatment of the subject, that covered every aspect of patching programs for ZPC, would easily fill a large book. And I'm also sure that the book would never make the 'best seller' list. We'll just have to do the best we can, with the resources and space that is available.

Patch Logic — Why Some PC Programs Won't Run Under ZPC

ZPC is a software solution to the problem of PC compatibility for the Z-100. And although the emulation provided is amazingly complete, considering the differences between a Z-100 and IBM-PC, some things just can't be emulated. These 'un-emulatable' things tend to fall into specific categories. See Figure 1.

Port Accesses
Unsupported Interrupts
The Graphic Character Table
BIOS Data Segment Accesses
Specific Hardware Constraints

Figure 1
General Patching Categories

Far and away the most common reason a program won't run under ZPC is because it makes accesses to ports which don't exist on a Z-100. The I/O port map of an IBM-PC doesn't bear any resemblance to that of a Z-100. So any port accesses made by PC software are potential troublemakers.

Another area of concern is the handling of interrupts. Since interrupts are processed by software interrupt routines, there is a great potential for software emulation of interrupts, and in fact, this is how ZPC provides most of its PC compatibility. But there are a few PC interrupts that cannot be supported because of hardware incompatibilities.

The graphic character table used by many PC programs is held in PC memory at an address already used by the Z-100's MTR-100 monitor ROM, which precludes moving a copy of the table there for an emulated environment. This problem is easily overcome by patching the address of the table in PC programs.

Some programs access the IBM-PC BIOS data area directly to learn about system parameters and addresses. Where possible, ZPC has tried to replicate the PC data area, but there are still problem areas.

And finally, there are some hardware devices and accessories for PC compatible computers that just don't lend themselves to software emulation. Good examples would be EGA/VGA video cards, sound generation hardware, and direct access of I/O hardware by PC programs.

A Note About ZHS Circuits and ZPC

Many of the patches we will be discussing may be unnecessary if you have a ZHS circuit (Scottie board) installed in your Z-100. This simple hardware device, described in previous issues of REMark, allows many PC compatible programs to run under ZPC, which would otherwise require patches. My discussion here, however, will assume that no Scottie board is installed.

General Patching Strategy

Before we dive into the actual process of locating patches to correct specific problems, an explanation of our overall strategy would be in order. The basic methods we will use will be the same, regardless of what particular patch we are going to make.

The hardest part of the patching process may be in deciding what type of patches are required. There will be some clues. For instance, if text on the screen is unrecognizable, that's a good indication that the address of the graphic character table needs patching. But in many (if not most) cases, the program will simply hang, or crash. Trying to decide what to look for will be difficult, and may be a matter of experience and intuition, more than anything else.

Once we know what we're looking for, we'll use DEBUG to do the searching. Sometimes the search will be as simple as looking for a text string. But other times, we'll need to use the unassemble command to find the byte values of likely Assembly Language instructions. In many cases, there will be more than one way the Assembly code we're searching for could be written. This will require some skill in Assembly Language, and a great deal of patience.

As an example, we may want to search for accesses to the PC ports. These accesses might be done with Assembly Language instructions like this:

```
MOV   DX, 3D8
OUT   DX, AL
```

We can use the DEBUG unassemble command to determine that these Assembly instructions have the following byte values:

```
BA D8 Ø3      MOV    DX, 3D8
EE            OUT    DX, AL
```

Now we can use the DEBUG search command to search for occurrences of these bytes, like this:

```
SØ FFFF BA D8 3 EE
```

This sounds simple enough. (If it doesn't, then you need to go back and read Z-100 Survival Kit #10, which was a quick DEBUG tutorial.) But there is a big catch. The Assembly code doesn't have to look like that shown above. The DX register could be loaded from another address. Or the MOV DX instruction might be separated from the IN instruction by other code.

Of course, you could be sure of finding all of the port accesses by simply searching for all occurrences of the port access instructions:

```
IN    AL, DX
IN    AL, (port number)
OUT   DX, AL
OUT   (port number), AL
```

The problem with this is that you would have to sort out the long list of prospective candidates to see which ones are significant, and which ones are simply random data that happen to have that byte value.

I think you can begin to see why a fair amount of Assembly Language knowledge is necessary to become proficient in ZPC patching.

After the patch location (or locations) is found, then we need to make the changes to the program using the DEBUG enter or assemble commands. The program can now be written back to disk and tested, if desired. Of course, you should also make a note of the changes you have made, so that you will not have to go through this laborious procedure again in the future. And if you are finally successful in getting the program to run under ZPC, you should send a copy of the patches to me, so that I can spread the news around a bit.

Port Access Patches

The way a program communicates with its hardware environment is by reading and writing to ports. Ports are accessed by a particular port address. In the Z-100, all of the I/O ports have addresses which must range between 0 and 0FFh. In an IBM PC, the I/O ports may have addresses anywhere between 0 and 3FFh. Special instructions are used to communicate with ports. And port addresses should not be confused with RAM memory addresses.

The Z-100 has no I/O ports in common with the IBM PC. (See Figures 2 and 3 for a breakdown of the Z-100 and IBM PC ports.)

000-00F	DMA Controller
020-021	Interrupt Controller
040-043	Timer
060-063	PIA, Keyboard
080-083	DMA Page Registers
200-207	Game I/O Adaptor
210-217	Expansion Unit
278-27F	Parallel Printer 2
2F8-2FF	Secondary COM Port
320-32F	Hard Disk Controller
378-37F	Parallel Printer
3BC-3BF	Monochrome Display Adaptor
3C0-3CF	EGA Adaptor
3D0-3DF	CGA Adaptor
3F0-3F7	Floppy Disk Controller
3F8-3FF	Primary COM Port

Figure 2
IBM PC/XT Port Usage

So strictly speaking, ANY port address by a PC program is a likely troublemaker. We are, however, lucky in two respects. First of all, direct port accesses are not necessary if a program takes advantage of the IBM PC BIOS services, so not many programs make port accesses except for the video ports. Secondly, many of the common port addresses do not conflict with the Z-100 ports, or cause no trouble if they do. If there is an attempt to write to a non-existent port, nothing happens.

There is one major problem that is caused by the two different addressing schemes used by the Z-100 and IBM PC. Whenever a PC port with an address of greater than 0FFh is accessed, the high byte is ignored, causing an access to the port whose address is indicated by the least significant byte. In other words, all four pages of PC ports are mapped into the single page of Z-100 ports.

A8-AB	Primary Hard Disk Controller
AC-AF	Secondary Hard Disk Controller
B0-B7	Primary Floppy Controller
B8-BF	Secondary Floppy Controller
D8-DB	Video Control Port
DC-DD	CRT Controller
DE	Light Pen
E0-E3	Parallel Port
E4-E7	Timer
E8-EB	Serial Port A
EC-EF	Serial Port B
F0-F1	Slave Interrupt Controller
F2-F3	Master Interrupt Controller
F4-F5	Keyboard
FB	Timer Status
FC	Memory Control Latch
FD	High Address Latch
FE	Swap Port
FF	DIP Switch

Figure 3
Z-100 Port Usage

If you want to become an expert at the art of ZPC patching, you'll need to compile all the information you can about how each I/O port in the IBM PC and the Z-100 is programmed, and what each bit of each byte does. This way, you can go through a particularly troublesome patching problem, and analyze which port accesses are benign, and which are serious.

Reading the Port Status

One of the most common problems associated with port accesses involves programs that try to read the status of a port. Reading a port will never cause anything to crash directly — it is a harmless act. The problem comes when the program insists on waiting for a particular port status to change.

A good example of this is when a program wants to write directly to video RAM on a CGA display. On some IBM PC models, if the video memory is accessed at the same time the screen is being refreshed, interference will appear on the screen. In order to avoid this problem, many programs check the video status register (port 3DAh). Bit 0 of this port indicates that it is okay to access video RAM. The code used to check the video status may look like this:

```
      MOV    DX, 3DA
LOOP: IN     AL, DX
      TEST   AL, 1
      JZ     LOOP
```

The problem here is that when the program tries to read port 3DAh, it really reads port 0DAh, which is the CRT-C address latch port on a Z-100. Chances are good that bit 0 of the address latch port will never satisfy the test in the code above, so the program will loop forever. The result, from the operator's point of view, is that the computer is locked up.

The patch that is required to correct this situation is really simple. Simply replace the IN, TEST, and JZ instructions with NOP's. Or you could just replace the JZ instruction with NOP's, since reading the status isn't going to hurt anything. Or you could replace the IN instruction with a JMP to the end of the loop. Whichever way you do it, the result is that we eliminate testing the port status. This simple example should demonstrate that there may be several ways to make a patch to a program.

Now I know a few of you are asking "how will the program operate with this code removed?" That's a good question, and one which you will always have to consider when making patches. In this particular example, the patched out code won't make any difference, because the Z-100 doesn't have any problem with interference when writing to the screen. But in other cases, the status read from the port may be used by subsequent instructions. If this is the case, you may have to tailor your patch to provide the desired response. In our example above, suppose that the program saved the status byte and used it for some other test. Then we might make a patch that looks like this:

```
MOV   DX, Ø3DA
MOV   AL, 1
JMP   LOOPEND
```

Writing to a Port

A completely different set of problems crop up when a program tries to write data to a port. Now, instead of just being a harmless access which may result in an endless loop, we have the program trying to tell the hardware what to do. And in every case, the expected hardware is not there. The best we can hope for is that the port instruction will be ignored.

Suppose that a PC program wants to change the graphics mode, and elects to do it by writing directly to the video mode control register (port 3D8h) instead of using the BIOS services. The access to port 3D8h will be mapped to port 0D8h in the Z-100, which is the video control register. It is only a coincidence that both of these port addresses happen to be video control registers. The sad fact is that any access to the PC's video mode port is going to make strange things happen to the Z-100's screen display.

