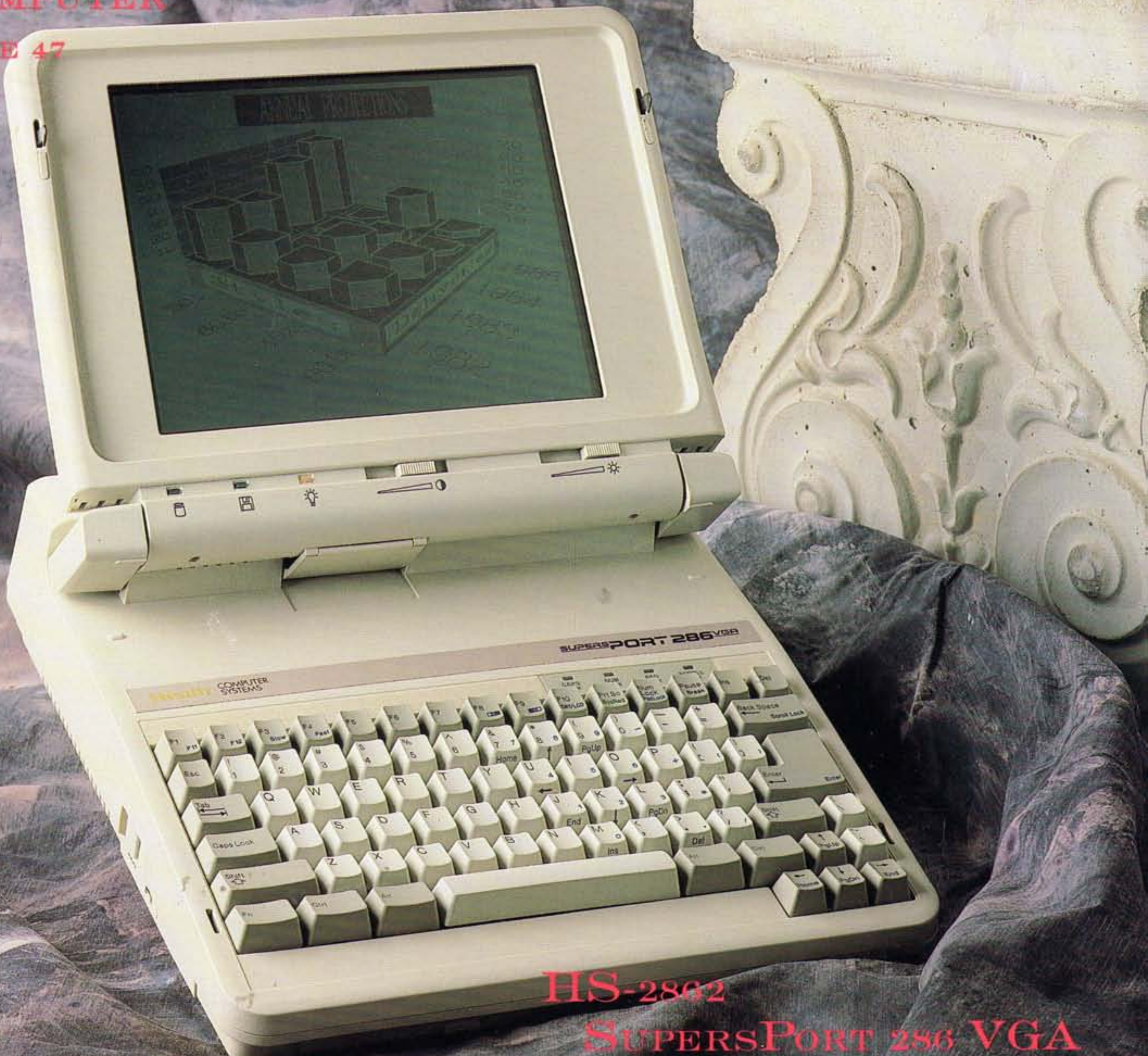
The Official Heath/Zenith Computer Users Magazine

REMark®

February 1990

Diagnostics for a Heath/Zenith Personal Computer
Page 47



HS-2862
SupersPort 286 VGA
Page 7

"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 2 • February 1990

Want New & Interesting Software? Check Out HUG Software

Reader Service No.		Page No.
196	Covox, Inc.	11
197	DMA Technologies, Inc.	24
104	FBE Research Co., Inc.	17
136	Lindley Systems	11
117	Payload	34
193	QuikData, Inc.	4
188	Serendipity Associates	43
149	WS Electronics	40

