The Official Heath/Zenith Computer Users Magazine

REMark®

May 1990

HUG Software Now Available On 2" Disks
See Page 3

Guide to Insanity
See Page 7

Cartoon Contest Winners Announced!
See Page 41

"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 5 • May 1990

Resources

Reader Service No.		Page No.
104	FBE Research Co., Inc.	6
136	Lindley Systems	16
114	Micronics Technology	27
107	Paul F. Herman	23
117	Payload	24
193	QuikData, Inc.	20
149	WS Electronics	42

PC Compatibles

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

PC Compatible

H/Z-100 and PC Compatible

H/Z-100 Only

General

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

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

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

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

HUG

		OPERATING		
PRODUCT NAME	**PART NUMBER**	**SYSTEM**	**DESCRIPTION**	**PRICE**
		H8 - H/Z-89/90		
ACCOUNTING SYSTEM	885-8047-37	CPM	BUSINESS	20.00
ACTION GAMES	885-1220-[37]	CPM	GAME	20.00
ADVENTURE	885-1010	HDOS	GAME	10.00
ASCIRITY	885-1238-[37]	CPM	AMATEUR RADIO	20.00
AUTOFILE (Z80 ONLY)	885-1110	HDOS	DBMS	30.00
BHBASIC SUPPORT PACKAGE	885-1119-[37]	HDOS	UTILITY	20.00
CASTLE	885-8032-[37]	HDOS	ENTERTAINMENT	20.00
CHEAPCALC	885-1131-[37]	HDOS	SPREADSHEET	20.00
CHECKOFF	885-8010	HDOS	CHECKBOOK SOFTWARE	25.00
DEVICE DRIVERS	885-1105	HDOS	UTILITY	20.00
DISK UTILITIES	885-1213-[37]	CPM	UTILITY	20.00
DUNGEONS & DRAGONS	885-1093-[37]	HDOS	GAME	20.00
FLOATING POINT PACKAGE	885-1063	HDOS	UTILITY	18.00
GALACTIC WARRIORS	885-8009-[37]	HDOS	GAME	20.00
GALACTIC WARRIORS	885-8009-[37]	CPM	GAME	20.00
GAMES 1	885-1029-[37]	HDOS	GAMES	18.00
HARD SECTOR SUPPORT PACKAGE	885-1121	HDOS	UTILITY	30.00
HDOS PROGRAMMERS HELPER	885-8017	HDOS	UTILITY	16.00
HOME FINANCE	885-1070	HDOS	BUSINESS	18.00
HUG DISK DUPLICATION UTILITIES	885-1217-[37]	CPM	UTILITY	20.00
HUG SOFTWARE CATALOG	885-4500	VARIOUS	PRODUCTS THRU 1982	9.75
HUGMAN & MOVIE ANIMATION	885-1124	HDOS	ENTERTAINMENT	20.00
INFO. SYSTEM AND TEL. & MAIL SYSTEM	885-1108-[37]	HDOS	DBMS	30.00
LOGBOOK	885-1107-[37]	HDOS	AMATEUR RADIO	30.00
MAGBASE	885-1249-[37]	CPM	MAGAZINE DATABASE	25.00
MAPLE	885-8005	HDOS	COMMUNICATION	35.00
MAPLE	885-8012-[37]	CPM	COMMUNICATION	35.00
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)		
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 7 ISSUES 72-83	885-4007	N/A	1986	25.00
SEA BATTLE	885-1211-[37]	CPM	GAME	20.00
UTILITIES BY PS	885-1226-[37]	CPM	UTILITY	20.00
UTILITIES	885-1237-[37]	CPM	UTILITY	20.00

Price List

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
X-REFERENCE UTILITIES FOR MBASIC	885-1231-[37]	CPM	UTILITY	20.00
ZTERM	885-3003-[37]	CPM	COMMUNICATION	20.00
	H/Z-100 (Not PC) Only			
ACCOUNTING SYSTEM	885-8048-37	MSDOS	BUSINESS	20.00
CALC	885-8043-37	MSDOS	UTILITY	20.00
CARDCAT	885-3021-37	MSDOS	BUSINESS	20.00
CHEAPCALC	885-3006-37	MSDOS	SPREADSHEET	20.00
CHECKBOOK MANAGER	885-3013-37	MSDOS	BUSINESS	20.00
CP/EMULATOR	885-3007-37	MSDOS	CPM EMULATOR	20.00
DBZ	885-8034-37	MSDOS	DBMS	25.00
DUNGEONS & DRAGONS (ZBASIC)	885-3009-37	MSDOS	GAME	20.00
ETCHDUMP	885-3005-37	MSDOS	UTILITY	20.00
EZPLOT II	885-3049-37	MSODS	PRINTER PLOTTING UTILITY	25.00
GAMES (ZBASIC)	885-3011-37	MSDOS	GAMES	20.00
GAMES CONTEST PACKAGE	885-3017-37	MSDOS	GAMES	25.00
GAMES PACKAGE II	885-3044-37	MSDOS	GAMES	25.00
GRAPHIC GAMES (ZBASIC)	885-3004-37	MSDOS	GAMES	20.00
GRAPHICS	885-3031-37	MSDOS	ENTERTAINMENT	20.00
HELPSCREEN	885-3039-37	MSDOS	UTILITY	20.00
HUG BACKGROUND PRINT SPOOLER	885-1247-37	CPM	UTILITY	20.00
KEYMAC	885-3046-37	MSDOS	UTILITY	20.00
KEYMAP	885-3010-37	MSDOS	UTILITY	20.00
KEYMAP CPM-85	885-1245-37	CPM	UTILITY	20.00
MAPLE	885-8023-37	CPM	COMMUNICATION	35.00
MATHFLASH	885-8030-37	MSDOS	EDUCATION	20.00
ORBITS	885-8041-37	MSDOS	EDUCATION	25.00
POKER PARTY	885-8042-37	MSDOS	ENTERTAINMENT	20.00
SCICALC	885-8028-37	MSDOS	UTILITY	20.00
SKYVIEWS	885-3015-37	MSDOS	ASTRONOMY UTILITY	20.00
SMALL-C COMPILER	885-3026-37	MSDOS	LANGUAGE	30.00
SPELL5	885-3035-37	MSDOS	SPELLING CHECKER	20.00
SPREADSHEET CONTEST PACKAGE	885-3018-37	MSDOS	VARIOUS SPREADSHEETS	25.00
TREE-ID	885-3036-37	MSDOS	TREE IDENTIFIER	20.00
USEFUL PROGRAMS I	885-3022-37	MSDOS	UTILITIES	30.00
UTILITIES	885-3008-37	MSDOS	UTILITY	20.00
ZPC II	885-3037-37	MSDOS	PC EMULATOR	60.00
ZPC UPGRADE DISK	885-3042-37	MSDOS	UTILITY	20.00
	H/Z-100 and PC Compatibles			
ADVENTURE	885-3016	MSDOS	GAME	10.00
ASSEMBLY LANGUAGE UTILITIES	885-8046	MSDOS	UTILITY	20.00
BACKGROUND PRINT SPOOLER	885-3029	MSDOS	UTILITY	20.00
BOTH SIDES PRINTER UTILITY	885-3048	MSDOS	UTILITY	20.00
CXREF	885-3051	MSDOS	UTILITY	17.00
DEBUG SUPPORT UTILITIES	885-3038	MSDOS	UTILITY	20.00
DPATH	885-8039	MSDOS	UTILITY	20.00
HADES II	885-3040	MSDOS	UTILITY	40.00
HELP	885-8040	MSDOS	CAI	25.00
HEPCAT	885-3045	MSDOS	UTILITY	35.00
HUG EDITOR	885-3012	MSDOS	TEXT PROCESSOR	20.00
HUG MENU SYSTEM	885-3020	MSDOS	UTILITY	20.00
HUG SOFTWARE CATALOG UPDATE #1	885-4501	VARIOUS	PROD 1983 THRU 1985	9.75
HUGMCP	885-3033	MSDOS	COMMUNICATION	40.00
ICT 8080 TO 8088 TRANSLATOR	885-3024	MSDOS	UTILITY	20.00
MAGBASE	885-3050	VARIOUS	MAGAZINE DATABASE	25.00
MATT	885-8045	MSDOS	MATRIX UTILITY	20.00
MISCELLANEOUS UTILITIES	885-3025	MSDOS	UTILITIES	20.00
PS's PC & Z100 UTILITIES	885-3052	MSDOS	UTILITY	20.00
REMARK VOL 5 ISSUES 48-59	885-4005	N/A	1984	25.00
REMARK VOL 7 ISSUES 72-83	885-4007	N/A	1986	25.00
REMARK VOL 8 ISSUES 84-95	885-4008	N/A	1987	25.00
SCREEN DUMP	885-3043	MSDOS	UTILITY	30.00
UTILITIES II	885-3014	MSDOS	UTILITY	20.00
Z100 WORDSTAR CONNECTION	885-3047	MSDOS	UTILITY	20.00
	PC Compatibles			
ACCOUNTING SYSTEM	885-8049	MSDOS	BUSINESS	20.00
CARDCAT	885-6006	MSDOS	CATALOGING SYSTEM	20.00
CHEAPCALC	885-6004	MSDOS	SPREADSHEET	20.00
CP/EMULATOR II & ZEMULATOR	885-6002	MSDOS	CPM & Z100 EMULATORS	20.00
DUNGEONS & DRAGONS	885-6007	MSDOS	GAME	20.00
EZPLOT II	885-6013	MSDOS	PRINTER PLOTTING UTILITY	25.00
GRADE	885-8037	MSDOS	GRADE BOOK	20.00
HAM HELP	885-6010	MSDOS	AMATEUR RADIO	20.00
KEYMAP	885-6001	MSDOS	UTILITY	20.00
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.

Now Available!

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

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

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

ORDERING INFORMATION

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

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

REMark's Remarks

Henry E. Fale
QuikData, Inc.
2618 Penn Circle
Sheboygan, WI 53081

In the last column, we discussed the Bull purchase of Zenith and the introduction of several of Zenith Data Systems' new computers. The purchase has now been finalized, but there are some problems lurking. We will discuss that along with some other Zenith news including the new dealer Medallion program, and we will talk about the introduction of the new EISA system, the ZCM-1492 VGA monitor, and the MinisPort RAM upgrade. That should bring us up to date. After that, this column may not appear every month, but as news and events accumulate enough for this column.

Zenith/Bull Sale Finalized

On December 28, 1989, Zenith Electronics Corporation completed the sale of its computer business to Groupe Bull.

At the closing, Zenith received $496.4 million in cash, representing 90 percent of the estimated purchase price based on the adjusted net book value of the computer business as of the end of November.

The net book value as of the end of November was about $55 million lower than at the end of September, primarily because of reductions in inventories and receivables. The final purchase price will be based on the adjusted net book value of the computer business as of December 28.

Zenith is using proceeds from the sale to repay short-term obligations and a portion of its long-term debt. At closing and before repayment of debt, the company's total interest bearing obligations were about $560 million.

Following approval at a special meeting of Zenith Stockholders, representatives of Zenith and Groupe Bull met in New York and signed the final closing documents, completing the transaction.

The final vote tally showed that 17.4 million shares of Zenith common stock, representing more than 96 percent of the shares voted (more than 65 percent of the shares outstanding), were cast in favor of the proposed sale.

With the sale of its computer business, Zenith is now a $1.5 billion firm in consumer electronics (color TVs, VCRs, camcorders, picture tubes, color monitors and cable products) and electronic components (power supplies, monochrome displays and monitors, and automotive electronics.

Zenith/Bull Dispute

Well, it looks like Zenith and Bull are off to a good start! Bull asks for a $49 million refund on the Zenith Data Systems buy, and Zenith says Bull owes them 49.5 million additional payment for the computer business.

Groupe Bull is demanding a $49 million refund on its 496.4 million purchase of ZDS after a significant fall in ZDS business that started in late 1989. Zenith Electronics Corporation disputes the request claiming that post-purchase asset valuations made by Bull appear to be in conflict with the provisions of the purchase contract.

A settlement of the dispute in Bull's favor would lower the purchase price for ZDS for the second time. It was described as being worth as much as $635 million in October 1989, then reduced to $550 million at the time of closing. Bull was expected to make an additional payment beyond the $496.4 million after determining the net asset value of ZDS as of December 28th, the closing date. Bull now, however, is asking for a $49 million refund plus interest.

The new valuation was made due to both a decrease of inventories, as well as the Zenith Data Systems market itself. Read on for more on inventory reduction via auctions and liquidations.

Bull said the late 1989 decline in demand for Zenith PCs caused a lower value. Zenith acknowledged that ZDS had a substantial deterioration of business during December, expecting a $22 million fourth-quarter gain while it in fact said a net loss was anticipated. Bull said that this should be reflected in the final sale price.

The ZDS business decline follows their loss of a $700 million desktop Air Force PC contract to Unisys. They also recently had a $534 million PC upgrade contract from the Defense Department suspended because of protests by other bidders. It was thrown out by the General Services Administration (GSA).

If this cannot be settled within 60 days, the matter would be referred to the Ernst & Young auditing firm for arbitration.

Zenith Electronics Corporation concluded that Groupe Bull owes an additional $49.5 million for Zenith's computer products business. As provided for in the purchase agreement, an adjustment to the closing date payment is to be made, including interest from the closing date, based on application of the contract terms to valuations of the net assets sold.

For 1989, discontinued operations (the computer products business sold to Groupe Bull) had an estimated new loss of $51.4 million, or $1.92 per share. Included in this figure are estimated computer operating losses of $70.4 million and the estimated gain on the sale of $19 million, net of costs and expenses. We'll find out what happens in the next installment.

Zenith Data Systems Contracts

On November 1, 1989 a U.S. government contract to support the Defense Department's installed base of Zenith Data Systems (ZDS) personal computers (PCs) was awarded, as of this date, to the computer products subsidiary of Zenith Electronics Corporation.

However, due to protests by other bidders, the $534 million contract was suspended. This will hurt Zenith since they usually do about a third of their sales to the government.

Under the terms of the Standard Desktop Computer Companion Contract, ZDS was expected to supply advanced peripherals and software, plus training and maintenance services, to the U.S. Navy, Marine Corps., Army, Air Force, Defense Logistics Agency and other Defense Department agencies over five years. The government also can receive maintenance, training and support services from ZDS for an additional two years.

They also suffered a major setback in the loss of an anticipated $700 million contract to the Air Force for computers and related accessories. With ZDS' track record of strong government business, usually amounting to about a third of their total computer related sales, the loss of these two contracts is going to hurt.

Liquidation and Auctions

Noticed that ZDS computers have been showing up all over the place in liquidation catalogs and being auctioned off? Inventory reduction!

An example is the major public auction: Multi-million dollar valuation surplus assets to ZDS held at the Rosemont/ O'Hare Expo Center in Rosemont, IL on Thursday, October 26, 1989. It was hosted by Ross-Dove Company, Inc., auctioneers. The 6 page heavy stock full color brochure that Ross-Dove put out on this says "More than 1100 new portable and desktop personal computer (386, 286, AT, XT, PC) compatibles with manufacturer's warranty, hundreds of new laser and dot matrix printers, scanners, tremendous quantities of terminals, peripherals, disk drives, and much more!"

Among things being auctioned were over 300 new TurbosPort 386 laptops with 100 MB hard drives and 2 MB RAM; over 200 new 386 compatible personal computers with 80 MB hard drives and 1 or 2 MB RAM; over 200 new AT compatible personal computers; over 200 new XT compatible personal computers; over 200 new PC-style computers. From what I could tell, they look like the Z-159s, Z-248/12s, Z-386s, Eazy PCs, TurbosPorts, EGA color monitors and more. Large quantities of hard drives; over 5,000 Interdyne 20 MB tape backup systems, memory expansion cards, high resolution video cards, modems, math co-processor boards and much more.

What happened to the Z-386 TurbosPort computers? I understand DAMARK, the mail order liquidators in Minneapolis received about 5,000 of them and were liquidating them at $2,999. There were DAMARK ads in the October 4th issue of the Wall Street Journal, and my last few DAMARK catalogs featured them also. Then I see the November Computer Shopper and see a place called TREDEX in Los Angeles advertising them for $2,999. I did not verify this, but somebody called me and said a place called Hi-Tech Liquidators in Atlanta, Georgia had about 5,000 and they were selling them to dealers and mail order firms for $2,700.

And whatever happened to the Z-386/16 MHz machines? They went pretty fast also, to be replaced by the 20 MHz models. Strange, I find out a place called Under-Ware Electronics in Wichita, KS was selling a Z-386 with VGA card and color monitor (looked like the flat screen monitor), 1 MB RAM, 1.2 MB floppy, 1.4 MB floppy, and DOS for $1,999! By the time I found out about this and contacted them, they were already all gone. They did tell me, however, that they were expecting another batch of Zenith Data Systems computers in a few weeks. They didn't know what they would be, however.

And did those of you with the Z-386/16s know that ZDS has discontinued their memory cards for them? Kind of early seeing how recent that machine is, but just try to get a 4 MB Z-515 memory card.

So enough of that, now on to the new stuff.

Zenith Data Systems Introduces ZCM-1492

Zenith Data Systems has introduced the ZCM-1492 FTM VGA analog monitor. The new FTM is similar to the original ZCM-1490, but lighter, quieter and has more conveniently located controls, making the 'most ergonomic' monitor even more superior to conventional monitors. It has replaced the ZCM-1490 FTM.

The new monitor, which will retail at $999, the same as the ZCM-1490, features streamlined cabinet styling, improved electrical design (requires no cooling fan), and auto-sensing 115/230V power supply. It is a 14" fully VGA compatible analog input, RGB monitor with OCLI HEA non-glare treatment and Flat Tension Mask CRT for high brightness and contrast performance. Active display area is approximately 9.84" × 7.09", and uses P22 phosphor. Bandwidth is 28 MHz, with a Horizontal scan frequency of 31.49 KHz. VGA text resolution of up to 720 (H) × 400 (V) and VGA graphics resolution of up to 600 × 480. Weight of the unit is down to 34 pounds.

MinisPort RAM Upgrade

The MinisPort, Model 1 (1 MB RAM) can be upgraded to a Model 2 by a factory installation of the optional ZA-1-9 1 Meg. RAM card. The MinisPort ZA-1-9 upgrade forms are available from ZDS dealers to pass on to customers. The form includes a statement to be signed by the customer that they recognize that the data stored in RAM will be lost and they must back up the unit before sending it in. The 1 MB RAM add-on card retails for $799, and the installation and mailback is another $200. The customer must send in the computer at his expense. When all finished, the unit will come back as a MinisPort, Model 2. As you can see, like any upgrade, you would have been much better off purchasing the Model 2 from the start.

Zenith Data Systems Ships First EISA

Although Compaq made the most noise about EISA, ZDS was the first to bring the product to market. An 80486 was expected, but it would have delayed the EISA to the market. I guess they compromised with a 33 MHz 80386 so EISA would get out there and start gaining some ground. With its reasonable prices and its fast 32-bit disk cache controller, it's a very fast machine and gives lots of bangs for the bucks. These systems should be available in late March.

The system is based on the powerful Intel 33 MHz 80386 microprocessor and features the ultimate in I/O throughput with Zenith Data Systems' EISA Mass Storage Controller.

Zenith Data Systems' Z-386/33E pushes the envelope of technology by increasing overall system throughput by as much as 200% with its 32-bit direct memory access, 32-bit addresses, and data transfer capabilities. The flexibility of the Z-386/33E will more than meet a user's demand for networking and file servers while executing sophisticated applications.

System Architecure

Overview: The Z-386/33E is Zenith Data Systems' first EISA product introduction. EISA, which stands for the Extended Industry Standard Architecture, is a 32-bit extension to the industry standard expansion bus. It is compatible with the thousands of boards on the market designed for the Industry Standard Artchitecture (ISA) prevalent today. This evolutionary approach allows for a smooth transition to new 32-bit peripheral devices as demonstrated by Zenith Data Systems' new 32-bit Mass Storage Controller.

The Z-386/33E is based on Intel's 33 MHz 80386 processor and ZDS' new EISA Mass Storage Controller; this high end personal computer is designed to meet the demands of today's disk intensive application environments.

The Z-386/33E challenges all competing 386 AT, MCA, and EISA systems in performance by continued enhancements in the hard disk subsystem.

A design component of the EISA specification is the definition of bus arbitration. By arbitrating the usage of the bus, all components are given an equal opportunity to utilize the bus and the amount of time they can maintain control.

Automatic system configuration is another EISA feature providing significant benefits for the end user. This feature will save users time by not having to set and reset system and expansion board jumpers and dip switches.

EISA Mass Storage Controller: Zenith Data Systems and Data Technology Corporation invested two years in the development of the 32-bit Mass Storage Controller. This innovation is so important to the computing industry that ZDS has filed for five separate patents on its design.

The Mass Storage Controller features a 15 Mbit per second transfer rate which provides a 50% increase in the communication cycle between the disk controller and the hard disk drives. This reduces the disk I/O time and allows for the use of larger and faster ESDI hard drives.

The EISA Mass Storage Controller has several features which allow the Z-386/33E to access information faster than ever before.

This mass storage controller is a 32-bit EISA compatible card which supports up to 13 storage devices: two floppy disk drives (either 3.5" or 5.25"), four ESDI hard disk drive devices, and seven SCSI devices.

The 1 Mbyte of hardware cache stores frequently used data, therefore, reducing the amount of time spent accessing hard drives. This effectively reduces the average access time allowing for higher sustained transfer rates to the EISA bus.

The multi-tasking mode allows operating systems to queue commands to multiple hard drives. Therefore, in multiple hard drive environments it improves subsystem performance.

The SCSI port provides a way of accessing up to seven secondary storage devices (i.e., tape backup device, WORM, CD ROM).

1:1 interleaving reduces the amount of time the hard disk needs to spin to reach a required sector of information. This results in improved system bit compared to the input/output card with the 16 MHz Z-386. Today, the new 32-bit I/O card supports two serial ports, one parallel port, and a real time clock/calendar. The LED diagnostic lights which help provide quick diagnosis of problems with the computer's key sub-assemblies is also supported by this controller.

Fast 16-bit VGA Video Card: The new high performance 16-bit VGA card is standard in the Z-386/33E. This VGA card is one of the fastest video cards on the market today and is hardware compatible with the VGA video standard.

Video performance is enhanced with Zenith Data Systems' "Slushware" technique whereby slow 8-bit video ROM is copied into fast 32-bit RAM at system boot-up.

The Z-386/33E incorporates a 115/230V switch-mode power supply which provides 200 watts of power to the system. This large power supply provides ample power for the addition of third party peripheral cards and drives.

Models Available

Z-386/33E Model 150: 3.5" floppy disk drive, 4 Mbyte memory, EISA Mass Storage Controller, 150 Mbyte 16ms ESDI hard drive, 16 Kbyte cache with 16 level write queue, four open EISA slots, VGA video card, two serial ports and one parallel port, MS-DOS 3.3 PLUS, and MS-WINDOWS/386. List price is $11,999.

Z-386/33E Model 320: 3.5" floppy disk drive, 4 Mbyte memory, EISA Mass Storage Controller, 320 Mbyte 16ms ESDI hard drive, 16 Kbyte cache with 16 level write queue, four open EISA slots, VGA video card, two serial ports and one parallel port, MS-DOS 3.3 PLUS, and MS-WINDOWS/386. List price is $13,799.

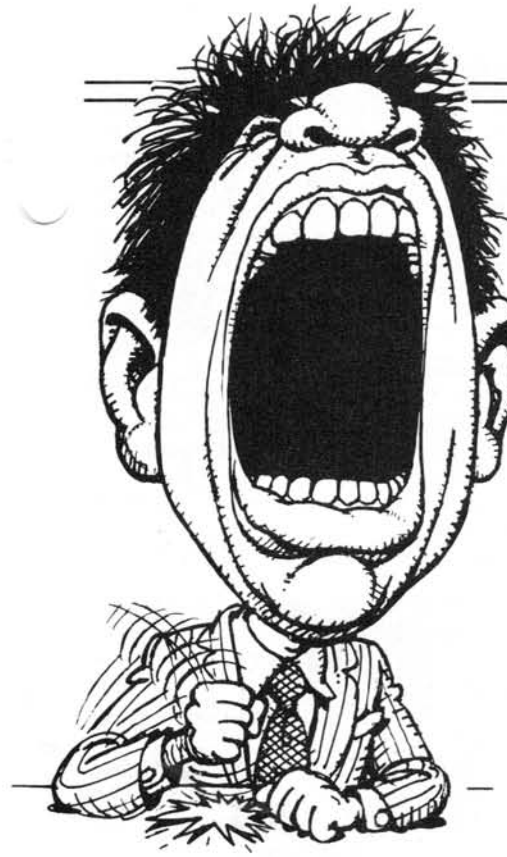
Speed

Many tests were conducted on this system versus others and the speed of this machine is very impressive. Perhaps it has something to do with ZDS' choice to use a dedicated microprocessor for administrating the cache. Another dedicated microprocessor developed by DTC interfaces with the ESDI hard drive. They work concurrently so data can be retrieved from the cache by one CPU and passed along to the system while data is being read from the hard drive by the other CPU. For instance, if the system requests a 15-sector read and 12 of them are already in the cache, the first 12 are transferred to the system at the same time the remaining three are retrieved. The resulting performance level is the same as having all 15 sectors in the cache. In addition, the Bus Master Interface Chip (BMIC) allows 32-bit transfers providing information to the EISA bus as speeds never before achieved in a microcomputer. And this is all done without help from the CPU freeing the CPU to proceed with other tasks, making the system more efficient. There are also other reasons for the high speed, but I will not go into these here.

Speed tests were conducted by Infonetics and I'll print some results. On Novel with 24 and 36 users, the Zenith Data Systems EISA was more than twice as fast as the IBM Model 70-A21 in all tests. While the IBM is only a 25 MHz system, this only accounts for about 30% of the Z-386/33E advantage.

In DOS tests with a standard 386/33 and Exel spreadsheets and CAD programs, the EISA performance index was over 12 times larger than the other machines. With UNIX SCO UNIX ver. 3.2 with 13 active users, the ZDS was almost twice as fast as the ISA-based machines. Running OS/2 using AIM benching software with random disk reads and writes compared to the IBM 70-A21, and a Compaq Deskpro 386/33, the ZDS clearly outperformed both systems by a wide margin.

Fact is, this system outperforms the more expensive i486-based fileservers. ✱



Computer Operator's Guide to Insanity

Salli Brackett
P.O. Box 5137
Moraga, CA 94575

Through my experience with many different computers, I have developed several suggestions to help keep the operator at his/her best. The following is a series of "tips" to help the amateur or professional computer user to utilitize his/her computer to its ultimate potential.

- Never turn on a computer unless you plan to spend at least four hours on a half-hour job.
- If you smoke, have plenty of cigarettes on hand. If you drink, have plenty of ice cubes ready. If you are a junk food junkie, bring in a TV tray, loaded. If you have no such vices — don't buy a computer in the first place.
- Items needed to start a project:
  - A computer — Preferably one that has a keyboard large enough for your fingers.
  - At least 5 disks for one file. (You will need 4 backups for all the wrong commands you use that wipe out your work.)
  - One table — Large enough to hold a computer, 2 or 3 books spread out at one time, scratch paper, and 4 or 5 floppy disks.
  - One chair for every member of the household. (You will get kibitzers.)
  - A printer. (A printer will pay for itself in 3 months in all the scratch paper it generates.)
  - A massager (preferably one of the opposite sex) for the knots in your shoulders after you have said, "I'm almost done, honey." 50 times.
  - A patient family (one who won't miss you for days).
  - Two or three boring books and/or sleeping pills. (Use these after your massager gets through. They will help turn off your mind.)

Now that you have all your necessary items, here are some tips on "how to be organized"; being organized helps you learn more quickly.

1. Only read the chapter in the manual referring to a specific job. (It's much more exciting to learn the "hidden foibles" of your computer accidently.)
2. Never start at the beginning of a manual. (The first 30 pages are BORING!)
3. If you have a computer with the keyboard and CRT separate, plug them into separate sockets. (It's much more fun remembering to turn them off separately.)
4. If you have dual drives, interchange working and data disks from drive "a" to drive "b". (This way you have the fun of figuring out why the commands don't work because your on the wrong drive.) If you have a hard disk, put everything in the root directory. (This will give you hours of fun looking for a file in the directory that has 300 files listed.)
5. Be sure and work with three or four programs at once. (This way you have time to smoke, drink, and/or eat while figuring out which program your in.)
6. Don't bother to learn basic computerease. (It will only slow down your typing time — anyway computerease is easy — it's English, isn't it?)
7. The last thing you should study is your keyboard. Most computers have shorcut keys, but they are designed for use by experts. (An expert being one who accidently hit the key and found out how it worked.)
8. Be creative with your file names. Logic should play no part. (This gives you the opportunity to be a cryptographer, deciphering the names when you need to use the file again.
9. Ignore what manuals say about backups; it's time consuming. (If you lose the file, you have a chance to practice your typing by re-entering the file.)

Now that you have all your "equipment" and you are totally organized, there are a few basic words to learn before you approach your computer (altered slightly for publication):

1. Darn it!!
2. Oh Heck!!
3. Shoot!!

You faint hearts out there, don't despair, these are vital to all beginning computer operators. If you have not learned these phrases already, practice them in the mirror. They will be VITAL TO YOUR SANITY!!

A Word About Manuals — "INSANITY"

I believe writers of manuals stay up nights finding ways to confuse. The worst kind are the "efficient, well-written" ones with tutorials. They give you step-by-step directions with pictures (marvelous inventions, those pictures). When you finish a chapter your so proud!! You actually made the computer do what you wanted!!

Next day, you decide to try the same project from memory — foolish person! You get half way through and forget how to type in a command; you start looking for it in the tutorial; you can't find the spot. "Oh, Heck with it!" (See, I told you those words would come in handy.) "I know, I'll use the 'reference quide'. That's easy, it's alphabetical by command."

Now the "gotcha!" You find the command you want easily — watch it — there's that sense of security. You look at the sample. SELECT <DB> WHERE <FN> is ( ). Now the fun begins. You are going to learn another skill, decoding. Oh, you say your not a cryptographer. SURPRISE!! You are about to become one.

Step One — You stare at the command line (that's computerease for "the line that tells the computer what you want to do). 10 minutes — Time for a vice.

Step Two — AHA! DB stands for data base; FN stands for field name. So, what's a data base? What's a field?

Step Three — Five minutes — Second "vice" EUREKA!! Data base is your file and field is . . . Well! One is better than none.

Step Four — You type in the command. "Syntax error" stares at you from your screen. 10 minutes — Oh! This means typo! Now, what did I do wrong?" You check the screen. NO typos! Now you go back to Step One.

Step Five — You read the explanation of the command. First time: Greek! (If you speak Greek — Russian!) Second time: "I think I understand." Third time (if you're still at the machine): "Why this is a shorter way to use this command than the tutorial taught . . . or is it different . . . or is it . . . ? Shoot, I used to have a good memory!"

Step Six — Back to the tutorial searching for the command. (Isn't this where we started?) You *must* know if your memory is still in tact! Two trips to the bathroom, 2 "vice" breaks, and at least 3 "words" later you finally find the command. Your memory still works. The reference manual and the tutorial show different methods of the same command. Of course! Did anyone say this was simple?

Footnote — Another Session: You are flipping through the manual for some other information. You find a page that has a series of symbols. You glance at them. "Well, I'll be darned!" (See, you even learn variations of "those words.") "This symbol (<>) isn't supposed to be typed, they just needed something to put around the DB, but this symbol ([]) you do type for 'options'. How simple??"

Note: Even if you were one of those disorganized people who read the manual from the beginning — DID YOU "REMEMBER" ALL THAT?

Postscripts

About Error Messages . . .

Some manuals put them in front; Some manuals put them in the back; Some are really clever and hide them in the middle; Some manuals let you guess where they are; Some are numbered; Some are alphabetical; Some are by categories (categories known only to the author). Authors want you to keep reading their manuals for the life of the computer.

Definitions

Bad Command — "A typist you ain't!" or "Wrong drive, dummy!" or "This is your data base program not your word processing program."

Title Not in Dictionary — Type it again, dummy!!

Wild Interrupt — Boy, did you blow it!!

No Disk Space — Brother, all that typing for nothing!

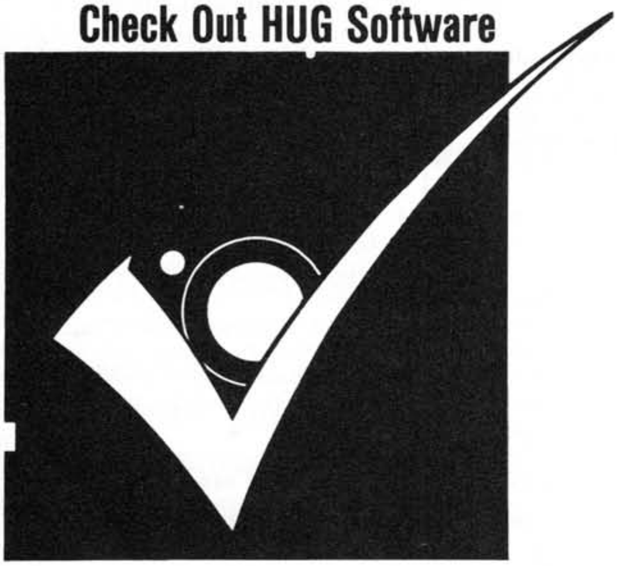
Missing Operand — Operator went out to lunch.