There are hardly any programs that write to port 3D8h to change the video mode, since a lot of other overhead is required in addition to this simple register access, and since it is so easy to use the PC's ROM BIOS to make the mode change. But there are other video ports that are used commonly, and can cause just as disastrous of effects.

The solution for cases which cause problems is simply to patch out the offending instructions with NOPs, and hope for the best. Obviously, since the PC program was trying to write something to a port, there may be ramifications if it doesn't get done. In some cases, the program will work correctly under ZPC anyway.

The purpose of this discussion has been to try to give you a 'feel' for the reasoning behind the patches. The explanation has been conspicuously shy of concrete examples, mainly because it would be impossible to cover all the possibilities. And I don't want to mislead you into thinking that you can look for a few fixed samples of code. You MUST understand the logic of what needs to be patched . . . then you can worry about the specific code fragments that need to be located.

Reading the ZPC User's Manual is an important part of the learning process. Along with a description of which interrupts and other services that ZPC emulates, are discussions of program patching. For some of you who would like to see some concrete examples of program patches, the ZPC manual has a few.

Before we finish our discussion of patching accesses to I/O ports, I need to call your attention to the section "Patching Programs: the Keyboard Interrupt" in the ZPC manual. This is required reading for all prospective patchers. The keyboard port (port 60h) naturally does not exist at that port address in the Z-100, but it is such an important service, that some provision had to be made to emulate it. This was handled by using two interrupt handling routines; INT 90h and 91h. Generally speaking, you can patch any instructions which read from ports 60h and 61h with a software interrupt to INT 90h or 91h, respectively, and the code returned in register AL will be a value appropriate for the original port access.

Unsupported Interrupts

Another thing which may cause problems when running PC programs under ZPC is an unsupported interrupt. Of course since the program is really running on a Z-100 computer, there won't be any unsolicited PC hardware interrupts to worry about, but the program may issue software interrupts that cause problems.

Figure 4 shows a list of IBM PC/XT interrupts.

Int	Type	Description
00	System	Divide by Zero
01	System	Single-Step
02	System	Non-Maskable Interrupt
03	System	Breakpoint
04	System	Overflow
05	BIOS	Print Screen
08	Hardware	Timer
09	Hardware	Keyboard
0B	Hardware	COM2
0C	Hardware	COM1
0D	Hardware	Hard Disk
0E	Hardware	Floppy Disk
0F	Hardware	LPT1
10	BIOS	Video Services
11	BIOS	Equipment Check
12	BIOS	Memory Size
13	BIOS	Disk I/O
14	BIOS	Serial I/O
15	BIOS	Cassette I/O
16	BIOS	Keyboard I/O
17	BIOS	Parallel I/O
18	BIOS	Resident BASIC
19	BIOS	Bootstrap
1A	BIOS	Timer
1B	BIOS	Keyboard Break
1C	BIOS	Timer Tick
1D	BIOS	Address of Video Parameters
1E	BIOS	Address of Disk Parameters
1F	BIOS	Address of Graphics Characters
20	DOS	Program Terminate
21	DOS	Function Request
22	DOS	Terminate Address
23	DOS	Control-C Address
24	DOS	Fatal Error Address
25	DOS	Absolute Disk Read
26	DOS	Absolute Disk Write
27	DOS	Terminate/Stay Resident

Figure 4
IBM PC/XT Interrupts

00	Hardware	Divide by Zero
01	Hardware	Single-Step
02	System	Non-Maskable Interrupt
03	System	Breakpoint
04	System	Overflow
05	BIOS	Print Screen
20	DOS	Program Terminate
21	DOS	Function Request
22	DOS	Terminate Address
23	DOS	Control-C Address
24	DOS	Fatal Error Address
25	DOS	Absolute Disk Read
26	DOS	Absolute Disk Write
27	DOS	Terminate/Stay Resident
40	Hardware	Parity Error
41	Hardware	Processor Swap
42	Hardware	Timer
43	Hardware	Slave IC
44	Hardware	Serial Port A
45	Hardware	Serial Port B
46	Hardware	Keyboard/Light Pen/Vertical Retrace
47	Hardware	Parallel Port

Figure 5
Z-100 Interrupts

The interrupt table for a Z-100, running in native Z-100 mode is shown in Figure 5.

You should note that in both of these interrupt table descriptions, there are other interrupts which have been defined. The ones shown are those which are most commonly used.

Of these, the first five (0 through 4) are defined by Intel for an 8088 system, and therefore, are equivalent between a PC and Z-100. Interrupt 5 is for support of the Print Screen key, and will typically not be found in a program. Interrupts 6h through 0Fh are generated by the PC hardware, and are difficult to emulate with software, but ZPC does synthesize an interrupt 9h to indicate keyboard activity.

Interrupts 10h through 1Fh are all vectored to entry points in the PC's BIOS ROM, and are typically called via a software interrupt in a program. These interrupts are very commonly used, and luckily, they are undefined in the Z-100's normal interrupt table (compare with Figure 5), and therefore, available to use for emulation purposes. ZPC does a good job in emulating these interrupt services. The specific interrupts which are supported by ZPC are listed in the ZPC User's Manual.

The Interrupts between 20h and 27h are MS-DOS defined interrupts, and are compatible between the PC and Z-100.

Even though many of the interrupts have been emulated successfully by ZPC, many have not. It is these cases where you will receive the infamous "WILD INTERRUPT" message when attempting to run PC software. If you run into a program that uses an unsupported interrupt, about the only thing you can do is patch the instruction out with NOPs and hope for the best. If you're lucky, the interrupt call will not affect the operation of the program, and everything will work okay. But if the program is trying to perform Disk I/O through the BIOS, or other critical functions, the program will almost certainly crash.

The Graphic Character Table

Whenever a PC program writes text on the screen while in graphics mode the text character font designs are looked up in a table in PC memory. This is known as the graphic character table, and it is located at address F000:FA6E. It would be nice if this table could be put in that location for emulation purposes, however, the Z-100's MTR-100 ROM occupies that space already. (An experimental MTR-100 ROM version has been independently developed which includes this table, but it is not yet available — see Z-100 LifeLine Issue #4 for further discussion.)

The cure for this problem is relatively simple. Simply find where the PC program is accessing this table and patch the address so that it points to the table ZPC has provided, which is at memory location B000:0000. There are a number of different instructions that a PC program might use to access this table, but it is usually pretty easy to find the table references by searching for the FA6E address. Remember when you are conducting your search with DEBUG, that the byte order that Intel uses places the least significant byte first in a word. Therefore, you should search for the bytes 6Eh, 0FAh, in that order.

BIOS Data Segment Accesses

In a real IBM PC, the segment beginning at paragraph 40h is used by the BIOS as a data segment. This area is normally used as a BIOS jump table by the Z-100, but Pat Swayne has figured out how to make this area available for use by ZPC for PC BIOS data. Most of the data values maintained in this area by ZPC are typical of what you would find on a real PC, and won't cause any problems. But if you run into a program that is trying to manipulate the PC BIOS type ahead buffer directly, or trying to determine hardware status, it will have problems. About the only thing you can do in these cases is patch the program so that it returns with the expected value. This may or may not affect a cure.

Specific Hardware Constraints

Luckily, most PC software is fairly well behaved. That is to say that most software uses the BIOS or DOS for communication with the hardware. The major exception is direct access of the video RAM, but ZPC has solved this problem nicely.

Some programs, however, will insist on communicating with the hardware directly. Any attempt to communicate directly with the video ports, serial or parallel ports, interrupt controller, DMA controller, keyboard chip, etc. must be patched. In most cases, patching out an access to a hardware device will cause the program to crash anyway. The most likely candidates for patching are the video port accesses. If you have a ZHS circuit with COM ports installed, direct accesses of the COM ports should be okay.

Tough Dogs

When you consider how many things could prevent a program from being run successfully under ZPC, it is incredible that any programs work at all. As it turns out, most programs can be made to work under ZPC, if you spend enough time trying. And you will find that most programs only require simple patches for video port accesses or the graphic character table.

There are times, however, when extraordinary measures may be necessary. If you have gone through a program and patched everything in sight that looks illegal and the program still refuses to run, you may want to reconsider just how important it is to use this program on the Z-100. If you still want to proceed, then you can try to single-step through a program, using DEBUG or SYMDEB, until you find the cause of the problem. This can (probably will) be a laborious task which requires real programming expertise. Even so, if a program uses overlays, is self modifying, or is memory resident, you may be destined for failure. *

OS/2 on the Zenith Computer

Harold C. Ogg
357 W. Diversey Avenue
Addison, IL 60101-3508

A revolution in PC operating environments was promised by the authors of Operating System/2 (OS/2). True multitasking and sophisticated memory management were finally within reach. But will OS/2's RAM-hungry tendencies inhibit buyer acceptance?

The ability of OS/2 to handle simultaneously running programs and tasks in a foreground/background mode has launched one of the longer running editorial debates of the late 1980's. It is not the intent of this article to weigh the pros and cons of OS/2, but rather, to share some experiences on the use and exploitation of this software. The emphasis of this essay will be the encouragement of the reader to seek further reading where detailed product specifications are required.

In December 1987, Zenith Data Systems (ZDS) became the first non-IBM microcomputer manufacturer to ship its release of Microsoft OS/2. IBM had earlier shipped the IBM OS/2 — its slightly different version of what is, in effect, the multitasking version of the Microsoft Disk Operating System (MS-DOS).

Tagged (among other labels) the "DOS 5.0" package, OS/2 was intended to achieve simultaneous handling of both related and unrelated processes. At the same time, it was supposed to run most of the existing MS-DOS programs from its so-called compatibility box, in order not to render obsolete the exhaustive user base of DOS applications.

OS/2 was to be a major step in unleashing the full potential of the Intel 80286 and 80386 processors and breaking the memory and speed limitations of the long established 8088 and 8086 chips. Further, it was to close the gap between the IBM-compatible world and the rival Motorola 68000-based Amigas and Macintoshes — which had already achieved some degrees of success in the multitasking realm.