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

PC Compatibles

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

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

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

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

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

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

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

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

Contributing Editor...... William M. Adney

Contributing Editor...... Robert C. Brenner

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

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

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

*U.S. Funds

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

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

HUG

H8 - H/Z-89/90

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

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

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
ADVENTURE	885-1222-[37]	CPM	GAME	10.00
BASIC-E	885-1215-[37]	CPM	LANGUAGE	20.00
CASSINO GAMES	885-1227-[37]	CPM	GAME	20.00
CHEAPCALC	885-1233-[37]	CPM	SPREADSHEET	20.00
CHECKOFF	885-8011-[37]	CPM	CHECKBOOK SOFTWARE	25.00
COPYDOS	885-1235-37	CPM	UTILITY	20.00
DISK DUMP & EDIT UTILITY	885-1225-[37]	CPM	UTILITY	30.00
DUNGEONS & DRAGONS	885-1209-[37]	CPM	GAMES	20.00
FAST ACTION GAMES	885-1228-[37]	CPM	GAME	20.00
FUN DISK I	885-1236-[37]	CPM	GAMES	20.00
FUN DISK II	885-1248-[37]	CPM	GAMES	35.00
GAMES DISK	885-1206-[37]	CPM	GAMES	20.00
GRADE	885-8036-[37]	CPM	GRADE BOOK	20.00
HRUN	885-1223-[37]	CPM	HDOS EMULATOR	40.00
HUG FILE MANAGER & UTILITIES	885-1246-[37]	CPM	UTILITY	20.00
HUG SOFTWARE CATALOG UPDATE #1	885-4501	VARIOUS	PRODUCTS 1983 THRU 1985	9.75
KEYMAP CPM-80	885-1230-[37]	CPM	UTILITY	20.00
MBASIC PAYROLL	885-1218-[37]	CPM	BUSINESS	60.00
NAVPROGSEVEN	885-1219-[37]	CPM	FLIGHT UTILITY	20.00
REMARK VOL 3 ISSUES 24-35	885-4003	N/A	1982	20.00
REMARK VOL 4 ISSUES 36-47	885-4004	N/A	1983	20.00
REMARK VOL 5 ISSUES 48-59	885-4005	N/A	1984	25.00
REMARK VOL 6 ISSUES 60-71	885-4006	N/A	1985	25.00
REMARK VOL 7 ISSUES 72-83	885-4007	N/A	1986	25.00
SEA BATTLE	885-1211-[37]	CPM	GAME	20.00
UTILITIES BY PS	885-1226-[37]	CPM	UTILITY	20.00
UTILITIES	885-1237-[37]	CPM	UTILITY	20.00

Price List

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
X-REFERENCE UTILITIES FOR MBASIC	885-1231-[37]	CPM	UTILITY	20.00
ZTERM	885-3003-[37]	CPM	COMMUNICATION	20.00
H/Z-100 (Not PC) Only				
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 6 ISSUES 60-71	885-4006	N/A	1985	25.00
REMARK VOL 7 ISSUES 72-83	885-4007	N/A	1986	25.00
REMARK VOL 8 ISSUES 84-95	885-4008	N/A	1987	25.00
SCREEN DUMP	885-3043	MSDOS	UTILITY	30.00
UTILITIES II	885-3014	MSDOS	UTILITY	20.00
Z100 WORDSTAR CONNECTION	885-3047	MSDOS	UTILITY	20.00
PC Compatibles				
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.

Magazines everywhere, and no way to reference the wealth of information they hold? Not anymore! Now there's ***MAGBASE***; a database designed specifically for referencing magazine articles. Don't let those one-hundred-and-some back issues of REMark, or C Users Journal, or Veternary Medicine, (or any magazine) gather dust, use ***MAGBASE***, and *find* that article you read two years ago! ***MAGBASE*** is available for ***MSDOS HUG P/N 885-3050*** or ***CP/M (P/N 885-1249-[27]***.

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

Make the no-hassle connection with your modem today! ***HUGMCP*** doesn't give you long menus to sift through like some modem packages do. With *HUGMCP*, YOU'RE always in control, not the software. Order ***HUG P/N 885-3033-37*** today, and see if it isn't the easiest-to-use modem software available. 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.

QUIKDATA - 13 YEARS OF H/Z SUPPORT!

For all your H/Z 8-bit and PC/XT/AT needs

MEMORY UPGRADES

Note: All memory upgrades come without memory chips. Call for current chip pricing. Memory chip prices have come down. Example: 256K 150ns DRAM is $2.95 as of this printing.

Z150MP - $19 Will allow you to upgrade your H/Z150/160 to up to 704K on the main memory board, using up to 18 256K DRAM chips.

MEGARAM - $43 Upgrades your H/Z150160 series with up to 704K of main memory, and about 512K for RAMDRIVE memory. Includes documentation, software RAMDRIVE disk, PAL and jumper wire. For the full 1.2 megs total memory, 45 256K DRAM chips are required.

ZMF100 - $53 Will allow you to upgrade your H/Z110/120 (old motherboards; with p/n less than 181-4918) to 768K system RAM. Requires 27 256K DRAM chips.

Z100MP - $76 Similar to ZMF100 above, but for new motherboards with p/n 181-4918 or greater.

WINCHESTER UPGRADE KITS

Note: All hard drives are thoroughly tested before they are shipped out. Hard drive upgrade kits are tested as a complete unit with drive, controller and cables.

PCW20 - $289 Complete winchester setup for a H/Z150, 148, 158, 159, 160, PC etc. Includes 21 meg formatted half-height Segate ST-225 65ms drive, WD WX1 winnie controller board, cable set, documentation.

PCW30 - $349 32 meg with 38ms Segate ST-138.
PCW40 - $409 42 meg with 37ms Segate ST-251.
PCW80 - $649 80 meg with Segate ST-4096 full size drive.