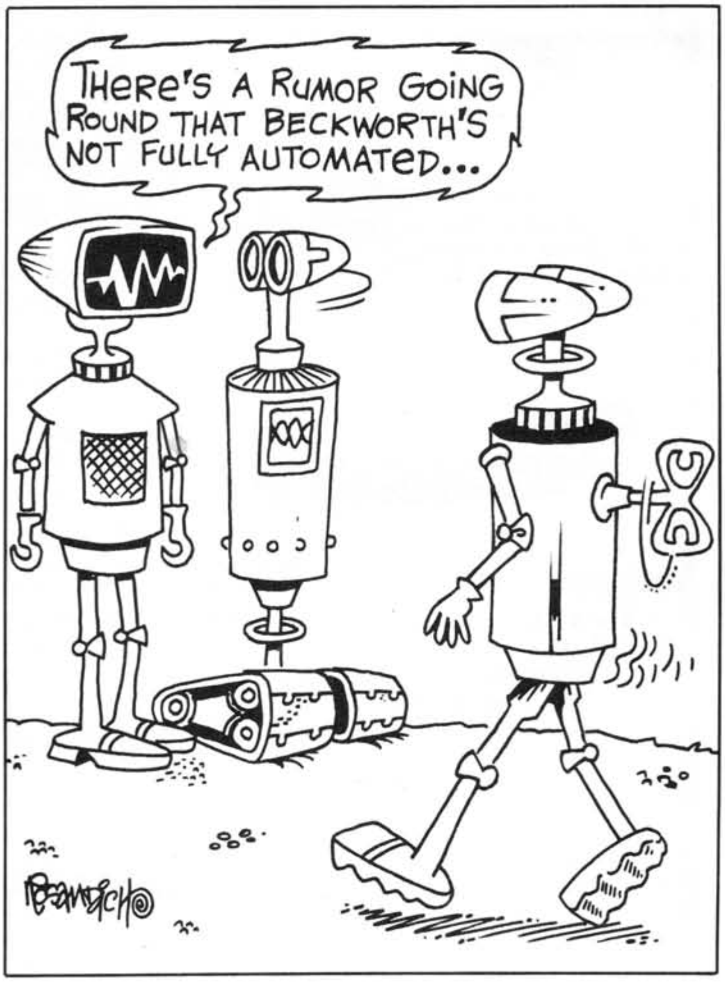
Write Protect Error — You ain't writing on me, Turkey!

Disk Reading Error — This disk doesn't make sense!

Disk Writing Error — I don't like this disk. Give me another one.

Missing Literal or More Than 15 Spaces — "Be more specific!" or "Don't be so verbose!!" ✱




THERE'S A RUMOR GOING ROUND THAT BECKWORTH'S NOT FULLY AUTOMATED...

ASSEMBLY LANGUAGE

PART 5
THE INSTRUCTION SET (PART 3)

PAT SWAYNE
HUG SOFTWARE ENGINEER

This is the fifth part in my series on Assembly Language. In this installment, I will continue my discussion of the instruction set, beginning with compare and logic instructions.

Compare and Logic Instructions

There are two Compare instructions in the 8086 family instruction set, CMP (compare) and TEST. They use the same addressing modes as the two-argument arithmetic instructions, such as ADD and SUB.

CMP — Compares the number in the source to the number in the destination. Some people group the CMP instruction with the arithmetic instructions (discussed last time), because it is actually a non-destructive version of the SUB (subtract) instruction. In other words, it subtracts one number from another, but does not save the result of the subtraction. It does, however, affect the result flags in the same way as the SUB instruction. Whenever you are trying to figure out what flags will be set or reset by a particular CMP instruction, it is helpful to remember that it is doing a subtraction.

TEST — Compares the number in the destination to the bit pattern in the source. The TEST instruction is not classified as a compare instruction by some people. Like CMP, it is also a non-destructive version of another instruction, which in this case, is the AND instruction. See the AND instruction for more information.

All of the logic instructions except one can be classified into two groups — the Boolean (logic) operation group and the shift/rotate group. The Boolean instructions use the same addressing modes as the two-argument arithmetic instructions. They affect all of the result flags, and their affect on the carry flag is that it is always cleared.

AND — The number in the source is logically ANDed with the number in the destination, and the result is placed in the destination. The Assembly Language AND instruction works just like the GW-BASIC AND logical operator (as do OR and XOR), so you may want to look up AND in your GW-BASIC manual. The BASIC operation A = A AND B is just about an exact duplicate of the Assembly Language AND instruction. The result flags are set or reset depending on the result of the AND. The TEST instruction is identical except that the number that results from the ANDing of the original two numbers is not saved.

OR — The number in the source is inclusively OR'ed with the number in the destination, and the result is placed in the destination.

XOR — The number in the source is exclusively OR'ed with the number in the destination, and the result is placed in the destination.

The shift/rotate instructions are a useful group of instructions for manipulating individual bits. They all shift bits to the left or right in a number. The destination argument (the number to be shifted) can be specified using the register, direct, or index addressing modes. The source argument (the number of places to shift, called the count) is either the implied number 1, or a number specified in the CL register. The shift instructions affect the carry, overflow, parity, sign, and zero flags. The rotate instructions affect the carry and overflow flags.

SAL/SHL (Shift Arithmetic Right) — This is a single instruction that has two names. The bits in the destination are shifted to the left by the number of times in the count. Zeros are shifted in on the right. If the count is one, the overflow flag will be set if the sign bit changes as a result of the shift, and it will be cleared otherwise. If the count is not one, the overflow flag is undefined.

SAR (Shift Arithmetic Right) — Shifts the bits in the destination to the right by the number of times in the count. If the sign bit is one (a negative number), ones are shifted in on the left. If the sign bit is zero, zeros are shifted in on the left. Normally, the overflow flag is cleared by this instruction. However, if the count is one and the next to the high order bit is not the same as the high order bit after the shift (I cannot imagine how that could happen), the overflow flag will be set.

SHR (SHift logical Right) — Shifts the bits in the destination to the right by the number of times in the count. Zeros are shifted in on the left. The overflow flag is set if the count is one and the next to the high order bit is not the same as the high order bit. If the count is not one, the overflow flag is undefined.

ROL (ROtate Left) — Rotates the bits in the destination to the left by the number of times in the count. A rotate is different from a shift in that the number acts like it is "bent" into a circle. As bits are shifted out of the high end of the number, they are shifted into the low end. In other words, if the high order bit is one before a shift to the left, the low order bit will be one after the shift. A copy of the last bit shifted out is placed in the carry flag. The carry flag will, therefore, be set if the last bit shifted out is one, or cleared if the last bit is zero. To look at it in another way, the carry flag will always be in the same state as the low order bit in the number after the rotation. The overflow flag is affected as per the SHL instruction. Figure 1 shows a graphical illustration of the ROL instruction with an 8-bit argument. Keep in mind that the number could also be 16 bits.

RCL (Rotate through Carry Left) — Rotates the bits in the destination to the left by the number of times in the count. In this instruction, the carry flag is considered to be part of the number. In order words, if the destination is an 8-bit number, a 9-bit number consisting of the destination and the carry flag is rotated. The carry flag will be set or cleared depending

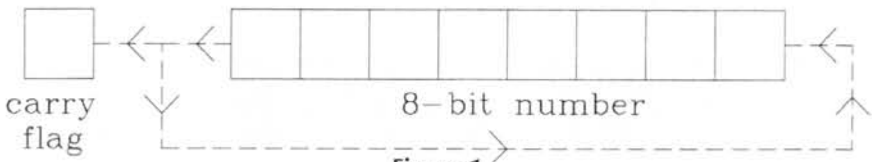


Figure 1
Graphical Illustration of the ROL Instruction

on whether a bit has been shifted into it when the rotate is completed, and the overflow flag will be affected as per the SHL instruction. Figure 2 shows a graphical illustration of the RCL instruction with an 8-bit argument.

ROR (ROtate Right) — Rotates the bits in the destination to the right by the number of times in the count. This instruction works like ROL, except that the bits are rotated to the right, and the carry flag contains a copy of the low order bit shifted out. The carry flag will always be in the same state as the high order bit in the number after the rotation. The overflow flag will be affected as per the SHR instruction.

RCR (Rotate through Carry Right) — Rotates the bits in the destination to the right by the number of times in the count. This instruction works like RCL, except that the bits are rotated to the right. The overflow flag is affected as per the SHR instruction.

NOT — This instruction negates the number in the argument. This instruction is like the NEG instruction (listed with the arithmetic group last time), except that it computes the one's complement of the argument rather than the two's complement. In other words, every one bit is changed to a zero, and every zero bit is changed to a one. The listing of this instruction with the logical group, and the NEG instruction with the artithmetic group is the way Intel lists them, so I left it that way.

String Manipulation Instructions

The string manipulation instructions are a special group of instructions that make string handling easier. The term "string" in this case, means more than just ASCII characters, because these instructions can work with either byte or word elements. The string manipulation instructions use the index addressing mode, with the SI pointing to the source string, and the DI register pointing to the destination string.

MOVS/MOVSB/MOVSW (MOVe String/MOVe String by Bytes/Move String by Words) — This instruction moves (actually copies) a string element (byte or word) from the source area to the destination area. Then, if the direction flag is clear, it increments both the SI and DI registers to point to the next elements. If the elements are words, the registers are incremented by two, and by one otherwise. If the direction flag is set, the SI and DI registers are decremented. You can specify whether a byte or word is to be moved by using the specific form of the instruction applicable. Most assemblers will also allow you to use the general form (MOVS) if you specify dummy arguments that tell the assembler what you want to move. Consider this example.

```
START:  MOV     SI,OFFSET BUFFER1
        MOV     DI.OFFSET BUFFER2
        MOVS    BUFFER2,BUFFER1

BUFFER1 DB      10 DUP (?)
BUFFER2 DB      10 DUP (?)
```

Here, the assembler knows to use the byte form of the instruction, because the buffers are defined using the DB (define byte) directive. We could have used MOVSB with no argument to get the same result. If the buffers had been defined using the DW (define word) directive, then the assembler would use the word form of the instruction.

LODS/LODSB/LODSW (LOaD String/LOaD String by Bytes/LOaD String by Words) — Places a copy of the element pointed to by the SI register in AL (for a byte) or AX (for a word). Then the SI register is incremented or decremented depending on the direction flag. As with MOVS, you can use the general form of the instruction if you supply a dummy argument to tell the assembly what you are loading.

STOS/STOSB/STOSW (STOre String /STOre String by Bytes/STOre String by Words) — Places a copy of the element in the AL or AX register in the area pointed to by the DI register. Then the DI register is incremented or decremented depending on the direction flag. The general form can be used if a dummy argument is supplied.

SCAS/SCASB/SCASW (SCAn String/ SCAn String by Bytes/SCAn String by Words) — This instruction works like the CMP (compare) instruction. It subtracts the element pointed to by the DI register from the number in the AL or AX register, and sets the result flags. Like CMP, it does not store the number resulting from the subtraction. The DI register is incremented or decremented depending on the direction flag. This instruction violates the destination/source rule followed by the other string manipulation instruction, because if you compare it strictly with the CMP instruction, the AL or AX register is actually the destination, and the DI register points to the source. If it did a real subtraction rather than just compare, the answer would be in AL or AX. The general form can be used if a dummy argument is supplied.

CMPS/CMPSB/CMPSW (CoMPare String/CoMPare String by Bytes/CoMPare String by Words) — This instructionn subtracts the string element pointed to by DI from the one pointed to by SI and sets the result flags. The number resulting from the subtraction is not saved. The SI and DI registers are incremented or decremented depending on the direction flags. This instruction also violates the destination/ source rule because CMP subtracts the source from the destination. That makes DI the source, and SI the destination. The general form can be used if a dummy argument is supplied.

REP (REPeat) — This instruction is a special prefix that can be used with the MOVS, LODS, or STOS instructions. (Actually, there is not much point in using it with LODS, although it can be done.) It causes those instructions to repeat for a count specified in the CX register. This instruction and two other prefix instructions give the string manipulation instructions their real "power". Going back to the example given for MOVS, suppose you wanted to copy all 10 elements in BUFFER1 to BUFFER2. You could set up a loop that would execute the MOVS 10 times, but with REP you don't have to. Here is all you have to do.

```
START:  MOV     SI,OFFSET BUFFER1
        MOV     DI,OFFSET BUFFER2
        MOV     CX,10
        REP MOVS BUFFER2,BUFFER1

BUFFER1 DB      10 DUP (?)
BUFFER2 DB      10 DUP (?)
```

You can move or store up to 65,536 string elements with one instruction by using the REP prefix. The CX register is decremented each time the instruction repeats, but the result flags are not affected by this.

REPZ/REPE (REPeat while Zero/REPeat while Equal) — This instruction is a prefix used with the SCAS and CMPS in-

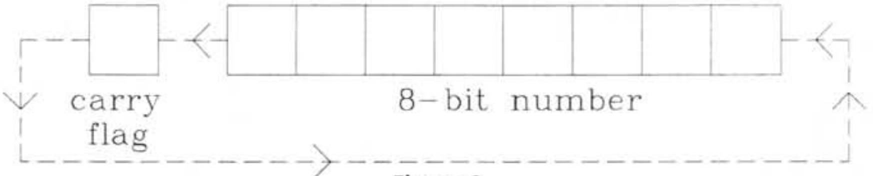


Figure 2
Graphical Illustration of the RCL Instruction

Continued on Page 16

Fractals on the H/Z-100 (PC or NOT PC) in C

Robert W. Rasch
1504 Chickees Street
Johnson City, TN 37604

Exploring the edges of the Mandelbrot Set can be exciting, but the many calculations required to create a display make the process painfully slow when done with interpreted or compiled BASIC.

My first contact with Fractals came with issue #23, July-August 1986 of SEXTANT "BASIC Fractals" by Edward A. Byrnes. Unfortunately, even a compiled BASIC program is too slow for the exploration of Fractals and I quickly tired of "looking around the Mandelbrot Set."

What was needed was the assistance of an 8087 numerical processor and a language that would use it for speedier computation. Ecosoft "C" does that precisely, and displays that took a whole day to complete with compiled BASIC are now finished in several hours.

The "C" listing presented here is a translation of the algorithm by Byrne (originally A.K. Dewdney: The Scientific American 257:140-146 November 1987) into Ecosoft "C" and it can be used on the H/Z-100 (NOT PC) with the "nbpoint pixel setting routine" published in the Z-100 Notebook, William N. Locke, #54 July-August 1987 SEXTANT nbpoint(x, y, color). NBPOINT is in my GRAFIX.LIB.

If you don't want to make your own GRAFIX.LIB for the H/Z-100, you could use graphics routines already in the Eco-C88 library and emulate the IBM under Pat Swayne's ZPC. ZPC does not handle the getch() routine very well and you must use getchar(); which requires a carriage return to complete an entry. I think that an IBM compatible will use getch() without any difficulty.

I would be pleased to copy my GRAFIX.LIB, this C listing and the compiled program for the H/Z-100 (NOT PC) to a formatted 5.25" disk that is accompanied by an SAS disk mailer.

IBM compatibles can probably use the ECOS.LIB routines directly. The Eco Compiler comes as part of the Heath Course "C Programming: A Hands-On Approach." Or you can order it as: ECO-C88 C Compiler Ecosoft, Inc. 6413 College Avenue, Indianapolis, IN 46220. The com-

Listing 1

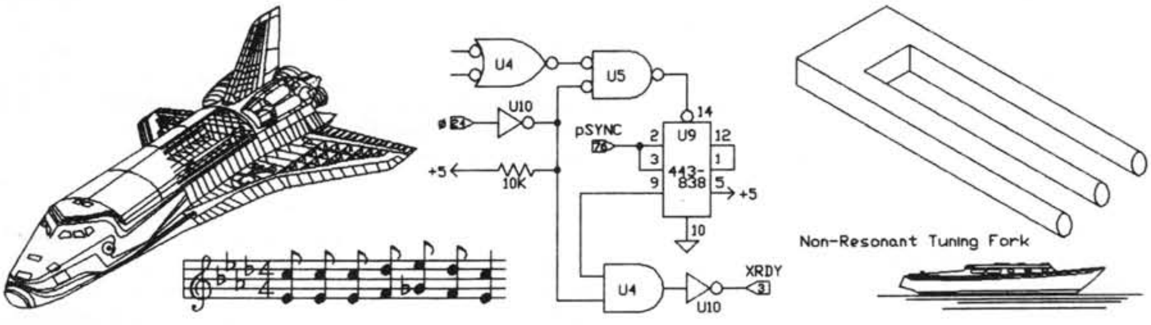
```
/* Displays FRACTALS on Heath/Zenith 100, Heath/Zenith 100 under ZPC
   or any IBM compatible.  Note what is required
   for each system and then compile with ECOSOFT C88 or make the
   changes necessary for your compiler.                                */
#define Z100 1 /* compile for H/Z100 with grafix.lib */

/* #define ZPC 1 compile for running under ZPC
   #define IBM 1 compile for IBM compatible
   take your choice and make the definition. */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <ctype.h>
#ifdef Z100
#include <grafix.h>
#endif
#include <dos.h>
#define KBSTATUS 0x0b /* Look at keyboard status */
#define EMPTY (-2)
#ifdef Z100
#define SET25 "\033Y8 " /* put cursor on 25th line */
#endif
static double acorner, bcorner, side, gapx, gapy;
static double ac, bc;

main()
{
int is_mandl();
int get_color();
double get_ac();
double get_bc();
char c, buffer[30];
int color, iter, m, n;
int explain();
#ifdef Z100
cls();
#else
clrscr();
#endif
#ifdef Z100
enable();        /* enable the 25th line */
#endif
explain();
printf("Enter ACORNER");
        if((acorner = atof(gets(buffer))) == NULL) acorner = -2.0;
        ;
printf("ACORNER is %20.6f\n",acorner);
printf("\nEnter BCORNER");
        if((bcorner = atof(gets(buffer))) == NULL) bcorner = -1.25;
        ;
printf("BCORNER is %20.6f\n",bcorner);
        printf("\nEnter SIDE");
        if((side = atof(gets(buffer))) == NULL) side = 2.9;
        ;
printf("SIDE is %2Ø.6f\n",side);
printf("depress any key to continue");
#ifdef ZPC
        c = getchar(); /* ZPC emulation has a problem with this */
#else
        c = getch();
#endif
gapx = side/64Ø.Ø;
gapy = side/225.Ø;
#ifdef Z1ØØ
cls();
#else
clrscr();
scr_smode(6); /* 64Ø x 2ØØ black and white */
scr_spal(1,1); /* black back and white is fore */
#endif
m = 1;n = 1;
do {
if(chkkb() != Ø) c = getch(); /* see if they want to quit */
if((c == 'Z') || (c == 'z')) exit(Ø);
ac = get_ac( n, &gapx, &acorner);
bc = get_bc( m, &gapy, &bcorner);
iter = is_mandl( &ac, &bc);
#ifdef Z1ØØ
if(iter > Ø) {color = get_color(iter); nbpoint( n, m, color);}
        ;
#endif
#ifdef IBM
if(iter > Ø) {color = get_color(iter); scr_wdot( m, n, color);}
        ;
#endif
#ifdef ZPC
if(iter > Ø) {color = get_color(iter); scr_wdot( m, n, color);}
        ;
#endif
if( n >= 639 ) {n=1;m=m+1;}
         else n=n+1;
} while ( m < 2ØØ );
#ifdef Z1ØØ
        alpha();         /* set Z1ØØ for alphanumeric characters */
        puts(SET25);    /* put cursor on 25th line */
#endif
} /* end of main */
/* finds out if this location is within the Mandelbot set
   tries to escape with a value of sizeq > 4, if not returns
   how many iterations were required                         */
   int is_mandl( double *ac, double *bc)

{
double ax, bx, az, bz, sizeq;
int iter;
az = Ø;
bz = Ø;
iter = Ø;sizeq = Ø.Ø;
        do {
        ax = (az * az) - (bz * bz) + *ac;
        bx = (2.Ø * az * bz) + *bc;
        az = ax;
        bz = bx;
        sizeq = (ax * ax) + (bx * bx);
        iter = iter +1;
        } while( iter < 1Ø1 && sizeq < 4.Ø );
        return iter;
}
/* sets the color as black or white depending on the value
   of the iteration that allowed an escape - count is odd or even */
get_color(count)
int count;
{
int color;
#ifdef Z1ØØ
if((count % 2) == Ø) color = 7;
if(count > 1ØØ) color = Ø;
#else
if((count % 2) == Ø) color = 3;
if(count > 1ØØ) color = Ø;
#endif
return color;
}

double get_ac( int n, double *gapx, double *acorner)
{
double ac;

ac = ((double) n * *gapx) + *acorner;
return ac;
}

double get_bc( int m, double *gapy, double *bcorner)
{
double bc;

bc = ((double) m * *gapy) + *bcorner;
return bc;
}

int chkkb(void)
{
        union REGS ireg, oreg;

        ireg.h.ah = KBSTATUS;
        int86(Øx21, &ireg, &oreg);
        return(oreg.h.al);
}
int explain(void)
{
int c;
printf("\tExplore the Mandelbrot Set\n");
printf("\tCourtesy of Robert W. Rasch\n");
printf("\t\tVersion (12.31.1989 B&W)\n");
printf("\tUses an 8Ø87 Numerical Processor if present.\n");
printf("\n");
printf("The default settings are:\n");
printf("Lambda: ACORNER = -2.Ø BCORNER = -1.25\n");
printf("SIDE = 2.9 they will locate the Mandelbrot Set.\n");
printf("Various values of SIDE will magnify or reduce that image.\n");
```

Continued on Page 32

AutoSketch®

Power Drawing for Less

Pat Swayne
HUG Software Engineer



If you have a computer, but you are still using a pen or pencil for drawing, you are missing out on one of the best things your computer can do for you. This is especially true if you do practical drawing rather than artistic drawing. Programs designed for practical drawing are called CAD (Computer Aided Design or Computer Aided Drafting) programs. The first CAD programs to become available to micro computer users were intended for design professionals, and they sold for big bucks. Later, inexpensive CAD programs appeared, making it possible for non-professionals to do "power drawing" with their computers.

AutoSketch is one of the least expensive of the non-professional CAD programs available. It was produced by one of the professional CAD program companies. The first version of AutoSketch came out two or three years ago, and I was not impressed enough with it to write about it here. However, the current version (version 2.0 as of this writing) is a competent, complete 2-D (two dimensional) CAD program that compares favorably with others that cost more. In fact, it is a more powerful drafting program than the original AutoCAD.

Product Description

AutoSketch comes with the disks and manual packaged together in a spiral binder with a hard cover. The disks (5 of them if 5.25", 3 if 3.5") are in vinyl sleeves in the binder. There are actually two versions of AutoSketch included in the package. The standard version will work on systems that do not have a math coprocessor, and it will work slightly faster on systems that have one. The speed enhanced version requires a math coprocessor, and works up to ten times faster than the standard version without a coprocessor. Each version takes up two 5.25" disks, or one 3.5" disks, and there is also a disk containing sample drawings.

The 180-page manual is logically arranged into installation, tutorial, and command reference sections, with an introduction at the beginning, and appendixes, glossary, and an index at the end. It is about as well written as any CAD manual I have seen.

Like most CAD programs, AutoSketch is an object oriented drawing program. That means that if you draw a circle, for example, on the screen, AutoSketch stores a mathematical representation of the circle in memory. If you move or change the size of the circle on the screen, AutoSketch uses the stored mathematical representation to redraw the circle. Therefore, the circle is always drawn using the best resolution on the screen regardless of how large or small you make it. A tiny circle that is enlarged does not become jagged as happens with non-object oriented drawing programs.

Because AutoSketch comes from the AutoCAD people, it is somewhat compatible with AutoCAD. In addition to its own format, it can read and write files in the DXF (tm) format supported by AutoCAD. Therefore, you can transfer your drawings to AutoCAD if you decide to upgrade, or if you need to supply drawings to someone who uses AutoCAD. You can also transfer some drawings from AutoCAD to AutoSketch. The space shuttle below the title of this article (the title was produced with AutoSketch) is a well known AutoCAD sample drawing that I was able to transfer. Some AutoCAD objects are ignored by AutoSketch, so a drawing may have parts missing after the transfer. In some cases, an AutoCAD drawing with untransferable objects can be modified before it is output as a DXF file so that it can be transferred. Specifically, (for you AutoCAD users out there), AutoCAD "blocks" (equivalent to groups in AutoSketch) will not transfer, but they can be "exploded" first to make transfer possible. Appendix C discusses reading AutoCAD DXF files, and lists what will not transfer.

Both AutoSketch and AutoCAD can write text in several fonts, and the font files are identical and interchangeable. The fonts supplied with AutoSketch include Roman, Italic, and Script, and there are also symbol fonts containing map, math, and music symbols. For example, with the music font loaded, the letter O makes a treble clef. AutoCAD comes with more font files than AutoSketch, so if you have AutoSketch, and you have a friend with AutoCAD, ask him to copy all of his font files to a disk for you. Some of them will be duplicates of what you already have, but some will be useful additions.

Probably the best feature that AutoSketch shares in common with AutoCAD is the ability to use ADI (tm) (Autodesk Device Interface) drivers. ADI drivers provide a way to use video cards, pointers (digitizers), plotters and printers that are otherwise not supported. They also provide a way to improve the use of supported devices. For example, AutoSketch supports Epson FX compatible printers, but only at 120 x 72 dots per inch resolution. I have written an ADI driver that can drive an FX compatible printer at 240 x 216 dots per inch. The title of this article was printed on an Alps P2000G printer (which is FX compatible) using my driver. I have written several other printer drivers, and also a driver for the SummaSketch graphics tablet (which is not supported by AutoSketch, but is by AutoCAD). These drivers are available for download from the HUG Bulletin Board in a file called ADRIVERS.ZIP.

Many manufacturers of super high resolution big screen display systems provide drivers that allow AutoCAD to work with their systems. These drivers will also work with AutoSketch. If you have that oddball Z-449 video card that ZDS supplied for a while, the 640 x 480 driver for that will work with AutoSketch, provided that it is the driver that works with AutoCAD release 10. The old AutoCAD driver originally supplied with the card will not work.

Autodesk provides a package called the ADI Driver Development Kit that contains a book explaining the technical details of ADI drivers, and some samples on disk. This kit normally costs $100, but they sometimes give it away. They gave me one after I uploaded some of my printer drivers to the Autodesk SIG on CompuServe (R). If you are wondering how I was able to write printer drivers before I got the kit, it is because they used to supply an older version of their kit for download on the Autodesk SIG. I still have this older kit, and if you would like one, send me a blank disk and stamped disk mailer c/o HUG. The specifications in it for video drivers are obsolete, but the specifications for printer, plotter, and digitizer drivers are still good.

ADI drivers can take many forms. Sometimes they are TSR (Terminate and Stay Resident) programs that you load into memory before you run AutoSketch or AutoCAD. In the case of printer or plotter drivers, however, they can be programs that process data after you run AutoCAD or AutoSketch, which can be set up to write printer or plotter output to a file. That means that you can write a driver using any computer language that can process data and send it to a printer, including BASIC. To illustrate, here is a driver for a Heath IR-5208 (Sweet P) plotter, which is not normally supported by AutoSketch.

```
10 PRINT "PROGRAM TO PLOT ON SWEET P"
20 LINE INPUT "ENTER FILE TO PLOT: ";F$
30 F$=F$+".PLT"
40 OPEN "I",1,F$
50 IF EOF(1) THEN 200
60 LINE INPUT #1,L$:L=LEN(L$)
70 C$=LEFT$(L$,1):C=ASC(C$)-48
80 ON C GOTO 100,200,300,400,500,600,700,800
100 LPRINT "PU;":LPRINT "HO;"
110 GOTO 60
200 LPRINT "MA 2500,1838;":CLOSE #1:STOP
300 L$=RIGHT$(L$,L-2):L$="MA "+L$+";"
    :PRINT L$
310 GOTO 60
400 L$=RIGHT$(L$,L-2):L$="DA "+L$+";"
    :LPRINT L$
500 GOTO 60
600 GOTO 60
700 GOTO 60
800 LPRINT "PU;"
810 GOTO 60
```

To use this driver, run AutoSketch in the configuration mode (SKETCH /R), and set up the plotter as follows:

```
Select plotter: 2
Select output format: 1
Does the plotter have multiple pens? <N>: N
Specify plot size in millimeters? <N>: N
Maximum horizontal (X) plot size in inches
    <11.00>: 10
Plotter steps per inch in the horizontal (X)
    direction <1000.00>: 250
Maximum vertical (Y) plot size in inches
    (8.50): 7.45
Plotter steps per inch in the vertical (Y)
    direction (250.00): 250
```

When you start a plot, AutoSketch will ask you for a file name. You can enter the same name as the file you are plotting, without the extension, and AutoSketch will supply .PLT as the extension. When the plot is finished (the pointer arrow will appear), exit from AutoSketch and run the BASIC program listed earlier, using BASICA or a compatible BASIC interpreter. Enter the same name you supplied to AutoSketch (without the extension) when the program asks for the file to plot.

Using AutoSketch

AutoSketch uses pull-down menus to select commands, and the user interface is similar to that of Microsoft Windows (tm). It can be operated with a keyboard only, but it is much easier to use if you have a mouse or other pointing device. That is because the menus can only be pulled down by moving the pointer to them. Some of the commands can also be executed directly (without having to go through a menu) via function keys. This feature comes in very handy, because some commands can be executed while you are in the middle of other commands. For example, you can turn a feature called "snap" on or off while you are drawing lines or other objects. (Snap causes the pointer to only go to preselected coordinates on the screen.) Two of the sample drawings that came with version 1 of AutoSketch were templates showing the function key commands (one template for 101-key keyboards, and one for 84-key keyboards). You could print or plot one of these templates, cut it out, and lay it over your function keys. For some reason, these drawings are not included with version 2, but I retained copies of them. If you send me a self-addressed stamped envelope, I will run off a copy of the (plotted) drawings and send it to you. There are 20 function key commands (F1 through F10 and Alt-F1 through Alt-F10), so a template is helpful.

The command reference section of the AutoSketch manual is laid out in the same order as the menus are arranged on the screen. There is also a fold-out quick reference that shows all of the menus pulled down, with all commands listed. There is no on-screen help built into AutoSketch, so the foldout reference and the key template (described above) are useful aids. Actually, AutoSketch is so easy to use (many of the commands are self-explanatory) that you may be tempted to start using it without reading the manual. I suggest that you read it anyway, because you can miss out on some important techniques.

Many of the commands in AutoSketch automatically repeat. For example, if you select the Line command and then draw a line, the next thing AutoSketch will prompt you to do is place the first point for another line.

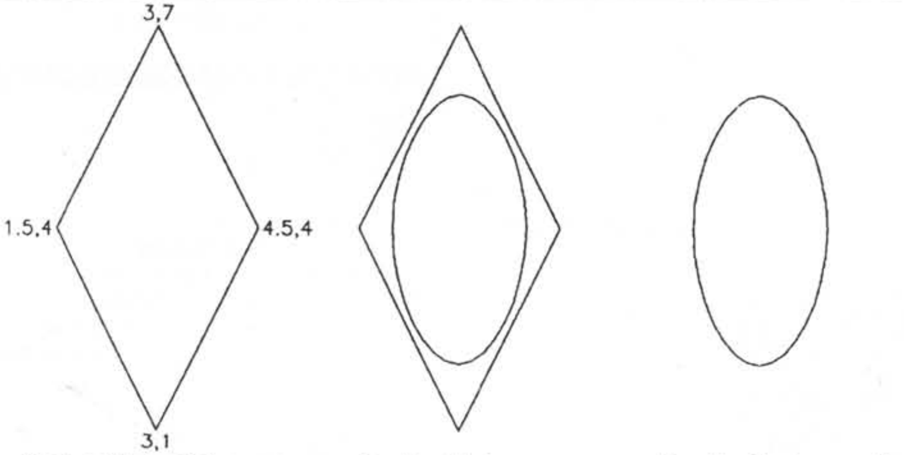
The pointer on the screen is normally a small arrow, but it will change to a hand figure if you are executing a command that requires you to select an existing object in order to do something with it (for example, to move or copy it). You can use the pointer to select points on the screen (for example, click a mouse button when the pointer is where you want it), or you can specify them as numerical coordinates. When you specify a point as coordinates, you can use absolute X, Y coordinates, relative (to the last point) X, Y coordinates, or relative polar coordinates. When you are selecting an existing object, you can point to it (which usually means pointing to anywhere on the object, as anywhere along a line), or you can draw a box around it. If you select a point that is not on any object when AutoSketch expects you to select an object, it will start a box. The box can be used to select more than one object (for moving, copying, erasing, etc.) If you move the pointer to the right to create a box, only objects fully within the box are selected. If you move the pointer to the left, any object that the box crosses is also selected. The manual refers to these two types of boxes as a window box and a crosses box.

The drawing screen in AutoSketch is divided into units called "drawing units", and when you specify coordinates, it is in terms of these units. The default screen, when you first start AutoSketch, is 9 units high and 14 units wide, and the origin (point 0,0) is at the lower left corner. You can, of course, zoom in or out to have any number of drawing units or fractions thereof on the screen, and you can pan to place the origin just about anywhere. With AutoSketch, as with any good CAD program, your computer screen is just a window looking at a virtually infinite drawing area.

The basic capabilities of AutoSketch include the ability to draw lines, circles, arcs, polygons, curves, and fill regions. To draw a curve, you place 3 or more points, called "control points", and AutoSketch fits a curve to them. If the last point is the same as the first point, autoSketch creates a closed curve. If you are using a pointing device and you want to make a closed curve, the last point only has to be reasonably close to the first point. Unless you have "snap" turned on, it is difficult to point to an exact coordinate, but AutoSketch makes up for this in some cases, as when you close a curve.

While you are drawing the points for a curve, AutoSketch connects them with "rubber band" lines, and does not show you the actual curve until you place the last point. Then the lines go away and the curve appears. If you want to see what the control points were after you have drawn the curve, you can turn on a toggle called "frame". AutoSketch has a "stretch" command which can be used to, among other things, adjust a curve after you have drawn it. By using stretch, you can move one or more of the control points after the curve has been drawn. In order to select a curve to stretch, or to move or copy it, you should toggle frame on so that you can see the control points. You cannot move or copy a curve with frame off unless you put a window or crosses box around the entire curve, including the area occupied by its control points.