The schedule of releases, according to IBM, indicated a logical progression: the initial offering of OS/2 (which Zenith released to its users) would be version 1.0. A subsequent, or "enhanced" 1.0 issuance would follow (and did, in July 1988), and the version 1.1 with the Presentation Manager inclusion was slated for release in late 1988. Ultimately, the "standard" OS/2 version 1.1 with the Presentation Manager was released in October 1988, followed by the Extended Edition, version 1.1, the following month.

Zenith Data Systems released its version 1.1 in the second quarter of 1989. Although the trade literature states that IBM's upgrade from version 1.0 to 1.1 is free to registered users, ZDS has placed a $135.00 update fee on its release. Only the standard editions will be released for resale by non-IBM vendors, because the extended editions contain three components — the data base manager, the communications manager, and the LAN workstation software — that are proprietary to IBM equipment.

The Presentation Manager resembles Microsoft Windows, although software written with the MS-DOS Windows Developer's Edition is not guaranteed compatible under OS/2. The significant feature of the Presentation Manager is a user interface like that of Windows version 2.0. Also included with OS/2 version 1.1 is a graphics programming language. Current users of Windows will find available the same features, including mouse support, under the Presentation Manager.

Zenith offers OS/2 version 1.1 under part number OS-51-4 (OS-31-4 if 3-1/2" media are desired), and sets its price at $339.00 — against IBM's tariff of $325.00. However, Heath Users' Group members can take advantage of their usual 20% discount, giving them an actual price of around $271.20 plus shipping and tax. I obtained my copy by mail order; and actual shipping time was a bit over three weeks because the product was backordered.

If you get OS/2 on 5-1/2" floppy diskettes, two of the 1.2-megabyte variety are required. However, even with high-density disks, OS/2 is virtually unworkable on a floppies-only setup. You'll need a hard drive. For example, the SWAPPER .DAT file needed on an OS/2/MS-DOS dual-boot setup requires 650K of disk space. Even with 1.44-MB floppies, OS/2 would guarantee a considerable amount of disk swapping to invoke its full power.

OS/2 requires a machine with a 80286 or 80386 CPU. I have been running OS/2 on a Z-248, which is one of the first 80286-based machines released by Zenith. There is not a separate version of OS/2 for the different AT-class machines that Zenith manufactures. Although a 20-MB hard drive is recommended, my experience has been that a 40-MB drive is the smallest usable configuration. And you'll need at least 2.5 megabytes of RAM (inclusive of the 640K used by MS-DOS) for a minimally sufficient work area. OS/2 shunts operations for which there is not sufficient memory onto workspaces on the hard drive. Obviously, with more RAM available, OS/2 makes fewer disk accesses. Version 1.0 of OS/2 can address up to 16 megabytes of memory, so your pocketbook will dictate quantities over the 2.5MB threshold in your particular machine.

Documentation and Installation

Documentation accompanying the software consists of three spiral-bound manuals containing a setup guide, a startup handbook, and a user's reference. The Programmer's Reference, although mentioned in the manuals, is not included. The manuals' format is typical of ZDS, readable but not reflecting the thorough consideration to detail and rewrite of Heath of yesteryear.

Each manual has its own separate index, and there are no cross references to the other two volumes. ZDS says that OS/2 is "for experienced users"; however, the manuals are written in a tone similar to a business office tutorial for less advanced users.

Users seeking more advanced information will, at least for the present time, be somewhat disappointed with the literature in the bookstores. Most third party books are rehashes of the user manuals, and there isn't the detail base of "tips and techniques" material that exists for MS-DOS. I have found a couple of references which are useful, however, and these are listed at the end of this article.

The installation procedure is straightforward. A few pages into the setup manual, you'll see that some prior experience with matters of installation will prove advantageous. A ZDS staffer said that this OS/2 "was for the accomplished programmer," but I think that this is an overstatement. The setup is reasonably quick with only a few uncomplicated menu choices of drivers, ports, and translation tables that must be entered.

The installation is much easier to accomplish if MS-DOS is already present on the hard disk. You can then begin to run OS/2 after about ten minutes of installation work.

Situations and Solutions

Near the end of the installation process, I initially experienced a lockup that defied all attempts to correct. After the "commercial" for Microsoft, the screen would blank and the keyboard would go dead; it would require a reboot.

Fortunately, the documentation came with some addenda sheets that alluded to the Zenith Enhanced Graphics (EGA) board — the Z-449 video card. There is a possible conflict due to an interrupt routine it uses; apparently, this is related to the board's ability to sense the monitor type you have and automatically switch to support it. There is not much explanation regarding other EGA cards, but essentially you must disable the auto-switch feature on EGA video boards that support such a convention. If you do so, be careful that the manual switch settings match your type of monitor.

Other than the initial run-in with the video card, the remainder of the installation process unfolded smoothly.

The very first time through, the install program asks whether you wish to make a backup copy of the distribution diskettes. Until you do, the message will keep reappearing at each bootup, so you probably ought to make a backup copy as soon as possible to stop the nagging. You must also run SETCOM40 to let OS/2 know of the COM: port's existence.

There are only a few specifics asked of the installer, such as the choice of mouse driver and boot system (on a dual operating system setup). There are several mouse drivers provided, and redundant ones can be erased. You can also use MOUSE.COM, which is provided with certain other Microsoft products. If you are using a Logitech mouse and wish to take advantage of Logitech's preset menuing utilities, note that its driver must also be installed for use with certain programs.

Running on AT-class (80286- and 80386-based) machines, Logitech's bus mouse has the advantage of allowing its port assignment to be LPT2:, thus freeing a COM: port for other uses. (This isn't allowed on 8086/8088-based machines.) I created two partitions on a 40-MB hard drive. I found it simpler to place the OS/2 supplied MOUSE03.COM driver on the OS/2 partition and the Logitech driver on the MS-DOS side.

When you specify the default bootup system at setup time, you can choose OS/2, MS-DOS, or a manual selection. For the manual selection, you will be offered a prompt for choice at boot time. And OS/2 comes with a utility that allows you to change your mind about the default boot system. If you wish temporarily to boot from the alternate system, you can override the default setup choice by holding down the ALT key at powerup.

Physically removing MS-DOS (i.e., to eliminate the dual boot setup) is difficult and involves a reinitializing of the hard disk. The Zenith OS/2 does not make provisions for you easily to change your mind, and you should decide at installation what kind of setup (OS/2-only or dual system) is best for you. The version 1.0 documentation reassures that an update to a newer version of OS/2, however, is a much simpler matter of installing the new software over the old.

Familiarities

When running, OS/2 version 1.0 operates from a session management screen. This display lists all currently running software, and can be configured as a selector list for commonly used programs. Entries on the list are mouse or keyboard selectable. Version 1.1, with the Presentation Manager, operates (obviously) from a more windowlike display.

The OS/2 utilities are remarkably similar to those in MS-DOS, and deliberately so. One added feature is that popup help screens are available both in real and protected modes (discussed below). On an error, the OS/2 error message is returned with its corresponding number. For further information, you just type 'help' plus the error number. The help feature can be turned on and off. As in MS-DOS, a command that produced an error can be retried, aborted, or you can return to the operating system or running program.

Batch files of operating system commands are similar in appearance, except that they will have the extension .CMD rather than MS-DOS's familiar .BAT. You can call one .CMD file from another, and embedded comments (remarks) are allowed.

Hardware Considerations

On a Z-248 computer, ZDS OS/2 requires version 1.9 or later of the boot routines stored in read-only memory (ROM). Two ROM chips are required; these are available from ZDS as part numbers 444-423-10 and 444-424-10 at $13.05 each (as of summer 1989) plus shipping.

Even if your machine contains exactly version 1.9, you will probably want to update to take advantage of some added firmware enhancements. For example, the later ROM version supports 1.44-MB 3-1/2" floppy drives. Version 2.0f is the earliest ROM that will support 1.44-MB drives, and to run 1.44-MB drives under MS-DOS you need ZDS's 3.21 update to version 3.2; OS/2 already supports the 1.44-MB drives.

OS/2 also supports 360-kilobyte 5-1/4" floppies under the same format as MS-DOS. Because of this compatibility, and a similar compatibility with hard drives, it's not necessary to reformat your data and program disks upon installing OS/2.

Also, if you haven't done so, recall the setup ROM and tell it that there is a B: floppy drive — even if you don't have one. Some RAM-drive utilities (such as AST's Superdrive) need to think that there is a physical drive attached before they can create a logical one in its place. This isn't a replacement of an existing drive, only an assignment of an address for the RAM-drive to use. Most RAM-drive utilities allow you to configure a particular logical/physical setup, and you don't lose any existing drives in the process.

One other hardware consideration needs to be taken care of. Some of the older keyboards have a decoder ROM that causes erratic typing behavior with the new operating system. You can experience both dropped characters and double characters. The easy solution is to order a new keyboard ROM (version 1.1, part number 444-400-2, $12.50) from ZDS.

Not so welcome is the fact that many of the older keyboards have the ROMs soldered directly to the printed circuit board. It took me about two hours with a solder vacuum to replace the chip, but this did indeed cure the typing problems.

Life on the Hard Disk

One area that ZDS (and OS/2) did not address is the old, familiar limitation: You still can't have any more than 32 megabytes per logical disk partition.

To be sure, the FDISK program, an OS/2 utility, lets you set up an extended partition on physical drive C: — which OS/2 can access as logical D:. But you still cannot set up a physical or logical drive greater than 32 megabytes.

The process is rather straightforward: using the PREP and FORMAT utilities provided with MS-DOS, prepare the hard disk in the usual manner. If multiple partitions are created, only one will be designated as the bootup partition. MS-DOS PREParation is compatible with the OS/2 environment, and the only restriction is that two operating systems can't be assigned to the same partition once the process is complete. Access violations will abound if you try.