We also have the DTC controllers and daughter board expansions to place a hard drive in the H/Z148 computer.

BARE WINCHESTER DRIVES

ST-125 - $265 37ms 3.5" drive in 5" frame with 21 megabyte storage capacity. Good in low power applications especially for Z100 and Z148 computers.

ST-138 - $295 37ms 3.5" drive in 5" frame with 32 megabyte storage capacity. Good in low power applications especially for Z100 and Z148 computers.

ST-225 - $225 65ms half height bare winchester with 21 megabyte storage capacity.

ST-251 - $325 37ms autopark half height bare winchester with 42 megabyte storage capacity. Good choice for -AT compatibles.
ST-251-1 - $349 Same as ST-251 but 28ms speed.

ST-4096 - $595 28ms autopark full height bare winchester with 80 megabyte storage capacity. Best choice for -AT compatibles.

ANY DRIVE IN YOUR PC/XT/AT

With the CompatiCard, you can install up to four additional drives, of any type in your PC/XT/AT computer. Add a 1.2 meg 5" floppy, or a 1.44 meg 3.5" floppy, or any other drive, including 8" to your system. The CCARD2 will handle up to two drives, and the CCARD will handle up to 4 drives. Additional cables and external enclosures may be required.
CCARD2 - $89
CCARD - $109

COMPATICARD IV is a new high speed one megabit per second data transfer rate card to support the new 2.8MB 3.5" drives and high speed tape drives. On-board ROM allows booting from any attached drives. All drives supported, up to four drives on a card.
CCARD4 - $139

DOES YOUR COMPUTER KNOW THE TIME?

We carry the SMART CLOCK for any PC/XT computer, including the Z100 series. Simply plug it in the system ROM socket, and plug the ROM into the clock module. Self contained sealed lithium battery for 5-10 year life. Includes software for the clock/calendar.
SMARTCLK - $35 (add $2 for Z100 installation for spacers)

ZENITH COMPUTER SAMPLE

Zenith AT STARTER. Start with this basic unit and build your own. Includes 12Mhz Z248 (80286 processor) with two serial ports, one parallel port, 1MB RAM, 5" 1.2 meg floppy, no video card. If you desire we can install a 42mb 28ms hard disk for an additional $345, or a 80MB 28ms hard disk for an additional $589. VGA card for $189, EGA card for $119.
Z24812S1 - **$1495** (starter system)

Zenith 386 STARTER. Similar to above, but this one features the 20Mhz 80386 processor. $2895

Zenith Z159 8Mhz, 640K RAM, 101 keyboard, EGA video, dual floppies.
Z159-S1 - $1295 ($1549 for single floppy 21mb hard disk unit)

Zenith SupersPort 184 Laptop with 20MB hard disk - $2145

Zenith MinisPort notebook computer with 1MB RAM - $1395

Call or write for a quote on any Zenith computer system.

OTHER STUFF

Quikdata also carries spike protection filters, backup power supplies, modems, printers, disk drives, drive enclosures, cables and connectors, laptop batteries, monitors, memory cards, memory chips and ICs, joysticks, accessory cards, serial and bus mouse, the complete Zenith line of computers, parts and accessories, a wide variety of useful and most popular software and much more!

8-BIT

We carry a full line of hardware and software products for the H8/H89/90 computers. This includes diskettes, printers, H89 working boards and parts, some H8 parts, etc. Refer to our other ads in this issue.

H-SCOOP NEWSLETTER

Of course, don't forget about the only independent source of Heath/Zenith related information you can obtain - our monthly newsletter, **H-SCOOP**! Just $24 for a 12-issue year ($28 Canada, $35 foreign), it will help you get the most from your computer investment. Get sound technical advice, helpful hints, find out what the problems are, fixes, reports, reviews, information from other subscribers, classifieds and much more.

Call or write in to place your order, inquire about any products, or request a free no obligation catalog. VISA and Master Card accepted, pick up 2% S&H. We also ship UPS COD and accept purchase orders to rated firms (add 5% to all items for POs). All orders under $100 add $4 S&H. Phone hours: 9AM-4:30PM Mon-Thu, 9AM-3PM Friday. Visit our bulletin board: (414) 452-4345

QUIKDATA, INC.

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

BUGGIN' HUG

Wants to Expand Z-171

Dear HUG:

I have a Z-171 Zenith portable PC computer expanded to 640K. The lunch bucket style.

I am looking for hardware that is compatible with the Z-171. It has an External Expansion Bus Port and I am wondering what current upgrade accessories are available for this port.

I am particularly interested in . . .
- an expansion chassis to take standard cards, etc.
- or an expansion chassis with winchester storage (hard disk) capabilities, external or internal.

Should anyone have any information on other products compatible with my computer, please send also.

Sincerely,
Larry Verheyden
B.P. 76 Postes Francaises
Andorra La Vella Principat D'Andorra
Europe Via France

The Evolution of DOS

Dear Mr. Brenner:

Thank you for the article "The Evolution of DOS". I really enjoyed reading this. In your article you mentioned Digital Research, but nothing was said about their version of DOS sometimes known as Easy DOS. Can you furnish a continuation of your article analyzing the difference between MS-DOS, PC-DOS and EASY DOS?

Thank you.