While AutoSketch has a built-in command for drawing circles, it does not have a command for drawing ovals (ellipses). However, the curve command can be used to draw ovals if you draw a closed curve with the control points arranged in a diamond shape. The width and height of the diamond should be 1.5 times the desired width and height of the oval. The illustration on this page shows an oval 4 drawing units high by 2 units wide.



Draw a four-point curve using these points.

Result with frame on.

Result with frame off.

To draw an oval with AutoSketch, draw a four-point curve with the points arranged in a diamond pattern.

The "polygon" in AutoSketch is a series of connected lines. Like the curve, it can be open or closed. The "fill region" is a closed polygon that is filled in (painted) after it is closed. The fill region command is the only command available for drawing solid objects. If you want to draw a solid colored circle or closed curve, you must use the circle or curve command first, and then start the fill region command and put several points around the edge of the circle or curve. You can zoom in as much as you want for precision placement of the points. A fill region, polygon, or curve can have as many as 100 points.

To help you make multiple copies of a figure arranged in a pattern, AutoSketch provides the "block array" and "ring array" commands. With the block array command, you can draw a brick wall by drawing one or two bricks, and then making an array of them. The ring array command makes it easy to draw something like the HUG symbol. You just have to draw one of the six segments, and then make a rotating ring array. You can also make non-rotating ring arrays, where the repeated figures are all at the same angle, but still arranged in a circle. Any figure, no matter how complicated it may be, can be repeated in an array. That is because there is a "group" command, that lets you combine several drawing objects into one object, so that when you point to it for the purpose of duplicating in an array, etc. , the whole figure is selected.

As with any good drawing program, it is possible in AutoSketch to copy any part of a drawing to another part of the "page". If you are drawing a schematic diagram that has several AND gates in it, you only need to draw one, and then copy it to where it is needed. You can also load a previously made drawing from a disk and use it as a "part" of another drawing. So you can make a library of electronic or other symbols that can be used whenever you need them. If the symbols are the wrong size, they can be scaled after you load them. I would suggest that if the symbols are much larger than you need, that you load them away from your drawing on a blank part of the "page", scale them, and then move them to where they are needed.

The scale command can be used to scale any part of a drawing or a whole drawing to another size. Scaling is somewhat limited in that you can only scale in tenths. For example, if you are scaling upwards, you can scale to 1.1 times the original size, 1.2, 1.3, etc. If you are scaling downwards, you can scale to .9, .8, etc. However, you can scale in stages if you need a result that is not an even tenth. For example, if you need to scale a figure to 1/4 its original size, which is .25 in decimal, you can scale by .5, and then scale again by .5. If you need to scale to 1.25, scale to 12.5 and then to .1. You can use the group command on a complex figure before you scale it to make it easy to select for multiple scaling. The scaling amount is selected by moving the pointer while you watch either the figure (if you want to scale by sight) or a numerical indicator. If snap is on when you begin to move the pointer to begin scaling, it can cause scaling to jump in greater steps than one tenth. Therefore, be sure to turn off snap when you are scaling.

When it is time to put your completed drawing on paper, AutoSketch really shines out among other CAD programs. Two features, called the "plot box" and the "clip box", are provided to help you print or plot your drawing. The plot box allows you to scale your drawing precisely (drawing units to inches or millimeters on the page) and position it on the printed page. You can even view on the screen the way it will be positioned on the page, and move the plot box around until the drawing is where you want it. The clip box allows you to print a portion of a drawing that is smaller than the plot box. When I first started using AutoSketch, I found the concept of the plot box and clip box a bit strange, but now I wish other CAD programs had those features also.

Evaluation

I have tried Generic CADD, Prodesign (now called Design CAD), AutoCAD, CADKEY, and AutoSketch, and the one I use most is AutoSketch. Unless you need a high powered CAD program, or 3-dimensional capability, I see no need to pay more or put up with the higher learning curve of other CAD programs. There are, however, some improvements that could be made to AutoSketch. The scaling command could be improved, and the ability to scale drawings brought in as parts before they are placed on the page, as can be done with AutoCAD, should be added. Text can only be left justified, and must be moved after it is placed if you want it to be centered or right justified. The screen height to width ratio is not adjustable as it is with AutoCAD and some other CAD programs, so if circles do not appear circular on your screen, you either have to put up with it, or adjust your video monitor.

In spite of these shortcomings, AutoSketch is an excellent low cost CAD program. I have been using version 2 for 3 or 4 months now, and I have not encountered a single "bug". Autodesk maintains a technical support hotline for those who have problems, and they will return your call (if you are a registered user) if their technical support people are all busy when you call. I usually don't have trouble with software (I read the manuals), but I did call them about adjusting the screen ratio, so I can attest to the friendliness and willingness of their support people to try to help. I find this remarkable, because the company had only been selling software that costs in the thousands of dollars before they produced AutoSketch. There is a tendency to think that a company like that would be a bit "snooty" when it comes to the low cost software customer, but Autodesk is not. For additional support, Autodesk maintains a forum on CompuServe(tm). There you will find ADI device drivers and other support software and information.

Where to Get It

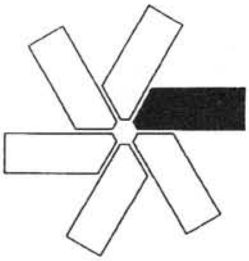
I got AutoSketch version 1 back when it was in the Heathkit catalog, and then I upgraded to version 2 for $25 directly from Autodesk. If you have version 1, call Autodesk using the phone number in your AutoSketch manual (it is the support number, but they can direct you to the proper people for upgrading).

If you are not upgrading from version 1, I would suggest that you get AutoSketch from one of the larger mail order houses. Here are three, with their phone numbers, addresses, and their prices for AutoSketch.

Telemart
(Order AutoSketch by name)
$85 plus $7 shipping (Federal Express)
(800) 426-6659
8804 N. 23rd Avenue
Phoenix, AZ 85021

PC Connection
(Order No. 4519)
$95 plus $3 shipping (UPS 2 day)
(800) 243-8088
6 Mill Street
Marlow, NH 03456

TigerSoftware
(Order No. ZX109)
$95 (free Federal Express shipping)
(800) 888-4437



The Ring Array command makes it easy to draw the HUG symbol with AutoSketch.

800 Douglas Entrance
Executive Tower, 7th Floor
Coral Gables, FL 33134

Continued from Page 10

structions that causes them to repeat for a count in CX, or until the Z flag is not set as a result of the subtraction (which would indicate that two elements are not equal). This instruction is physically the same as the REP instruction, but it has a different meaning when used with SCAS or CMPS than when used with MOVS, LODS, or STOS. REPZ is most often used with CMPS, to see if two strings contain the same elements. The Z flag will be set when the instruction is completed if the two strings are equal, and it will be cleared otherwise.

REPNZ/REPNE (REPeat while Not Zero/REPeat while Not Equal) — This instruction is a prefix for SCAS and CMPS that causes them to repeat for a count in CX, or until the Z flag is set as a result of the subtraction (which would indicate that two elements are equal). REPNZ is most often used with SCAS, to search a string for a particular element. The Z flag will be set when the instruction is completed if a match is found, and it will be cleared otherwise.

If REPNZ is used with MOVS, LODS, or STOS, it has the same effect as REP (which is the same thing as REPZ).

Segment Overrides

As I stated in Part 2 of this series, the SI register is "linked" to the DS segment register, and the DI register is "linked" to the ES register when they are used with string manipulation instructions. However, it is possible to override the DS register using another special prefix instruction (but you cannot override the ES register in string manipulation instructions). That means that you can link the SI register to the CS register or to the ES register. To specify a segment override in an Assembly Language instruction, you just put the name of the new segment register you want to use in front of the argument to the instruction with a colon (:) after it.

Segment overrides can also be used with data transfer instructions (covered in Part 3). For example, if you have the instruction:

```
MOV     AX,[SI]
```

the value pointed to by the SI register, in the DS segment, will be copied to the AX register. But if the data is in the segment referenced by the ES register, you could access it with

```
MOV     AX,ES:[SI]
```

To specify overrides in string manipulation instructions, you can use dummy arguments, as in

```
REP MOVS BUFFER2,ES:BUFFER1
```

or

```
LODS    CS:BUFFER1
```

Note: If you have two prefixes in a repeated instruction, as the REP MOVS example above, you can run into trouble with some members of the 8086 processor family. Normally, the processors allow interrupts to occur while repeated instructions are in progress, and they will resume after the interrupt is over. However, if a repeated instruction having a segment override is interrupted, it may not be resumed correctly after the interrupt is over. For that reason, you should disable interrupts (using the CLI instruction, which will be discussed later) before you execute the repeated instruction, and enable them afterwards.

This concludes this part of my Assembly Language series. Next time, I will present the branch instructions and machine control instructions.

PART 2 DOS AND UNIX

TOM BING
2755 CAROLYN DRIVE
SMYRNA, GA 30080

FEATURES OF MKS TOOLKIT

Due to an oversight on our part, we ran Part 3 of this series before running Part 2. Here is the second part. We're sorry for any inconvenience this may have caused.

In my first article, the intent was to show some of the more subtle ways in which the 'same' DOS and Unix commands produce somewhat different results. In this article, we will talk about a way to have the best of both worlds — the DOS programs and utilities that you've come to rely on, plus some of the more powerful tools from Unix. One package for DOS that adds Unix functions is the MKS Toolkit, available from Mortice Kern Systems (see addresses at the end of the article). This package provides over 110 commands and utilities. A few of the major ones are surveyed below, just to whet your appetite. Please bear in mind that the examples given are from actual runs on a personal computer with the Toolkit installed; this article will not deal with vaporware. The Toolkit isn't a multitasking system like Xenix. It will, however, provide an almost exact match with System V Unix in those commands it includes, both in the way the commands are issued and in the results they produce. The areas where a perfect match can't be attained are carefully explained in the Toolkit manual. For instance, we have to stick with the DOS limits on file names (8 + 3 characters). The Toolkit is installed on systems with existing DOS files and 'understands' the DOS file system layout. Here are some major parts of the Toolkit:

sh: This is the Korn shell, so named after its Bell Labs author. This shell is usually named 'ksh' on System V Unix machines. It is 'sh.exe' in the Toolkit. This is the Unix command interpreter and replaces 'command.com'. The shell is also a complete programming language in its own right. Users may create 'shell scripts', similar to DOS batch files. Shell scripts are more powerful because they can prompt the user for input, set internal variables of their own, perform computations, test for a wide variety of conditions, and do conditional execution. For example, the script below was written and executed on my PC using the Toolkit:

```
# newer.ksh - by Tom Bing 7-26-88
# Reports on which of two files was
# changed most recently.

echo "First file name: \c"
read file1
echo "Second file name: \c"
read file2

if [ $file1 -nt $file2 ] # if $file1 is
   newer than $file2
then
     echo "\nFile $file1 is newer."
elif [ $file2 -nt $file1 ]
then
     echo "\nFile $file2 is newer."
else
     echo "\nFiles $file1 and $file2 are
   exactly the same age."
fi
```

Here's how a run of this script looks:

```
c:/>newer
First file name: autoexec.bat
Second file name: config.sys
```

File config.sys is newer.

This example could have been a lot fancier. We could have verified that file1 and file2 were in fact existing files (or directories, if need be). The shell is a powerful language for everyday tasks involving file manipulation and copying. For instance, I use a shell script to fire up my Procomm communications program. Procomm logs the session to a file with the same name every time ('mainfram.log', for example). After the Procomm run ends (or 'exits' as we say in Unix), the script renames the 'mainfram.log' file to something like '88072601.log', which is the first log file created on July 26, 1988. If I call the host computer back the same day, the next log file would automatically be named '88072602.log'. 'sh' can do this because it can check for the existence of files (like the last log file) and do the simple math involved in 'bumping' the sequence number (01,02,etc.) that appears in the log file name after '880726'. The shell gets the '880726', by the way, from the 'date' command. It then pastes together ('concatenates' is the fancy term) the '880726', '02', and '.log' to form the file name. The script that does all this has about 35 lines of shell code and 22 comments. All shell scripts are interpreted, not compiled. The neat part is that your 'interpreter', sh.exe, is there all the time and doesn't have to be loaded into memory, like BASICA. This means you can be running your shell script right after you finish writing it. If it 'bombs', the shell will give an error message showing the number of the line in the script that it can't understand.

Now let's look at the three text editors in the Toolkit: ed, sed, and vi.

ed: 'ed' was the original Unix editor. The present 'ed', though much refined, is still a line-oriented editor; it neither knows nor cares whether your terminal is a PC or a teleprinter, because it doesn't use screen attributes. 'ed' is useful for making the same kind of change throughout a file. If we wanted to remove trailing blank spaces from every line in a file, the 'ed' command is simply this:

```
1,$s/ *$//
```

Let me translate that into English: "Starting at line one through the last line of the file (1,$), replace the pattern after the first slash (/) with the pattern after the second slash". The pattern after the first slash is three characters: a space, an asterisk, and a dollar sign (/ *$/). 'ed' understands the space and asterisk to mean 'zero or more spaces'. The closing '$' means that the pattern will only be replaced if it occurs at the end of a line. In other words, blank spaces followed by other characters on the line won't match the pattern. Notice that the '$' in a line range means 'the last line'. In a pattern, it means 'the end of a line'. The closing slash marks the end of the second pattern. Since there is nothing between slashes two and three, that is what will replace trailing blanks on the line — no characters, a zero-length string.

The advantage of 'ed' over EDLIN is that 'ed' understands regular expressions — the patterns mentioned above. Instead of representing only a fixed series of characters, regular expressions can also describe a pattern to be matched in a more general way, such as "all digits followed by either a period or a space". As a regular expression, this becomes:

```
/[0-9][. ]/
```

This ability to understand regular expressions is a common feature of Toolkit (and Unix) programs that process text files. As you have seen above, some characters in a regular expression aren't taken literally, but are used to indicate a range of characters, a repetition of characters, or a location (start or end of line). These are called 'metacharacters' and they are listed below:

```
\ ^ $ . [ ] | ( ) * + ?
```

We can turn off the special meaning of a metacharacter by quoting it with a backslash. Thus, the expression /\\/ is 'looking' for a single backslash. Also, inside the square brackets, the metacharacters (generally) have their literal meaning.

'ed' can also be used with a script of editing commands to make a mass change to a group of files at one time. For example, if we had a group of C programs that printed reports, we might want to change a header-printing function so that it printed not only a page number, but the date also. Before the change, a program line would look like this:

```
header(pgno);
```

Afterwards, it would be:

```
header(pgno, today);
```

We could create a file (an editing script) named 'changes' with the following contents:

```
1,$s/header[(]pgno/&, today/g
w
q
```

The "[(]" above turns off the special meaning of "(", so we are searching for a literal left parenthesis. The special character "&" means "use whatever matched the first regular expression here". Thus, the first line above will cause every occurrence of "header(pgno" to be replaced with "header(pgno, today". The 'g' at the end of the line means that the substitution is to be done 'globally' — everywhere it occurs on a line. Without the 'g', if we had two calls to 'header' on a single line, only the first would be changed.

The 'w' on the second line means 'write the preceding changes to the file'. It's like '^KS' in WordStar. The final 'q' means 'quit, return to system prompt.'

These editing instructions could be applied to each of our C programs in turn using a shell script like this:

```
for i in *.c
    do
    edit $i < changes
    done
```

We can walk off and get coffee now while the shell script is running, and when it completes, all our C code will be changed. It's good practice to check out the 'ed' script thoroughly using a single test file. Once we're sure the 'ed' script does only what we want, we can kick off a shell script like the one above.

Another way to do editing 'untouched by human hands' is via 'sed'. The 'sed' (stream editor) program is useful when making simple changes in files which may be too large for other editors. Many 'sed' commands and concepts originated with 'ed'. The word 'stream' refers to the fact that input to 'sed' is often the output of other programs. Before this output goes in a permanent file, it is piped through 'sed'. The editing script for 'sed' may be either a file or a short command string as in the example below.

```
rmctl < mainfram.log | sed '/ENTER PAS
SWORD:/c\ ' > log2
```

This command is part of a larger shell script mentioned earlier which runs Procomm. Its meaning is "Run the rmctl program, using mainfram.log as its input. Feed the resulting output to sed, and write the output from sed to the log2 file". The 'mainfram.log' is the 'session log' file. It contains all the commands given by me and the responses from a remote computer during a phone call to that computer. The 'rmctl' program (which I wrote) removes control characters from 'mainfram.log'. The output of 'rmctl' is passed to 'sed'. 'Sed' gets its editing instructions from the string in single quotes, which tell it to replace any lines containing 'ENTER PASSWORD:' with a blank line. There will still be two files after this command finishes, named 'mainfram.log' and 'log2'. 'Sed' does not change the original input, because it could be either a 'stream' (as in this case) or a file. This idea of feeding the output of one program to the input of another is basic to both Unix and the Toolkit.

vi: This is the Unix, and Toolkit, full-screen editor. While it has some sterling qualities, you may not want to run out and dump WordStar or whatever PC editor you have in favor of 'vi'. People make a heavy intellectual investment in learning a good editor and becoming productive with it. Unless your current editor simply won't do your bidding or is too clumsy, you should probably stick with it.

Now for the 'pros' about 'vi'. It's handy to have it on the PC if you also use it on a Unix machine. It's a programmer's editor; it recognizes bigger units of text than lines, words, and characters. For instance, you can move between sentences, paragraphs, and sections. You can also make sure that punctuation in a program which always comes in pairs, such as {}, [], and (), really does balance out.

The Toolkit version of 'vi' allows graphics characters in a text file. It also permits you to change the standard 'vi' characters for cursor control to the PC's cursor movement keys. There is a wealth of option settings which can be placed in the 'ex.rc' configuration file for 'vi' governing cursor size, auto-indent, and so forth. The Toolkit documentation, generally high quality, is exceptionally thorough in this area, telling you things about 'vi' setup that are not disclosed in most Unix manuals.

sort: Vastly more powerful than DOS SORT, this Toolkit utility can sort on either columns, fields, or sections of fields. You can use multiple sort keys, allowing descending sorts on key 1 and ascending sorts on key 2. If sorting is done by fields, the fields are assumed to be separated by tabs, unless the '-t' option indicates another field separator. Let's look at 'member.lis', a file giving the names, ages, and hometowns of members of a senior citizens' club:

```
Spyros, Effie         :  67:Vista, KS
Meier, Ralph          :  65:Beantown, MA
Kringle, Kris         :1988:North Pole
Wyrtzen, Emilio       :  67:Biglake, NY
Stone, Judith         :  65:Holiday, MO
```

To sort this list by age first, with ages in descending order, and names in alphabetical order by age groups, the command is:

```
sort -t: +1r -2 +0 -1 member.lis >
   member.srt
```

The output 'member.srt' file is:

```
Kringle, Kris         :1988:North Pole
Spyros, Effie         :  67:Vista, KS
Wyrtzen, Emilio       :  67:Biglake, NY
Meier, Ralph          :  65:Beantown, MA
Stone, Judith         :  65:Holiday, MO
```

In fact, if we had used '+1rn' above as our first sort field key, we wouldn't have needed spaces in the age field.

It takes a few experiments like the one above to realize the power and versatility of 'sort'. The Toolkit manual section on this command gives good examples and explains the '+' and '-' options (defining sorting keys) quite thoroughly.

cut: This utility is often used with 'sort' to pull out particular fields in a file for further manipulation. We could take the ages out of the 'member.srt' file above with 'cut' and pass them to another utility called 'uniq' to give us a count of the members in each age group. The command would be:

```
cut -d: -f2 member.srt | uniq -c >
   age.grp
```

The file 'age.grp' will look like this:

```
1 1988
2   67
2   65
```

paste: This command takes two or more input files and pastes them together side-by-side. The same file can appear more than once in the input list. As an example, let's say we have a file called 'spell.err' containing misspelled words, one per line. If we want to create a two-column list with misspellings on the left and correct spellings on the right, we would do this:

```
paste spell.err spell.err > cross.ref
```

The 'cross.ref' file now contains lines that look like this:

```
definate        definate
seperate        seperate
```

Of course, 'paste' simply outputs two copies of the misspelled word on each line. Using a word processor, we could edit the right column in 'cross.ref', correcting the spelling of the second word only, and have a list useful to editors and proofreaders. By default, 'paste' puts a tab character between columns. This can be changed with the '-d' option. More than two files can be pasted together at a time:

```
paste -d: file1 file2 file3
```

pr: If there had been nothing else in the Toolkit, this would have almost made it worth the money. 'pr' is a print formatter that can add margins, arbitrary headings (or the file name), and other format controls when a file is being printed. Take a look at 'pghp.ksh', a shell script for printing files with page breaks, left margins, and headers on a HP Laserjet printer:

```
cp c:/s/dos/escnorm.dat prn
pr -fF -o7 -l60 $* > prn
```

'cp' is the file copy command in Unix and the Toolkit. The 'prn' file is the same as DOS PRN. The 'escnorm.dat' file simply contains the literal Escape-E sequence needed to set the Laserjet back to its default print mode. The 'pr' options used are:

-f Use form feeds between pages.
F Fold lines (don't cut them off) if they go past 72 columns, printing the folded part on the next line.
-o7 Left margin offset 7 spaces; printing starts in column 8 instead of column 1.
-l60 Only 60 lines per page; that's all the Laserjet can print with default font and pitch.
$* When the 'pghp' command is entered on a line, interpret everything after 'pghp' as a list of file names to be printed.

If we typed 'pghp member.srt', using the above shell script and the file described earlier, here's what would print on the Laserjet:

```
Sat Jul 30 12:53:36 1988 member.srt
                             Page 1, Line 1

Kringle, Kris        :1988:North Pole
Spyros, Effie        :  67:Vista, KS
Wyrtzen, Emilio      :  67:Biglake, NY
Meier, Ralph         :  65:Beantown, MA
Stone, Judith        :  65:Holiday, MO
(Form feed)
```

With other 'pr' options, we could have used a label other than the file name in the header, or suppressed headers and bottom margins altogether. The date and time used is the file's modification time, or if input to 'pr' is piped from another program, the current system date and time.

awk: This utility deserves a whole article to itself, and may get one. The easiest way to describe 'awk' is to say that if you have a text file that you want to transform in some way, and you can describe the transformation in a step-by-step process, then 'awk' (with help from 'sort' sometimes) can do it. The step-by-step process is written as an 'awk script'. A simple example will show how powerful 'awk' is. If we issue the command:

```
ls -l member.lis member.srt /s/dos/pghp.ksh > file.lis
```

the file 'file.lis' contains:

```
-rwxrwxrwa 1       56 Mar 15 16:19 /s/dos/pghp.ksh
-rwxrwxrwa 1      193 Jul 30 12:52 member.lis
-rwxrwxrwa 1      193 Jul 30 12:53 member.srt
```

Though the above list may look a little strange, remember that 'ls' is describing ordinary DOS files to us. We can still use DIR and other DOS commands after the Toolkit is installed. For the purpose of creating the list above, though, 'ls -l' is better. The third column (56, and so on) is the file size in bytes, just as DOS DIR gives us, and the last column is the file name. If we wanted just the file sizes and names in a Lotus spreadsheet, here's what we'd have to do:

1. Pull out columns three and seven (the last one) from 'file.lis'.
2. Put quotation marks around the file name.
3. Put the resulting two columns in a file with a '.prn' extension, so that the data can be imported into a spreadsheet.

Now, if we had an 'awk' script that would do this (let's call it 'makeprn'), we would run it like this:

```
awk -f makeprn file.lis > file.prn
```

Here's what's in 'makeprn'. Lines starting with '#' are comments.

```
# The 'quote' variable is given the
# octal value 042 (") before processing
# any lines of input.

BEGIN { quote = "\42" }

# If a line has exactly 7 columns
# (fields), print column 3 out as a whole
  number, six  digits max,
# put quotes around column 7 and print it
   beside column 3.

NF == 7 { printf("%6d %s\n",$3, quote $7
   quote) }
```

Now, finally, here's what's in 'file.prn' after the 'awk' command above is given:

```
 56 "/s/dos/pghp.ksh"
193 "member.lis"
193 "member.srt"
```

If you don't count comments, that was done by writing two lines of code and issuing one 'awk' command. I don't think I could do it that easily in BASIC.

A historical note: 'awk' is named from the initials of its creators, Alfred V. Aho, Peter J. Weinberger, and Brian W. Kernighan, all of Bell Labs, and mighty gurus indeed in the realm of Unix. As near as I can tell, the Toolkit version of 'awk' has all the features of the latest and greatest version that AT&T began distributing with Unix System V Release 3.1. Addison-Wesley has published a definitive new (1988) book by these three gentlemen titled "The AWK Programming Language". Though we haven't touched on all the capabilities of 'awk', that's what it is: a programming language with a whole raft of math, string manipulation, conditional execution, looping, output formatting, and other features.

I have used 'awk' for creating cross-reference lists from source code files, changing the order of data in a file (swapping columns 4 and 6, etc.), printing size of files in both characters and disk sectors — the list is endless. If you can get data into a text file, 'awk' can do things with it that the creators of the data never anticipated. 'Awk' is another item that by itself justifies the price of the Toolkit.

We could go on, and if you really liked these articles, maybe we will. A future article could be written describing Toolkit installation and choices of different setups, along with more of the utilities included. The MKS Toolkit is something I really wouldn't do without on my computer.

Sources

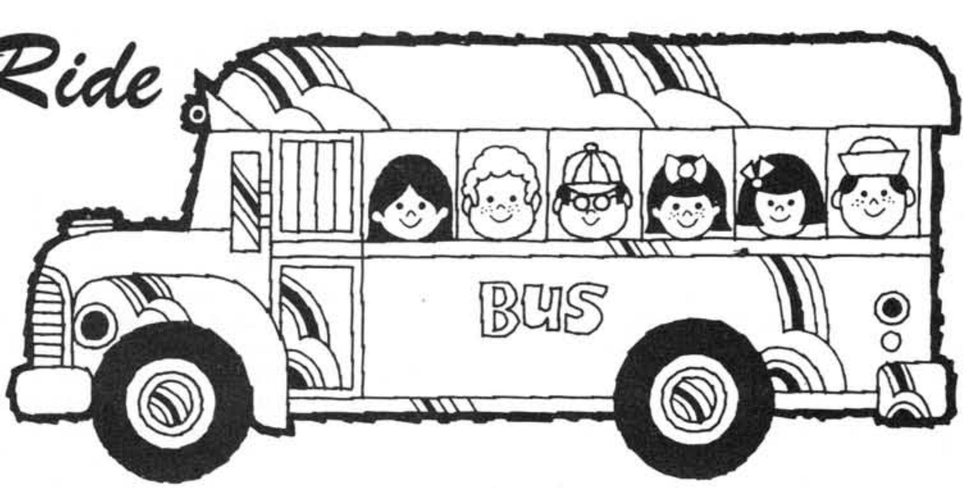
MKS Toolkit: Mortice Kern Systems, Inc.
35 King Street North
Waterloo, Ontario
Canada N2J 2W9
(519) 884-2251 (Canada)
(800) 265-2797 (US Order Line)

CompuServe User ID: 73260, 1043
Programmer's Connection
Order Processing Dept.
7249 Whipple Avenue NW
North Canton, OH 44720
(800) 336-1166 (US Order Line)

"The AWK Programming Language"
Addison-Wesley
(800) 447-2226

The list price of the Toolkit (Version 3.1) is $249.00 US at this writing. Programmer's Connection offers the product at $219.00 US in their Fall 1989 catalog. The Toolkit is not copy protected and requires MS-DOS 2.1 or later. ✱

Taking a Ride on the



Dennis L. Myers
717 Clover Lane
Temple, TX 76502

Z-180 Series Laptop Computer Bus

The Zenith Data Systems laptop computers are the best on the market. They offered a "Z-180" series of laptop computers for about two years, then supplanted them with the SupersPort/TurbosPort laptop line in the summer of 1988. Since dropping the Z-180 line, ZDS has also dropped support for those machines. In fact, they committed what I first felt was an unpardonable sin — failing to provide support for part of the Z-180 series while it was in production! (More about that later).

This two part article describes how I went about adding external floppy and hard drives to my Z-181 laptop and the difficulties I encountered. Part 1 reveals the year-long saga of my adventure, while Part 2 is a "how-to-do" description which allows using a Z-181 model 93 laptop with a 32 meg Seagate ST-238 1/2 height hard disk and an OMTI RLL (Run-Length-Limited) IBM-PC compatible hard disk controller. Users only interested in how I adapted the Z-181 to support non-ZDS external hard disks should still read Part 1 of this series. I will use my experiences with this project to support some conclusions about the prospects for feasibility of similar individual projects in the future. I believe there has been an evolution of the special relationship that has existed in the past between Heath/Zenith and its customers.

I bought my Z-181 Zenith Data Systems Laptop in January 1988. The Z-181 machines had been out for less than one year, but had already been upgraded by increasing clock speed and allowing the addition of an external hard disk drive (the original model only allowed adding an external 360k 5 inch floppy drive system). Three months later, ZDS announced that the Z-181 series machines were discontinued, to be replaced by the SupersPort line. I bought my machine expecting to be able to easily add an external floppy and hard drive when I was ready. As soon as I heard of the new SupersPort line, I checked into buying external ZDS drives for my Z-181 before it was too late. I was used to the high prices for peripherals, but the $400.00 asked by ZDS for a single external 360k floppy drive system, and $1300.00 for a 20meg external hard disk, floored me. It really didn't matter if I could afford it, because almost immediately after announcing the hard disk system was discontinued, it also became no longer available through the Heath/Zenith Computers and Electronics stores.

I quickly attempted to gather all available technical information on the Z-181 so I could put together a non-ZDS external floppy and hard disk system when (and if) I could afford them. I had built both a Z-89 and a Z-100 Heath computer, and had placed non-standard 5 inch and 8 inch drives in both of them, so I felt I had enough general understanding to complete the project. Also, the technical and hardware documentation for those machines had been outstanding. I naively thought the same would be true for the Z-181 laptop. I was wrong.

I chose as my first project adding an external floppy drive to my laptop. I found that my updated Z-181 (model 93) was very different from the original Z-181 (model 92), even though model 92 was only sold for little less than one year. Unfortunately, the only technical manual available for the Z-181 was (and still is!) a manual for the Z-181-92. It took me months to find someone at Heath/Zenith who had any information on the Z-181-93. The only papers available then (and now!) was a single sheet listing pin-outs for the external connectors on the laptop itself. That was enough to allow working out the details of adding a non-Zenith Data Systems external 360k 5 inch floppy drive.

The November 1988 issue of REMark has an article by me describing in detail how it can be done.

After my rather unsettling experience with the floppy drive project I decided to re-read the owner's manual that came with my Z-181, since it appeared to be the only documentation written especially for the Z-181-93. It accurately described my Z-181 in the scant details it provided about hardware. Many sections of the manual were identical to areas of the model 92 technical manual, but there were areas of disagreement as well. For example, the model 92 technical manual section on disk drives stated that high-density 1.2 meg 5 inch floppy drives are not supported by the on-board Z-181 floppy disk controller, even though the operating system (MS-DOS 3.xx) allows them. The model 93 manual stated that both the controller and the operating system allows them. Other statements in the owner's manual sent me down a blind alley for the better part of a year as I tried to make a hard disk work with my Z-181.

I knew my model 93 would support some kind of external hard drive system since ZDS had sold both a 10 meg (briefly) and a 20 meg (even more briefly) system for it. The model 93 has an external 50-pin ultraminiature connector to which

(Installing External Floppy & Hard Drives on Z-181's)

the Zenith Data Systems external hard disk system attached. My owner's manual said only two things about this connector: "This connector provides IBM-PC compatible bus signals to an external hard disk drive" and "This 50 pin connector provides access to the system bus for future expansion or connection of a hard disk drive system". The manual also stated that the ZDS external hard disk system for the Z-181 housed the hard disk controller, so I knew the Z-181 motherboard did not have an on-board hard disk controller.

I thus would need to develop an external controller/disk system, not just an external hard disk. Based on my manual, I figured all I needed to do was connect the "IBM-PC compatible bus signals" present at the bus connector to an IBM-PC compatible hard disk controller and drive. I worked out details such as providing power to the external controller circuits (which normally derive power from the motherboard . . . the Z-181 bus connector does not provide motherboard voltages). I obtained a single sheet of documentation from Heath that listed the pin-outs of the 50-pin ultraminiature bus connector on the back of the Z-181-93. Comparing it to specifications for the IBM-PC "I/O channel" expansion slots (found on all true PC-compatibles), I found that all lines required for a hard disk controller (as specified by IBM) were present. I hardwired those lines directly to an OMTI hard disk controller. I must have tried a thousand times, but could never even get the Z-181 to acknowledge the BIOS ROM I knew was present on the controller card at hex address C8000. I gave up.

All this took about a year, during which time I talked to every technical person at Heath/Zenith who would speak to me. As had been my past experience, the Heath people were extremely friendly. They just didn't have the information needed, since the Z-181 was a short-lived system built in Japan (obviously imported for a specified period of time to fill a market gap for Zenith Data Systems). ZDS technical support persons were almost impossible to approach. Once they knew I wasn't representing a company with large orders, their interest evaporated. However, I did find several friendly persons in the Dallas Zenith Data Systems technical support office (thanks, Jim & Rick) who checked and found that they had absolutely no documentation on the Z-181-93 machine.