Conspicuously absent from the OS/2 distribution diskettes are any hard drive preparation utilities. OS/2 developers apparently figured that AT-class hard disk manufacturers will provide them with the preparation software. An example of this is the Ontrack Disk Manager program that Seagate often packs with its hard drives. Disk Manager, through a process of increasing the disk sector size does allow partitions greater than 32 megabytes so that drives of up to 300 megabytes can be employed.

A La Mode: Which One to Choose?

As I mentioned above, the Zenith version of OS/2 supports two modes of operations. You can run an OS/2-only setup, or you can choose whether you'll boot OS/2 or MS-DOS. The dual personality is preferable — especially now while the DOS world continues to dominate.

Even if you choose the OS/2-only setup, however, OS/2 permits two modes: "Protected" mode uses the full power of the 80286 and 80386 processors, and is most important for multitasking operations. In "real" mode, an 80286 or 80386 emulates an 8086/8088 processor. The real mode is OS/2's MS-DOS compatible mode; running as a single-tasking operation, it should carry the ball for all the MS-DOS software.

Another reason that the dual-mode schizophrenia should be maintained is that many device drivers (.SYS and .DVD files) will work under MS-DOS but not under real mode OS/2 — even though the latter ostensibly functions as if it were MS-DOS. Also, some MS-DOS utilities whose counterparts are not supplied with OS/2, exhibit similar non- or misbehavior in real mode. (One example is the Zenith-supplied ZCOM.EXE file-transfer utility.)

That certain device drivers will function only under MS-DOS is easily verifiable — just try mouse and scanner drivers that predate OS/2. If you wish to abandon the DOS partition, you may have to set aside expensive hardware until the manufacturers play catch-up and provide you with compatible drivers.

However, the dual system has its problems, too. An example is the driver conflict that shows up when you use VDISK.SYS for a RAM drive. If the dual MS-DOS/OS/2 setup is employed, two appropriate copies of VDISK.SYS must be provided at boot time. (There are separate versions of VDISK for real and protected modes.) VDISK.SYS resides in the CONFIG.SYS file for MS-DOS and in CONFIG.OS2 for OS/2.

In the two CONFIG files, you may very well specify different sizes for each operating system's RAM disk. This sets up the possibility that you could forget which size RAM drive was currently present. And any RAM drive will reduce the amount of available extended memory for multitasking.

Also, VDISK assigns the drive letter for the RAM drive; it doesn't allow you to specify a letter of your own choice. Dissimilar letters can be assigned by the two versions of VDISK if you are using a dual-boot setup. This will be a problem if you make extensive use of batch files that target a specific drive for caching. A better solution is to use the RAM-drive software that was supplied with your extended (expanded) memory board so that the RAM drive will always carry the same letter designation.

Linear vs. Paged Memory

When it comes to memory above the 1 MB boundary, AT-class machines and OS/2 prefer to address extended and not expanded memory — "linear" memory with addresses above 1 MB on 80286/80386 machines. The OS/2 manual, in fact, specifies extended memory in most discussions.

But 80286/80386 machines and OS/2 can also use the (expanded) memory that was originally introduced to get around the 1-MB limitation of 8086/8088 machines. The so-called Lotus-Intel-Microsoft Expanded Memory Specification (LIM EMS) standard governs add-on memory that's paged into a vacant address above 640K and below 1 MB.

An example of a true extended memory board is the AST RAMvantage; this is linear memory, and the 80286/80386 chip family usually accesses it through interrupt services that invoke the paging process. Extended memory boards were once more popular than now; I have seen such boards at computer flea markets selling for as little as $40 (bare board).

Some of the newer memory boards, such as the RAMpage and Above Board, are hard wired for EMS. A .SYS driver must be installed to let the Z-248 access this as extended memory. If you use OS/2 with such cards, you should recheck a particular board's documentation to ascertain whether the EMS capabilities must be physically disabled for the board to appear as extended memory.

On the other hand, if you have an extended board that doesn't want to look like EMS memory, several utility packages take up the slack: AboveDisc (Teleware West, Seal Beach, CA), V-EMM (Fort's Software), and VRam (a shareware program by Biologic Company, Manassas, VA) install as terminate-and-stay-resident (TSR) programs and permit emulation of expanded memory for most software utilizing it. These programs emulate the Lotus-Intel-Microsoft 3.2 specification, and newer versions support the LIM 4.0 standard.

Some users claim that adequate response time cannot be guaranteed with such utilities; but my experience has been that none of the memory management packages discernibly impair the response of most applications programs. These so-called "LIM-ulators" perform the page mapping that expanded memory requires. They can fail if an attempt is made by the applications package to cause a page to be mapped at two different addresses. This occurrence is remote, however, and I am not aware of any programs with which the situation arises.

Given the availability of the various boards and MS-DOS utilities, OS/2 doesn't have a unique talent on the handling of large amounts of memory. OS/2's real power lies in its ability to handle memory management functions. OS/2 can "overload" the operating system and run more programs at one time than available memory would otherwise allow. The process is basically just a matter of swapping executable modules in and out of virtual memory, occasionally to and from the hard disk.

In protected mode, OS/2 manages memory by accessing an 80286- and 80386-only feature known as a selector. The selector, as its name implies, chooses a segment to which will be attached a running program and so allows executions above the 1-MB address threshold. (The 8086/8088 chips can theoretically access 16 megabytes of memory; but because of MS-DOS considerations, and the fact that these chips do not have the selectors in their architectures, they are cut off at the 1-MB boundary). When using the "overload" method, one should remember to leave some hard disk space for OS/2's spooler to use as a scratchpad.

Alternatives

One OS/2 drawback (although a fading problem) is that many of the smaller shops are not rallying to write applications programs specifically for the new operating system. But applications houses are beginning to yield to consumer demand for OS/2 compatibility, and many of the comprehensive software packages, such as dBASE IV, R:BASE, Lotus 1-2-3, Ventura Publisher and WordStar all have or have promised OS/2 compatible versions.

The question for an individual user, however, is whether new programs will bring benefits proportional to the cost — not only for the programs themselves, but for the memory they and OS/2 will require. The decision lies between the choice of settling for a 640K machine and one of the new multitasking utilities (mentioned below); or of purchasing a 2.5 MB (or greater) memory board for an AT-class computer to achieve multitasking through OS/2.

MS-DOS 4.01 is now available and multitasking (or at least the appearance of it) is available through such products as Concurrent DOS, Xenix, PC-MOS 386, Windows-386 and Desqview. Because of this, and because of the RAM-hungry nature of OS/2, many people are rethinking whether they'll move "up" to it.

Desqview and similar programs work by swapping executable files in and out of memory (expanded memory if available), but they can operate exclusively in the machine's lower 640K — which renders them usable on an 8086/8088-based IBM compatible. The process is slow yet effective — acceptable if precise timeslicing is not critical.

For programmers, too, options are available for non-OS/2 multitasking development: Lattice Software, for example, has announced a version of its native C compiler whose executables will work under Concurrent DOS; this language release is available along with version 3.4 of Lattice's MS-DOS/OS/2 C compiler.

Help for Developers

Programmers, especially C and Assembly Language authors, who wish to get the full benefit of OS/2, should remember that they need to use a linker that can generate protected mode programs. Under OS/2, it's possible to create ordinary stand-alone programs that are complete in themselves after your language compiles them. However, the documentation recommends that your programs use OS/2's dynamic linked libraries (DLLs). These files, resident on the user's disk, provide a runtime library much like the ones used heretofore by dBASE III Plus' dBRUN and C BASIC.

I expect that DOSCALLS.DLL will probably be the most popular dynamic linked library — especially where the C language attempts function and system calls that OS/2 doesn't recognize. Some C calls, like int86() are incompatible with OS/2 and are thus "real mode only." Applications programmers, especially those creating data bases and writing in the RPG language, will notice immediately the absence of file handles. (No 'files = ' appears in the CONFIG.OS2 file).

For the time being at least, users of Zenith equipment will have to wait for full network support. For network programmers, IBM for its Extended Edition version 1.1 has included networking capabilities. In other words, this IBM release (mentioned as the "enhanced" version above) is for IBM boxes only, not the compatibles (including those manufactured by Zenith).

In evaluating program development packages, such as compilers, one should ascertain whether the documentation compares the differences between previous releases and the one currently available. Watch for things like the reentrance capabilities of library functions and thread maintenance. Lattice C for Concurrent DOS, for example, has given much attention to looking in your source code for calls to MS-DOS' interrupt 33 hexadecimal, which handles pointer control functions. This interrupt is valid only under MS-DOS and has to be changed into Concurrent DOS' interrupt 224h.

Windows development and the Presentation Manager have already been mentioned in this article. It may come as no surprise to programmers that the tools for writing support modules for these two packages are steeped heavily in the C language. Compiler vendors have recognized the need to base their products in such a versatile language, and Lattice and Microsoft in versions 3.4 and 5.1, respectively, of their C compilers have aggressively and extensively addressed OS/2 interfacing. The literature bears this out, and fluency in C is essential for understanding both the DOS (windows) and OS/2 (Presentation Manager) interfaces at the code level.

For Further Information:

Two comprehensive works are Kris Jamsa's *"Using OS/2"* and Ed Iacobucci's OS/2 *"Programmer's Guide"*. The latter is quite essential for C and developmental language programming, and gives a thorough explanation of the OS/2 all-purpose interface (API) and dynamic linked libraries (DLL's).

To be able to handle OS/2 concepts, such as threads, semaphores, and time-slicing, the study of the more sophisticated literature is mandatory. Two other recently published references for these concepts are William Murray and Chris Pappas' *"Assembly Language Programming Under OS/2"* and Ray Duncan's *"Advanced OS/2 Programming"*.