Yours very truly,
Malcolm H. Aukerman
Datarite, Inc.
350 Poplar Street
P.O. Box 67
Newport, IN 47966

Current Contents on Diskette

Dear HUG:

H/Z-100 (not PC) owners that need to search the scientific literature may be interested in knowing that Current Contents on Diskette works on the '100 under ZPC with no patches required.

Set PC to 7 (the monochrome mode) and it is off and running. You need either a hard drive or 8 inch drives to store approximately 1 Meg of information in each weekly issue.

You can get further information about demonstration disks from:
ISI Institute for Scientific Information
3501 Market Street
Philadelphia, PA 19104
CURRENT CONTENTS on Diskette (for IBM PC/XT/AT and compatibles).
J-600 and J-1200 Life Science editions
Agriculture, Biology & Environmental Sciences (AB&ES)
Current Book Contents

The interface is quick, slick and easy to learn. It will even fill out reprint request cards: addressing them on your Laser Printer.

Selected search scans can be printed in one of three different formats for your card files.

It works quicker than leafing through the weekly printed edition.

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

Can't Get X-10 to Work with XT

Dear Jim:

Wonder if "Buggin' HUG" might be able to help me.

Purchased an X-10 Powerhouse CP290 for a clone AT. Worked fine for years. Removed and placed in my new H2526-A. All I receive is (ERROR: CHECK THE POWERHOUSE CONNECTIONS). Tried everything I could, slowed the speed down, removed mouse board. Still the same problem. Now X-10 (USA), Inc. advises me their software will not work with an XT. Wondering if anyone has written new software for the XT.

X-10 advises they will write new software some time in 1990. But in the meantime . . .

Thanks much for your help and time.

Sincerely yours,
Frederick D. Jenke
Rt. 2 Box 373
Eureka Springs, AR 72632

News for Brenner

Dear HUG:

Received your December issue of REMark in the mail yesterday. Upon reading the article entitled "The Evolution of DOS" by Robert C. Brenner I discovered a slight error.

He states that PC-DOS 4.0 will only interface with IBM hardware and can only be installed on a hard disk formatted with PC-DOS. He also states DOS 4.0 will only recognize an IBM mouse or IBM printer.

Have I got news for Mr. Brenner. I not only installed DOS 4.0 on my Heath 386 with hard disk formatted with Zenith MS-DOS 3.3+ I also used my Microsoft Mouse and MPI SX Printer. The only thing I had to do before using the DOS 4.0 sys and replace commands was to use the Norton Disk Doctor to put an IBM signature on the hard disk. I also have the patches to correct the bugs.

I used it for about a month till I grew tired of the shell and went back to Zenith's MS-DOS 3.3+. PC-DOS became shelfware.

For those really wanting DOS 4 it can be done and quite easily. Try it you might like it — I didn't.

Since then I changed printers and am now using an Epson LQ 850 with Ultra Script PC. Ultra Script PC is a Postscript interpreter for non-Postscript printers. I think it does a rather nice job.

Sincerely,
Norman G. Harris
Box 141
Milan, MI 48160

Wants News!

Dear Jim:

I have been a subscriber to REMark since 1984, when I bought a Z-151. In 1988, I bought a SupersPort 286. I have read REMark in order to remain current with hardware and software developments in the computer world and in order to get news of developments related to the Z-151 and SupersPort 286. Even though I have enjoyed and benefited from reading REMark, and even though I recognize that you probably operate on a shoestring budget, I have usually been disappointed in at least one respect: you don't have a news column. What I and other readers would benefit from tremendously, and what you ought to be offering, is a news column on software and hardware; i.e., what are the latest developments in the computer world pertinent to Heath/Zenith users? For example: give us information (and maybe, but not necessarily, some analysis) about the Bull corporate takeover; provide announcements of new software and especially new hardware — add on equipment, new computers in the Zenith line, prices, and sources. On the latter point: I learned from browsing through a portable computer magazine at my local newsstand that there is a machine known as the SupersPort 286e. I don't recall any mention in your magazine of this machine. I called my local Heath/Zenith store to inquire and learned that the difference between a 286 and a 286e is that the 286e has VGA video, but that I can't retrofit my 286. Why didn't I learn about this machine from my HUG magazine? Also, why is it that Zenith makes it impossible to upgrade my 286 to a 286e? These are only two examples of what I and others believe to be a huge gap in your user group coverage. Please add a news column. It can't be all that difficult to do. Many of your articles are informative, but most are too wordy. Give me **news**, please.

A faithful, but disappointed friend,
Jeffrey P. Kimball
Department of History
Miami University
Oxford, OH 45056

Jeffrey, care to take on the job as anchorman? -Ed.

Harvard Graphics Works on Z-100

Dear HUG:

I recently bought a copy of Harvard Graphics for use at work. As a Z-100 owner, I was, of course, very interested to see if it would run under ZPC. In short, it does and it provides a very useful capability. It is, however, a large program requiring about 1.67 M of disk space.