I also talked to people on the HUG BBS. Everyone was friendly, but none knew more than I. I was able to obtain a copy of the installation guide for the short-lived external 20 meg. hard disk system (model ZA-180-20) sold by Zenith Data Systems for the laptop, thanks to the efforts of a very kind lady in the Heath Parts Department. She photocopied the installation guide from one of the last drives in stock just before it was sold. The only information of value in the guide was a listing of pin-outs for the disk drive end of the ZDS cable that connected the laptop to the drive case. The documentation labeled them as "Interface Connector Pinouts". The significance of that label eluded me at the time.

Out of the blue in Spring 1989 (months afer I had stopped work on the project), Mr. Jim Collins contacted me wanting all the information I had pulled together. Mr. Collins also had a Z-181, and had been following the efforts of myself and others on the HUG BBS. He had found that Heath listed a replacement part for the 20 meg. ZDS ZA-180-20 external drive system for the Z-181. The part number was (Heath) #970-1809, and was listed as "Interface PC Assembly". Its price was $8.95. Mr. Collins obtained the part, which turned out to be a small circuit board with pins for power inputs, a few buffer-type ICs, a 62-pin connector to accept the cable from the laptop, and a 62-pin IBM-PC style male card-edge connector (but no documentation!).

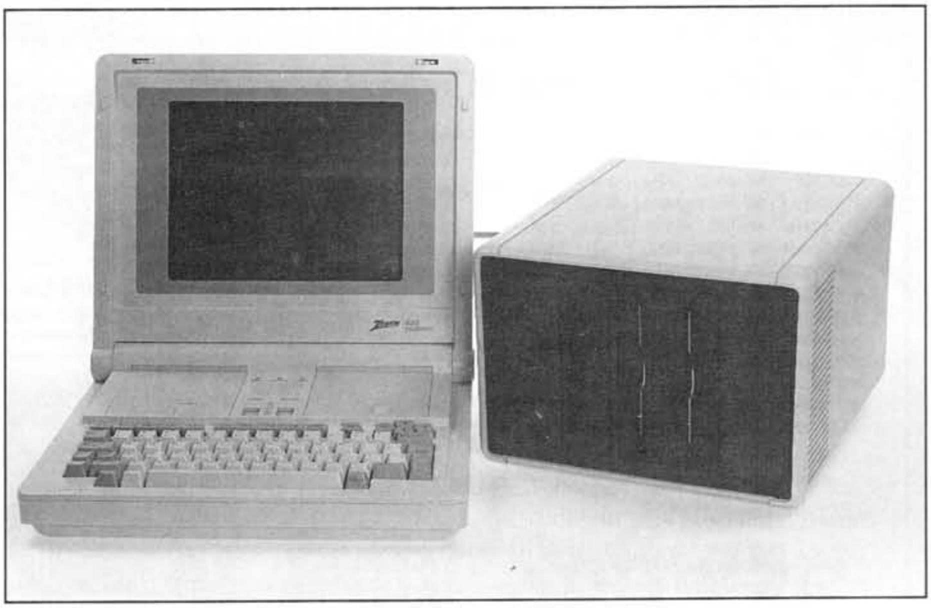
Eureka! The existence of this circuit board as a necessary component for the external hard disk drive system could only mean that the signals present at the back of the Z-181 laptop are NOT "IBM-PC compatible bus signals" as my manual had stated. They are unbuffered, "raw" lines from the motherboard. Add-on cards for IBM-PCs and compatibles are designed to plug into a 62-pin slot that accesses the IBM-PC computer motherboard bus via a BUFFERED interface that IBM calls the "I/O CHANNEL". The Heath Interface PC Assembly card clearly must be needed to emulate the "I/O Channel" expected by an IBM-PC compatible hard disk controller card, otherwise there would be no reason for it to be present in the ZDS drive system. Jim ran out of time to work on the project, and kindly sent his circuit board to me. The connections required seemed quite simple: Hook the laptop to the interface assembly using the ZDS cable sold to connect the Z-181-93 laptop to the external drive system, add power to the interface board, and connect a hard disk controller/drive to the board. The power pins were not hard to figure out (+5v, GND, +12v, -12v). I left a 4-position DIP switch on the board as it was shipped. I was concerned whether the 62-pin card-edge connector on the interface board was really pin-compatible with the IBM card definitions. I had no schematic to go by, but many easily verifiable pins (ground, various voltages, etc.) matched up. I wired it up and fired it up ... and it didn't work! Maybe the DIP switches were wrong. Sixteen tries later (2x2x2x2) I set them back the way they came. I then went back to basics ... did the laptop even know a disk controller was "out there"? DEBUG showed no memory existing at C8000 hex (the reserved starting address for hard disk controller BIOS ROM in IBM-PC compatible systems). Going over the Z-181-92 technical manual, I found how the Z-181 is told of the existence of external devices (EXT ON signal). Tracing that signal on the interface board, I found that it was not properly grounded on the board Jim sent to me. Thirty-five minutes and two soldered jumper wires later (and over a year since starting), I was booting the Z-181 from the Seagate drive. The system runs beautifully, and the 62ms access speed of the drive is fast enough for the 8088-based Z-181. The hard drive really extends the usefulness of my Z-181. I subsequently obtained another (new) Interface PC Board from Heath, and found that indeed the first board was faulty in its grounding of the EXT ON line. The replacement worked perfectly without modification.

Was it worth it? The answer is conditional, and not strictly related to cost. Using all Zenith Data Systems parts (cables and drive systems), it would cost $1700.00 to add a single 360k floppy and a 20 meg. hard disk to the Z-181 . . . if the systems were still available. My cost to add two 360k five inch floppy drives and a 32 meg. hard drive was:

External drive case and power supply	$60.00
Two 360k 5 inch floppy drives	140.00
RLL hard disk controller and 32 meg. hard disk	260.00
ZDS floppy drive cable	50.00
ZDS hard disk drive cable	100.00
Interface PC board, plus this & that	25.00
TOTAL	$635.00

But I didn't do it just to save money. I spent dozens of hours thinking, talking, writing, and testing. I did all that for fun, the same reason I built my Z-89 and Z-100. I doubt if many others (HUG hardware freaks like me excepted) would continue working on this type of project after being unable to find technical documentation for the computer system. This project made me realize just how much Heath/Zenith has changed as it has evolved from a small-scale producer of desktop computers (when nobody knew much about how to use them, and those who worked with them did so more for fun than utility) to a major market supplier driven by the needs of users (not enjoyers) of small computers. A business user could not afford to do what I did. It would be much more logical to write off the Z-181, and upgrade to a current generation laptop with a built-in hard disk.

Like others, I miss the old Heath days of computing. But the problem is not Heath/Zenith's, its mine. If the Heath computers I liked in the late 70's and early 80's still existed, it would be out of business. (Remember the premature death of the Z-100, a machine far superior to the



The completed external drive system houses a 32 meg. hard disk and two 360k five inch floppy drives.

IBM-PC it competed against?) Heath/Zenith has become a major force in computers exactly because it has been able to change with the times. The people it serves now are different from the early days. One thing does still exist, however. It is the commitment of Heath to support its buyers to the best of its ability. Even though the technical support group could not help me with the laptop as they had with machines past (through no fault of their own), every piece of information I came up with was directly due to the efforts of Heath people. My anger with ZDS for not producing technical manuals for my discontinued model Z-181 subsided when I logically asked myself "Why should they?". The decision to develop a new line of laptops after a very short production run of Z-181s was a good business decision. Other manufacturers were producing new generation laptops that blew the Z-181 out of the water. ZDS could not afford to mess around with technical support on a product it was discontinuing. How many Z-181 users (nearly all of whom are business people) would ever need the information I wanted? The owner's manual that came with my Z-181 covered most common service questions. In fact, compared to documentation I have seen from other major vendors, its level of hardware detail was outstanding. Heath/Zenith did not sell my laptop to me under the same circumstances as my kit version Z-89 & Z-100. Kits require generous technical support because kits are built by people who will want to do all kinds of crazy things with their machines after they build them. Business machines need clear operating instructions, intense applications support, and a good warranty (all of which the Z-181 had). Zenith Data Systems' approach makes good business sense: If it breaks, replace it for free . . . but don't waste time or money telling the customer how it's built or how to fix it . . . just how to use it.

This means we "early Heath users" should not expect to be able to manipulate Heath/Zenith machinery as we did in the past unless we are willing to accept new conditions for the game: "You're on your own, kid". For this reason, I expect to find fewer and fewer hardware modification projects for ZDS machines to be reported in REMark in the future. They will be supplanted by "how-to-do" software modifications, and other articles appropriate to users likely to be more interested in modifying the computing environment than the computer itself. I do find it ironic that the reason I was successful in completing this project without the help of Heath/Zenith was because I have the knowledge of hardware systems, circuit boards, power supplies, typical pin configurations, disk drives, and operating systems I gained as a direct result of the astonishing level of support provided by Heath and HUG in the past. I am grateful to Heath for the education. Part 2 of this article will include complete instructions on building the 32 meg. hard disk/dual 360k floppy system housed in one external case that I put together. ✱

FREE MICROSOFT WINDOWS * DETAILS BELOW

Seagate ⇨HARD

MODEL	CAPACITY/FORMAT/SPEED/SIZE	DRIVE ONLY	XT KIT
* ST-125	21 MEG / MFM / 40 MS / 3.5"	$224.00	$274.00
* ST-138	32 MEG / MFM / 40 MS / 3.5"	$272.00	$322.00
* ST-138-1	32 MEG / MFM / 28 MS / 3.5"	$302.00	$352.00
* ST-151	42 MEG / MFM / 24 MS / 3.5"	$348.00	$398.00
* ST-138R	32 MEG / RLL / 40 MS / 3.5"	$253.00	$308.00
* ST-157R	49 MEG / RLL / 40 MS / 3.5"	$281.00	$336.00
* ST-225	21 MEG / MFM / 65 MS / 5.25"	$199.00	$249.00
* ST-251-1	42 MEG / MFM / 28 MS / 5.25"	$307.00	$357.00
* ST-4096	80 MEG / MFM / 28 MS / 5.25" FH	$578.00	$628.00
* ST-238R	32 MEG / RLL / 65 MS / 5.25"	$213.00	$268.00
* ST-277R-1	65 MEG / RLL / 28 MS / 5.25"	$343.00	$398.00
* ST4144R	122 MEG / RLL / 28 MS / 5.25" FH	$618.00	$673.00

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

*** ZENITH PC COMPUTER UPGRADES ***

SmartWatch from FBE RESEARCH

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

Z-150/160 MEMORY UPGRADE

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

Z-150 SERIES HARD DISK DRIVE KIT

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

* ST-125/Z150 Kit	21 Meg, 40 MS,	$278.00
* ST-138/Z150 Kit	32 Meg, 40 MS,	$326.00
* ST-151/Z150 Kit	42 Meg, 24 MS,	$404.00

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,	$349.00
* ST-138/Z148 Kit	32 Meg, 40 MS,	$397.00
* ST-151/Z148 Kit	42 Meg, 24 MS,	$473.00

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

15718 SYLVAN LAKE * HOUSTON TX 77062

FLOPPY DISK DRIVES

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

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

PAYLOAD CUSTOM ASSEMBLED COMPUTERS

⇨We assemble 8088 XT, 80286 AT 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-1492	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	$ 89.00
⇨EGA480	PARADISE AUTOSWITCH 640x480	$ 99.00
⇨VGAPLUS	PARADISE AUTOSWITCH 800x600	$ 172.00
⇨VGAPLUS16	PARADISE AUTO 16 BIT 800x600	$ 207.00

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	SIM 1Mx9 80 ns ....$89.00.
⇨41256 256x1 120 ns $2.30	SIM 256x9 80 ns ....$34.00.
⇨1Mbit 1Mx1 80 ns $9.40	

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. 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. 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

Z-100 SERIES SOFTWARE

⇨PART NUMBER /DESCRIPTION	LIST PRICE	SALE PRICE
⇨CD-463-2 Condor File Manager	$175.00	$10.00
⇨MS-253-1Basic-80 (8-bit)	$175.00	$10.00

MICROSOFT WINDOWS version 1.04 for PC (not Z-100)

⇨FREE with any order from this ad over $30.00. Just ask for it and add $5.00 for
⇨shipping and handling. Includes 5 disks and 300+ page manual. Offer good until
⇨existing inventory given away (about 300 packages).

⇨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

The Perils of New Software

Ed Demaree
P.O. Box 23944
Tigard, OR 97223

There are a number of perils lurking for the unwary in the seemingly innocent process of upgrading your software to a new version or adding a new program to your hard disk. We'll look at the problems, what you can do to avoid them, and see how they applied to a PC Tools Deluxe upgrade to Version 5.5.

The upgrade process starts innocently enough. You receive a notice in the mail about all the latest enhancements to your program. So you decide to spend the bucks and get your upgrade package. (Central Point Software has an excellent upgrade policy and even sent a free disk to all registered users of PC Tools Deluxe version 5.0 to fix some bugs in the program.)

The first peril is copy protected programs. These dragons of the software world are a rare breed nowadays, but can still be found lurking on odd programs (mostly computer games). The best solution is not to buy copy protected software in the first place. Most software companies have finally recognized that copy protection isn't worth the cost to them or the problems it creates for the legitimate user. The next best rule is to check before you buy copy protected software to be sure you can back it up with COPY II PC or a similar back up program. Be prepared for unusual problems if you must install copy protected software on your hard disk. Dealing with the various copy protected installation procedures is beyond the scope of this article, so we'll be looking at normal installation without copy protection. As a general rule, if the original program isn't copy protected, an upgrade won't be either.

When you get a new software package the first thing that usually falls out of the package is a note telling you how to install the software. In the case of the PC Tools Deluxe upgrade, they tell you that "Some files have been compressed to save disk space and will automatically be decompressed by PCSETUP. If you do not run PCSETUP to install your PC Tools files, they will not work correctly." This seems to leave the user no alternative to using their install program (PCSETUP).

Most software houses don't bother to tell you exactly what the install program does. Instead, the user is told to just "Follow the instructions on the screen for installation." In other words, "Trust me". Depending on the quality of the software, this can be an invitation to trouble. As a long time user of PC Tools, I had every reason to expect that their PCSETUP installation program would work properly and be user friendly. Nevertheless, I took the following precautions which I recommend before installing any new or upgraded software:

Back Up Your Hard Disk

First, do a complete back up of everything you don't want to lose just in case the install program manages to wipe out something or the "enhanced" program has a major bug. No reputable software house would intentionally release a program that way, but nobody can test compatibility with everything else and a backup is cheap insurance against your system becoming a victim as two programs fight for control of the microprocessor.

Dueling TSRs

Another peril to avoid is dueling TSRs. This applies only if your new or upgraded software is a TSR. (A TSR is a Terminate-and-Stay-Resident program. TSRs are intended to remain unobtrusively in RAM while you run other software, awaiting a "hotkey" or some special condition before becoming active. Some TSRs are incompatible with each other under any circumstances. Others can coexist only if loaded in a specific sequence.) The symptoms of this peril range from complete lock up of the computer to some very subtle quirks. (I once found a combination of TSRs that would dump my spreadsheet after 3 to 5 minutes of operation.)

Disable TSRs in AUTOEXEC.BAT

If the new or upgraded program is one of many TSRs, I recommend placing a Remark statement in front of all other TSRs in your AUTOEXEC.BAT. That way you can satisfy yourself that the new software works without possible duels. Then edit out the REM prefix from one TSR at a time and reboot the computer to check for problems. For example, the AUTOEXEC.BAT line for the TSR SMARTKEY changes from "SMARTKEY" to "REM SMARTKEY" (REM causes DOS to ignore the rest of a batch file line).

Duplicate AUTOEXEC.BAT and CONFIG.SYS

Next, make copies of your AUTOEXEC.BAT and CONFIG.SYS files and save them with a different name, such as AUTOEXEC.BAK or CONFIG.OLD, in your root directory. These files are the most likely to be corrupted by an install program and renaming your duplicates gives you an easy way to recover from this common peril.

Back Up New Distribution Disks

Place write protect tabs on the disks with the new software and make backup copies. This protects against two more perils; wiping out a distribution disk by mistake and defective distribution disks. The defective distribution disk is more common than you might think. On two occasions, I have received a defective disk from major software vendors. Few things are more frustrating than having an install program die in midstream because it can't read one of its own files.

If the directory on a distribution disk is O.K., but some files won't copy on your computer, you probably have either copy protected software or a defective distribution disk. There is a small chance that your floppy disk drive has problems, but if it reads everyone else's distribution disks, this is unlikely. DOS won't tell you why a file won't copy, (Abort Retry Ignore isn't very informative). The Copy utility of PC TOOLS Deluxe or most similar programs does a much better job of telling you what went wrong.

If some files on a distribution disk won't copy on a 360K disk drive, you might try copying them on a 1.2M drive. Since the 1.2M drive uses a narrower track, it reads from near the center of each track on the 360K disk, which may still be intelligible. If that works, then format a brand new floppy disk on your 360K drive (this makes sure there are no vestiges of old data on your disk to confuse things) and copy the files to it using the 1.2M drive. It's easier to copy them first to the hard disk (into an empty subdirectory) unless you've got both a 1.2M and a 360K floppy on the same computer. This will sometimes allow you to make a workable backup from a marginal distribution disk. Note that you should avoid writing to or formatting old 360K disks on a 1.2M drive since the 1.2M drive can't erase old data from the edges of the wider 360K track. This combines old and new data on the same track making it unreadable to a 360K drive.

Read the Manual

Read everything the manual says about installation before you use the install program. Central Point Software did a surprisingly good job of explaining everything their PCSETUP is supposed to do. Unfortunately, they put pieces of the explanation in each of the three manuals that come with the upgrade. Even so, it's one of the best documented install programs I've seen.

Check README Files

Then display or print the contents of any README files on the new software disks. These files usually contain information that didn't make the printed manual and/or corrections to the manual. Some software packages include a disk program to print or scan the README file automatically. If your instructions don't tell you to use some other method, you can print most program's README files by placing the appropriate disk in drive A: and entering TYPE A:\README.TXT >PRN. (Note that the file name README.TXT varies from program to program.) The README file on the distribution disk gave the last piece of information needed (how to uncompress a file) so that installation could have been done without the PCSETUP program, although it would have been tedious and time consuming.

Delete the Old Version Files

If you're doing an upgrade, you should delete most or all of the old version's files unless you're told not to. Failure to do this with the PC Tools Deluxe upgrade will cost you some extra work figuring out which files were unneeded remnants of the old version and getting rid of them. This is more of a nuisance than a peril, but it could be a problem if the new program tried to use an old file such as an overlay or temporary file. Don't forget any relevant files you may have moved or copied to unlikely places on your disk.

If your program uses special configuration files, you may want to leave them intact. If the install program is unusually good, it will check for these files and either use them or translate their configuration information into the upgrade's format. PCSETUP has this capability.

Run the INSTALL Program

Having done all this, PCSETUP seemed to be one of the most user friendly install programs I've seen. The menus were well done and the program seemed to work well. When it finished, like most install programs, you're told to reboot the computer. **DON'T REBOOT THE COMPUTER.** If the install program has improperly altered either the CONFIG.SYS or AUTOEXEC.BAT file, some weird things can happen when you reboot, including not being able to reboot from the hard disk at all.

Some install programs will automatically reboot the computer. If the worst happens and it won't properly reboot from the hard disk, don't despair. Most Zenith Data Systems computers allow you to enter the ROM monitor by simultaneously pressing the Control, Alt, and Insert keys. Then place a bootable floppy disk in drive A: and enter the monitor ROM command BF0 to Boot from Floppy drive Zero (A:).

Check AUTOEXEC.BAT and CONFIG.SYS Before You Reboot

Instead of rebooting, use the TYPE command to check the contents of your CONFIG.SYS and AUTOEXEC.BAT files. Normally, I'd just use PC Tools to check these files, but since PC Tools Deluxe was just upgraded, it probably wouldn't work and might lock up the computer. That brings us to the next caution: Avoid using the program you're installing or upgrading before you've rebooted, since it may not work properly. Especially if it's a memory resident program.

I was unpleasantly surprised to find that PCSETUP had added a new and meaningless "= F" command to my CONFIG.SYS file when it changed FILES = 20 to FILES = 23 and that the loading order of my TSRs had been arbitrarily altered in the AUTOEXEC.BAT file. This performance was better than the average install program, but I didn't want to find out what DOS would do with "= F" in the CONFIG.SYS file, so corrections were in order. Ordinarily, I'd have used PC Tools, but in this case, Word Perfect was a better choice (most word processors will handle ASCII text files).

You could also reboot with the copies of your old CONFIG and AUTOEXEC files that you saved earlier. First, use the DOS Rename command REN to rename the questionable or corrupted files to CONFIG.NEW or something you can remember easily. Then rename the old file copies to AUTOEXEC.BAT and CONFIG.SYS.

If you use TSRs, you should check AUTOEXEC.BAT carefully and make sure their loading sequence still seems acceptable.

PCSETUP loads MIRROR early in AUTOEXEC.BAT. (It moved MIRROR to just after my clock updating program, which seems reasonable.) It also moves PC-CACHE, PCSHELL, and DESKTOP to the last three TSR locations in AUTOEXEC .BAT. This may or may not be best for you, depending on which TSRs you use.

Once you're satisfied with the critical CONFIG.SYS and AUTOEXEC.BAT files, the computer can be safely rebooted.

Check the Installed Files

After rebooting, the next step is to check to see what the install program put on the disk and make sure it's in the correct subdirectory. Then check for extraneous files (such as drivers for printers you don't have) that can safely be deleted. Note that if you couldn't delete the old files before installing an upgrade, you may have both old and new version files intermingled. In this event, the documentation should provide the names of the files that are now obsolete and should be deleted. If not, you'll probably find cleaning out the old files to be a time consuming chore.

To clean up old version files, compare the hard disk files against the distribution disk and delete anything with an old date that isn't found on the distribution disk. (Files with a current date may have been created by the install program and shouldn't be deleted.) Note that old files aren't always garbage. For some unknown reason, the PC Tools Deluxe program MEMCHK was omitted from the version 5.5 upgrade. I salvaged it as a useful utility.

Now compare the hard disk files with your manual, which should list the purpose of each file and delete the files you can be sure you don't need. (You can always copy them back from the distribution disk if you find you've made a mistake.)

Move Boot Files if Desired

If you want your computer to boot as fast as possible, you may want to move (or copy) some of the newly installed files to the root directory. (The boot process runs slightly faster if files used by AUTOEXEC.BAT are all in the root directory and in contiguous locations on the hard disk whenever possible.) I found that while PCSETUP had upgraded appropriate files in the C:\PCTOOLS subdirectory, it failed to find the old PC-CACHE and MIRROR which I had left in my root directory to see if PCSETUP would find them.

Look for Side Effects of the Upgrade

If you're feeling adventurous or in a hurry, running the new program is the fastest way to find major side effects. I suggest reading the manual first.

Sometimes program features can look like bugs. The Desktop portion of PC Tools Deluxe, for example, has three poorly documented hotkeys ^O, ^Insert, and ^Delete (where ^ represents holding the Control key down while pressing the named key) in addition to the ^Space hotkey used to call up Desktop. I found out about them the hard way when I accidentally pressed ^Delete. The cursor changed to inverse video, moved to screen center, and didn't respond to alphanumeric keys. I thought I'd locked up the computer and rebooted. This was a chance to find out how good Central Point Software's technical support group was. One phone call to their automated answering system and some patience got me to a real live person who actually knew something about the program, and even provided an undocumented way to disable the hotkeyed "features" since I use neither the autodialer nor the cut and paste functions. (The Utilities menu allows you to change hotkeys. selecting ^\ sets Control nothing as a hotkey and effectively disables the hotkey.)

Some side effects are subtle but documented in the manual. For example, PCSETUP had renamed the DOS FORMAT command to FORMAT! and substituted it's own FORMAT.BAT without asking permission. Central Point Software may like this arrangement, but I don't. I prefer my floppy disks completely erased by format, and ZDS computers are reasonably well protected against entry errors. My hard drives are format protected and Zenith Data Systems' FORMAT command (version 3.28) is easy enough to get out of if you enter FORMAT A: by mistake. Just press control-Break (^Break) when Format asks you to "Place a new disk in drive A: and press RETURN when ready".

PCSETUP was supposed to search for Word Perfect and add it to PCSHELL's Applications menu. It failed to do this with Word Perfect 4.2 located in a subdirectory of partition D:. Apparently, it just searches drive C: for these files.

After correcting these items, I found no problems with the upgrade itself.

Replace TSRs in AUTOEXEC.BAT

If you placed REM statements in front of some of your TSRs in AUTOEXEC.BAT, now is the time to test for dueling TSRs. Edit out the REM prefix from one TSR at a time and reboot the computer. Then run through your programs and exercise your TSRs to check for incompatibilities. If the computer locks up due to a problem, remember to boot from a floppy as described earlier.

I use three TSRs besides PC Tools; SS.COM (H.U.G. screen saver), DOSEDIT (a DOS command line editor), and SMARTKEY (an excellent keyboard macro and redefinition program). I found that some of my SMARTKEY macros inhibited some PCSHELL commands. This problem was easily solved by loading SMARTKEY after PCSHELL. This causes SMARTKEY to be ignored while PCSHELL is active. Otherwise, everything was nicely compatible.

After correcting these items, I found the upgrade well worth while. The shell has been improved considerably with a command bar and single letter selection of frequently used commands.

This restores the ease of use that was a hallmark of earlier versions of PC Tools and PC Tools Deluxe version 4.24, but lost when bells and whistles were added in PC Tools Deluxe version 5.0. I don't use most of the "enhancements," but as a file recovery and utilities package, PC Tools Deluxe is the best I've found. I have now installed PC Tools Deluxe on three Zenith Data Systems computers; a '151, a '158, and a '386/25. It runs well on all three and I recommend it as the software of choice for most users.

Products Discussed

PC Tools Deluxe Version 5.5
List Price $129.95
Update $20.00
Central Point Software
15220 NW Greenbrier Parkway #200
Beaverton, OR 97006
(503) 690-8090
(800) 888-8199 (Update orders only)

SMARTKEY
(I use version 5.11. Current version is 5.32)
List Price $89.95

DOSEDIT (DOS command line editor TSR using 1968 bytes RAM)

DOSEDIT.COM 2048 bytes 08/17/85 7:00 PM (my version)

Public domain program distributed with SMARTKEY also available on various bulletin boards.

(FBN Software sold Smartkey rights to:)
Command Software Systems, Inc.
28990 PC Highway, Suite 208
Malibu, CA. 90265
(800) 423-9147 (Orders only)
(213) 457-1789 (California)

SS.COM (Screensaver TSR vers 1.0)
Blanks screen after inactive time using 17K RAM
Heath Users' Group
Part #885-6009
(w/other utilities) $20.00
Heath Users' Group
P.O. Box 217
Benton Harbor, Mi 49022-0217
(Mail orders add 10% postage & handling)
(616) 982-3463 (Software orders)

CLASSIFIED ADS

WANTED: BUY OR RENT HERO JR. Basic ROM Cartridge. Jay (216) 456-6574.

FOR SALE: TWO SHUGART 1/2 HT. DISK DRIVES, $40.00 each. Hayes Smart Modem 1200, $80.00. CDR Disk Controller, $80.00. Panasonic parallel to series card, $40.00. CPM and HDOS original programs, $3.00 each. Postage extra. Call (416) 896-8329.

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) 429-3583.

'386 COMPUTER ZBV-3340 includes 3.5" 1.4 MB, 5.25" 1.2 MB drives, VGA card, 16 MB RAM, 80367 co-processor. 120 MB hard drive capacity, 2 parallel/2 serial ports, ZCM-1490 Monitor, mouse, internal modem. Almost $22,000 retail for only $10,900. 90 day guarantee. (616) 429-3292 weekdays after 5:30 pm E.S.T. Weekends anytime.

Taking the Mystery Out of Hard Drives

D. R. Cool
7421 Troy Manor Road
Huber Heights, OH 45424

FAT, boot record, partition table. These are all terms that occur in many computer articles, especially articles on hard drives. Probably many users know that the FAT (File Allocation Table) has something to do with how DOS allocates files. But, what is its actual structure, where is it located on the hard drive and how does it actually work? Other questions pop up now and then. "How is a hard drive actually structured? How does DOS keep track of a directory as you add and delete entries? What happens when you format a hard drive?"

Until a few months ago, I hadn't the foggiest notion of the answers to these questions. Finally, I decided to delve into the intricacies of my system and solve these mysteries once and for all. Here are the results of my probing.

First, let me state that what I discovered applies to a Z-248 computer with a 20 MByte hard drive and using version 3.1 of Zenith Data Systems' version of MS-DOS. I assume that much of this information will apply to other IBM AT-compatible computers with a 20 MByte drive, but I wouldn't guarantee it.

By decoding the boot record of my C: drive (more on this later), I determined that my drive has four sides. Therefore, I assume the Winchester unit contains two physical platters since each platter has two sides. Each side consists of 615 cylinders or tracks, of which only the first 613 are used for data storage. The first track is called track 0 and each track consists of 17 physical sectors, starting with sector 1. If you multiply this out using 512 bytes per sector, then total capacity equals 4*613*17*512 or 21,342,308 bytes. If you do a CHKDSK on a 20 MByte hard drive with a single partition, you should get a total disk space of 21,274,624 bytes. Which begs the question: "What happened to the missing 67,584 bytes?" The answer is what CHKDSK reports is the amount of space available for files (including the operating system). The rest is reserved for the partition table, the boot record, two copies of the FAT and the root directory.

Before I go further, let's make a distinction between physical sector and logical sector. Physical sector refers to the actual sectors that any track is divided into, starting with sector 1 and ending with sector 17; whereas logical sector is in relation to logical sector 0, which is the first sector that DOS has access to. Generally, logical sector 0 turns out to be side 1, track 0, physical sector 1. Logical sector is what a utility like PC-Tools* calls relative sector if you do a View/Edit by disk.

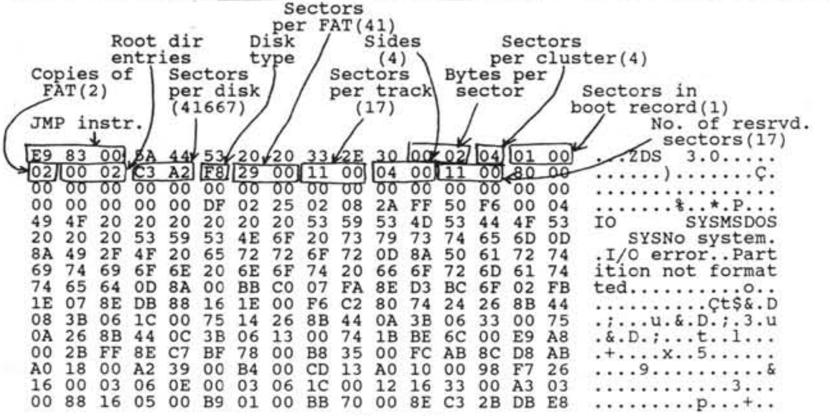
Since I had verified by sector dump using two different disk utilities that the data stored on relative sector 0 was actually located on side 1, track 0, physical sector 1, it occurred to me that an entire seventeen sectors on side 0, track 0 was being ignored by the one utility. A little digging turned up the information that these first seventeen sectors are reserved and are not accessible to utilities like PC-Tools. Paul Mace, in his book The Paul Mace Guide to Data Recovery**, states that the Master Boot Record is stored here. I may or may not dispute this, depending on what he means by the Master Boot Record. Being the curious person that I am, I modified a utility that, among other things, reads the partition table which is stored in the first of the reserved seventeen sectors. This program makes use of BIOS interrupt $13, which will read any valid side, track and sector combination of a hard drive. A very handy tool to have if you want to do some exploring. Figure 1 is a dump of the first physical sector of side 0, track 0. The first 30 bytes appear to be a machine language program. (I seem to remember reading somewhere that a boot program is stored on the hard drive, so maybe this is it. Perhaps someone can shed some light on this.) Also, as is evident from the ASCII side of this dump, three "error" messages are stored here.