On-line sharing of information is also available on the various electronic bulletin boards. Lattice Software, for instance, is currently running an OS/2 conference for registered users. And the Byte Information Exchange (BIX) has a similar conference.

Recent periodicals contain interesting and useful articles. The reader is directed to *"IBM OS/2 Standard Edition,"* a review by Eva M. White in Byte magazine, June 1988; and *"The Flexible Interface,"* by David A. Schmitt in PC Tech Journal, November 1987. The latter is particularly valuable for its summary of MS-DOS-related functions. PC Magazine's April 12, 1988 issue contained an article by Charles Petzold entitled *"OS/2: a New Beginning for PC Applications,"* which was quite valuable for its overview of the OS/2 operating system from the user's viewpoint. He followed with *"OS/2 Extended Edition"* in the January 31, 1989 issue of the same periodical with an essay that clarifies the features and differences of the four IBM releases.

In addition to the extended memory management utilities mentioned above, there was a series of articles in the Power Programming column by Ray Duncan entitled *"Using Extended Memory"* in PC Magazine of May 30, June 13 and June 27, 1989 which lent exhaustive treatment to the subject. Also in the June 27 issue in the Utilities column by Douglas Boling was another feature of extended memory management. Both his and Duncan's articles offered Assembly source code and documentation for two .SYS drivers which simulate expanded memory.

Finally, Tony Rizzo and Richard Hale Shaw published *"OS/2 — a New Perspective"* in their Lab Notes column in PC Magazine's July 1989 edition. This was a particularly informative article on the Presentation Manager, and conforms to ZDS's version 1.1 release.

Suggested OS/2 References

Advanced OS/2 Programming
by Ray Duncan $24.95
Programming the OS/2
Presentation Manager
by Charles Petzold $29.95
Microsoft Press
16011 NE 35th Way
Box 97017
Redmond, WA 98073-9717

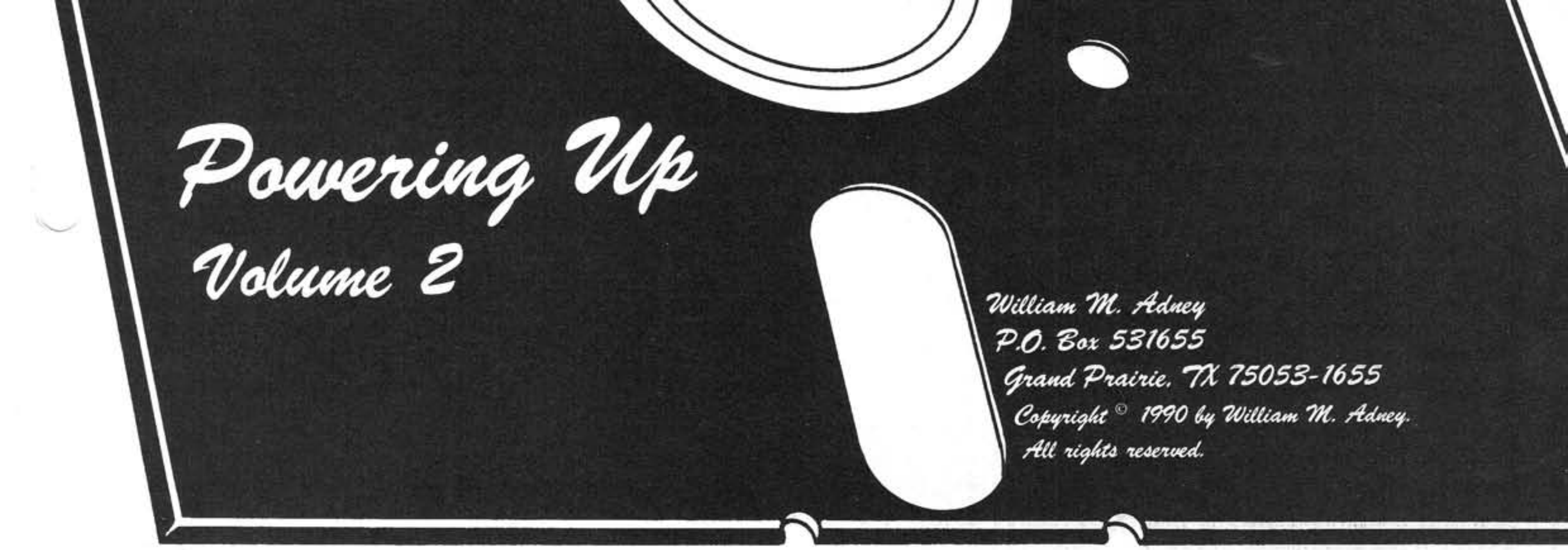
Assembly Language Programming
Under OS/2 by William H. Murray, III
and Chris H. Pappas $24.95
OS/2 Programmer's Guide
by Ed Iacobucci 24.95
Using OS/2 by Kris Jamsa 19.95
Osborne McGraw-Hill
2600 Tenth Street
Berkeley, CA 94710

OS/2 Operating System $339.00
Zenith Data Systems
Hilltop Road
St. Joseph, MI 49085

Users upgrading from version 1.0 to 1.1
Zenith Data Systems
Software Registration
St. Joseph, MI 49085

Various ROMs mentioned in this article
Heath Company
Parts Department
Benton Harbor, MI 49022
(616) 982-3571

*



What a Disk Really Contains

An understanding of what a disk really contains is important, especially when you just realized that you deleted an important file with the dreaded "DEL *.*" command. Or when you see a CHKDSK message like "5 lost clusters found in 1 chains. Convert lost chains to files (Y/N)?" Or, a message indicating that a file "contains invalid cluster, file truncated." Or, a file is "cross linked on cluster 9." These are all possible messages that CHKDSK can display, and what they mean is sometimes a mystery, even to experienced computer users. More importantly, they are trying to tell you that at least one file on a disk is partially "destroyed" because it cannot be completely and accurately read from the disk by DOS. For many users, their first inclination is to panic when they see a message like this because it is obvious that DOS is having a problem reading a file. And of course, an extension of Murphy's Law will be true because it will always be a problem with a critical file that occurs at the worst possible time, such as when your boss needs that latest information from your spreadsheet NOW!

Perhaps you have wondered why a 360 K, 5.25-inch floppy disk does not really hold 360 kilobytes worth of files. Or, why is it called a 360 K disk in the first place? In this article, you will see how that really works out, and you will see how to verify some of the information in this article using CHKDSK. For those of you who have computers with only 3.5-inch drives, I have also included information about them too. Although the general information in this article about disk formats also applies to hard drives, I have chosen to use only floppy disks because it is easy to see the results of what happens with CHKDSK and the results are always the same. Let's begin by looking at the four "areas" that are always created by the FORMAT command on all disks, hard drives and floppies.

How a Disk is Organized

Like everything else in your computer, there is a method in the apparent madness in the way a disk is organized. The primary reason for this organization is so that DOS can find a file on a disk when you ask for it. It may be a program file or a data file — it really does not matter. Nor does it matter whether you are talking about a floppy disk or a hard disk, the basic organization is still the same.

When you run the FORMAT command to initialize a new disk, FORMAT actually creates four major areas on a DOS disk: the Boot Sector, the File Allocation Tables (two of them), the Disk Directory, and the Data or File Area. Do not confuse this with the four DOS components that were discussed in the last article, although this article will include a discussion of where these components are actually stored on a disk. Here, we will be talking about the way a disk is organized, and where files are actually stored. Let's begin by looking at some basic disk terminology.

Disk Terminology

All information is stored on all disks in the same way, regardless of the type or size of disk. Disks are divided into a series of concentric circles, and each circle is called a *TRACK*. Each track is a complete circle, not a spiral like you will find on a phonograph record. Tracks are numbered from the outside track to the inside track, beginning with track zero (0), which is actually the first track on the disk. For example, a standard double-sided 5.25-inch floppy disk (360 K) has 40 tracks on each side, numbered from track 0 to track 39, and a standard 3.5-inch disk (720 K) has 80 tracks on each side, numbered from 0 to 79. If you have a relatively current Zenith MS-DOS version, you will find that the FORMAT command reports which CYLINDER number (same as track number) it is formatting, and if you watch carefully, you will find that FORMAT begins formatting cylinder 0 and ends when it reaches cylinder 39 on a 360 K disk or cylinder 79 on a 720 K disk.

On a double-sided disk, there are tracks on both the bottom and top sides of the disk. If you look carefully at the internal construction of a disk drive (either floppy or hard drive), you will find that the read/write HEADS for that drive are attached to a *single* movable arm. For a double-sided floppy drive, there are two read/write heads: the one on the bottom is head 0 (for side 0) and the one on the top is head 1 (for side 1). Because both heads are attached to a single movable arm, a command that moves head 0 to a specified track will also move head 1 to the SAME track. A group of tracks on a disk with the SAME track number is called a *CYLINDER*, and the name is based on the imaginary cylinder-shaped surface formed by tracks with the same track number. For example, if the read/write heads were positioned over track 9 on any side of the disk, your computer could read or write data anywhere on *cylinder* 9 without having to reposition the heads. Before we go on, let's carry this idea one step further.

Because of the way DOS works, it is possible to have a single file stored in various wide-spread locations on a disk, say at cylinders 27, 3, 31, and 9 in that order. Both floppy and hard drives are mechanical, and it takes some time to reposition the heads over each of those cylinders in order to read a single file. What that means to you is that it simply takes longer for your system to access that data. Although we will take a look at some of the details of this later in the article, you can significantly improve the apparent speed of your system by "reorganizing" the disk so that each piece of each file is in or near the same location on a disk. Depending on what software you use, this process is called optimizing or defragmenting, and it can improve the performance of file access considerably. I personally like the Mace Utilities UNFRAG program because it works on floppy disks, as well as hard drives. By the way, one of the major reasons for the use of utility programs, like UNFRAG, is that it helps reduce file access time AND minimizes wear and tear on the disk drive.