In order to operate in the Draw/Annotate mode, it is necessary to patch the main program. The following addition to PATCHER.DAT can be used with PATCHER to make the necessary change.

```
HARVARD GRAPHICS 2.12
Insert the disk containing HG.EXE
4174,0,B0
417A,0,0
z
```

The instructions below from HG.EXE which reference the PC graphics table differ a bit from those described in the ZPC manual. I found them by looking for the sequence GE FA with the debug search function.

```
<segmnt>:4270  MOV WORD PTR [8768],F000
<segmnt>:4276  MOV WORD PTR [876A],FAGE
```

Harvard uses some CTRL-keypad key combinations for editing which are not implemented on the PC in the normal manner; and are, therefore, not supported by ZPC. These are CTRL-Down Arrow. Even though, I found three occurences of INT 16 in HG.EXE including what appears to be an undocumented use of function 2, I was unable to work out how HG recovers these key combinations.

Fortunately, Harvard has a macro feature which provides a simple work around. MACRO.COM is a TSR with which one can record keystrokes in ASCII files, replay a file, or edit a macro file. Using the copy con command, I created four one-key macros as follows:

Keypad Key	**Macro Text**
Delete (Z100 .)	<CtrlDEL>
Insert (Z100 0)	<CtrlINS>
Up Arrow (Z100 8)	<CtrlUP>
Dn Arrow (Z100 2)	<CtrlDN>

Sincerely,
Vern Reisenleiter
5216 Franconia Road
Alexandria, VA 22310

Discovered Problems

Dear Jim,

I discovered two problems with the Backup version 3.30.06 and Restore version 3.30.06 programs supplied with MS-DOS 3.3+ from Zenith. One: backups made on the 10th of any month other than October would always cause Restore.com to say that it was restoring files recorded on October 13th. Two: backup data sets made in the month of October could not be restored at all: Restore.com would simply quit with an unlisted error message that the backup data set was invalid.

Investigation uncovered the fact that the file "Backupid.@@@" was being written incorrectly. This file, which is written on every diskette in the backup data set, contains the creation date of the backup data set. Instead of the desired 0x0a for the month of October, it was found that an 0x0d followed by an 0x0a was being written out to the file. When Restore.com was run to restore the data set, the program checked this file for valid data to insure that a real member of the backup data set was being used. Naturally, it choked when it saw a 13 in the month field. Note that if the day was the 10th, Backup.com would write out an 0x0d in the day field, followed by an 0x0a in the month field, followed by the actual month's value in an unused field. This would cause Restore.com to display a date of October 13th, regardless of the actual month and day.

The temporary fix for this problem, which would allow the October backup data sets to be restored, was:

```
debug restore.com
-e 22e4
xxxx:22e4  0c.
                0d
-w
-q
```

This changes:

```
xxxx:22e0  80 3e f8 13 0c 7f 95 80 3e
   80 00 00 74 16
to:
xxxx:22e0  80 3e f8 13 0d 7f 95 80 3e
   80 00 00 74 16
```

What this patch does is to allow a month of 13 to pass inspection. When Restore.com is run, however, a nonsense message is displayed which says that the file was created on "unable to create or open log file". Ignore this message; the files will be restored correctly.

The correction of the problem requires that Backup.com be patched. This is done by:

```
debug backup.com
-e 4a87
xxxx:4a87  75.
              eb
-w
-q
```

This changes:

```
xxxx:4a82  f6 06 1b 11 80 75 04 c6 46
   fc 80
to:
xxxx:4a82  f6 06 1b 11 80 eb 04 c6 46
   fc 80
```

This patch corrects the problem by preventing a 0x0a (a newline character in text mode) from being translated into a 0x0d followed by a 0x0a (a carriage return followed by a linefeed). Apparently, the file was opened in text mode (where the translation is supposed to take place), as opposed to binary mode (where the data being written out is supposed to be left untouched), which is fine, unless one is trying to write out binary information. This patch fixes the problem for the Backupid .@@@ file without adversely affecting either the writing of text to the display or data to the diskette.

As an aside, it is interesting to note that Backup.com and Restore.com both work just fine on H/Z-100s, under MS-DOS 3.10.

Yours without the Bull (at least so far),
Bill Rothman
200 Hoffman Avenue, Apt. 204
Cranston, RI 02920

Trying to Convert to a Hard Disk System

Dear HUG:

I am a very frustrated writer attempting to convert from a floppy drive system to a hard disk system, but with one very major problem. The system is:

- Z-100 with 768K ram
- Dual disk drive and hard card
- MS-DOS version 3.01
- Lotus 1A, purchased from Zenith 1984
- Okidata printer Microline 193 plus

When this system works from the floppy disks everything works well. But when the hard disk is activated at the boot, then the following problems arise:

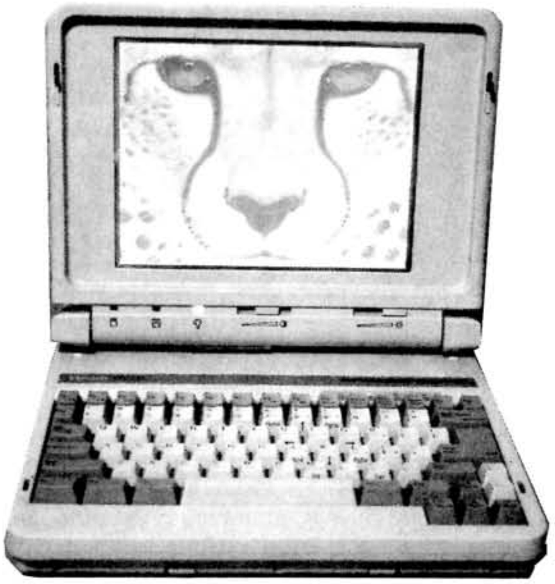
When using LOTUS 1-2-3 via the hard drive and not making any "set-up string" entries [/ppos (no entry)] everything prints as desired. But whenever anything is entered in the "set-up" string, then the only thing that prints to paper is the letter "n", in either normal or compressed print size.