According to Michael Hyman's book Memory Resident Utilities***, this first sector also contains the partition table. This table starts at the 446th byte and is 64 bytes long. It contains four 16-byte blocks, one for each of four possible partitions. Each 16-byte block can be decoded as follows:

Byte **Meaning**
0 This is the boot indicator: 80h for the active partition, 0 for all others.
1 Side where partition begins.
2 The lowest six bits of this byte

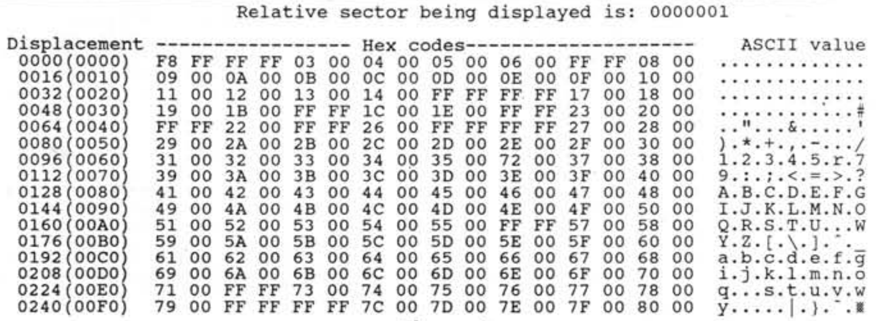
```
2B C0 FA 8E D0 BC 00 7C FB 8B F4 52 8E C0 0E 1F   +......|...R....
BF 00 06 B9 00 01 FC F3 A5 EA 1E 06 00 00 8E D8   ................
BE BE 07 B9 04 00 90 F6 04 80 74 06 80 7C 04 00   .........Çt.Ç|..
75 0A 83 C6 10 E2 F0 BE 7F 06 EB 36 5A 8A 74 01   u.......■..6Z.t.
8B 4C 02 8B EE BF 05 00 BB 00 7C B8 01 02 57 CD   .L........|...W.
13 5F 73 0C 2B C0 CD 13 4F 75 ED BE 96 06 EB 12   ._s.+...Ou......
BE BE 06 81 3E FE 7D 55 AA 75 07 8B F5 EA 00 7C   ....>.)U.u.....|
00 00 AC 0A C0 74 06 B4 0E CD 10 EB F5 EB FE 4E   .....t.........N
6F 20 62 6F 6F 74 61 62 6C 65 20 70 61 72 74 69   o bootable parti
74 69 6F 6E 73 00 55 6E 61 62 6C 65 20 74 6F 20   tions.Unable to
72 65 61 64 20 62 6F 6F 74 20 63 6F 64 65 20 66   read boot code f
72 6F 6D 20 70 61 72 74 69 74 69 6F 6E 00 4E 6F   rom partition.No
74 20 61 20 62 6F 6F 74 61 62 6C 65 20 70 61 72   t a bootable par
74 69 74 69 6F 6E 00 00 00 00 00 00 00 00 00 00   tition..........
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 01   ..............Ç.  Partition
01 00 04 03 91 64 11 00 00 00 C3 A2 00 00 00 00   .....d..........   table
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA   ..............U.
```

Figure 1
Dump of Side 0, Track 0, Sector 1



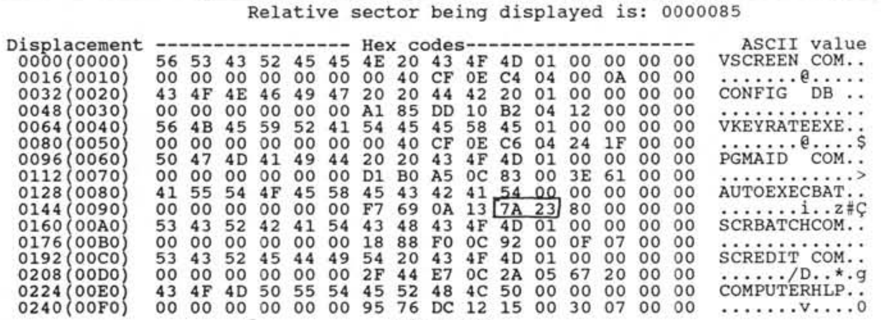
```
E9 83 00 5A 44 53 20 20 33 2E 30 00 02 04 01 00   ...ZDS  3.0.....
02 00 02 C3 A2 F8 29 00 11 00 04 00 11 00 80 00   ......).......Ç.
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 DF 02 25 02 08 2A FF 50 F6 00 04   .......%..*.P...
49 4F 20 20 20 20 20 20 53 59 53 4D 53 44 4F 53   IO      SYSMSDOS
20 20 20 53 59 53 4E 6F 20 73 79 73 74 65 6D 0D      SYSNo system.
8A 49 2F 4F 20 65 72 72 6F 72 0D 8A 50 61 72 74   .I/O error..Part
69 74 69 6F 6E 20 6E 6F 74 20 66 6F 72 6D 61 74   ition not format
74 65 64 0D 8A 00 BB C0 07 FA 8E D3 BC 6F 02 FB   ted..........o..
1E 07 8E DB 88 16 1E 00 F6 C2 80 74 24 26 8B 44   ..........Çt$&.D
08 3B 06 1C 00 75 14 26 8B 44 0A 3B 06 33 00 75   .;...u.&.D.;.3.u
0A 26 8B 44 0C 3B 06 13 00 74 1B BE 6C 00 E9 A8   .&.D.;...t..l...
00 2B FF 8E C7 BF 78 00 B8 35 00 FC AB 8C D8 AB   .+....x..5......
A0 18 00 A2 39 00 B4 00 CD 13 A0 10 00 98 F7 26   ....9..........&
16 00 03 06 0E 00 03 06 1C 00 12 16 33 00 A3 03   ............3...
00 88 16 05 00 B9 01 00 BB 70 00 8E C3 2B DB E8   .........p...+..
```

Figure 2
Boot Record Parameters



```
                Relative sector being displayed is: 0000001

Displacement ----------------- Hex codes-------------------    ASCII value
 0000(0000)   F8 FF FF FF 03 00 04 00 05 00 06 00 FF FF 08 00   .............
 0016(0010)   09 00 0A 00 0B 00 0C 00 0D 00 0E 00 0F 00 10 00   .............
 0032(0020)   11 00 12 00 13 00 14 00 FF FF FF FF 17 00 18 00   .............
 0048(0030)   19 00 1B 00 FF FF 1C 00 1E 00 FF FF 23 00 20 00   .............#
 0064(0040)   FF FF 22 00 FF FF 26 00 FF FF FF FF 27 00 28 00   .."...&.....'
 0080(0050)   29 00 2A 00 2B 00 2C 00 2D 00 2E 00 2F 00 30 00   ).*.+.,.-.../
 0096(0060)   31 00 32 00 33 00 34 00 35 00 72 00 37 00 38 00   1.2.3.4.5.r.7
 0112(0070)   39 00 3A 00 3B 00 3C 00 3D 00 3E 00 3F 00 40 00   9.:.;.<.=.>.?
 0128(0080)   41 00 42 00 43 00 44 00 45 00 46 00 47 00 48 00   A.B.C.D.E.F.G
 0144(0090)   49 00 4A 00 4B 00 4C 00 4D 00 4E 00 4F 00 50 00   I.J.K.L.M.N.O
 0160(00A0)   51 00 52 00 53 00 54 00 55 00 FF FF 57 00 58 00   Q.R.S.T.U...W
 0176(00B0)   59 00 5A 00 5B 00 5C 00 5D 00 5E 00 5F 00 60 00   Y.Z.[.\.].^._
 0192(00C0)   61 00 62 00 63 00 64 00 65 00 66 00 67 00 68 00   a.b.c.d.e.f.g
 0208(00D0)   69 00 6A 00 6B 00 6C 00 6D 00 6E 00 6F 00 70 00   i.j.k.l.m.n.o
 0224(00E0)   71 00 FF FF 73 00 74 00 75 00 76 00 77 00 78 00   q...s.t.u.v.w
 0240(00F0)   79 00 FF FF FF FF 7C 00 7D 00 7E 00 7F 00 80 00   y.....|.}.~.▓
```

Figure 3
Start of FAT #1. (Side 1, Track 0, Sector 2)



```
                Relative sector being displayed is: 0000085

Displacement ----------------- Hex codes-------------------    ASCII value
 0000(0000)   56 53 43 52 45 45 4E 20 43 4F 4D 01 00 00 00 00   VSCREEN COM..
 0016(0010)   00 00 00 00 00 00 00 40 CF 0E C4 04 00 0A 00 00   .......@.....
 0032(0020)   43 4F 4E 46 49 47 20 20 44 42 20 01 00 00 00 00   CONFIG  DB ..
 0048(0030)   00 00 00 00 00 00 A1 85 DD 10 B2 04 12 00 00 00   .............
 0064(0040)   56 4B 45 59 52 41 54 45 45 58 45 01 00 00 00 00   VKEYRATEEXE..
 0080(0050)   00 00 00 00 00 00 00 40 CF 0E C6 04 24 1F 00 00   .......@....$
 0096(0060)   50 47 4D 41 49 44 20 20 43 4F 4D 01 00 00 00 00   PGMAID  COM..
 0112(0070)   00 00 00 00 00 00 D1 B0 A5 0C 83 00 3E 61 00 00   ............>
 0128(0080)   41 55 54 4F 45 58 45 43 42 41 54 00 00 00 00 00   AUTOEXECBAT..
 0144(0090)   00 00 00 00 00 00 F7 69 0A 13 7A 23 80 00 00 00   .......i..z#Ç
 0160(00A0)   53 43 52 42 41 54 43 48 43 4F 4D 01 00 00 00 00   SCRBATCHCOM..
 0176(00B0)   00 00 00 00 00 00 18 88 F0 0C 92 00 0F 07 00 00   .............
 0192(00C0)   53 43 52 45 44 49 54 20 43 4F 4D 01 00 00 00 00   SCREDIT COM..
 0208(00D0)   00 00 00 00 00 00 2F 44 E7 0C 2A 05 67 20 00 00   ....../D..*.g
 0224(00E0)   43 4F 4D 50 55 54 45 52 48 4C 50 00 00 00 00 00   COMPUTERHLP..
 0240(00F0)   00 00 00 00 00 00 95 76 DC 12 15 00 30 07 00 00   .......v....0
```

Figure 4
Directory Listing of AUTOEXEC.BAT

are the sector where the partition begins. The high two bits are the high two bits of the track where the partition begins.

3 Lowest eight bits of the track where the partition begins.

4 System indicator:
4 = DOS partition w/16-bit FAT
1 = DOS partition w/12-bit FAT
0 = other

5 Side where partition ends.

6 Lowest 6 bits: sector where partition ends. High 2 bits: high two bits of ending track number.

7 Low 8 bits of ending track number.

8-11 Double word: number of sectors preceding the partition.

12-15 Double word: number of sectors in partition.

Decoding this for my C: drive, I have a 16-bit FAT DOS partition, starting at side 1, sector 1, track 0; ending at side 3, sector 17, track 612 and containing 41,667 sectors. Obviously, I have only 1 partition for physical drive 0.

The next sector of interest is physical sector 1, located on track 0 of side 1. This sector can be displayed using the Disk View/Edit Service of PC-Tools, which labels it as relative sector 0000000. (See Figure 2.) Hyman's book says the boot record parameters are stored on this sector. The first three bytes make up a machine language command to jump to the boot program. The remaining parameters are decoded as follows:

Byte	**Contents**
3-10	Name of the DOS which formatted the disk, in my case, "ZDOS 3.0".
11-12	Bytes per sector (512)
13	Sectors per cluster (4)
14-15	Sectors in boot record (1)
16	Copies of the FAT (2)
17-18	Maximum root directory entries (512)
19-20	Sectors per disk (41,667)
21	Disk type: F8h = hard disk
22-23	Sectors per FAT (41)
24-25	Sectors per track (17)
26-27	Sides per disk (4)
28-29	Number of reserved sectors (17)

As to the contents of the remainder of this sector, I have no information as to their significance. I do know that what gets written to this sector depends on what utility you use to format the disk. The next sector (relative sector 1) is the start of the first copy of the FAT (see Figure 3). This is a 16-bit FAT, which means that two bytes are used to define the status of every cluster. Since the FAT occupies 41 sectors, it can have entries for up to 10496 clusters or 41984 sectors. The first byte in the FAT is F8h, which is the disk type code. This is followed by three FFh bytes. According to Hyman, cluster numbers begin with cluster number 2; that is to say, there is no cluster 0 or cluster 1. The meaning of the values of each FAT entry is as follows:

Value(hex)	**Meaning**
0000	Cluster is free
0001	Invalid
0002-ffef	Cluster in use. Entry points to next cluster in chain
fff0-fff6	Cluster is reserved
fff7	Cluster is bad
fff8-ffff	Last cluster in a file chain (End of file marker)

To understand how the FAT works, we have to first have a basic idea of how space is allocated to files. The smallest possible space that can be allocated to a file is one cluster, even if the file is only one byte long. Cluster size can actually vary between one and eight sectors. DOS chooses the size as a trade-off between the number of clusters DOS needs to keep track of and the amount of space that is wasted by partially filled clusters. A four-sector cluster may or may not be the best compromise, depending on the average size of files you have on your system.

Starting with a file name, how does DOS know where to look for the beginning of the file? It starts with the actual directory entry. Figure 4 is a sector dump of a portion of a directory. Each directory entry consists of 32 bytes and begins with the actual name of the file. Look at the entry for AUTOEXEC.BAT. The 26th and 27th bytes (boxed in) are the first cluster allocated to the file. This is the starting cluster of the file chain, even if the file occupies only one cluster. The hex value 237A equals 9082 decimal. To translate a cluster number to relative sector, use the following equation:

```
Rel. sector = (cluster number - 2) * (s
ectors per cluster) + (number of first
sector in the data area)
```

For my C: drive, the first data sector is relative sector 115; therefore, relative sector = 9080 * 4 + 115 = 36435. If I now do a data dump of relative sector 36435, I can see the actual AUTOEXEC.BAT file as stored on disk. (See Figure 5.)

```
                 Relative sector being displayed is: 0036435

Displacement ----------------- Hex codes--------------------   ASCII value
 0000(0000)   45 43 48 4F 20 4F 46 46 0D 0A 43 4C 53 0D 0A 50  ECHO OFF..CLS
 0016(0010)   41 54 48 3D 63 3A 5C 3B 63 3A 5C 44 4F 53 3B 63  ATH=c:\;c:\DO
 0032(0020)   3A 5C 46 4F 58 3B 63 3A 5C 51 42 34 35 3B 63 3A  :\FOX;c:\QB45
 0048(0030)   5C 54 50 41 53 3B 63 3A 5C 45 4E 41 42 4C 45 3B  \TPAS;c:\ENAB
 0064(0040)   63 3A 5C 4D 41 43 45 0D 0A 53 45 54 20 43 4F 4D  c:\MACE..SET
 0080(0050)   53 50 45 43 3D 0D 0A 53 45 54 20 4D 41 43 45 20  SPEC=..SET MA
 0096(0060)   3D 20 63 3A 5C 6D 61 63 65 0D 0A 50 52 4F 4D 50  = c:\mace..PR
 0112(0070)   54 20 24 70 24 67 0D 0A 43 48 4F 49 43 45 0D 0A  T $p$g..CHOIC
 0128(0080)   FE 00 F0 C7 06 12 00 6E 00 2E C6 06 7E 01 00 BE  ........n....
 0144(0090)   20 00 26 8B 04 2E A3 7F 01 26 8B 44 02 2E A3 81   .&....■.&.D.
 0160(00A0)   01 FA 26 C7 04 AE 01 26 8C 4C 02 FB EB 22 1E 50  ..&.....&.L...
 0176(00B0)   33 C0 8E D8 C7 06 20 00 C5 01 2E A3 7C 01 58 1F  3..... .....|
 0192(00C0)   2E FF 2E 7F 01 2E C6 06 7E 01 FF 2E FF 2E 7F 01  ...■.... ....
 0208(00D0)   E8 32 00 2E FF 06 7C 01 2E 80 3E 7E 01 FF 75 F0  .2....|..Ç>~.
 0224(00E0)   2E A1 81 01 FA 26 89 44 02 2E A1 7F 01 26 89 04  .....&.D...■.
 0240(00F0)   FB 2E A1 7C 01 03 C0 A3 12 00 C3 8B C8 E3 05 E8  ...|.........
```

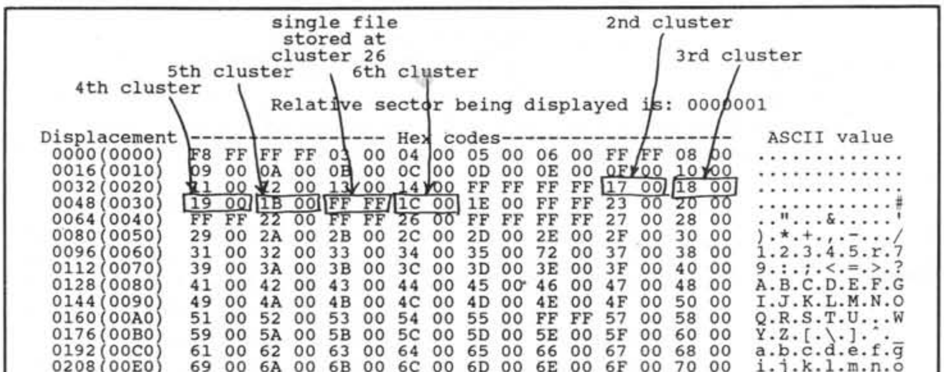
Figure 5
AUTOEXEC.BAT File

```
                 Relative sector being displayed is: 0000036

Displacement ----------------- Hex codes-------------          ASCII value
 0000(0000)   01 23 02 23 03 23 04 23 05 23 FF FF FF FF 08 23  .#.#.#.#.#...
 0016(0010)   09 23 0A 23 0B 23 0C 23 0D 23 0E 23 0F 23 10 23  .#.#.#.#.#.#.
 0032(0020)   11 23 12 23 13 23 14 23 15 23 FF FF 17 23 18 23  .#.#.#.#.#...
 0048(0030)   FF FF 1A 23 1B 23 1C 23 1D 23 1E 23 FF FF 20 23  ...#.#.#.#.#.
 0064(0040)   21 23 22 23 23 23 24 23 FF FF FF FF 28 23 2F 23  !#"###$#....(
 0080(0050)   29 23 2A 23 2B 23 2C 23 2D 23 FF FF FF FF 30 23  )#*#+#,#-#...
 0096(0060)   31 23 53 23 33 23 34 23 35 23 36 23 37 23 38 23  1#S#3#4#5#6#7
 0112(0070)   39 23 3A 23 3B 23 3C 23 3D 23 3E 23 3F 23 40 23  9#:#;#<#=#>#?
 0128(0080)   41 23 42 23 43 23 44 23 45 23 46 23 47 23 48 23  A#B#C#D#E#F#G
 0144(0090)   49 23 4A 23 4B 23 4C 23 4D 23 4E 23 4F 23 50 23  I#J#K#L#M#N#O
 0160(00A0)   51 23 52 23 FF FF 54 23 77 23 5D 23 57 23 58 23  Q#R#..T#w#]#W
 0176(00B0)   59 23 5A 23 5B 23 5C 23 FF FF 5E 23 5F 23 60 23  Y#Z#[#\#.. #_
 0192(00C0)   61 23 62 23 63 23 64 23 65 23 66 23 67 23 68 23  a#b#c#d#e#f#g
 0208(00D0)   69 23 6A 23 6B 23 FF FF 00 00 00 00 00 00 00 00  i#j#k#.......
 0224(00E0)   00 00 00 00 00 00 00 00 00 00 00 00 00 00 78 23  .............
 0240(00F0)   79 23 FF FF FF FF FF FF 00 00 00 00 00 00 00 00  y#...........
```

Figure 6

```
                 Relative sector being displayed is: 0000085

Displacement ----------------- Hex codes--------------------   ASCII value
 0256(0100)   43 48 4F 49 43 45 20 20 53 43 52 00 00 00 00 00  CHOICE  SCR..
 0272(0110)   00 00 00 00 00 00 2B 63 0A 13 8C 01 20 0F 00 00  ......+c....
 0288(0120)   24 53 43 4E 54 4C 20 20 53 59 53 02 00 00 00 00  $SCNTL  SYS..
 0304(0130)   00 00 00 00 00 00 A5 49 4E 12 FE 00 08 02 00 00  .......IN....
 0320(0140)   41 43 54 49 56 49 54 59 53 59 53 20 00 00 00 00  ACTIVITYSYS .
 0336(0150)   00 00 00 00 00 00 2E 77 17 13 16 00 80 FA 00 00  .......w....Ç
 0352(0160)   41 4E 53 49 20 20 20 20 53 59 53 01 00 00 00 00  ANSI     SYS..
 0368(0170)   00 00 00 00 00 00 8A 41 D9 0C 94 00 00 08 00 00  .......A.....
 0384(0180)   42 55 52 4E 44 45 56 20 53 59 53 01 00 00 00 00  BURNDEV SYS..
 0400(0190)   00 00 00 00 00 00 8B 8D 1B 0D A4 02 30 08 00 00  ............0
 0416(01A0)   43 4F 4E 46 49 47 20 20 53 59 53 00 00 00 00 00  CONFIG  SYS..
 0432(01B0)   00 00 00 00 00 00 5C 68 0A 13 43 17 94 00 00 00  ......\h..C..
 0448(01C0)   50 41 53 53 57 4F 52 44 53 59 53 01 00 00 00 00  PASSWORDSYS..
 0464(01D0)   00 00 00 00 00 00 C8 64 48 0F 95 00 82 02 00 00  .......dH....
 0480(01E0)   E5 48 4F 49 43 45 20 20 42 41 4B 00 00 00 00 00  .HOICE  BAK..
 0496(01F0)   00 00 00 00 00 00 0C 66 0A 13 40 03 30 02 00 00  .......f..@.0
```

Figure 7
Directory Listing of ACTIVITY.SYS



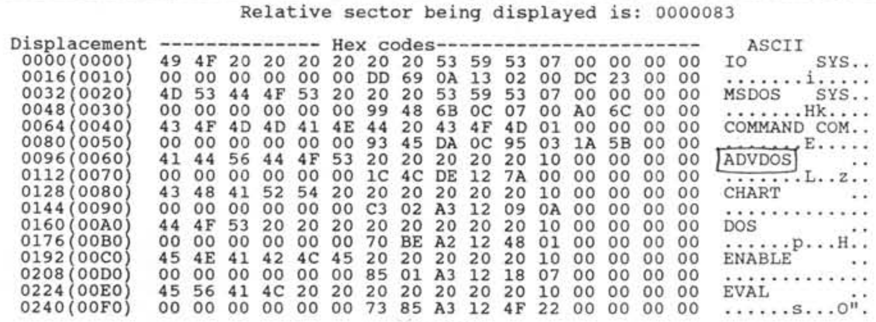
```
                 Relative sector being displayed is: 0000001

Displacement ----------------- Hex codes--------------------   ASCII value
 0000(0000)   F8 FF FF FF 03 00 04 00 05 00 06 00 FF FF 08 00  .............
 0016(0010)   09 00 0A 00 0B 00 0C 00 0D 00 0E 00 0F 00 10 00  .............
 0032(0020)   11 00 12 00 13 00 14 00 FF FF FF FF 17 00 18 00  .............
 0048(0030)   19 00 1B 00 FF FF 1C 00 1E 00 FF FF 23 00 20 00  .............#
 0064(0040)   FF FF 22 00 FF FF 26 00 FF FF FF FF 27 00 28 00  .."...&.....'
 0080(0050)   29 00 2A 00 2B 00 2C 00 2D 00 2E 00 2F 00 30 00  ).*.+.,.-.../
 0096(0060)   31 00 32 00 33 00 34 00 35 00 72 00 37 00 38 00  1.2.3.4.5.r.7
 0112(0070)   39 00 3A 00 3B 00 3C 00 3D 00 3E 00 3F 00 40 00  9.:.;.<.=.>.?
 0128(0080)   41 00 42 00 43 00 44 00 45 00 46 00 47 00 48 00  A.B.C.D.E.F.G
 0144(0090)   49 00 4A 00 4B 00 4C 00 4D 00 4E 00 4F 00 50 00  I.J.K.L.M.N.O
 0160(00A0)   51 00 52 00 53 00 54 00 55 00 FF FF 57 00 58 00  Q.R.S.T.U...W
 0176(00B0)   59 00 5A 00 5B 00 5C 00 5D 00 5E 00 5F 00 60 00  Y.Z.[.\.].^._
 0192(00C0)   61 00 62 00 63 00 64 00 65 00 66 00 67 00 68 00  a.b.c.d.e.f.g
 0208(00E0)   69 00 6A 00 6B 00 6C 00 6D 00 6E 00 6F 00 70 00  i.j.k.l.m.n.o
 0224(00E0)   71 00 FF FF 73 00 74 00 75 00 76 00 77 00 78 00  q...s.t.u.v.w
 0240(00F0)   79 00 FF FF FF FF 7C 00 7D 00 7E 00 7F 00 80 00  y.....|.}.~.■
```

Figure 8
Cluster chain for ACTIVITY.SYS

You may be wondering, "If this is all that's needed for DOS to find a file, what does it need the FAT for?" To help answer that question, let's look at the FAT entry for the AUTOEXEC.BAT file. The same number 9082 also tells DOS where in the FAT table to find the first cluster entry for the AUTOEXEC.BAT file. This offset from the start of the table equals the cluster number times two. For cluster number 9082, the offset to find the next cluster number is 18164, a rather unwieldy number if you are dealing in sectors. So let's do a conversion: 18164/512 = 35 with a remainder of 244. Since the FAT starts at relative sector 1, we have to look at the 244th byte of relative sector 36. Figure 6 is a dump of relative sector 36 with bytes 244 and 245 boxed in. Note that the values are ffh. As noted before, this signifies an end of file marker, telling DOS that for AUTOEXEC.BAT it need not look further since the file has only one cluster allocated to it.

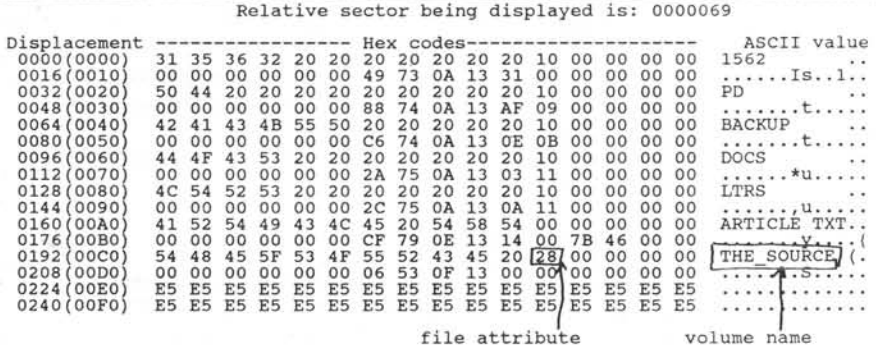
What about larger files? Files larger than a single cluster are the reason DOS needs a cluster chain to find all of the file. If files were always stored in consecutively numbered clusters, a cluster chain would not be necessary, since DOS would need only the starting cluster and the length of the file in bytes. However, not all files are stored in consecutive clusters. If you delete a file, the clusters that were allocated to that file are de-allocated. The next disk write may then use this de-allocated space, putting part of a file in that space and the remainder somewhere else. This occurs more and more often as the disk fills up. Thus files become fragmented and disk access starts to slow up. Thus, the reason for unfragmenting utilities. At any rate, DOS would have an impossible task of accessing a fragmented file without a cluster chain.

To see how this works, let's look at the cluster chain of an actual fragmented file. Figure 7 shows the directory entry for ACTIVITY.SYS, a file which I determined is definitely fragmented. The starting cluster for this file is 22 (0016h). This number multiplied by two also determines the offset from the start of the FAT where the next cluster number is located. Therefore, the next cluster number for ACTIVITY.SYS occurs at offset 44 from the start of the FAT. Figure 8 is a dump of the first sector of the FAT. At offset 44, the value of the next cluster for ACTIVITY.SYS is 0017h or 23 decimal. Remember that each cluster value serves two functions: (1) the value itself is part of the file cluster chain and (2) the value multiplied by two is the offset for the next cluster value. The second cluster then is cluster 23. To find the third cluster number, offset = 2 * 23 = 46. The third cluster has the value 0018h or 24 decimal. The fourth cluster number is therefore at offset 48. Its value is 0019h or 25 decimal. So far the file appears to be stored in consecutive clusters. But watch



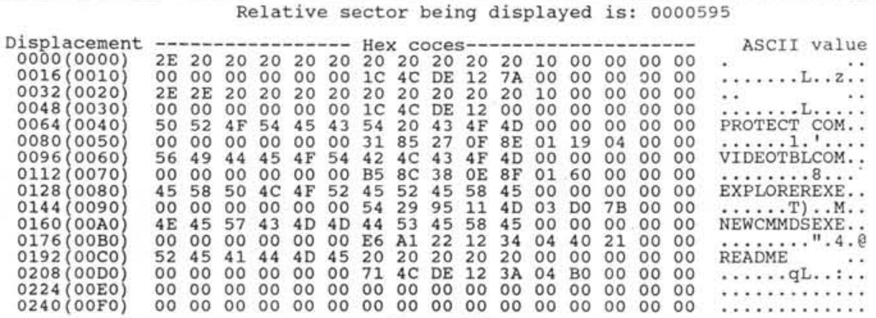
```
                    Relative sector being displayed is: 0000083

 Displacement  -------------- Hex codes-----------------------         ASCII
  0000(0000)   49 4F 20 20 20 20 20 20 53 59 53 07 00 00 00 00   IO      SYS..
  0016(0010)   00 00 00 00 00 00 DD 69 0A 13 02 00 DC 23 00 00   .......i.....
  0032(0020)   4D 53 44 4F 53 20 20 20 53 59 53 07 00 00 00 00   MSDOS   SYS..
  0048(0030)   00 00 00 00 00 00 99 48 6B 0C 07 00 A0 6C 00 00   .......Hk....
  0064(0040)   43 4F 4D 4D 41 4E 44 20 43 4F 4D 01 00 00 00 00   COMMAND COM..
  0080(0050)   00 00 00 00 00 00 93 45 DA 0C 95 03 1A 5B 00 00   .......E.....
  0096(0060)   41 44 56 44 4F 53 20 20 20 20 20 10 00 00 00 00   ADVDOS      ..
  0112(0070)   00 00 00 00 00 00 1C 4C DE 12 7A 00 00 00 00 00   .......L..z..
  0128(0080)   43 48 41 52 54 20 20 20 20 20 20 10 00 00 00 00   CHART       ..
  0144(0090)   00 00 00 00 00 00 C3 02 A3 12 09 0A 00 00 00 00   .............
  0160(00A0)   44 4F 53 20 20 20 20 20 20 20 20 10 00 00 00 00   DOS         ..
  0176(00B0)   00 00 00 00 00 00 70 BE A2 12 48 01 00 00 00 00   ......p...H..
  0192(00C0)   45 4E 41 42 4C 45 20 20 20 20 20 10 00 00 00 00   ENABLE      ..
  0208(00D0)   00 00 00 00 00 00 85 01 A3 12 18 07 00 00 00 00   .............
  0224(00E0)   45 56 41 4C 20 20 20 20 20 20 20 10 00 00 00 00   EVAL        ..
  0240(00F0)   00 00 00 00 00 00 73 85 A3 12 4F 22 00 00 00 00   ......s...O".
```

Figure 9
Root Directory Entry for Subdirectory ADVDOS



```
                   Relative sector being displayed is: 0000069

Displacement  ----------------- Hex codes-------------------         ASCII value
 0000(0000)   31 35 36 32 20 20 20 20 20 20 20 10 00 00 00 00   1562        ..
 0016(0010)   00 00 00 00 00 00 49 73 0A 13 31 00 00 00 00 00   ......Is..1..
 0032(0020)   50 44 20 20 20 20 20 20 20 20 20 10 00 00 00 00   PD          ..
 0048(0030)   00 00 00 00 00 00 88 74 0A 13 AF 09 00 00 00 00   .......t.....
 0064(0040)   42 41 43 4B 55 50 20 20 20 20 20 10 00 00 00 00   BACKUP      ..
 0080(0050)   00 00 00 00 00 00 C6 74 0A 13 0E 0B 00 00 00 00   .......t.....
 0096(0060)   44 4F 43 53 20 20 20 20 20 20 20 10 00 00 00 00   DOCS        ..
 0112(0070)   00 00 00 00 00 00 2A 75 0A 13 03 11 00 00 00 00   ......*u.....
 0128(0080)   4C 54 52 53 20 20 20 20 20 20 20 10 00 00 00 00   LTRS        ..
 0144(0090)   00 00 00 00 00 00 2C 75 0A 13 0A 11 00 00 00 00   ......,u.....
 0160(00A0)   41 52 54 49 43 4C 45 20 54 58 54 00 00 00 00 00   ARTICLE TXT..
 0176(00B0)   00 00 00 00 00 00 CF 79 0E 13 14 00 7B 46 00 00   ........y....(
 0192(00C0)   54 48 45 5F 53 4F 55 52 43 45 20 28 00 00 00 00   THE_SOURCE (.
 0208(00D0)   00 00 00 00 00 00 06 53 0F 13 00 00 00 00 00 00   .......S.....
 0224(00E0)   E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5   .............
 0240(00F0)   E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5   .............

                                         file attribute         volume name
```

Figure 10
Directory Entry for Volume Label



```
                   Relative sector being displayed is: 0000595

Displacement  ----------------- Hex coces-------------------         ASCII value
 0000(0000)   2E 20 20 20 20 20 20 20 20 20 20 10 00 00 00 00   .           ..
 0016(0010)   00 00 00 00 00 00 1C 4C DE 12 7A 00 00 00 00 00   .......L..z..
 0032(0020)   2E 2E 20 20 20 20 20 20 20 20 20 10 00 00 00 00   ..          ..
 0048(0030)   00 00 00 00 00 00 1C 4C DE 12 00 00 00 00 00 00   .......L.....
 0064(0040)   50 52 4F 54 45 43 54 20 43 4F 4D 00 00 00 00 00   PROTECT COM..
 0080(0050)   00 00 00 00 00 00 31 85 27 0F 8E 01 19 04 00 00   ......1.'....
 0096(0060)   56 49 44 45 4F 54 42 4C 43 4F 4D 00 00 00 00 00   VIDEOTBLCOM..
 0112(0070)   00 00 00 00 00 00 B5 8C 38 0E 8F 01 60 00 00 00   ........8...
 0128(0080)   45 58 50 4C 4F 52 45 52 45 58 45 00 00 00 00 00   EXPLOREREXE..
 0144(0090)   00 00 00 00 00 00 54 29 95 11 4D 03 D0 7B 00 00   ......T)..M..
 0160(00A0)   4E 45 57 43 4D 4D 44 53 45 58 45 00 00 00 00 00   NEWCMMDSEXE..
 0176(00B0)   00 00 00 00 00 00 E6 A1 22 12 34 04 40 21 00 00   ........".4.@
 0192(00C0)   52 45 41 44 4D 45 20 20 20 20 20 00 00 00 00 00   README      ..
 0208(00D0)   00 00 00 00 00 00 71 4C DE 12 3A 04 B0 00 00 00   ......qL..:..
 0224(00E0)   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   .............
 0240(00F0)   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   .............
```

Figure 11
Directory Entries for ADVOS Subdirectory

what happens. The offset to the fifth cluster is 50 and the value is 001bh or 27. Evidently, when this file was last written, DOS skipped cluster 26 and for good reason — another file, less than one cluster long, already occupies this space. The cluster chain resumes at cluster 27 which translates to an offset of 54. This points to cluster 28, etc.