A disk is further subdivided into pie-shaped wedges called *SECTORS*. The standard 360 K and 720 K disks we are using as examples are both subdivided into nine sectors. Although there are technical ways to change the size and capacity of each sector, virtually all DOS-based computers use a standard sector size of 512 bytes per sector. Now we know enough about each of these formats and terminology to do some calculations on disk capacity.

The 360 K disk has 2 sides with 40 tracks on each side for a total of 80 tracks. Each track has nine sectors, so there are 720 total sectors on this disk. And each sector has a 512-byte capacity, so the disk has a calculated capacity of 368,640 bytes. A kilobyte is defined as 1,024 bytes, so if you divide the 368,640 bytes by 1,024, you will find that this disk has EXACTLY 360 K of calculated capacity. For a 720 K disk, you can do the same calculation (with 160 tracks) to show that it has 737,280 bytes available, which is exactly 720 K.

An Important Tip

The whole idea of this is to demonstrate that disks are usually identified by their *calculated* capacity, NOT their actual capacity. In other words, the actual capacity of a disk is always less than the calculated capacity, and this fact is particularly important when you buy a hard drive. Many hard drives are advertised with numbers reflecting their calculated capacity, and you should always ask what the formatted capacity is before you buy, especially if you use mail order. And if a vendor is unable to answer your question about the formatted drive capacity, I would find another vendor who can. This is a simple question that anyone who answers a vendor's order line should be able to answer.

Back to the Disks

When you work with your system for a while, you will find there are a number of different disk sizes and formats that you can use. What that really means is that these disks also have a different number of sectors, tracks, and cylinders. For example, hard drives may have from 17 to 63 sectors per track, and virtually all hard drives have more than two heads. Also, hard drives typically have capacities that range from 300 to 1,000 cylinders or more, depending on the exact type and capacity of a specific hard drive.

If you have had your computer very long, you may have seen a message indicating a "Bad Sector" on a disk. What this means is that, for some reason, DOS is not able to read that sector. If you use a specific floppy disk a lot, you may begin to have this kind of problem. Or, if you find a real deal on some cheap floppy disks, you may find that the FORMAT command tells you that there are a number of bytes in bad sectors. On a new disk, you will frequently find that bad sectors tend to be physically located toward the center of the disk. Consider the physical length of the pie-shaped sectors on track 0 and track 39. A sector on track 0 is physically quite a bit longer than a sector on track 39, even though both sectors contain the same amount of data: 512 bytes. As you get closer to the center of a disk, the physical storage space for each sector is smaller, so the data are simply packed closer together. On a cheap floppy disk, you will probably find that most of the bad sectors are physically located toward the center of the disk because of the increase in data density. With this introduction to some of the physical characteristics and terminology used with disk drives, let's now turn to the first of the four disk areas: the Boot Sector.

The Boot Sector

The Boot Sector is located at sector 0, track 0 — the first sector on a disk. As its name suggests, it is exactly one sector (512 bytes) long, and it always is the first physical sector on a disk. Like each of the four disk areas, the boot sector is created by the FORMAT program when you initialize a new disk.

The boot sector contains a number of different types of information, but there are two significant items that are important for this discussion. First, the FORMAT program records a special block of information about the disk characteristics (e.g., sectors per track, etc.) in the "front" part of the boot sector. This information is used by DOS to determine how to read from and write to a disk. And you have probably guessed that the second item in the boot sector is the boot loader program, which is one component of DOS that was discussed in the last article.

The File Allocation Tables (FATs)

The File Allocation Tables (FATs) are the next disk area created by the FORMAT program. Although a lot of DOS documentation will refer to a single FAT, there are actually two FATs on every disk. Most documentation about disk formats from Microsoft says that one FAT is a backup of the other, which should tell you something about how critical the FAT is.

The FAT is really just a "map" of the File (or Data) Area on a disk. DOS uses this map to determine the physical location of a file (what space is used), as well as the available (i.e., free) space on a disk. Each FAT entry contains a specific code to indicate whether the space is being used, what space is available, and what is not usable (e.g., a bad sector due to a defect on the disk). Because the FAT is used by DOS to monitor the entire usable data storage area on a disk, two identical copies of it are stored in case one is damaged.

How the FAT is used by DOS is not difficult to understand if you ignore all of the technical details, but there is one thing you must know to see how the FAT is used. As it turns out, virtually all computers allocate storage space on magnetic media (either disk or tape) in "blocks" of data. And in a DOS-based computer, that block is called a *CLUSTER*. A cluster (or block) is the smallest amount of space that can be allocated to a file, regardless of how small the file is. For both of the disk sizes that have been used as examples, it happens that both have cluster sizes containing two sectors or 1,024 bytes. In other words, you might hear someone say that the *CLUSTER FACTOR* is 2, meaning that each cluster contains two sectors.

The number of sectors per cluster, or cluster factor, varies with different disk types and formats, especially hard disks. Without getting into all of the details, suffice it to say that the cluster factor is always some power of 2: 1, 2, 4, 8, 16, etc. The use of the cluster factor is also important in terms of understanding how DOS uses disk space. If the cluster factor on a disk is 8 (a common one for a hard disk), that means that no matter how small a file actually is, say 37 bytes, that file will still require one cluster (8 sectors times 512 bytes/sector or 4096 bytes) of storage space to store the file. DOS is more efficient using this type of space allocation technique, even though some space is wasted. The cluster factor cannot be adjusted on a floppy disk, but you can change it on a hard disk if you have Zenith MS-DOS by simply adjusting partition size with the PART command.

A cluster is the smallest possible unit of disk space allocation in DOS, and the FAT is a map of all available clusters on a disk, including those used and available. As you might expect, each FAT can be specifically identified on a disk (if you know how), and the FAT has a specific length in sectors, depending on the disk size and type. For a 3.5-inch 720 K disk, each FAT is three sectors long, so a total of six sectors is required for the two FATs. For a 5.25-inch 360 K disk, each FAT is two sectors long, so a total of four sectors is required for the two FATs. The larger-capacity 720 K disk clearly requires a bigger FAT because it has more storage space than the 360 K disk, and this is generally true as disk capacity increases. We will look at some additional information about the FAT after we finish a discussion of the two remaining disk areas.

The Disk Directory

The third disk area is the Directory, sometimes called the root directory. It is the "master" directory for the disk. Each *DIRECTORY* entry contains the file name, starting cluster number, and various other information about the file, such as date, time, and actual file length. Each directory entry is exactly 32 bytes (or characters) long.

DOS uses the disk directory to find the physical location(s) of a file on a disk. The first 11 characters (or bytes) in the directory are used for the file name, the first 8 for the filename and the last 3 for the file type. The filename and file type are both left justified in these positions, which explains why only one character is required by DOS for a complete file name.

Each directory entry contains a starting cluster number for a file. The starting cluster number is a physical location on the disk, as well as a "pointer" to the map in the File Allocation Table. DOS uses the starting cluster number in the directory as a starting point to locate a file and then reads the FAT to locate the remaining clusters in the file. Remember that the FAT is just a map of the disk space, and this map contains cluster numbers like 4, 5, 6, etc. If the starting cluster number was 3, then the all of a file's cluster numbers — say 3, 4, 5, and 6 — must be read by DOS to access the complete file. This string of cluster numbers has a special name: a *CHAIN* or file chain. In this example, the chain includes 4 clusters. Cluster 3 is found in the directory, which points to the first of the remaining 3 clusters in the FAT.

Like the FAT, the disk directory occupies a specific number of sectors on a disk, depending on the disk type and size. For both a 5.25-inch 360 K disk and a 3.5-inch 720 K disk, it happens that they both use 7 sectors for the directory. Since a directory entry for each file requires 32 bytes, you can easily calculate that the maximum number of files that can be stored on either disk is 112. The important point is that you can have a maximum of 112 root directory entries (or files) on these disks. It also means that, unless you know a trick or two, you are limited to a maximum of 112 files on the disk regardless of file size.

Fortunately, DOS has a way around that limitation by using something called a subdirectory. How to use this feature was covered in Chapter 3 of the first *Powering Up* book, but a discussion of how that works is beyond the scope of this article. One of the reasons for using a subdirectory on a system with a hard drive is because many hard disks have a maximum of 512 root directory entries, which is also determined by the number of sectors allowed for the directory.

The File Area

The fourth and last part of the disk is called the *File Area*, and it includes all of the remaining space. It is sometimes called the *Data Area* because it is where all of the actual program and data files are really stored. The amount of space available in the file area is easy to calculate because it is based on the calculated space less that required for the reserved sectors in the other three disk areas.

We have seen that the calculated capacity of the 5.25-inch disk is 360 K, and that disk contains a total of 720 sectors. We have also seen that the calculated capacity of the 3.5-inch disk is exactly 720 K, and it has a total of 1440 sectors. Other information about each disk format is shown in Figure 1.

Sectors Used

	3.50"	5.25"
Boot Sector	1	1
FATs (Both)	6	4
Directory	7	7
File Area	1426	708
TOTAL	1440	720

Figure 1
Sectors Used in All Disk Areas

By performing a couple of easy subtractions, it is quite simple to calculate that a 3.5-inch disk has 1,426 sectors of file space available, and a 5.25-inch disk has 708 sectors. And because each sector contains 512 bytes, regardless of the disk type or size, it is also easy to calculate the actual capacity. A 3.5-inch disk has a total file storage capacity of 730,112 bytes or 713 kilobytes (divide 730,112 by 1,024). A 5.25-inch disk has a total file storage capacity of 362,496 bytes or 354 kilobytes. So far, all of this is just theory, so let's see how to prove it.