Resetting the default settings changes nothing. If there is an entry in the set-up string everything else is overridden. After numerous attempts to make something different occur only frustration has won out.

Something is causing the printer not to receive the data for printing as displayed. Only the letter "n" is being output or deciphered by the printer.

Continued on Page 17

The HS-2862 SupersPort 286 VGA



A New Cat for the Road

Pat Swayne
HUG Software Engineer

Photo 1
The HS-2862 can display 320×200 256-color images in gray scale on its VGA-compatible screen.

Most of you probably know by now that Zenith Data Systems has introduced three new versions of its SupersPort and TurbosPort portable computers. All of these new models feature VGA-compatible screens that can display graphics with a resolution up to 640 × 480 pixels. As with the older SupersPort and TurbosPort models, Heathkit versions of these computers will soon be available.

I had the opportunity to assemble a "proof build" of the HS-2862 SupersPort 286 VGA, which is the Heathkit version of the SupersPort 286e. This machine is nearly identical to the original SupersPort 286 except for the new screen. Not only is it capable of higher resolution graphics, but it has a "page white" screen like the one originally only available on the TurbosPort. This new screen does a great job with CAD programs such as AutoSketch™ at the 640 × 480 resolution. But the really amazing thing about it (to me, at least) is the way it handles 320 × 200 256-color MCGA (Multi-Color Graphics Array) images. There is a routine built into the ROM that processes the 256 colors into the 16 levels of gray that the screen is capable of displaying, and it does a truly remarkable job of translating the colored images into black-and-white pictures. Photo 1 shows how a popular MCGA picture looks on the HS-2862.

Construction

Building the HS-2862 is as simple as building the original SupersPort kit, the HS-2860. Only mechanical assembly is required, with no soldering or other electronic assembly work. Photos 2 through 6 show the major assembly steps. The only tricky parts were routing the hard drive cable (the connector was drawn wrong in the pre-production assembly manual), fitting the keyboard in place (a tight fit), and fitting the cabinet halves together without pinching the video cable. The completed HS-2862 looks a lot like its ancestor except that the "lid" containing the LCD screen is a bit thicker, as you can see in the side view in Photo 7.

Setup and Operation

Those of you familiar with the original SupersPort 286 know that it has two separate Setup screens: one for portable operation, and one for operation with the ZA-3034 expansion box. The new SupersPort has three setup screens: one for battery operation, one for AC operation (without the expansion box), and one for operation with the expansion box. This arrangement allows users who run the computer on batteries and also on AC power, but without the expansion box, to set such things as hard disk power down and backlight timeout separately for each situation.

There is a new option on the setup menu — the screen size. The screen size can be set to normal, centered, partly stretched, or fully stretched. To explain what these options do, and the need for them, I will have to explain something about the various video modes supported on a VGA system.

A VGA display system can display a variety of graphic and text modes, because it is downward compatible with previous video systems. The graphic modes range from the original 320 × 200 pixel four color CGA graphic mode to the 640 × 480 16 color high resolution VGA mode. On a conventional CRT display such as the ZCM-1490 monitor, the low resolution modes are displayed using two "scan lines" on the screen for each horizontal graphic line so that the picture is actually "painted" using 400 scan lines. However, the picture on the screen is just as tall as when the system is operating in the 640 × 480 mode, in which there are 480 scan lines on the screen. The reason why the image on the screen is the same height for both modes is that the monitor automatically adjusts the space between scan lines. When the system is operating in the 640 × 350 EGA graphic mode, even more space is added between the scan lines, so that this mode also fills the screen. So much space is used in this mode, in fact, that you can easily see the individual scan lines on the screen.

The liquid crystal screen on the HS-2862 is capable of displaying the highest resolution VGA graphics mode, so there are 480 rows of pixels, and each row contains 640 pixels. These pixels are at fixed