Following the first copy of the FAT, which ends at relative sector 41, a second identical copy is stored in relative sectors 42 through 82. (The actual length of the FAT on a hard drive depends generally on the size of the partition. A smaller partition will have a smaller FAT and a larger partition a larger FAT.) In the event the first copy of the FAT becomes unreadable, the second copy can be read.

Following copy two of the FAT at relative sector 83, I found a recognizable directory. If you are looking at a bootable drive (drive C: normally), you will see directory entries, starting with the two hidden system files, which in my case are IO.SYS and MSDOS.SYS (See Figure 9). (In version 3.21 of DOS, these files are called IBMBIO.COM and IBMDOS.COM, respectively.)

Speaking of hidden files, whenever I did a CHKDSK, it would always report three hidden files instead of the expected two. Even when using special directory utilities, I could never find this third hidden file. I finally solved this mystery when I decided to assign a label to my D: drive using the DOS LABEL command. The next time I did CHKDSK on my D: drive, it came back with "0 bytes in 1 hidden files"! A hidden file with zero length?! I had to find that file. As you can probably guess, that "file" turned out to be the volume label, which is listed just like any other directory entry. (See Figure 10.) How does DOS distinguish this entry from a file? The 12th byte of any directory entry is called the file attribute. If the lower four bits of this is an "8", the entry is a volume label. Similary, if the upper four bits is a "1", the entry is a subdirectory.

Which brings up another point — subdirectories are treated exactly like files in regard to their location on the disk. For example, refer again to Figure 9. My first subdirectory entry is ADVDOS. The first cluster is indicated as 007ah, which translates to sector 595. Figure 11 shows the directory entries within this directory. Note the two entries that begin with a period (2eh). The first entry acts as a pointer to access the current directory. The second entry (two periods) stands for the parent directory, which in this case is the root directory. This is why you can always return to a parent directory by simply typing "CD .." plus ENTER. By the way, since directory entries are treated the same as files, you can use a three-character extension as part of a directory name, such as "MYDIR.NEW".

Finally, we come to the root directory itself. This is the only directory that has a specified limit for the number of directory entries, which for a 20 MByte drive with one partition is 512. The root directory is stored as one contiguous block of sectors immediately following the second copy of the FAT. Since it has room for 512 entries and each entry is 32 bytes long, it occupies 32 sectors, starting at relative sector 83 and ending with sector 114. The reason DOS must restrict the root directory should be obvious. If it were allowed to expand indefinitely, it would have to be treated as a file with a cluster chain to keep track of all the different parts as it grew larger. Obviously, file and subdirectory location becomes a lot simpler if DOS knows exactly where the root directory is located and exactly how big it is.

After the root directory comes the first sector where files can be stored. On a single-partitioned 20 MByte drive, this normally is relative sector 115. If the drive is bootable, the first three files should be the two system files followed by COMMAND.COM. When you do a CHKDSK, DOS start with this sector when totaling storage space.

Finally, what happens when you format a hard drive? Unlike formatting a floppy disk, the DOS format (Version 3.04) command does not write over any files in the file area, so it is possible to recover files with certain utilities, particularly word processing files. However, the DOS format command does do the following:

1. Rewrites the boot record.
2. Overwrites the FAT with zeros.
3. Overwrites the root directory completely.

In contrast, the MACE FORMAT-H**** utility does not completely over-write the root directory. What it does is to change the first letter of every non-deleted directory entry to E5h, which indicates a deleted entry, thus making it easier to recover files. The FAT, however, is completely overwritten, as would be expected.

I hope this article has been informative and helpful. It certainly has been an educational process for me and has removed some of the mystery of hard drives and how they work.

* PC-Tools is a registered trademark of Central Point Software.

** Published by Simon and Schuster, Inc.

*** Published by Management Information Source, Inc.

**** FORMATH is a registered trademark of Paul Mace Software, Inc.

Continued from Page 12

```
    printf("Smaller SIDES will enlarge the image.\n");
    printf("To stop the program type Z.\n");
    printf("\n");
    printf("Depress any key to continue\n");
    #ifdef ZPC
            c = getchar();
    #else
            c = getch();
    #endif
    }
```

piled program will use an 8087 if you have one and will emulate one in its absence.

The fractal generating program explains itself. It produces a black and white image (I can't print a colored image); however, I think that you will need a color compatible monitor to get an effective display.

How to print or save the display? I use Hogware's "Z100grab utility" and I print the result from the captured picture file using "SHOWOFF." Hogware has these programs for the IBM compatible in addition to the H/Z-100 (NOT PC). There are also some Screen Dumping utilities available from HUG that will send completed pictures to your printer.

Under ZPC, I use the "SNAPSHOT" utility that comes with PFS: First Publisher. It makes the image into an "art" file that can be transferred into the "MAC" picture format. The Hogware SHOWOFF utility can read a "MAC" picture file and print it on your printer.

Probably the increased speed experienced with this program is due to the presence of the 8087 numerical processor. If you do a lot of computation, then you need one. Plug it in (just under the 8088 chip) on the H/Z-100 motherboard. After you type the listing, #define either Z-100, ZPC or IBM 1. The #define compiler directive makes Z-100, ZPC or IBM true which triggers different options for the compiler.

Compile with the command line:

```
cc -lgrafix bwfract
```

or

```
cc -lecos bwfract
```

Depending upon which of the graphics pixel setting routines you have available (ECOS.LIB or GRAFIX.LIB). BWFRACT .C will then be compiled to BWFRACT .OBJ and linked to form BWFRACT.EXE. Type BWFRACT <return> and you are ready for a fractal exploring adventure.

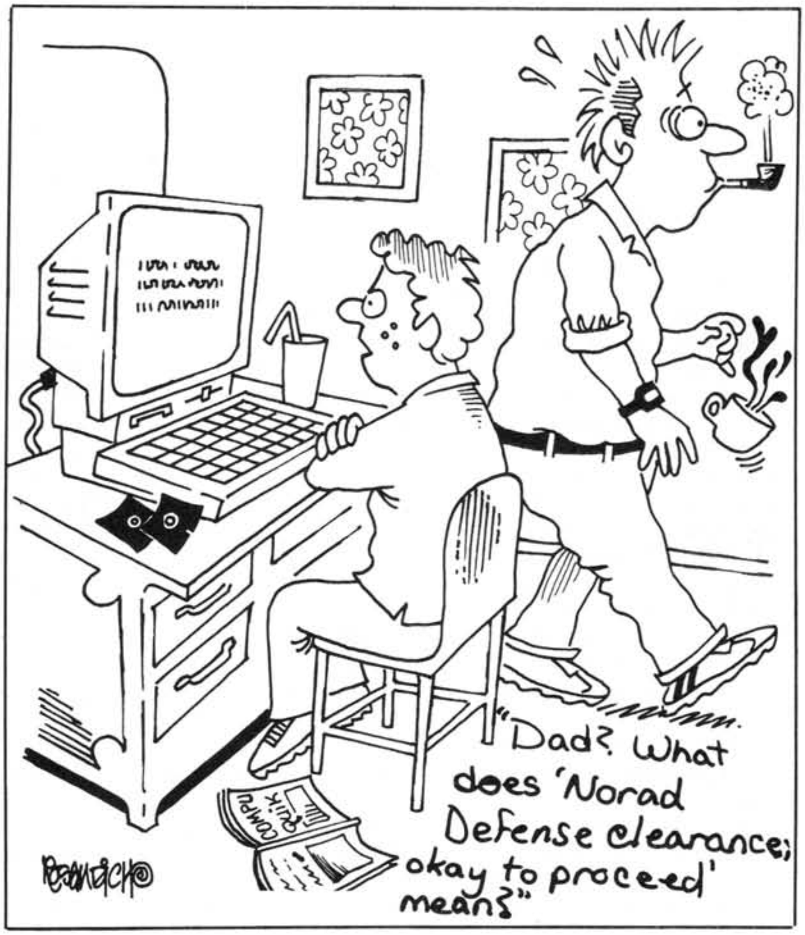
HUG Discount List

HA-2860-2	10%
HA-2860-4	10%
HA-2862-2	10%
HA-2862-4	10%
HA-3286-2	10%
HA-3286-4	10%
HV-2000	10%
HVB-550	10%
HWD-440	10%
HWD-4028	10%
SWA-4300	20%
TM-140	10%
TM-150	10%
TM-158	10%
TM-159	10%
TM-170	10%
TM-180	10%
TM-240	10%
TM-2300	10%
TM-2500	10%
TM-3034	10%
TMP-200	10%
Z-207-7	10%
Z-416-2	10%
Z-416C	10%
Z-416-SS	10%
Z-445	10%
Z-449	10%
Z-505	10%
Z-515	10%
Z-516	10%
Z-525	10%
Z-549	10%
Z-605-1	10%
ZA-1-4	20%
ZA-180-64	10%
ZA-180-65	10%
ZA-180-66	10%
ZA-180-67	10%
ZA-180-83	10%
ZA-180-85	10%
ZA-180-86	10%
ZA-180-87	10%
ZA-180-89	20%
ZA-181-7	10%
ZA-181-19	10%
ZA-181-23	10%
ZA-181-24	10%
ZA-3034-CS	10%
ZA-3034-EB	10%
ZA-3034-ME	10%
ZA-3034-NP	20%
ZA-3600-CI	10%
ZA-3600-CW	10%
ZA-3600-ME	10%
ZA-3600-MG	10%
ZA-3600-MQ	10%
ZA-3700-CI	10%
ZCA-2300-EF	10%
ZCA-2300-MG	10%
ZCM-1390Z	20%
ZCM-1400-1	10%
ZCM-1492	20%
ZD-12	10%
ZD-14	10%
ZD-800	10%
ZKB-2	20%
ZMM-149A	20%
ZMM-149P	20%
ZMM-1470G	20%
ZVM-1200-1	10%
ZVM-1240	20%
ZVM-1300-1	10%
ZVM-1330	20%

FOR ALL ASSEMBLED COMPUTERS,
CALL FOR PRICE.

Z-100 Survival Kit #12

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


COMPU QUIK
"Dad? What does 'Norad Defense clearance; okay to proceed' mean?"

This issue I'll devote the entire column to questions and answers I've received from Z-100 users. Here we go . . .

⁂

Question: *How can I write a keyboard input routine in Assembly, 'C', or Pascal, which will respond to the non-ASCII keys on the keyboard. Also, how can I determine if the SHIFT, CONTROL, and CAPS LOCK keys are down?*

Answer: This is a pretty broad question, about which an entire column could be written. I'll try to give a few pointers here. First of all, if you are using the 'C' language, or Pascal, or any high level language, standard library functions should be available to read the keyboard. Today's modern language implementations usually give the programmer several different types of character input routines, each of which may be useful in different situations. Some languages attempt to filter unwanted characters from the input stream (BASIC is notorious for this) and make reading non-ASCII characters difficult. But most languages allow you to read the actual characters which are read by the DOS input routine. Obviously, if you are using Assembly Language, you won't have any filtering problems since you will be using DOS function calls for keyboard input.

In any case, if you need to use non-ASCII keys in your program, it is usually easier if you disable the key expansion feature. This is done by sending an ESC ? y (hex codes 1B 3F 79) to the console. When key expansion is disabled, a single key code is returned for each key that is pressed. (See the Z-100 User's Manual, Appendix B, for key codes and escape sequences.) The default DOS mode (key expansion enabled) causes many of the keys to return an ESC code, followed by another key code. It is easier to handle special non-ASCII keys when they only return one unique key code. If your program does disable it again before exiting member to enable it again before exiting to DOS. This can be done by sending ESC ? x (hex codes 1B 3F 78) to the console.

In its normal ASCII scan mode of operation, the keyboard encoder considers the SHIFT, CONTROL, and CAPS LOCK keys to be modifier keys, and they do not generate a key code when they are pressed. This makes it impossible to monitor the state of these keys, as is done in an IBM-PC. There is a way that you can tell if the keys are down, and that is by using the up/down (event driven) mode of the keyboard. In up/down mode, the SHIFT, CONTROL, and CAPS LOCK keys generate separate up and down codes when they are pressed and released (just like any other key). In order to take advantage of this feature, your entire keyboard input routine would need to be written to use up/down mode. This would be a challenging project which is beyond the scope of this question and answer section. Even using up/down mode, you will not be able to tell whether it is the left or right SHIFT key that is down, like on a PC clone. The keyboard encoder chip itself would need to be reprogrammed to make this possible.

⁂

Question: *I'm using the BIOS__CONOUT routine to output text to the Z-100 screen. But I'm having trouble positioning the text on the screen. I've tried changing the HORZ__CHAR and VERT__LINE variables in the MTR-100 data segment before writing text to the screen, but the text still appears right where it left off the last time. Plus, when control returns to DOS, the cursor is still at the same spot it was before my program took over. What do you suggest?*

Answer: I'm not sure why the text location doesn't change when you manipulate the HORZ__CHAR and VERT__LINE variables in the MTR-100 data segment . . . I would need to look at your code. The D__CRT and S__CRT routines in the MTR-100, which are eventually called by the BIOS__CONOUT routine, do use these variables to determine the next character position on the screen. Likewise, the problem with the cursor is also mysterious. Whenever BIOS__CONOUT is called to output a character, the cursor is automatically updated. There is more to this than meets the eye. I suspect that you may be using a wrong value for the offset to the variables in the MTR-100 data segment, or you may be using some other routine to output the characters on the screen. For instance, if you are calling the MTR-100 DFC (Display Font Character) routine directly from your program, the symptoms you describe would result.

Sorry I can't give definitive answers without seeing some code, but there is an important point to note here. The MTR-100 ROM program is a very complex program that makes it possible for the Z-100 to operate. Many of the functions provided by the MTR-100 are intertwined, so that changing one thing causes problems elsewhere. If you plan to get into the MTR-100 data segment and change values, you need to know how each of those variables is used by the ROM program. Of course, it won't hurt anything to play around a bit. But if you want to write reliable software, you might be better off doing things the 'well-behaved' way.

The best way to change the cursor location is to use the ESC Y escape sequence. This is well documented in the Z-100 User's Manual. Using ESC Y automatically takes care of all the overhead associated with cursor and text positioning.

* * *

Question: *I have written a pop-up memory resident utility which needs to save the existing screen when it takes control. When 64K video RAM chips are installed, there is supposed to be enough memory for two pages of video memory, so I thought I might avoid using a big chunk of system RAM for buffer space by using this second page of video memory to save the screen. The problem is that when I move the contents of the screen memory from the beginning of each video plane to the top half of the video plane, it overwrites the existing screen starting about 16 lines down. Can you tell me what I'm doing wrong? Here is some of the code I'm using to save the screen:*

```
        MOV     AX, ØE000h              ; point to green video plane
        MOV     DS, AX                  ; point DS and ES to video plane
        MOV     ES, AX                  ;
        MOV     SI, Ø                   ; SI points to first page
        MOV     DI, 8000h               ; DI points to second page
        MOV     DX, 225                 ; will mcve 225 scan lines
   NEXTLINE:
        MOV     CX, 8Ø                  ; 8Ø bytes at a time
        CLD                             ;
        REP     MOVSB                   ; move 8Ø bytes now
        ADD     SI, 48                  ; skip to next scan line
        ADD     DI, 48                  ;
        DEC     DX                      ; decrement scan line count
        JNE     NEXTLINE                ; repeat until done
```

Answer: I'll give you an 'A' for effort, but your description of the problem, and the code sample, indicates that there are a lot of things you don't understand about the Z-100 video layout.

First of all, let me say that you are right in assuming that 64K video RAM chips will provide more than enough memory for two pages of video memory. A simple calculation shows that 225 scan lines of 80 bytes each only takes 18,000 bytes of memory, so theoretically at least, there should be enough room for three pages of video using 64K chips. But unfortunately, the way the Z-100's video memory is organized precludes this possibility. First of all, each 80 byte scan line includes 48 additional bytes at the end which are not used. This makes the math faster for scrolling and address calculations, since each line is 128 bytes long. You have taken this into account in your sample program by adding 48 to the DI and SI registers after each line is moved. Another complication in the video mapping scheme is that each group of nine scan lines (representing a text row) is followed by 7 non-displayed scan lines. This makes the beginning of each text row start at an even 800h byte boundary. This was also done to accommodate faster text scrolling.

When you take this odd video mapping scheme into account, you find that the normal 25 line text screen on the Z-100 appears to take 51,200 bytes of video RAM. This figure is arrived at by multiplying 16 scan lines/text row (9 displayed, 7 non-displayed) by 128 bytes/scan line by 25 text lines. The question naturally arises: "How can you address up to 51,200 bytes of memory, in a system that may only have 32K RAM chips?" The answer lies in the way the video memory is mapped between the CPU and the CRT-Controller. This subject is way too complicated to discuss here. Suffice it to say that the video RAM mapping module allows the CPU to see video memory in a way that is convenient for scrolling, while at the same time allowing the CRT-Controller to access the memory in a manner appropriate to screen refreshing. For those of you who want to know more, the Z-100 Technical Manual has an in depth (but barely understandable) discussion of this mapping scheme.

The end result of all this video memory mapping business is that you can't simply move 32K bytes from the start to the end of each video plane. It is possible to write to 'page one' or 'page two' of video memory, but it is done by changing the value of the video address latch. In other words, the second page of video memory is accessed at exactly the same memory address as page one, but the value of the address latch changes the actual location for the memory access. This is pretty deep stuff which needs to be explained at length. If there is interest, I'll cover it more thoroughly in a future installment of Z-100 Survival Kit.

But let's get back to your original problem of saving the screen for a pop-up utility. Even if you knew everything there was to know about saving the existing screen in the second page of video memory, this may not be the best way to go. What if your memory-resident program was popped-up while you are running a program that uses interlaced video? Since the high-resolution screen used in interlace mode uses more video memory, there would not be enough room to save the entire screen in video memory. You would also be precluded from using your pop-up utility with programs that make use of two pages of video memory. I'd say the best all-around solution, especially if you are writing your program for others to use, would be to byte the bullet and simply reserve a block of system RAM large enough for the screen buffer.

* * *

Question: *I would like to learn more about Assembly Language programming, and using MS-DOS function calls on the Z-100. There are lots of books available that cover this topic for PC compatibles, but nothing for the Z-100. Any suggestions?*

Answer: Any book which covers Assembly Language programming for the IBM-PC will be useful for Z-100 users, as well. The key here is that both the IBM-PC and the Z-100 have an 8088 CPU, and they both use the MS-DOS operating system. All of the Assembly Language instructions, and all of the DOS functions are identical between the IBM-PC and the Z-100.

A good place to start is by reading Pat Swayne's "Getting Started with Assembly Language" series in the last few issues of this magazine. Or, if you prefer a book, these are good tutorial introductions:

Assembly Language Primer
for the IBM-PC & XT
Robert Lafore, © 1984 the Waite Group
Published by New American Library

Peter Norton's Assembly Language Book
for the IBM-PC
Peter Norton and John Socha, © 1986
Brady Communications Co.
Published by Prentice Hall Press

Even though these books say they are for the IBM-PC and XT, almost everything in them is also applicable to the Z-100. Another valuable reference is:

The iAPX88 Book
Intel Corporation © 1981 or later
Published for Intel by Reston Publishing
Company

More advanced books about MS-DOS and Assembly programming would include:

Advanced MS-DOS
by Ray Duncan, © 1986 Ray Duncan

Published by Microsoft Press
MS-DOS Developer's Guide
John Angermeyer and Kevin Jaeger, © 1986 the Waite Group
Published by Howard W. Sams & Company
Tricks of the MS-DOS Masters
John Angermeyer, Rich Fahringer, Kevin Jaeger, and Dan Shafer
© 1987 the Waite Group, Published by Howard W. Sams & Company
MS-DOS Papers
Waite Group, © 1988 the Waite Group
Published by Howard W. Sams & Company
Memory Resident Utilities, Interrupts, and Disk Management with MS-DOS
Michael Hyman, © 1986 Michael I. Hyman
Published by Management Information Source, Inc.

There will be portions in these, and other books, which will not be applicable to the Z-100. Many authors also include a discussion of how to access the IBM-PC BIOS routines from Assembly Language. This type of info won't do you any good because the Z-100 has BIOS routines which are different from the IBM-PC.

Information about the Z-100's BIOS can be found in the Heath MS-DOS Programmer's Utility Pack (sometimes referred to as the PUP). If there is any book that can legitimately lay claim to being the Z-100 programmer's bible, this is it. Any serious Z-100 programmer MUST have this reference guide.

* * *

Question: *I know that the Z-100 maintains a font table in system RAM which it uses for text characters. I also know that this table is created by copying the table in the MTR-100 monitor ROM. Can I replace the RAM version of the table with my own character font?*

Answer: Sure, that's why the font table is moved into RAM memory to begin with. As a matter of fact, if you have an ALTCHAR.SYS file on your boot disk, it is already being done for you. The Z-100 version of DOS will automatically look for a file by that name in the root directory, and if found, that font will be copied into memory and used for screen text. This feature is commonly used to gain access to the H-19 style block graphics characters, but any font could be copied to ALTCHAR.SYS for loading at boot time.

After you have booted up, the font may be changed at any time by using the DOS FONT program. This program will allow you to change fonts, to redesign an existing font, or to create an entirely new font. The FONT program also lets you change the keyboard mapping. See your Z-100 MS-DOS manual for more information about this program.

If you want a program to load a special font, this is also possible. One way would be to have your program EXEC the DOS FONT program to load the new font. Or your program can load the font itself. In order to create and load your own font, you'll need to know the exact layout of the font table. This is shown in the source listing for the MTR-100 ROM (which is included with the Z-100 Technical Manual set). To find the start address of the font table in memory, follow this procedure:

1. Find the start address of the MTR-100 data segment. This is stored as a double-word pointer at address 0:3FCh in the interrupt table.
2. The double-word pointer to the start of the font table is at offset 6Fh in the MTR-100 data segment.

* * *

Question: *There are beginning to be a lot of cheap 8 inch drives available. Which ones will work with the Z-100?*

Answer: About the only thing I can say for sure is that the 8 inch drive interface of the Z-207 floppy controller is designed for use with a standard 50 pin Shugart compatible (SA801 or SA851) drive. I expect that other drive manufacturers could tell you if their drive meets this specification. The controller and BIOS software will support single- or double-sided drives. Be careful when you go shopping, because some of the older Heath drives (lovingly referred to as boat anchors) are not double-sided, double-density, and have a limited storage capacity. The more recent style drives allow 1.25 megabytes of storage, and still command a fair price (although that situation will change dramatically as more people switch to 5-1/4 and 3-1/2 inch high density drives.

* * *

Question: *How do I calculate the video address for a pixel on the screen, or for a text character on the screen?*

Answer: The video RAM offset for a text character at row R, column C, may be calculated as follows:

```
VOS = (R * 2Ø48) + C
```

This equation assumes that the row and column indices begin with row 0, column 0, and that the Z-100 is programmed for a standard 640 × 225 resolution screen with 25 text lines. The VOS number calculated above is the offset to the top byte in the text character. Each text character consists of 9 bytes of displayed information. Each successive byte of the character design is offset 128 bytes. This means that the ninth (and last) byte of the character design will be written to VOS +1024. Keep in mind that each plane of video memory may need to be updated independently, depending on the foreground and background colors of the font, and the status of the video control register bits.

When the MTR-100 ROM program writes text to the screen, it actually writes 11 bytes of information for each character (to the green plane only). The additional two bytes are not displayed on the screen. Byte number 10 is the ASCII code for the character, and byte number 11 are the character attributes.

The video RAM offset for a single pixel at coordinate X, Y may be calculated as follows:

```
TR = INT(Y / 9)
VOS = (TR * 2Ø48) + ((Y - (TR * 9)) *
   128) + INT(X / 8)
```

Additionally, the bit number of the pixel in the byte may be calculated as:

```
BIT = 7 - (X - INT(X / 8) * 8)
```

Again, we are assuming that a standard 640 × 225 screen is being used, and that the origin for the X and Y coordinates is 0,0 at the top left of the screen. We're also assuming that bits are numbered with bit 7 as the most significant bit in the byte.

There are more efficient ways of doing the arithmetic for these calculations. Typically, shift operations and modulo arithmetic should be used instead of multiply and divide instructions. But you get the idea.

* * *

Question: *The source listing for the MTR-100 ROM, and the documentation in the Programmer's Utility Pack, show quite a few different variables in the ROM data segment. I understand what many of them do, but some are elusive. There are some that I can't find referenced in the ROM code. What are they used for?*

Answer: Most of the variables in the MTR-100 monitor ROM data segment are used by the ROM to hold system status flags, address pointers, or to pass information to other routines. I'm sure Heath didn't plan for any of them to be changed by user programs, although clever programmers can do tricks by playing with them. Modifying the MTR-100 variables is definitely not for the faint-hearted, however.

Not all of the variables are used in current versions of the MTR-100 ROM. For instance, many of the variables in the COLOR structure are not used. Apparently, these variables are holdovers from earlier versions, or might even have been included in a trial version, and then never used. Being a programmer myself, I know that it is easy to forget to go back and remove unneeded trash before a product is released.

Not only are some of the documented variables unused, but there are many undocumented variables that are used. Only the variables up to about offset 300h are documented by Heath, but the data segment is 400h bytes, almost all of which is used for some purpose or another. Most of the undocumented variables are simply temporary storage locations which are used to hold loop indices or transient results.

* * *

Question: *The Z-100 Technical Manual gives a procedure to clear the Z-100* *screen by using the CLRSCRN bit of the video control port. One of the steps involves waiting for 16.7 milliseconds to elapse. How can this be done?*

Answer: There are two basic ways you can delay for the right amount of time, as described in the Technical Manual. You can use the timer, or you can wait for two consecutive vertical sync pulses. The way you count the video sync pulses is by hooking into interrupt vector 5Ah. This is a software interrupt generated by the BIOS when the vertical retrace interval begins. Your program should hook into interrupt 5Ah, wait for two interrupts, and then restore the interrupt vector to its original value.

The other way of forcing a delay is by using the system timer. It would probably be easier to use the timer that MS-DOS maintains, than to access the interval timer itself. This can be done by getting the DOS time, and then looping until 16.7 seconds have elapsed.

I personally think that either of these techniques is overkill when it comes to clearing the screen. It is easier, and still fast enough to simply let the 8088 CPU do the screen clearing by writing zeros to all memory locations. If you are willing to write code which allows your program to continue executing while it waits for the 16.7 seconds to elapse, then that's a different story. But if you're going to loop and wait anyway, you might as well just clear the screen manually, and forget about the CLRSCRN feature of the video control port.

Here is a sample code fragment which will clear the Z-100 screen (all three video planes) regardless of whether you are using normal or interlaced video.

```
IN      AL, ØD8h        ; get video port status
MOV     AH, AL          ; save status
AND     AL, 8Fh         ; enable all multiple access bits
OR      AL, 8Øh         ; enable CPU access of video RAM
OUT     ØD8h, AL        ; so all video planes are written
MOV     CX, ØEØØØh      ; get green plane
MOV     ES, CX          ;
MOV     DI, Ø           ; begin at start of video plane
MOV     CX, 8ØØØh       ; will clear 8ØØØh words of memory
CLD                     ;
REP     STOSW           ; do it to it
MOV     AL, AH          ; get original video port status
OUT     ØD8h, AL        ;
```

If we ignore the possibility of interrupts and video arbitration, this routine will clear the screen in about 60 milliseconds, which is fast enough for me. I mean how often do you clear the screen in a program anyway? The routine could be further optimized by only clearing the displayed portion of the video memory, at the expense of some additional code.

✳ ✳ ✳

Question: *Is anyone out there still willing to repair Z-100 computers?*

Answer: That's getting to be a real good question. At the risk of offending Zenith Data Systems (referred to as *.* on the January cover of REMark), I know that many of the Heath/Zenith Computer & Electronics Centers are shying away from Z-100 repairs. I'm in a position where I receive a lot of feedback about this type of thing, and I have received numerous reports of service being refused for such minor technicalities as the existence of an FBE memory expansion or a CDR speed-up kit. In other words, some of the Heath/Zenith centers are using the existence of non-Zenith modifications as an excuse to refuse service. This didn't use to be the case. But it seems to depend almost entirely on the management at the local store. I've also gotten reports of Heath Stores charging enormous up front fees (like $85.00) just to open the case and look. If you have a good relationship with the local Heath Store, and you can stand all those Apples looking at you, then by all means that's where you should take your Z-100.

Another alternative, providing you or a friend doesn't repair computers, is to try an independent Zenith Data Systems dealer. Again, this is a hit and miss situation, and you'd better check out the local dealer before you trust him with your baby. Very few ZDS dealers even do any service on computers. Don't even bother calling the local Zenith TV shop. Some of the ZDS dealers that have been serving the Heath community for years would be a good bet — you know — the ones who have advertised in REMark over the years. Good examples are First Capital Computer, Payload Computers, and Quikdata, Inc. I'm not sure about First Capital or Payload, but I do know that Quikdata will repair Z-100s, even if you didn't buy it from them originally.

The only other alternative, and one which is getting to be more popular as time goes by, is to simply junk your old Z-100, and buy another one. The going price for used dual-floppy Z-100s seems to be about $250. Of course, most used models are loaded with goodies, so the price might be higher for a particular system. And in the future, the price is going to go through the floor as bunches of Z-100s start coming back into the private sector through government auctions. Yep, I'd say if there is anything major wrong with your Z-100, you might be just as well off buying another, and keeping the old one for spare parts. ✳

PART 1

Getting Started With . . .

GDU

(ZDS' GENERAL DISK UTILITIES)

JAN AXELSON
2209 WINNEBAGO STREET
MADISON, WI 53704

If you have Zenith Data Systems' MS-DOS version 3.3 Plus, you also have something called GDU, which stands for General Disk Utilities. Much of GDU is dedicated to disaster relief — helping you recover files or directories that were deleted by mistake. Other functions let you sort directories and examine and edit a disk. In short, GDU has many of the same capabilities as popular programs like the Norton Utilities, Mace Utilities, and HUG's HADES II. Though GDU may not have all of the features of these other programs, it does have one advantage, at least for owners of MS-DOS 3.3 plus: since GDU is part of DOS, no additional purchase is required.

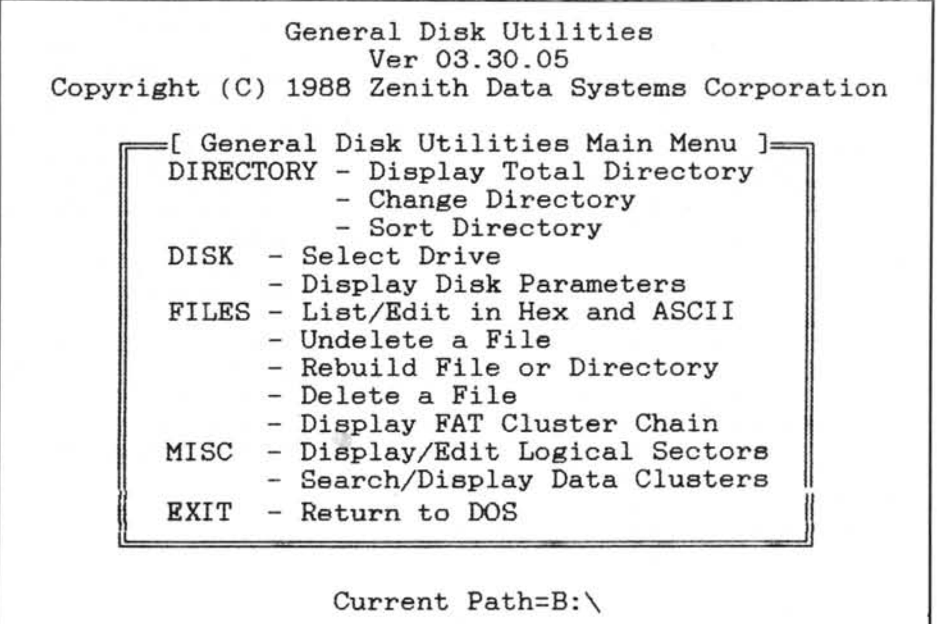
Through its menus and on-screen messages, GDU does a pretty good job of letting you know what you can do and how to do it. But it's not always obvious when each of GDU's functions might come in handy. This two-part article will introduce you to when, as well as how, to use GDU.

In this first installment, we'll look at the functions of GDU that you're likely to use most often, including undeleting files and sorting directories. I'll also include some background about disks and how they're structured, since this will help in understanding some of GDU's functions.

Next time, we'll explore GDU's ability to examine and edit any part of a disk, and how this can be used in recovering deleted, copied-over, or otherwise unrecoverable files and directories.

Starting Up

Activating GDU is easy to do. On a floppy-disk system, insert the DOS system disk containing GDU.EXE. On a hard disk system, change to the directory containing GDU.EXE. Type GDU, press ENTER, and you'll see GDU's main menu, shown in Figure 1.



```
                 General Disk Utilities
                      Ver 03.30.05
  Copyright (C) 1988 Zenith Data Systems Corporation

      ╔═[ General Disk Utilities Main Menu ]═╗
      ║ DIRECTORY - Display Total Directory  ║
      ║           - Change Directory         ║
      ║           - Sort Directory           ║
      ║ DISK  - Select Drive                 ║
      ║       - Display Disk Parameters      ║
      ║ FILES - List/Edit in Hex and ASCII   ║
      ║       - Undelete a File              ║
      ║       - Rebuild File or Directory    ║
      ║       - Delete a File                ║
      ║       - Display FAT Cluster Chain    ║
      ║ MISC  - Display/Edit Logical Sectors ║
      ║       - Search/Display Data Clusters ║
      ║ EXIT  - Return to DOS                ║
      ╚══════════════════════════════════════╝

                  Current Path=B:\
```

Figure 1
GDU's main menu presents you with 13 options.