Using CHKDSK

CHKDSK can be used for any number of things, but let's take a look at an example display for a 3.5-inch disk as shown in Figure 2.

```
730112 bytes total disk space
126976 bytes in 47 user files
603136 bytes available on disk

655360 bytes total memory
487760 bytes free
```

Figure 2
CHKDSK Example for 3.5-inch Disk

The example shown in Figure 2 was taken from one of my own disks, but the important information is that the predicted 730,112 bytes of total disk space (in the File Area) are reported by CHKDSK as expected. If we run CHKDSK on a 5.25-inch disk, the results are shown in Figure 3.

```
362496 bytes total disk space
 77824 bytes in 3 user files
284672 bytes available on disk

655360 bytes total memory
487760 bytes free
```

Figure 3
CHKDSK Example for 5.25-inch Disk

As predicted, the 360 K 5.25-inch disk has 362,496 bytes of total disk space reported by CHKDSK.

CHKDSK is an extremely important program, and you should run it often — at least once a week if you use your computer a lot. Although CHKDSK was used in these examples to show that the calculated disk space was what is actually available, CHKDSK does many other tasks. You will not see any results of everything CHKDSK does unless you have a problem with a disk.

Fixing Problems with CHKDSK

I have long maintained that DOS has one of the most fragile file systems that was ever developed. For example, the CP/M operating system keeps the cluster numbers (they are called blocks in CP/M) in the directory entry, so there is no need for anything like a FAT. Larger CP/M files that have more blocks than a single directory can hold simply use another directory entry. Because CP/M does not have to go through the process of locating a file's directory entry, and then go to the FAT, file access time is also reduced because it's simpler. Of course, if you lose a directory entry, then you have lost part or all of a file, but that is also true in DOS. The interesting point about all of this is that it is much more difficult to have a directory problem in CP/M, which explains why there is no command equivalent to CHKDSK provided with CP/M and many other operating systems. On the other hand, DOS really *needs* a command that can check directory and FAT information because the DOS filing system is so much more fragile than many others.

How fragile the DOS filing system really is has been demonstrated by thousands of users who find that they must reboot the system while they are in the middle of an application program, say a word processor. Depending on the specific application, rebooting the system can actually scramble the FAT entries, making it difficult or impossible to recover the file using any technique. That's why I specifically stated that you should NEVER reboot a computer system, except at the DOS command line if you can possibly avoid it in the first *Powering Up* book (page 23).

At the beginning of this article, I mentioned a couple of error messages that CHKDSK can display. Let's take a look at what each means and what you must do to fix it.

Now that you understand something about a FAT, a message indicating that a file has a "cross link on cluster 9" is not as big a problem. Consider two files that have file chains of *3 5 7 9* and *6 7 8 9*. Notice that both file chains contain cluster 9. In other words, cluster 9 is "cross linked" between two different files. Obviously, one chain or the other is an error because, by definition, two different files cannot contain the same cluster number. One of the things that CHKDSK does is verify that the FAT contains unique clusters for each and every file chain on a drive. And by the way, this kind of thing can happen if you reboot the system at the wrong time.

If you do not have a backup for these files, the only effective cure for this problem is to COPY both files to another drive, preferably a floppy, and then edit them using whatever program was used to create the files. In this example, the COPY command will actually copy the data in cluster 9 to both files (using a different cluster number on the new copy of course). This trick works because the COPY command does *not* check the FAT for cross linked clusters, and it will blindly copy whatever cluster numbers it finds. After you manually fix the file, you should then copy it back to the original drive, and that will fix the cross link problem. Just to be sure, I suggest running CHKDSK again on the original drive.

Problems with lost chains can be very difficult to fix, primarily because you will never know for sure why they were lost. Lost chains may be part of a real file, either a program or a data file, and there is no way to determine what the original file name using that chain was. Unfortunately, you may not find out that the lost chain was part of a program file until you run that particular program and find that it does not run properly or at all. If the lost chain was a part of a data file, you have a much better chance of recovering it, assuming you can recognize what file the data might be from.

If you see a message like "5 lost clusters in 1 chains" when you run CHKDSK, this means that there is a file chain in the FAT that has no associated directory entry. Remember that the directory entry contains the starting cluster that points to the rest of the chain in the FAT, and if there is no directory entry with a pointer to a file chain, DOS will never find that chain. Perhaps you can see why I said the DOS file system is so fragile. This is also another thing that can happen if you reboot the system at the wrong time.

One of the details that I intentionally omitted in the previous discussion of the FAT is that each file chain has an "end of chain marker" in the FAT. If you looked at a FAT, you would actually find that a complete file chain is something like *3 4 5 6 7 FFF*. In this example, the *FFF* is technically accurate because DOS really uses *FFF* (in the hexadecimal numbering system) as the "end of chain marker." Although it is a little more complex than that, this description will work just fine as an example. Now let's assume that these five clusters are the "lost" ones in the chain CHKDSK reported in the previous paragraph.

Despite the fact that many CHKDSK error messages will ask you if you want to correct the problem, CHKDSK will *NOT* actually fix the problem unless you run the command with the /F (Fix) switch. In fact, some of the older versions of CHKDSK will report that some problems (e.g., Allocation error, size adjusted) have been corrected when they have not. And they will not actually be fixed until the /F switch is used. By the way, I should mention that it is poor practice to always run CHKDSK with the /F switch. You should always run CHKDSK without the /F switch first, and then, if an error is reported, run CHKDSK again WITH the /F switch. For safety, it is also a good idea to make a backup of all files on the disk BEFORE running CHKDSK with the /F switch.

When CHKDSK finds lost clusters in one or more chains and the /F switch is used, CHKDSK will also provide a prompt: "Convert lost chains to files (Y/N)?" If you enter Y (for Yes), CHKDSK will convert EACH cluster to a different file name using the general format of FILEnnnn.CHK. Because each cluster becomes a file name, you will see file names like FILE0001.CHK, FILE0002.CHK, etc. The file type of CHK indicates that the file was created by CHKDSK. You may be able to see the contents of the file using the TYPE command, depending on what application software originally created it. Of course, if the clusters were a part of a program, looking at each file with the TYPE command will usually cause a lot of beeps and display strange characters on the CRT. For data files, what you can actually see with the TYPE command depends on what software created it, but it is usually enough to help you make an educated guess as to what kind of file the data is from.

Do not make the assumption that all of the recovered clusters came from the same file, especially if the CHKDSK message reported that more than one chain is involved. Based on what you see with the TYPE command, try to guess what file that cluster came from, and then use the appropriate application software to check what you think is the "master" file. Do this for each of the CHK files (that used to be clusters) which were identified by CHKDSK. If you are satisfied that your "master" file is intact, then delete each CHK file as you verify it.

Although this kind of problem most often occurs when a system is rebooted in the middle of an application program, sometimes that is beyond your control. An unexpected system reboot can also occur when you have a momentary power failure in the middle of a command. For example, a power failure that occurs during a DEL command can cause the same kind of problems. Depending on what happens, the directory entry and/or the FATs may not be updated correctly, and that can cause all kinds of strange problems.

Getting Tricky with CHKDSK

Aside from using CHKDSK to fix problems, it is also a very useful utility program in at least one other way. Have you ever wished that you could have a complete list of ALL files (including the system files that are hidden from the DIR command) on a disk, regardless of what subdirectory the files were in? Many so-called power users will tell you that you must have a special utility to do that. Not true. You can get a complete list of ALL files, regardless of their file names or attributes, by using CHKDSK.

All you have to do is run CHKDSK in the root directory of any drive with the /V (Verbose) switch. The /V switch will display a complete file name and path (including the drive letter) on the CRT. Or, you can use I/O redirection to send the list to the printer (with >PRN). Or, you can use I/O redirection to create a file with that list using the command "CHKDSK/V >A:DRIVEC", which of course requires that you insert a floppy disk in drive A before you run the command. This trick is especially useful if you KNOW that a file is on your hard drive, but you can't remember enough of the file name to use the Zenith SEARCH command to find it. And because this trick creates an ASCII file, you can use just about any word processor (or EDLIN) to look through the file. Now that you know how to create a complete file list for any drive, I leave other possible uses of it up to your imagination. I will suggest that one good use of this trick is to use the file list as a basis for writing a batch file where you want to do something specific with a bunch of files.

Earlier in this article, I mentioned the problem of file fragmentation and how you can improve the performance of your system with some kind of "optimizing" program. I also suggested that the whole idea of optimizing a disk is to help improve system performance by keeping an entire file located within a cylinder to help minimize head movement and improve access time. That's still true, but let's take a look at what file fragmentation really is.

Consider a file chain made up of cluster numbers: 5, 9037, 47, 23742, and 4982. Although the actual cluster numbers in the FAT are usually displayed in the hexadecimal numbering system, I am using decimal numbers to keep the explanation simple. The point is that this file has clusters spread all over the disk, which means that DOS must "collect" all of them in order to access the file. If the clusters were closer together — 5, 6, 7, 8, 9 — then DOS could access them much faster because they would most likely be in the same cylinder, and the additional time for moving the head(s) would not be required. And because the cluster numbers are sequential, each cluster is adjacent to the next one in the chain, so we can say that the file is comprised of *CONTIGUOUS* (sequential) clusters. Because this file's clusters are contiguous (5, 6, 7, 8, 9), your system can read the file much faster than if the file contained non-contiguous clusters (5, 9037, 47, 23742, 4982). Most optimizing programs simply examine the FAT for non-contiguous file chains, and then rearrange (by copying) the FAT so that each file contains contiguous clusters. By now, you may be wondering why I mentioned this when this part of the article has been talking about CHKDSK.

Many people don't know that CHKDSK can be used to check a disk to see if the files are contiguous, and you really do not need an optimizer to tell you that. All you need to do is enter the *CHKDSK* *.* command. If all the files in the current directory are contiguous, then CHKDSK will display a message like "All specified files are contiguous." If one or more files in the current directory are not contiguous, then CHKDSK will display a message like "*filename.typ* contains *nn* non-contiguous blocks." Unfortunately, this trick only works in a single directory at a time, which may give you another clue as to another way to use the /V switch with I/O redirection so that you can run CHKDSK on all directories on a disk.