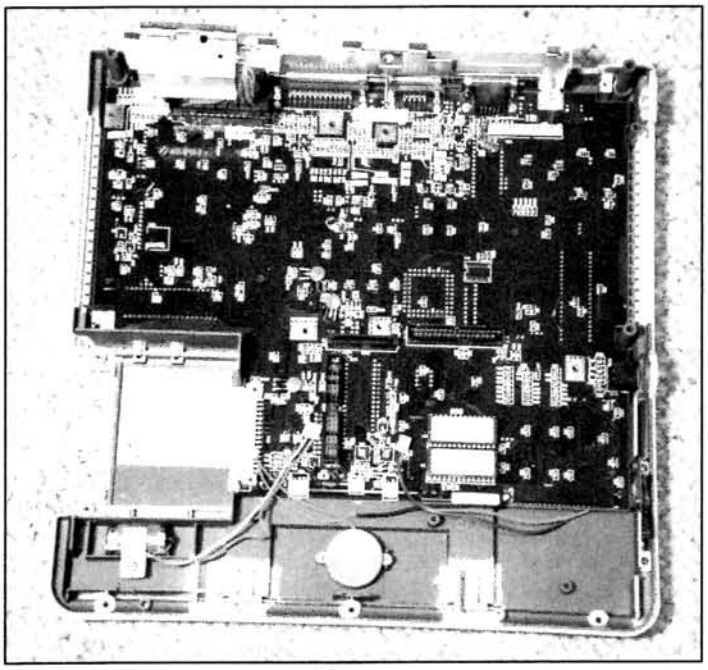


Photo 2
The first step in building the HS-2862 is to install the main circuit board in the cabinet bottom.

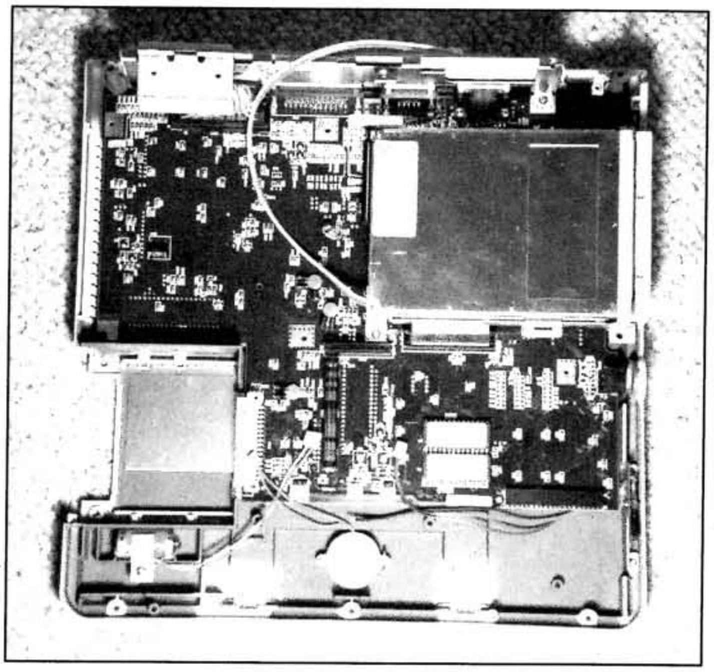


Photo 3
The floppy disk drive is mounted on the main circuit board.

locations on the screen, which means that the screen cannot adjust the space between "scan lines" when it displays the lower resolution modes. Instead, it normally just uses the number of lines required for the particular mode and leaves the rest of them blank (white). This is where the video size adjustment comes in. When the size is set to normal, the display always starts at the top of the screen, and any unused space is at the bottom. When the setting is "centered", the display is in the middle, with equal blank space at the top and bottom. The two stretch settings cause extra scan lines to be added at even intervals to partially or fully fill in the screen. These extra scan lines are simply copies of the lines above or below them (I'm not sure which). This method of filling the screen works out fine for some of the graphics modes. The display in Photo 1 was made with the video size set to fully stretched. Photo 8 shows the same display with stretching off, and the display centered. The number of scan lines added to the display is adjusted depending on which video mode is set. If the 640 × 480 mode is set, no extra lines are added because they are not needed.

The problem with the screen stretching method is that it looks lousy in the normal text mode. When a real VGA system is in the text mode, it forms the screen using 720 × 400 pixels. The characters are formed in boxes that are 16 pixels high and 9 pixels wide. The HS-2862 screen is only 640 pixels wide, so it cannot display the real VGA text mode. It

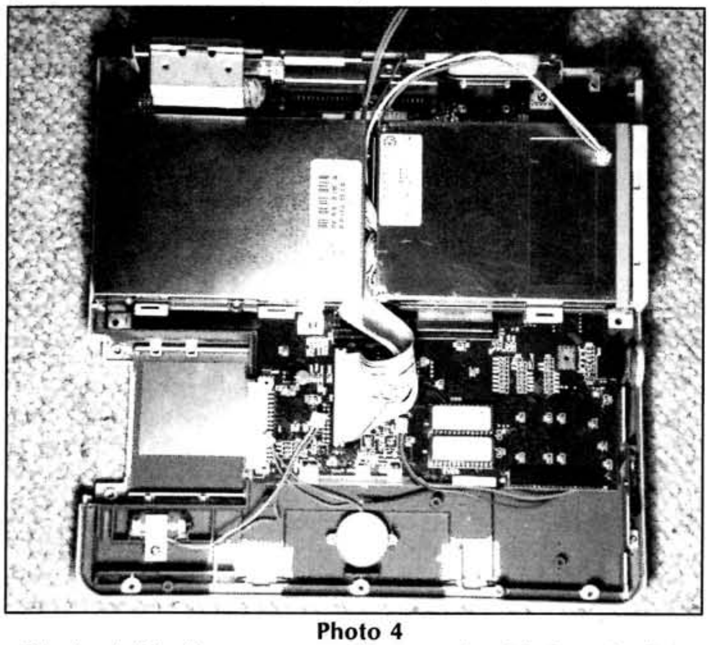


Photo 4
The hard disk drive mounts on the opposite side from the floppy drive. My hard drive cable is probably routed incorrectly, but the connector was drawn wrong in the pre-production assembly manual.