(GDU is easy enough to type, but how to pronounce it? "Gee-dee-you" isn't bad, but I've taken to calling it "G'doo," which has sort of an Australian ring to it.)

To highlight each of the menu items in turn, press the Down-Arrow or Up-Arrow key repeatedly. To select the currently highlighted item, press ENTER. For all items except EXIT (which returns you directly to DOS), you'll then see a new screen showing the information requested, or a menu of the options available for that item, or a request for more information.

One caution: Viewing the options is harmless, but GDU does include some powerful disk editing capabilities that can cause big problems if they're misused. In particular, do not press F8 (Edit Sector) after selecting a file in LIST/EDIT IN HEX AND ASCII, or after selecting a sector in DISPLAY/EDIT LOGICAL SECTORS, unless you know what you're doing.

You can return to GDU's main menu by pressing ESC (do so more than once if necessary). Any editing or changes that haven't yet been saved to disk will be ignored when you press ESC. From the main menu, you can return to DOS by highlighting EXIT and pressing ENTER.

Unless you tell it otherwise, GDU will work with the current default drive and directory — the ones you're in when you run GDU. To select a different drive to examine or edit, select SELECT DRIVE from the main menu and enter the letter of the drive you want. You also can select your drive by typing its letter on the command line when you run GDU. For example, the command GDU A: will run GDU and select drive A.

To change the directory to examine or edit, select CHANGE DIRECTORY from the main menu. Unfortunately, GDU lets you move up or down just one directory level at a time. In other words, from the CHANGE DIRECTORY screen, you can move only up to the current directory's parent directory (if there is one), or down to any subdirectory of your current directory. More complex moves require several steps.

Often it's simpler to change to the drive and directory you want to examine before you run GDU. If you do this, include GDU's drive and directory on the command line, if necessary. For example, if GDU is in the DOS directory of drive C, you can run GDU from the A: prompt by typing C:\DOS\GDU. GDU's selected directory will then be the current default directory of drive A.

Total Directories

After selecting a drive or directory, select DISPLAY TOTAL DIRECTORY to verify that you've selected the directory you want.

GDU's directory display is similar to what you see when you type DIR in DOS, with some additional information. GDU shows hidden files (H), system files (S), erased files (Erased), and volume labels (V). For example, if you display a bootable disk's root directory, you'll see the files IBMBIO.COM and IBMDOS.COM. Each will be labeled RHS to show that it's a Read-only, Hidden, System file. DOS's DIR command won't show you these files at all.

Read-only, as you might guess, means that you can't write to, or change, the file with conventional DOS operations. Read-only is normally used to prevent accidentally writing to important, unchanging files. You can make a file read-only or not with DOS's ATTRIB command.

Hidden and system files are files that are not displayed by DIR, for whatever reason. Erased files are files that have been deleted with DOS's DEL or ERASE command. These include most files deleted from within applications programs. Volume labels are the optional disk labels you may enter when you format a disk, or with DOS's LABEL command.

One thing GDU does not show is the status of a file's archive bit (used with XCOPY and BACKUP).

Sorting Directories

GDU can sort a directory in any of several ways. For a menu of sorting options, select SORT DIRECTORY from the main menu. You can sort by file name including or excluding extension, by extension first, then file name, by date and time, by type (read-only, hidden, etc.), or by size.

If you prefer your directories in one of the above orders rather than in the order they otherwise end up in, GDU can help. A directory sorted by GDU will stay sorted after you exit GDU. A drawback is that you can sort only the current selected directory — there's no way to sort several directories in one step.

You also can use DIRECTORY SORT as a tool for temporary use. For example, to quickly find the oldest or newest files in a directory, sort the directory by date, then return to the order you prefer when you're finished.

About Disks

On request, GDU will show you some technical information about a disk. From the main menu select DISPLAY DISK PARAMETERS, and you'll see a screen of information about your current selected disk. Figure 2 shows the disk parameters displayed by GDU for a MicroScience HH-1050 40-Megabyte hard disk. You can probably compute for a lifetime without knowing these parameters, but GDU will show them to you if you want or need to know them. To print out your parameters, press SHIFT-PrtSc while they are displayed.

The parameter FORMAT ID refers to the type of disk, with an ID of F8 signifying a hard disk. DATA CAPACITY tells how many bytes of data the disk can store. Figure 2 shows that the MicroScience 1050 can actually store 44,363,776 bytes, or over 44 Megabytes of data. (Data capacity depends on the disk and disk controller. Different controllers may allow different capacities for the same disk.)

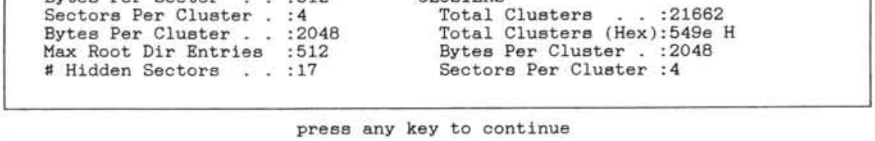
To understand most of the other disk parameters, we need to define some terms relating to disks. Both hard disks and floppy disks are magnetic recording media in the shape of platters, or disks. Each recording surface on a disk requires a read/write head for reading and writing to the disk. A double-sided floppy has two surfaces, or sides, and thus two heads. Hard disks typically have more than one platter and a recording surface on each side of each platter, though one surface may be reserved for use in positioning the heads. Figure 2 shows that the MicroScience 1050 has 5 heads.

Data is recorded on a disk in circular paths called tracks. The tracks are divided into sectors, usually 512 bytes in length. Figure 2's disk has 512-byte sectors, with 17 sectors per track.

Hard disks may be divided into more than one partition, or logical drive, each with its own letter designation (C, D, E, etc.). But it's common to format a disk with a single partition that uses the entire disk, and this is the type of disk we'll assume for this article. (Partition type is set with DOS's PREP and PART commands.)

In formatting, most of the disk is divided into clusters, which are the units used when storing files on the disk. Each cluster consists of one or more sectors, with cluster size dependent on disk type. Figure 2 shows a cluster size of 4 sectors, or 2048 bytes, which is typical for a hard disk. We'll see later how the use of clusters helps keep track of where a file is stored on a disk.

As you would expect, if you multiply Figure 2's sector size (512 bytes) times its sectors per cluster (4) times the number of clusters (21,662), you get the disk's data capacity (44,363,776 bytes).



```
                         --- DISK PARAMETERS ---

DISK DRIVE = C                          SYSTEM ID : C ZDS3.3
  Partition Type :6                       Fixed Disk Partition
  Data Capacity . . :44363776
  Number of Heads . :5
  Format ID . . . . :F8                 SECTORS
  # Bits/FAT Entry  :16                   Total Sectors on Disk  :86853
                                          Root Directory Sectors :171 - 202
DISK FORMAT                               File Allocation Table  :1 - 170
  Sectors Per Track . . :17               Data Sectors . . . . . :203 - 86853
  Sectors Per FAT . . . :85               # Reserved Sectors . . :1
  Number of FATs  . . . :2
  Bytes Per Sector  . . :512            CLUSTERS
  Sectors Per Cluster . :4                Total Clusters   . . :21662
  Bytes Per Cluster . . :2048             Total Clusters (Hex):549e H
  Max Root Dir Entries  :512              Bytes Per Cluster . :2048
  # Hidden Sectors  . . :17               Sectors Per Cluster :4

                        press any key to continue
```

Figure 2

On request, GDU will display a screen of information about a disk. The above display is for a 40-Megabyte hard disk.

The first sectors on a hard disk partition or floppy disk make up three areas called the boot record, file allocation table (FAT), and root directory.

The boot record (sector 0) contains a boot loader, which is a short program that loads (or boots) DOS. The boot record also contains information about the disk. In fact, most of what is displayed by GDU's DISPLAY DISK PARAMETERS is contained in the boot record.

The FAT

The next group of sectors (sectors 1 through 170 in Figure 2) contains the FAT, which keeps a record of each cluster on the disk and whether it is bad (unusable), in use (holding data), or free for use. The disk keeps two copies of the FAT, so that its information is saved even if one copy is damaged.

Following the FAT is the root directory (sectors 171-202 in Figure 2), which contains information about the files and subdirectories of the disk's root, or parent, directory. Unlike subdirectories, which can grow to any size, the root directory's size is limited (to 512 bytes in Figure 2).

The remaining sectors (203 through 86,853 in Figure 2) are the data sectors, which make up the clusters that store the disk's files and subdirectories. The clusters are numbered, beginning with cluster 2. If a file uses more than one cluster, its clusters need not be in numerical sequence, but each cluster may be used by only one file. Two files may not share the same cluster.

How does the computer keep track of which clusters are used by a file? Although you don't see it when you type DIR, a file's directory entry contains the number of the first cluster used by that file. Meanwhile, the FAT contains an entry (a numerical code) for each cluster on the disk. The FAT entry for a file's first cluster stores one of two things: either a code signifying that this is the last (and only) cluster used by the file, or the number of the file's next cluster. If the FAT points to a new cluster, the entry for this cluster will in turn point to the next cluster or contain a code signifying the final cluster, and so on, until the final cluster is reached.

Normally, all of this is invisible to you when you use your computer. To run a program, you need only type its name — you don't have to be concerned with where on the disk the program is physically located. But knowing this information can be useful in certain situations, like when you're trying to recover a deleted file with GDU and its standard undeleting techniques fail. We'll see examples of this later on.

Experimenting With GDU

Several of GDU's functions can help you recover files or directories that have been deleted by mistake, or that you can no longer access for some other reason. To experiment and practice with GDU, you're safest using a "practice" floppy that contains only files you no longer need. This way you can practice undeleting without having to worry if you make a mistake. At the very least, create a temporary directory on your hard disk and copy into it several files to practice with.

Text files are the easiest to analyze, so include some of these on your practice disk. Any file created by a word processor or text editor is a good candidate. After copying files onto your disk, use DOS's DEL command to delete some of your practice files (at least three), so you can try undeleting them.

Undeleting

From its name, you might logically conclude that DEL deletes, or erases, a file from a disk. But this isn't strictly true. When you delete a file, the file actually remains on the disk, for a while at least. Two things do change: The first letter of the file name in the directory is changed (to ASCII character E5, shown as a Greek sigma). This causes the file to no longer be displayed by DIR. And the FAT entry or entries for the file are changed to zero, to signify that the clusters containing the deleted file are now free for use by other files.

The number of the file's first cluster remains in the file's directory entry, and this is what allows GDU to begin to restore deleted files.

The quickest way to undelete a file is to type the name of the file on the command line when you run GDU. For example, if GDU is stored in the DOS directory of drive C and you want to undelete PRACTICE.FIL on drive A, at the C: prompt, type \DOS\GDU A:PRACTICE .FIL.

Try the above command, substituting the appropriate drive and directories and the name of one of your deleted practice files. With luck, you'll see the message "File <filename> appears to have undeleted successfully." To verify that the undelete was successful, exit to DOS and view the file using the DOS TYPE command or a text editor such as EDLIN.

For a non-text file, or any file that you can't analyze conveniently by viewing it, you can test for a successful undelete by trying to use the file as you normally would. For example, if the file name has a COM or EXE extension, try to run it. If the file seems to work OK, chances are you've undeleted successfully.

You also can undelete from GDU's main menu. Run GDU and select your practice disk and/or directory. Verify that you've selected the directory you want by selecting DISPLAY TOTAL DIRECTORY from the main menu. Your deleted files will be marked "Erased."

To undelete an erased file, from the main menu select UNDELETE A FILE. You're now presented with three options, selectable with the arrow keys. (We'll look at a fourth option, REBUILD FILE, later.)

The first two options are similar in how you use them, though the processes they use differ slightly. CLUSTER ANALYSIS PRIORITY searches the disk for erased clusters of the correct data type (either binary or text) and restores the file from these. CLUSTER SEQUENCE PRIORITY rebuilds the file from any erased clusters it finds, regardless of type.

The manual doesn't explain how GDU tells the difference between ASCII and binary clusters. Generally, binary files include most program files, such as files with COM or EXE extensions, while ASCII files are those that consist mainly of ASCII codes, such as files created by a word processor or other text editor. At any rate, it isn't essential to know what type of file you have in order to use GDU.

The manual recommends trying CLUSTER ANALYSIS PRIORITY first. This is the option used when you delete by typing the file name on GDU's command line, as we did above. If you select this option from the menu, you'll see a directory of the erased files in your chosen directory, with the first letter of each file name replaced by a question mark. Figure 3 shows an example of this screen. If the file you want to undelete isn't listed, be sure you've selected the correct drive and directory.

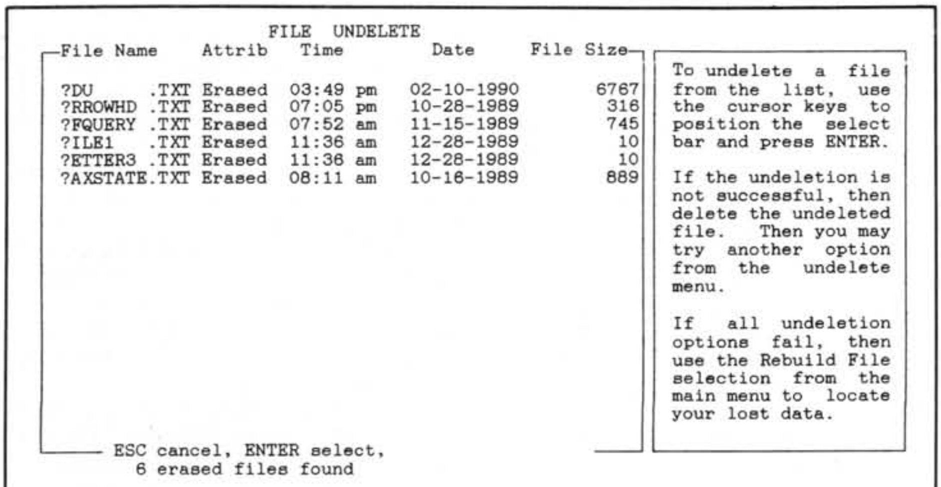
To undelete, use the arrow keys to highlight the file you want, press ENTER to select it, and type a character to replace the question mark. You can use the same first character that the original file used, or you can choose something different. With luck, you'll again see "File <filename> appears to have undeleted correctly" along the bottom of the screen. As before, you can check for a successful undelete by viewing or using the file.

In some situations, GDU may undelete a file incorrectly. In attempting to undelete, it will create a new file that may contain some, but not all, of the deleted file's clusters. If this happens, it's time to try CLUSTER SEQUENCE PRIORITY.

Before doing so, use GDU's DELETE A FILE option to erase any incorrect file that GDU created. Even if your first undelete was successful, try undeleting the same file, or another of your practice files, with this option. Select UNDELETE A FILE from the main menu, then CLUSTER SEQUENCE PRIORITY. From there, the procedure to follow is the same — view the erased files, select the one you want, type the first letter of the file name, let GDU go to work, and examine the results.

Using GDUTSR

The third option for undeleting is FILE



```
                         FILE  UNDELETE
┌File Name    Attrib   Time          Date       File Size┐ ┌─────────────────────┐
                                                            │To undelete  a  file │
 ?DU      .TXT Erased  03:49 pm   02-10-1990       6767│ │from the  list,  use │
 ?RROWHD  .TXT Erased  07:05 pm   10-28-1989        316│ │the  cursor keys  to │
 ?FQUERY  .TXT Erased  07:52 am   11-15-1989        745│ │position the  select │
 ?ILE1    .TXT Erased  11:36 am   12-28-1989         10│ │bar and press ENTER. │
 ?ETTER3  .TXT Erased  11:36 am   12-28-1989         10│ │                     │
 ?AXSTATE.TXT Erased  08:11 am   10-16-1989        889│ │If the undeletion is │
                                                        │ │not successful, then │
                                                        │ │delete the undeleted │
                                                        │ │file.   Then you may │
                                                        │ │try  another  option │
                                                        │ │from  the   undelete │
                                                        │ │menu.                │
                                                        │ │                     │
                                                        │ │If   all  undeletion │
                                                        │ │options  fail,  then │
                                                        │ │use the Rebuild File │
                                                        │ │selection  from  the │
                                                        │ │main menu to  locate │
                                                        │ │your lost data.      │
└────── ESC cancel, ENTER select, ─────────────────────┘ └─────────────────────┘
          6 erased files found
```

Figure 3

GDU will show you the names of all deleted (erased) files in a directory. The first character of each file name is replaced by a question mark .

RESTORATION FROM BACKUP DATA. This can be used only if you loaded GDU's companion program GDUTSR (General Disk Utilities Terminate and Stay Resident) before you deleted. GDUTSR creates the file GDINDEX.$$$, which saves the cluster numbers and path (directory location) of each file you delete.

To load GDUTSR, exit from GDU, change to your practice disk, and type GDUTSR, including its drive and path as necessary. This loads into memory a terminate-and-stay-resident program that instructs your computer to store the cluster numbers and path of each file you delete. It also checks to see if GDINDEX.$$$ exists in the root directory of the current selected disk and if not, creates it.

GDINDEX.$$$ is a hidden file, so you won't see it when you type DIR. You will see it listed with GDU's DISPLAY TOTAL DIRECTORY, however, and you can delete it with DEL.

Delete one of your practice files. As it's deleted, you'll see a window with the message "DATA BACKUP (GDUTSR)," to let you know that the cluster numbers and path of your deleted file are being saved in GDINDEX.

After you delete a file using GDUTSR, run GDU and select UNDELETE A FILE, then FILE RESTORATION FROM BACKUP DATA. You'll see a directory of the files that have been deleted while GDUTSR was loaded into memory. Select the file you want to restore, and GDU will go to work. Instead of having to guess, GDU looks in GDINDEX to find out which cluster(s) contained the file. The file will be restored to its original directory, regardless of what directory is currently selected.

To find out before you delete a file whether GDUTSR has been loaded, press ALT-U, or type GDUTSR. If GDUTSR is already loaded, you'll be informed of this. If you change disks after loading GDUTSR, you do not need to run GDUTSR again. If GDINDEX doesn't yet exist on a disk, GDUTSR will create it when a file is deleted.

The manual gives no command for unloading GDUTSR once it's been installed. If necessary, you can reboot your system and the automatic backups will no longer be made, until you again type GDUTSR.

In its default size, GDINDEX will hold information about the last ten files deleted. As you delete your eleventh file, information about the first file is dropped. You can change the capacity of GDINDEX to anywhere from 1 to 99 files. Just enter the number of files on the command line when you load GDUTSR. (For example, GDUTSR 5 will store locations for five files.) If you change the size of GDINDEX, any previous cluster locations saved are wiped out and you start fresh.

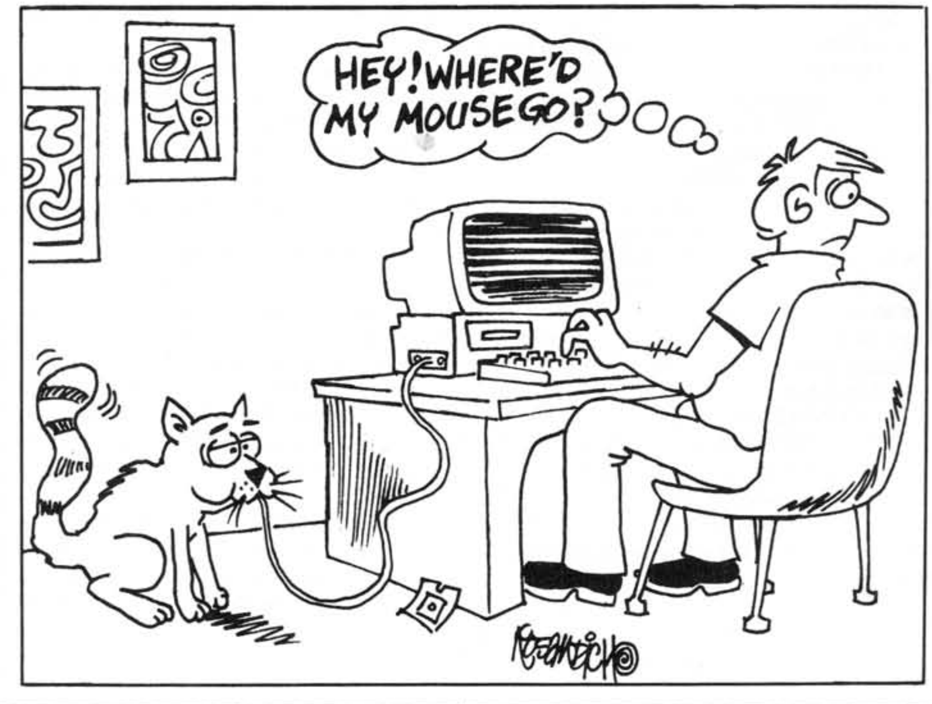
A smaller GDINDEX may be desirable to save disk space, though the effect isn't as dramatic as you might expect. On my hard disk, a 1-file GDINDEX is 23,040 bytes, while a 99-file GDINDEX is 33,280 bytes. (On floppy, GDINDEX is much smaller, perhaps because it doesn't expect to have to store as many cluster numbers.) Once you create GDINDEX, the same index file is used whenever a file is deleted on that disk, if GDUTSR is loaded.

Although GDUTSR can make successful undeletes easier, it has disadvantages, including GDINDEX's use of disk space. Also, GDUTSR can cause problems, including system crashes, when files are deleted from within some applications programs. And GDUTSR can't guarantee successful undeletes. If you write to a disk after deleting, you run the same risk as always of overwriting your deleted files' clusters. Since GDU's other undelete options are effective in many cases, you may decide to use GDUTSR infrequently or not at all.

A Final Tip

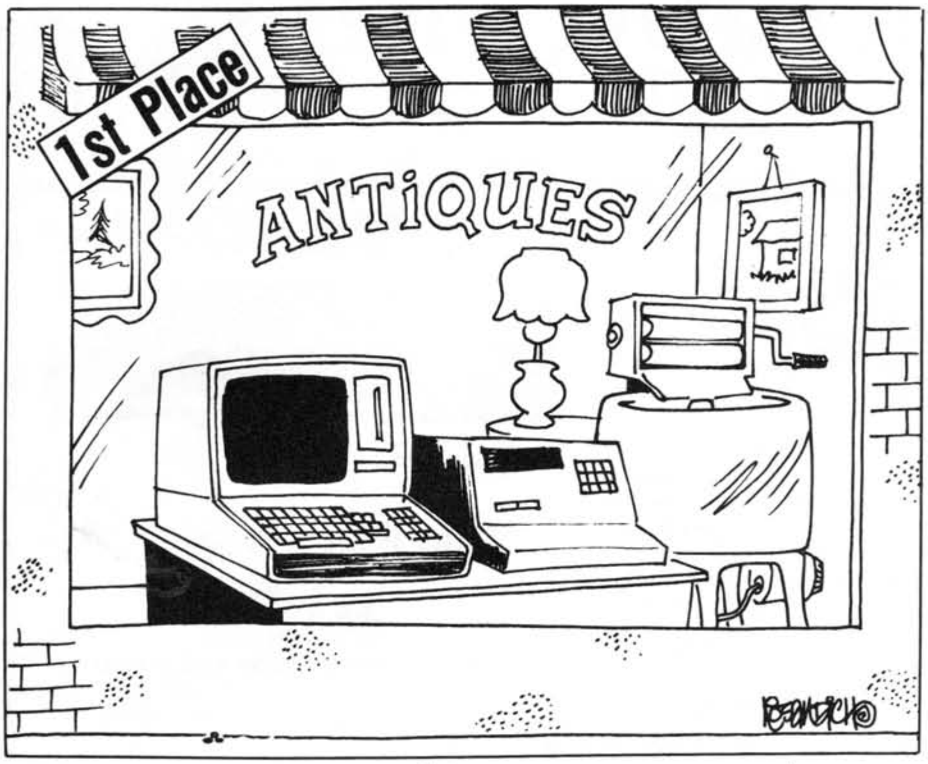
Whether or not you use GDUTSR, always undelete as soon as possible after deleting. It's OK to run GDU, view directories, and perform other functions that don't write to the disk. But the more times you save information to a disk after deleting, the more likely it is that your deleted file's clusters will be reused and you won't be able to get the file back.

This completes Part 1 of our introduction to GDU. Next time we'll look at how to examine files, and how to rebuild files and directories that the regular undelete options can't handle. ✱



CARTOON CONTEST WINNERS!

IT WAS DIFFICULT, BUT WE'VE DECIDED!



"Sure, you guys. I know what a backup is!
I once had a sock caught in my drain hose."

Thomas Bing
Smyrna, GA 30080

Way back in the November 1989 issue of REMark, we ran a contest. We were looking for the best caption for our cartoon. Well, here it is, the final selection on the winners.

The HUG staff chose our First Place winner wtih a unanimous decision. Thomas B. Bing (Smyrna, GA) gave us the caption for our winning cartoon entry. He will receive a ZWL-200-21 SupersPort 286e, 1.4 MB, and 20 MB winchester.

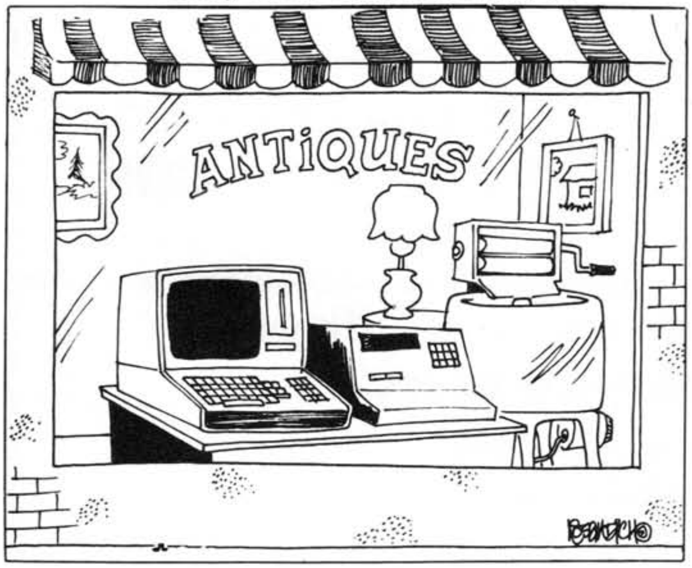
Our Second Place winner was harder to select because we had two entries that were very good. Since we here at HUG were at a split decision, we declared a tie. Our two Second Place winners are: Henry Hosek, Jr. (Crown Point, IN) and Jim Gilliland (Clovis, CA). They both will be receiving a ZWL-200-2 SupersPort, 1.4 MB, and 20 MB winchester.

We have run the winning entries here, and will be running some of the other entries in upcoming issues.

Thank you for the great response to this contest, and there will be more chances to win in the future.

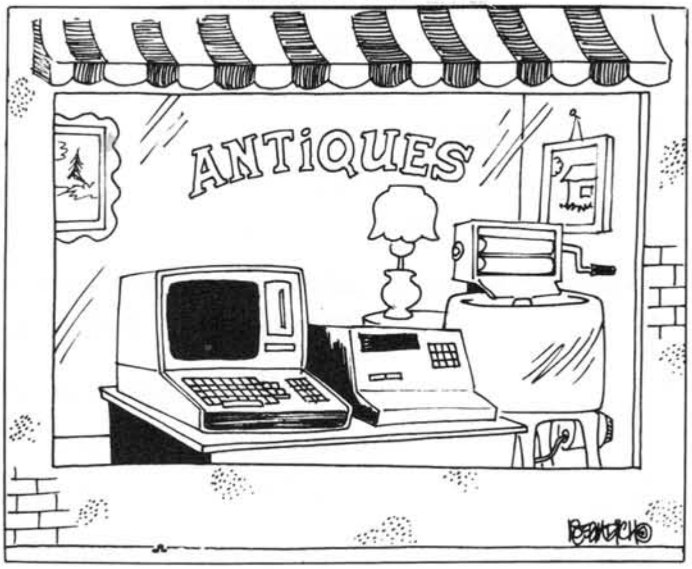
2nd Place

Henry Hosek, Jr., Crown Point, IN



Three Early Cleaners:
One for clothes: Two for wallets.

Jim Gilliland, Clovis, CA

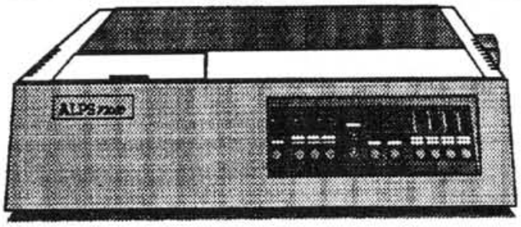


One uses an old motherboard:
The other was used by an old bored mother.

W S Electronics

(513) 376-4348 ** Since 1975 **** (513) 427-0287**
1106 State Route 380, Xenia, Ohio 45385

WHAT MAKES THE ALPS SO SPECIAL, ANYWAY?



It's simple. The ALPS P2000™ Dot Matrix Printer has everything it takes to replace your old business printer.

For instance:

- Dependability—full one-year limited warranty, 5 years normal use without a breakdown.
- Speed—250 draft, 125 correspondence, 50 letter quality cps.
- Versatility—letter quality, graphics, multiple type fonts and sizes, paper-saving push/pull tractor feed.
- Compatibility—runs with most popular PCs and software.

So you see, now there's a very special reason for replacing your old printer. A better printer.

***Reconditioned with 90 day warranty, Quantities are limited**

Retail Price $995.

Sale Price $ 395. *

ALPS
AMERICA

IT'S TIME YOU SAW THE ALPS.

P2000 is a trademark of ALPS Electric Co., Ltd.
©1987, ALPS America

enable™

2.0

***Word Processing**
***Spreadsheet**
***Graphics**
***Database Management**
***Telecommunications**

Retail Price $695.

Sale Price $99.*

*Quantities are limited

Attention: Federal Government Offices

We stock ALL ALPS Printer Models

We stock ALPS PARTS and RIBBONS for all ALPS models including your P2000's and ASP 1000's

****** Government Discounts Offered ******

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

Reader Service #149

A 6.6 Meg. Floppy Drive in the Z-151

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

When I bought my Z-151 computer in 1984, I knew I was getting a state-of-the-art machine. I anticipated that sooner or later it would become obsolete (it was already my third computer). However, I did not anticipate the degree of flexibility that was built in, both in the design and in the amount of continuing support by Zenith Data Systems. ZDS does an excellent job of making available upgrade monitor ROM chips. As a result, I have been able to keep my computer very close to state-of-the-art.

This computer now uses the Intel 80386 chip (Intel Inboard 386/PC) with one meg. of 32-bit high-speed memory (soon to be expanded to 5 meg.). The computer also has a high resolution video and a 20 meg. hard drive (soon to be expanded to 40 meg.). My latest addition is a Kodak 6.6 MB floppy disk drive. This drive uses special 5-1/4" diskettes, 17 sectors, 384 TPI. The diskettes must be formatted by a special program provided by Kodak that leaves 5.5 MB of usable disk space.

Although I have never had a hard disk failure, or even a hint of hard disk trouble, I am very conscientious about keeping current backups. I do a full hard disk backup at least once a month (using Fastback Plus) and a differential backup once a day (backups of those files changed since the last full backup). Using 360K floppy drives require between 35 and 40 diskettes for a full backup. Using the Kodak drive, the same backup can be done on only 3 diskettes!

I will describe my experiences in installing and using this high capacity drive. I will then give some background and describe some of my experiences with the Intel Inboard 386/PC in my Z-151.

I first heard about the Kodak drive at a meeting of the San Diego HUG. One of our members obtained these drives through a group purchase for $175.00 each plus $3.00 for each of the special diskettes. I installed the drive without difficulty. In order to save space, I replaced the B floppy drive with the Kodak drive. The drive connects to a special controller card supplied by Kodak. A device driver is installed in the CONFIG.SYS file, the system is rebooted and gives a message "Kodak 6.6 Drive Installed as Logical Drive D" (DOS treats the drive as a hard drive). The drive worked flawlessly right from the start, except for one complication — my second serial port had become dead. The external modem connected to Com Port 2 had to be moved to Com Port 1 (displacing my mouse in the process) and the troubleshooting began.

A switch block on the Kodak controller card could be set to disable the Winchester, to disable the high capacity drive or to disable the 360K floppy. None of the switch settings would enable the second serial port. I called Kodak Technical Support and was told that the XT type of computer had a limited number of channels. They suggested that I run the 360K floppy off the Kodak controller (in the "replacement" mode) instead of off the system controller (in the "coexist" mode). When I told Technical Support that my cable would not fit the Kodak controller, they sent me a new cable. They said that using the "replacement" mode had solved a similar problem in the IBM XT. In my computer, however, neither floppy would work in the "replacement" mode. Technical Support said it seemed that both controllers were competing for the same address. They didn't know anything about the Zenith Data Systems setup and had no further advice. In the "coexist" mode, the disk controller takes over interrupt request line 3, the line generally used by Com Port 2.

I went back to using the "coexist" mode and began to feel discouraged. I decided to try one more avenue, and I called FBE Research, the mail order company in Seattle, Washington. They market many useful products for Zenith Data Systems computers and had been very helpful to me in the past. I asked them if they had a way of changing the address for Com Port 2. They said that it wasn't necessary and reminded me that the IRQ line number for Com Port 2 was jumper selected on the floppy controller card. Sure enough, I moved the jumper block from IRQ3 to IRQ4, loaded my modem software program (I use Mirror III), and changed the Port 2 address from 2Ch (IRQ3) to 30h (IRQ4). The modem started working again and the mouse went happily back to live at Com Port 1.

The Kodak drive can read from, but not write to, standard 360K floppies. That means giving up the Diskcopy and Diskcompare utilities. Also, the computer cannot be booted from this drive. Files can still be read and copied from standard 360K floppies in this drive using the Copy or XCopy commands. Some users I know have installed their Kodak drive in an external chassis and use it concurrently with both 360K floppies and even with 1.2 meg. floppies.