What a Disk Really Contains

Many people have asked me why the FORMAT command must be run on any new disk, floppy or hard, and I hope it is obvious by now why that is required. FORMAT sets up and initializes a disk by creating the four disk areas: the boot sector, FATs, directory, and file area. Virtually all computer systems have some kind of initialization command, like FORMAT, that does essentially the same thing. What any initialization command like FORMAT does in detail depends on the specific operating system and its file management system.

So far, we have only looked at what the FORMAT command does when no system files are transferred to a disk. As part of the initialization that FORMAT does, I should also mention in passing that FORMAT also creates track and sector headers (essentially "labels"), but that is only important to system programmers who write this kind of program. But what does FORMAT with the /S switch do?

First, FORMAT looks at the disk drive hardware to determine the type of disk you want to format. And unless you add a switch to FORMAT (e.g., /4 to format a 360 K disk in a 1.2 MB drive), the program will attempt to format a disk at the maximum density of which the drive hardware is capable. When FORMAT creates the boot sector, it also creates disk-specific information so that DOS will know how to read the disk as previously mentioned. Then FORMAT creates the FATs, the directory, and the file area. As a last step, FORMAT finally copies the system files — the BIOS, System Kernel, and Command Interpreter — to the disk being formatted. FORMAT must wait until the disk is completely initialized before copying these files.

What a disk really contains is not difficult to understand when you recognize that there really is a method and procedure for doing it. As you have probably discovered by now, very little of this kind of information is included in any DOS manual, including Zenith MS-DOS.

Powering Down

The famous "Abort, Retry" messages . . . if you have not seen one yet, then you have simply not been working with a computer long enough. The next article will explain where these messages come from, and what to do when you see them. If you have looked at any DOS manual, you have probably found that the explanations are not even close to helpful when it comes to correcting a problem.

If you have any questions about anything in this column, be sure to include a self-addressed, stamped envelope (business size preferred) if you would like a personal reply to your question, suggestion or comment.

Products Discussed

Mace 5 Utilities	$99.00
Mace Gold Utilities	149.00

Fifth Generation Systems, Inc.
11200 Industriplex Boulevard
Baton Rouge, LA 70809
(800) 873-4384 (Orders only)

Classified Ads

REACH THOUSANDS OF DEDICATED HEATH/ZENITH COMPUTER OWNERS!!

Classified Ad Rates

There is a minimum of $5.00 for 10 words or less. Each additional word is $.40. Payment for all classified advertising must be in advance.

Display Classified is available at $35.00 per column-inch. Minimum one inch. Display Classified is commissionable to recognized agencies.

How To Count Words

Count one word each for initials, standard abbreviations, whole numbers, name, address, city, state, zip, area code and telephone number. Dimensions (such as 6 × 9) are counted as one word. Box or department numbers are counted as one word each. All classified ads (not display) are set in the same size type. The first several words, depending on the ad, are set in all caps.

Continuity Discounts

Run your ad at least 3 times during the year; and receive a discount off the regular price.

Run 3 insertions — 3% off, 4 insertions — 4% off, 5 insertions — 5% off, . . . 12 insertions — 12% off.

Issue & Closing Date

Issue Date	Closing Date	Issue Date	Closing Date
January	November 15	July	May 15
February	December 15	August	June 15
March	January 15	September	July 15
April	February 15	October	August 15
May	March 15	November	September 15
June	April 15	December	October 15

REMark Magazine has the right to refuse any ad for any reason.

Classified Order Blank

First	10	words	at	a
cost	of	$5.00	each	insertion.
11-$5.40	12-$5.80	13-$6.20	14-$6.60	15-$7.00
16-$7.40	17-$7.80	18-$8.20	19-$8.60	20-$9.00
21-$9.40	22-$9.80	23-$10.20	24-$10.60	25-$11.00
26-$11.40	27-$11.80	28-$12.20	29-$12.60	30-$13.00
31-$13.40	32-$13.80	33-$14.20	34-$14.60	35-$15.00
36-$15.40	37-$15.80	38-$16.20	39-$16.60	40-$17.00
41-$17.40	42-$17.80	43-$18.20	44-$18.60	45-$19.00
46-$19.40	47-$19.80	48-$20.20	49-$20.60	50-$21.00

Enclosed is a check or money order of $________ for ________ words.
(Minimum order: 10 words for $5.00. Each additional word $.40.)
Please insert this advertisement in the________________________issue.

Signature________________________________

(Please type or print)

Name________________________Phone________

Company________________________________

Address________________________________

City____________________State________Zip________

MAILING ADDRESS: Make check or money order payable to: Rupley's Advertising Service, Dept. — REM, 240 Ward Avenue, P.O. Box 348, St. Joseph, MI 49085, (616) 983-4550.

FOR SALE: 2 Dual Floppy Disk Drives, H25 Printer, 8 Ribbons, Excess of 200 disks. Call (601) 384-5886.

Z-171, TWO 5.25 360K DRIVES RGB Carrying Case. Recharge Battery Pack, Software, Backlighted LCD, AC. Excellent Condition. $900.00. Jacob, (901) 853-2010 Days. (901) 743-2565 Night/Weekends.

Z-100 FOR SALE: Low Profile Full Color Video, Color Monitor, 2-5-1/4", 2-8" Drives, Complete Manuals, Hundreds of Disks, Software Packages. Best Offer Takes It All. Call Roy (704) 884-2314 Evenings, or Write P.O. Box 2046, Brevard, NC 28712.

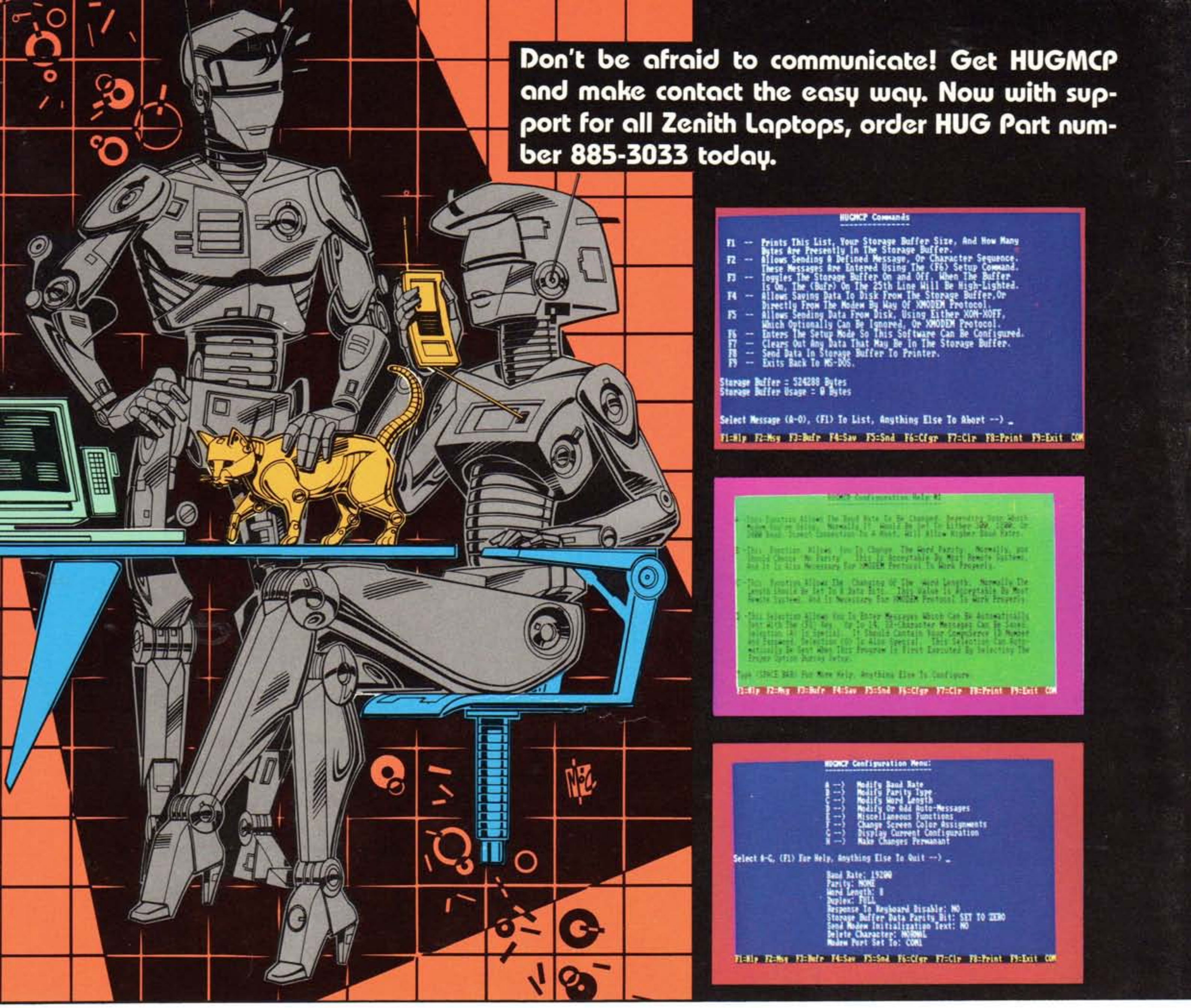
WANTED OLD WESTERN DIGITAL FILE CARD 10 and 20 Meg. 1/3 Height Hard Drive New or Used. Call Jim Evenings, E.S.T. (616) 420-3583.

HADES II

It's 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
- 512 MegaByte Drive Size Limit
- File Attribute Display/Edit
- Automatic Erased File Recovery
- Manual Rebuild File Recovery
- Works with Headerless MS-DOS Disks
- PC-Compatible or H/Z-100

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



Heath Users' Group

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

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

POSTMASTER: If undeliverable, please do not return.

$2.50
P/N 885-2122