Each enhancement to this computer has been both an adventure and a learning experience. Computers are a hobby with me. Although I have been building various Heathkits for the past 20 years, I am not a professional programmer. I am a Psychoanalyst by profession. My interest in how the mind works extends to computers as well. In 1975, I bought a Micro-86 microcomputer by Electronic Product Associates of San Diego. It was based on the Motorola 6800 microprocessor and came with integral numeric keypad input and LED output. The system was controlled by the John Bug monitor program contained in a 512 word PROM. It was similar to the Heath ET-3400 that was introduced in 1976. With it I was able to learn about machine language and a little about assembler. I later donated the computer to "The Computer Museum in Boston" where it now resides.

In 1977, I bought an SWTP 6800 (Southwest Technical Products) that was built from a kit (although not by me). This was a popular computer, and I found many people with whom I could share information. The operating system was on a PROM and the I/O was via a cassette tape player. I got a BASIC interpreter and began to learn BASIC programming. Some years later I bought a floppy disk drive that used an operating system by Smoke Signal Broadcasting (SS DOS). Much software was available for that operating system until the IBM PC came out. After that, new software became very scarce. The SWTP computer was used jointly by me and my then 13 year old son. He is now a programmer with NASA at the Ames Research Center. Our joint interest in computers helped to maintain a strong bond between us.

It was eight years before I switched to the Heath/Zenith, and during that time I learned about machine language, assembly level and BASIC programming. I also accumulated many game programs, a simple data base program, a sophisticated check ledger program (written in BASIC), and a complicated, but powerful, line editor and text processor. In 1984, I decided to buy an IBM compatible computer because I could no longer find support for my computer. I could diagnose problems, but had trouble finding someone who could help repair them. The computer was no longer fun to use.

The Z-151 was delivered with a CGA monitor, 360K of memory and two 360K floppy disk drives. From an ad in REMark Magazine, I learned about the ZP640 Plus Rampal ROM from FBE Research. Using this ROM I could, with 256K memory chips, increase the memory on the main memory board to 704K. I also installed a 20 meg. half-height hard drive in the space underneath the two half-height floppies.

In the June, 1986 issue of REMark Magazine, I read an article by Dante Bencivengo: "H-150 Speed-up Modification." I contacted Mr. Bencivengo and obtained and installed his modification kit. I could then run my computer at 4.77 MHz or 7.38 MHz. In another issue of REMark Magazine, an article by Joseph Katz described the HUG dual screen utility, "DS.COM," written by Jim Buszkiewicz. I installed a Zenith Data Systems high-resolution monochrome video board and connected it to a high-resolution monochrome monitor. With DS.COM I could run both monitors simultaneously, the CGA monitor was used for graphics and the monochrome monitor for word processing. After doing word processing on a high-resolution monitor, it was impossible for me to go back to a CGA monitor. It was the same with the speed-up mod; running at 4.77 MHz seemed glacial.

Each addition made the computer more enjoyable to use, and as I got used to each new feature, I realized how hard it would be to do without. By now the computer was in use daily. I don't have a secretary and I use the word processor for all my correspondence and professional writing. Wine collecting is another hobby, and I use PC-File:dB to keep track of my wine cellar inventory. An accounting program keeps track of all household and office expenses, and my collection of game and utility programs has continued to expand.

I became intrigued by articles on the various '286 accelerators, but the prices seemed too high for the benefits offered. When the Intel Inboard 386/PC was announced, my interest was piqued, and when mail order prices (PC Connection (800) 243-8088; Jameco Electronics (415) 592-8097) dropped to $600, I decided to buy.

The installation was tricky. For some reason in the Heath/Zenith, there has to be 256K of memory physically present on the RAM board (even though this memory is not used). Then the switch on the CPU board has to be set to indicate a total memory of 640K! The ribbon cable that runs from the CPU board to the Inboard is somewhat stiff and requires some effort in folding it properly so that the boards can be lined up. I found it easier to line the boards up first and then plug them into their respective slots simultaneously. The first time I booted up I got an error message. It turned out that I had to upgrade the monitor ROM. The monitor ROM I was using was V.2.3. I am now using V.3.0. The ROM came on two chips: p/n 444-229-16 and p/n 444-260-16, and I ordered them directly from Heath/Zenith Parts Dept. at (616) 982-3571.

There were no other snags or hitches. David Caranci, President of the National Inboard 386/PC Users Group, has an installation manual specifically for the Z-151 that I found helpful. He can be reached at (203) 567-5188 (voice) or (203) 567-3331 (BBS). Installation of the software was straightforward, and I found the Intel installation manual to be excellent.

The Inboard comes standard with a 16 MHz 80386 chip and 1 MB of memory. Additional options include an 80307 chip and piggyback memory boards providing up to 4 meg. of additional high-speed memory. Do not be misled about the amount of memory. All original system memory must be disabled or removed. Of the 1 MB of memory provided by the Inboard, 640K is used for system memory. 128K is reserved for internal functions, such as caching the system BIOS (rewriting the system BIOS into high-speed memory considerably speeds up all BIOS operations). Only 256K is left for use as extended memory. I use this 256K of memory for the hard disk cache.

The performance improvement is amazing. The Norton SI went from 2.6 (1.8 without the Bencivengo speed-up mod) to 16.9. The more modest PC Tools rating went from 155% to 790%. Most impressive has been the improvement in the performance of my Seagate ST-225 hard drive.

Before the Inboard, I used a disk caching program provided by Zenith Data Systems (ZCACHE.SYS) with a 256K cache in expanded memory using an AST RAMpage! board. With the Inboard, I use the ICache program provided by Intel (ICACHE.COM, it is a subset of Super PC-Kwik from Multisoft) with a 256K cache in the extended, high-speed memory. Using the PC Magazine Benchmark program for testing disk access speed, there is an 86% improvement in performance using ICache over ZCache! (Both programs are a big improvement over no cache.)

There is a lot of confusion between extended memory and expanded memory. What is important for the ZDS user is that the expanded memory on the AST RAMpage! (or the Intel Above Board) is slow, 8-bit memory. It can be used for programs such as DESQview or Windows, but it is too slow to be very useful. The Inboard has high-speed, 32-bit extended memory. (This can be converted into expanded memory, but I won't go into that here.) The AST boards and the Intel Above Boards can be used along with the Inboard, but they are best used for print spooling and RAMdisks. For programs that swap data into expanded memory, however, one must use either the Above Board or the Inboard. One cannot use both.

My Z-151 computer can now run most, but not all, software written for the 80386 machine. However, to run most 80386 software, a piggyback memory expansion board is essential. According to information posted on the Intel Technical Support Bulletin Board, (503) 645-6275, DESQview 386 runs without difficulty and is quite popular. Windows/386 requires a special version of the program called: "Windows/386 for the Inboard PC." This is a complete Windows 386 package and it also requires that one of the piggyback memory boards be installed. Windows/386 is very popular with Inboard PC users.

Other programs of note include Concurrent DOS 386 which is not compatible; PC-MOS 386 which is now compatible with release 3.0; VM/386 which is not compatible, but has announced that a compatible version will soon be available; and OS/2 which is not compatible and is not going to be compatible. Intel has made it very clear that they do not expect to support OS/2 in the PC environment.

If there is an OS/2 in your future, you will have to replace your '151. For me, however, tomorrow is here today. As long as ZDS continues to provide upgraded monitor ROMs and REMark Magazine continues to publish these excellent articles, I expect to have this computer for a long time to come.

Products Mentioned

Eastman Kodak Company
Verbatim 6.6 MB Internal Subsystems
Mass Memory Division

Continued on Page 48

dBASE III

D.R. Cool
7421 Troy Manor Road
Dayton, OH 45424

Part 3

In this article, I am going to discuss the creation of custom data entry screens and simple programming. As you know from using the EDIT and APPEND commands, the standard data entry screen has the field names aligned on the left side of the screen. In the normal start-up mode, dBASE III displays the field names as light characters on a dark background with the fields in inverse video. This limited mode of editing a data base can be improved on by using a special text file called a format file. A format file consists of only @...SAY...GET commands. With a format file you can design your own screen layout, including borders, messages and more descriptive prompts than the field names. Also, you have control over what fields will be edited and in what form the data may be entered. With date and numeric fields, you can even restrict the permissible range of values.

Format files can be written with any word processor or text editor that can produce ordinary ASCII text or, you can use the built-in text editor provided with dBASE III. To invoke the dBASE III editor, type MODIFY COMMAND followed by the name of the format file. When you use the dBASE editor for writing format files, be sure to include the extension .FMT in the file name; otherwise, the editor will give it the default extension of .PRG. Let's use the editor to create a format file for the PROJECTS data base. At the dot prompt type:

```
MODIFY COMMAND PROJEDIT.FMT
```

You can now type the file as shown in Listing 1. The first three lines of Listing 1 begin with asterisks. These are comment lines and will be ignored by the dBASE III command processor. I have made it a habit of starting every program or format file with the name of the file, a brief description of the purpose and the author and date written. Also, when I revise a file, I use comment lines to keep note of the date of revision, what was revised and why. The liberal use of comments within a program is never a bad idea, especially if you or someone else has to revise or debug a program months or years later.

As stated before, a format file consists only of @...SAY...GET commands. The general syntax as used in Listing 1 is:

```
@ row,col SAY expression GET variable
                          PICTURE clause
```

Row and col are numeric expressions. For a 25×80 screen, row can vary from 0 to 24 and column from 0 to 79. Expression is a text string of your own choosing. The expression must be delimited with single or double quotation marks or square brackets. The GET option displays information obtained from the data base; therefore, the GET variable must match one of the field names. Finally, the PICTURE option is used to control the format of the output and restrict how data may be entered. The PICTURE clause may consist of a function and/or template. If a function is used, the @ symbol must appear as the first character in the clause. Some functions are restricted according to data type. For example, the functions C, X, (, B and Z can be used with numeric data only. The functions A, ! and R apply only to character data.

Functions can be combined: XC displays DB after negative numbers and CR after positive numbers.

Whereas a function applies to the GET variable as a whole, a template is used to define each character of the GET variable. As an example of a template, using the command:

```
@ 1, 0 SAY AMOUNT PICTURE "9,999,999.99"
```

would cause the value 123456 to be displayed as "123,456.00". Functions can be combined with templates. For example, combining the XC functions with the previous template example:

```
@ 1, 0 SAY AMOUNT PICTURE "@XC 9,999,999.99"
```

results in the value 8734 displayed as "8,734.00 CR" and -12980 displayed as "12,980.00 DB". Note the required space between the function and the template. Another example is the use of the "R" function to format a data entry such as a telephone number. The telephone number is stored as the field TELENR in a data base as an unbroken 10 digit character string "7142381209", but you want the screen display to appear as "(714)-238-1209". You can accomplish this using the command:

```
@ 10,15 SAY "Telephone: " GET TELENR
         PICTURE "@R (999)-999-9999"
```

The "(", ")" and "-" characters are displayed on the screen, but are not stored in the data base, a savings of 4 bytes per record. The best way to see the effect of combining the various function and template symbols is to simply experiment.

The first non-comment line of Listing 1 uses only the SAY option to display the screen heading. The next line includes a prompt for the project drawing number, GETs the contents of the DWGNR field and formats the output using the template "9999-99999". This template allows only digits for the first four and last five positions, inserting a hyphen at position 5. Note that in this instance, since the "R" function is not used, the hyphen becomes part of the actual data. This template prevents non-digit characters from being entered into the DWGNR field. The third @ statement combines a function

Listing 1

```
* PROJEDIT.FMT
* FORMAT FILE FOR PROJECT UPDATE AND INQUIRY
* WRITTEN BY:  D.COOL        11/15/89

@  1,21 say "DRAWING PROJECTS - UPDATE AND INQUIRY"
@  3, 8 say "Drawing number: " get DWGNR picture "9999-99999"
@  3,38 say "Rev: " get REV picture "@! A"
@  3,48 say "Proj. nr: " get PROJNR picture "@! 9999-A9999"
@  4, 8 say "Engineer: " get ENGINEER picture "@! AAA"
@  6,12 say "Start date: " get STARTDATE
@  6,38 say "Est. compl. date: " get ESTDATE
@  7,12 say "Edit-in date: " get EDITIN
@  7,38 say "Edit-out date: " get EDITOUT
@  8,12 say "Approval date: " get APPDATE
@  8,38 say "Document date: " get DOCDATE
@ 25, 8 say "Comments: " get COMMENTS picture "@!"
```

Listing 2

```
* XREFEDIT.FMR
* FORMAT FILE FOR PROJECT CROSS REF UPDATE AND INQUIRY
* WRITTEN BY: D.COOL 11/15/89

@  2,20 say "PROJECTS CROSS REF - UPDATE AND INQUIRY"
@  5,20 say "Project Nr: " get PROJNR picture "@! 9999-A9999"
@  7,20 say "Device type: " get DEVTYPE picture "99"
@  7,38 say "Vendor PN: " get VENDPN picture "@!"
```

with a template. The @! function converts any lowercase input into uppercase, while the "A" template symbol permits entry of letters only. The fourth @ statement, which displays the project number, also combines a function and template. The effect of this PICTURE clause is — allow digits only for the first four and last four characters, allow letters only for the sixth position and insert a hyphen for the fifth position. The @! function is an added control to allow only uppercase for the letter at position six.

When you are done typing the format file, save it to disk by pressing CTRL-W or CTRL-END. Now you are ready to test the file. At the dot prompt type:

```
SET FORMAT TO PROJEDIT
```

(The SET FORMAT TO command assumes the extension .FMT.) Now, open the project data base with USE PROJECTS. Next, enter the command CLEAR to clear the screen, then type EDIT. If you typed Listing 1 exactly as printed, dBASE III should now be responding with the error message:

```
"SAY/GET position is off the screen"
```

I introduced a deliberate error in the listing to illustrate a point. Press ENTER to cancel the operation then type:

```
MODIFY COMMAND PROJEDIT.FMT
```

and correct the last @...SAY statement to read as follows:

```
@ 10, 8 say "Comments: " get COMMENTS
                           picture "@!"
```

When you have made the correction, save the corrected file to disk with CTRL-W or CTRL-END. Once again enter the following sequence of commands:

```
USE PROJECTS
CLEAR
EDIT
```

You should now see the same syntax error that you thought you just corrected! Why should this occur? Think a minute about how your computer operates. Programs and files are stored on disk, but they operate only from RAM. The corrected copy of PROJEDIT.FMT has been written to disk, but the original copy is still in RAM and this is the copy that dBASE runs. To get the corrected version into RAM you have to repeat the SET FORMAT TO PROJEDIT statement.

When you have created the format file from Listing 1 and corrected all errors, you can create a similar format file for the projects cross reference from Listing 2. As you did with the projects format file, test the cross reference format file with the SET FORMAT TO and EDIT commands until you are satisfied that the program works correctly.

Now that we have two format files for editing the projects and cross reference data bases, we can develop a short program that will facilitate the editing process and minimize the number of commands that must be entered at the dot prompt. A dBASE III command program, in its simplest form, is nothing more that an ordinary text file consisting of a sequence of dBASE III commands. DBASE III programs have the extension .PRG. To run a program from the dot prompt, you type:

```
DO program name
```

dBASE will then read the program line-by-line and execute each command exactly as if it had been entered at the dot prompt.

When you write dBASE III programs, is is a good idea to adopt certain notational conventions. My own convention has been to use lowercase for all command words and uppercase for file names, field names and memory variables because the memory variables and field names stand out when you read the program. Also, I spell out command verbs in full, although at the dot prompt you need only type the first four characters of a command. For delimiting character strings, I use double quotation marks ("), although you can also use single quotation marks (') or square brackets ([]). The one thing you can't do is start a character string with one delimiter and end it with a different one.

Extra spaces for readability can be used anywhere within a program line. The same goes for blank lines to separate function units of a program.

As stated in the discussion of format files, comment lines begin with an asterisk (*). They can also begin with the word NOTE. Comments can also be placed at the end of a command line if preceded by "&&". I use comments at any point in a program that I feel the logic of a programming statement may not be clear just from reading the program. If you ever have to revise or debug a program months later, you might be wishing for even more comments!

dBASE III programs can be written with any word processor or line editor that can produce ordinary ASCII text; that is, text without special control codes such as tab stops and page breaks. For your convenience, dBASE III provides a simple built-in editor which you invoke by typing:

```
MODIFY COMMAND program name
```

If you do use the dBASE editor, be aware of certain limitations:

1. You are limited to a maximum of 5000 characters.
2. It has no search and replace command.
3. It has no block copy, move or delete functions.

If you choose to use an external editor or word processor, there are three ways to go:

1. You can QUIT dBASE III and invoke your editor or word processor at the DOS system level. (If you have insufficient memory, you may be forced to do this.)
2. You can invoke your editor at the dot prompt using the RUN command:

   ```
   RUN editor/WP command name
   ```

   For example, to run Multimate, you would type RUN MM. If you use this option, you must have enough additional memory over and above the 256K required for dBASE III to load COMMAND .COM, plus the editor or word processor you are using, plus whatever memory you think you will need for your program.
3. You can invoke your editor/WP with MODIFY COMMAND.

But, you say, doesn't this invoke the dBASE III editor? Ordinarily, yes. However, if you have a favorite editor that you wouldn't think of being without and you are certain you will always be using it to write your dBASE III programs, you can get dBASE III to always load your editor when you use MODIFY COMMAND. Here's how.

When you invoke dBASE III, it always looks for a file in the current directory called CONFIG.DB. If it does not find this file, it starts up in a default configuration. In this default configuration, when you type MODIFY COMMAND, it loads its own internal editor. If, on the other hand, a CONFIG.DB file exists, dBASE III will modify its default setup according to the instructions in the CONFIG.DB file. One of the modifications you can make to the default setup is to substitute your own editor or word processor for the internal editor. Suppose, for example, you had an editor called MYEDITOR which you wanted to substitute for the dBASE III editor. Creating a CONFIG.DB file with the single line:

```
TEDIT = MYEDITOR
```

will cause dBASE III to load MYEDITOR whenever you use MODIFY COMMAND. As with the RUN command, you must have sufficient memory beyond the 256K minimum required by dBASE III. Also, if your editor program is not located in the default directory or one of the directories in your PATH statement, you must include the full path name in the TEDIT statement.

To create the CONFIG.DB file, type

the following at the dot prompt:

```
MODIFY COMMAND CONFIG.DB
```

Now type the TEDIT statement with the command name of the editor or word processor you wish to use. Finally, type CTRL-W to write the file to disk. Now you can QUIT dBASE III and re-enter. The next time you use MODIFY COMMAND, your own editor will be loaded. One other thing you will have to remember — if your word processor or editor supplies its own default extension (such as .DOC), you will always have to rename your program with the .PRG extension in order to make it recognizable to dBASE III. You can do this at the dot prompt with the command:

```
RENAME filename.DOC TO filename.PRG
```

The program we are going to start with is shown in Listing 3. At the dot prompt, type MODI COMM PROJUPDT. Type Listing 3 and save the file to disk.

This program uses a few commands not yet discussed. The first of these, SET TALK OFF, prevents the response to many dBASE III commands from being displayed on screen. SET TALK is normally ON. Therefore, if you don't want dBASE III responses to destroy a screen display, you must include the command SET TALK OFF somewhere at the start of your programs.

The next command, CLEAR, simply clears the screen. The next series of commands, the @...SAY commands, were discussed in the development of the format files. The new command, ACCEPT, has the syntax:

```
ACCEPT prompt TO memvar
```

This command is used primarily in command files to prompt the user for keyboard entry. The variable prompt is a character string delimited by single quotes, double quotes or brackets. The variable memvar stands for memory variable. A memory variable is a named data item that stores a value outside of the data base structure, providing a convenient means of temporary storage of variables. Memory variables can be of four types: character, date, numeric and logical. Memory variable names can be up to 10 characters long and can contain letters, numbers and underscores. They must, however, begin with a letter. Thus, BALANCE_3 is a valid memory variable but 3_BALANCE is not.

A character memory variable can contain up to 254 characters. Numeric variables are generally used to store numbers that may be used in calculations. The other two types of memory variables, date and logical, are not normally used with the ACCEPT command. When used with the ACCEPT command, the memory variable is always a character type.

The DO CASE command is our first encounter with a genuine programming structure. The various programming structures of dBASE III are what gives dBASE III its power, almost to the point of equaling the power of a high level language, such as BASIC or Pascal. The DO CASE command is used in the situation where you want the program to perform one of several functions dependent upon certain selection conditions. It is ideal for a menu type program in which you wish for certain functions to be performed dependent upon the selection made by the user in the form of key input. The general syntax for the DO CASE command is as follows:

```
DO CASE
    CASE <condition>
        <commands>
    CASE <condition>
        <commands>
    ...
OTHERWISE
    <commands>
ENDCASE
```

In the PROJUPDT program, the condition is the value of the memory variable SELECT. If SELECT has the value "1", then all of the commands following 'case SELECT = "1"' up to 'case SELECT = "2"' are executed. If SELECT does not equal one of the specified values (the user presses "6", for example), then the commands following "otherwise" up to the command "endcase" are executed. If the OTHERWISE statement is missing and all CASE conditions are evaluated as false, dBASE III will execute the first command following ENDCASE. Every DO CASE statement must end with an ENDCASE statement if you want your program to execute properly.

Note the indentations in the program listing for the CASE conditions and commands. Note also that the ENDCASE statement is lined up with the DO CASE statement. While this is not a requirement for the program to run properly, this practice of indenting makes the logic of complex programs easier to follow than if all statements start at the left margin. This is particularly true if you have nested DO CASE structures or IF-ENDIF structures nested within DO CASE structures. I will be discussing more on this in future arti-

Listing 3

```
* PROJUPDT.PRG
* A SIMPLE PROGRAM TO UPDATE THE PROJECTS/CROSS REF DATA BASES
* WRITTEN BY:  D.COOL       11/18/89

set talk off
clear              && CLEAR THE SCREEN
* PAINT MENU DISPLAY:
@  5,26 say "PROJECTS MAIN MENU"
@  7,21 say "1. Edit projects data base."
@  8,21 say "2. Append projects data base."
@  9,21 say "3. Edit cross reference data base."
@ 10,21 say "4. Append cross reference data base."
@ 11,21 say "5. Exit program."

* GET INPUT FROM USER:
accept "Make selection: " to SELECT
do case

    case SELECT = "1"
        use PROJECTS index PROJDWRV
        set format to PROJEDIT
        edit

    case SELECT = "2"
        use PROJECTS index PROJDWRV
        set format to PROJEDIT
        append
    case SELECT = "3"
        use PROJXREF index PROJXPJN
        set format to XREFEDIT
        edit

    case SELECT = "4"
        use PROJXREF index PROJXPJN
        set format to XREFEDIT
        append

    case SELECT = "5"
        close databases
        set talk on
        clear

    otherwise
        ? chr(7)
        wait "Invalid selection -- Press any key to continue"

endcase
```

cles.

To finish the discussion of the PROJUPDT program, the commands following 'case SELECT = "5"' and 'otherwise' need a bit of explanation. The first of these, CLOSE DATABASES, closes all open data base, index and format files. SET TALK ON simply resets TALK back to its normal default state. Whenever you use SET commands within a program to modify the dBASE III environment, it is always a good idea to use whatever SET commands are necessary at the termination of a program to return dBASE III to its default state. You can determine the default state of dBASE III by checking the dBASE III manual for each SET command or by typing SET at the dot prompt when you first start up dBASE III. DBASE III will then respond with a series of screens showing you the state of all the SET commands, the screen attributes, function key assignments, default drive and search path, alternate, format and index files, left margin and number of decimal places for numeric variables.

The ? command (yes, it is a command!) evaluates and displays at the current cursor position the value of the expression which follows it. For example, typing "? 100/4", which translates into "display the result of dividing 100 by 4", will result in the value 25.00 being displayed. In the case of the command statement "? chr(7)", chr is a function which converts a number to a character. The numbers are the ASCII code values for the keyboard characters. For example, a capital A has the ASCII code of 65; therefore, the command "? chr(65)" would display an "A". ASCII code 7, which is a carry over from teletype machines, rings the bell on most computer systems. The command "? chr(7)" is used to sound the system bell. I use this command generally only for invalid keyboard input, such as an invalid menu selection.

The WAIT command, which immediately follows the "? chr(7)" statement, causes dBASE III processing to pause until a key is pressed. It has the syntax:

```
WAIT prompt TO memvar
```

It's syntax is almost identical to the ACCEPT command, the main difference being that for WAIT, the memory variable is an option. Another difference is the WAIT will process the first key struck without waiting for an ENTER. The prompt, as for ACCEPT, may be any character string. If no prompt is specified, dBASE III displays the prompt "Press any key to continue...". If you want your program to pause without any prompt, you can use the form:

```
WAIT""
```

In this case, the prompt is a null string.

The overall result of the commands following "otherwise" in the DO CASE structure is:

1. System bell sounds;
2. "Invalid selection" message is displayed on screen;
3. Program waits for user to press a key.

We are now at the point where we can test the PROJUPDT program. At the dot prompt, type DO PROJUPDT. Assuming you made no typing errors, the screen should clear and you should see a menu. Also, the program should be prompting you to make a selection. If you now press "1" followed by ENTER, the screen should display the edit screen with the data for the first record in PROJECTS. This is exactly what you would get had you typed the following sequence of commands at the dot prompt:

```
USE PROJECTS INDEX PROJDWRV
SET FORMAT TO PROJEDIT
EDIT
```

While you are in the EDIT mode, you can bring up the next record by pressing PgDn or retrieve the previous record by pressing PgUp. You can exit from the edit mode by 1) pressing ESC or 2) pressing CTRL-END. Option 1 abandons any changes made to the current record whereas option 2 saves the changes. If you exit from the edit mode, you will be immediately back to the dot prompt, which means that you have exited from the program. At this point, the only way to bring back the menu is to rerun the program with DO PROJUPDT. If you continue to test this program with each of the five valid selections, you will always exit back to the dot prompt once the EDIT or APPEND process is terminated. (As a matter of fact, you will exit back to the dot prompt even if you make an invalid selection.) While this may be more convenient than typing a series of commands and having to remember the names of all the data base, index and format files, the program as written would hardly be considered to be user friendly. A normal menu-driven program should return you to the calling menu after completion of the selected function. A user would exit from such a program only when selecting the "exit" function. Something is obviously missing from this program. That something is the DO WHILE structure. This command and other programming structures will be discussed in Part 4 of the series. Also, we will develop a validation program to check on the accuracy of the data entered into PROJECTS. ✱

Continued from Page 44

343 State Street
Rochester, NY 14650
(716) 724-588
Tech. Support: (619) 587-4831

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

Mirror III
Softklone Distribution Corporation
327 Office Plaza Drive
Tallahassee, FL 32301
(904) 878-8564

Fastback Plus
Fifth Generation Systems, Inc.
11200 Industriplex Boulevard
Baton Rouge, LA 70809
(504) 291-7221

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

DESQview 386
Quarterdeck Office Systems
150 Pico Boulevard
Santa Monica, CA 90405
(213) 392-9851

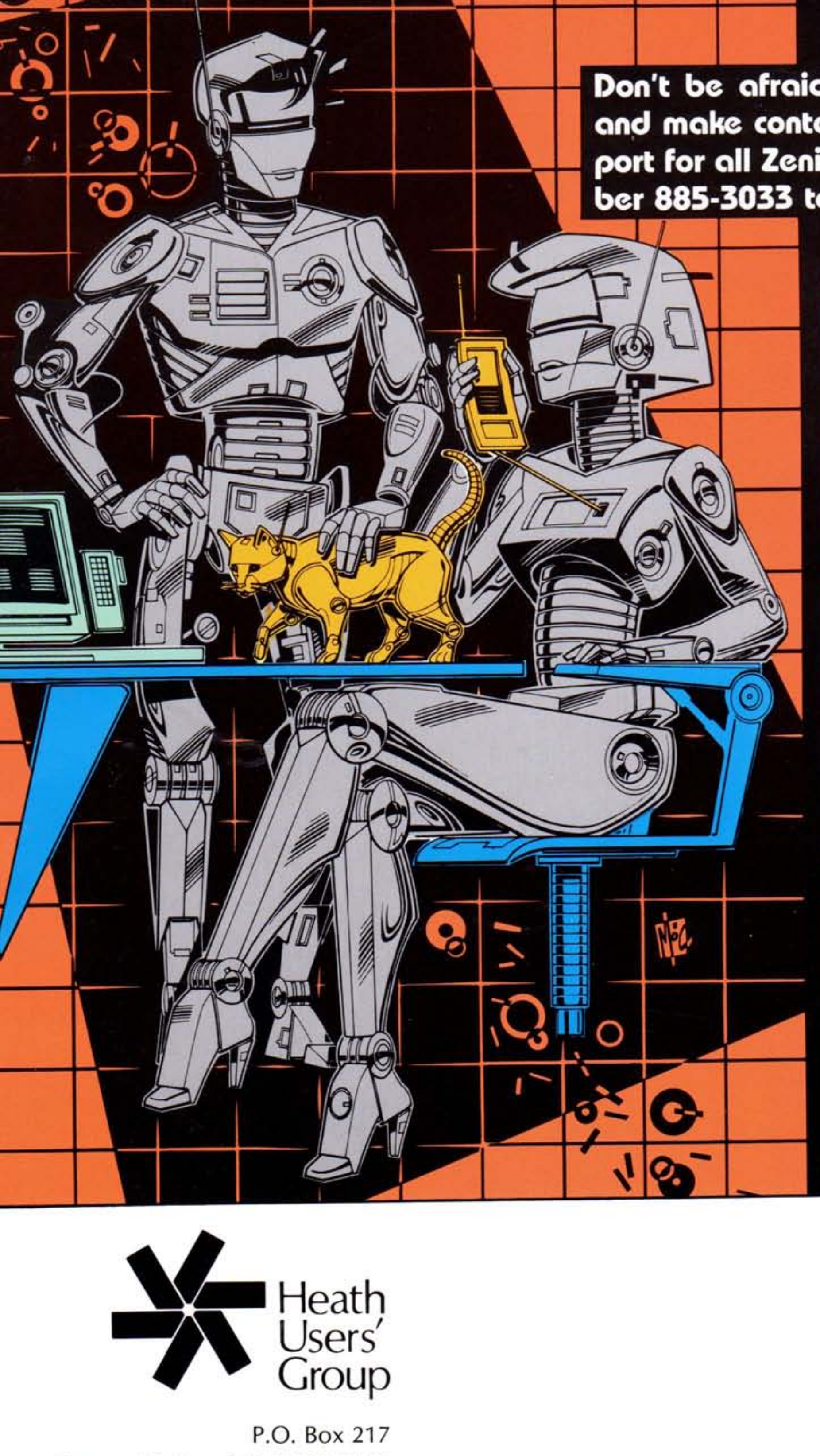
Microsoft Windows/386
Presentation Manager for Personal Computers
Using an Intel Inboard 386/PC
Microsoft Corporation
16011 N.E. 36th Way
Redmond, WA 98073
(800) 426-9400 ✱

HADES II

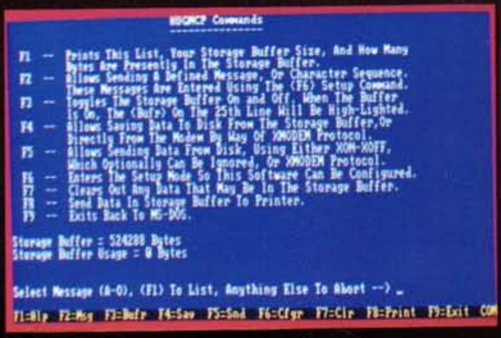
It's HOTTER than ever! Jam-packed with new features, HADES II still remains the easiest-to-use disk editor ever! Just look at some of the features:

- 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

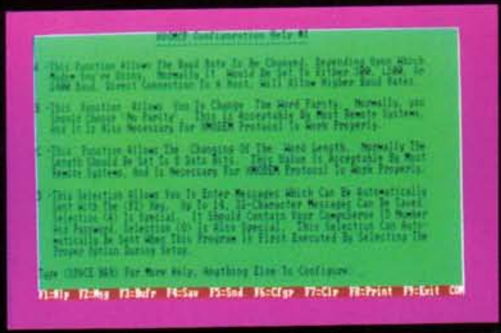
HADES II is still only $40, and original HADES owners can upgrade their distribution disk for only $15. Call HUG today at: (616) 982-3463.



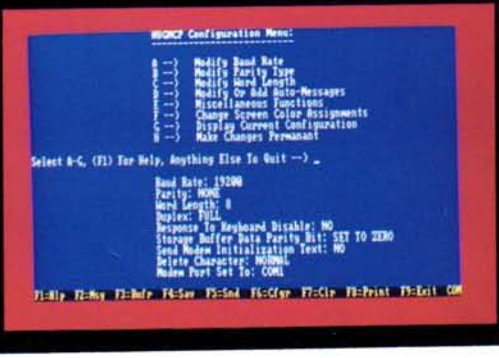
Don't be afraid to communicate! Get HUGMCP and make contact the easy way. Now with support for all Zenith Laptops, order HUG Part number 885-3033 today.



```
                    HUGMCP Commands
                    ---------------

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

Storage Buffer = 524288 Bytes
Storage Buffer Usage = 0 Bytes

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



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

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

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

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

D -This Selection Allows You To Enter Messages Which Can Be Automatically
   Sent With The (F2) Key. Up To 15, 32-Character Messages Can Be Saved.
   Selection (A) Is Special. It Should Contain Your CompuServe ID Number
   And Password. Selection (O) Is Also Special. This Selection Can Auto-
   matically Be Sent When This Program Is First Executed By Selecting The
   Proper Option During Setup.

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



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

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

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

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

Heath
Users'
Group

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

$2.50

P/N 885-2124