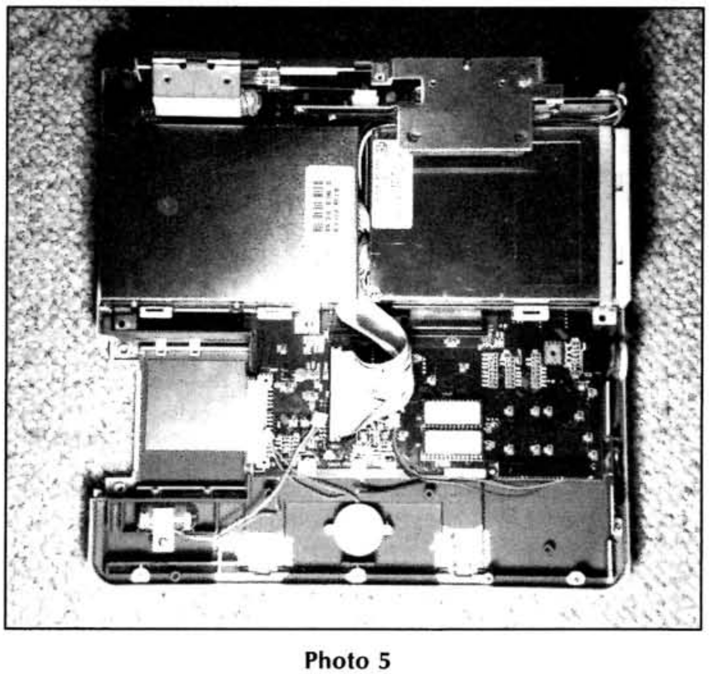


Photo 5
The power supply module mounts behind the disk drives. The power supply heat sink covers part of the floppy drive.

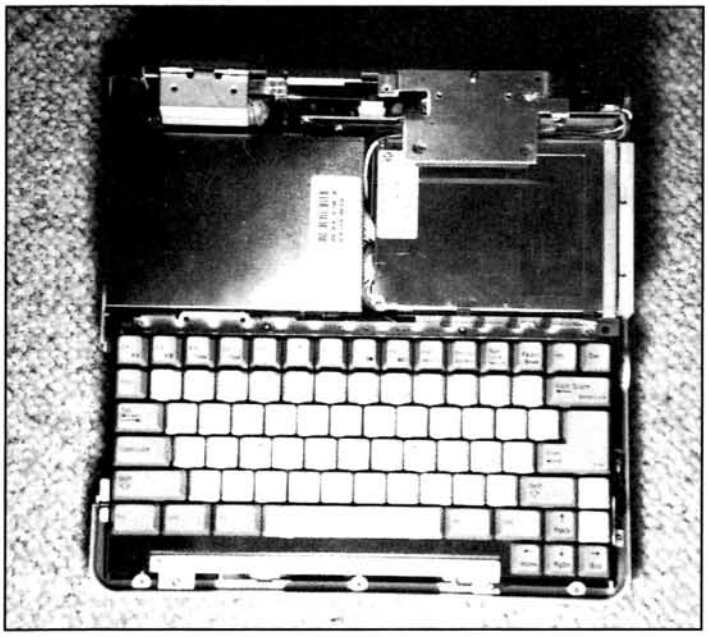


Photo 6
After you have mounted the keyboard, your new computer is nearly finished.

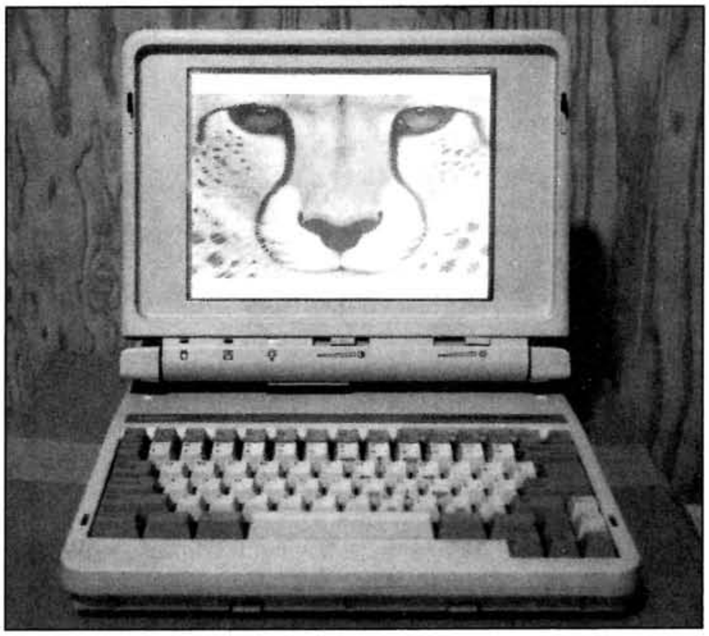


Photo 8
Here's how the image in Photo 1 looks with "stretching" turned off.

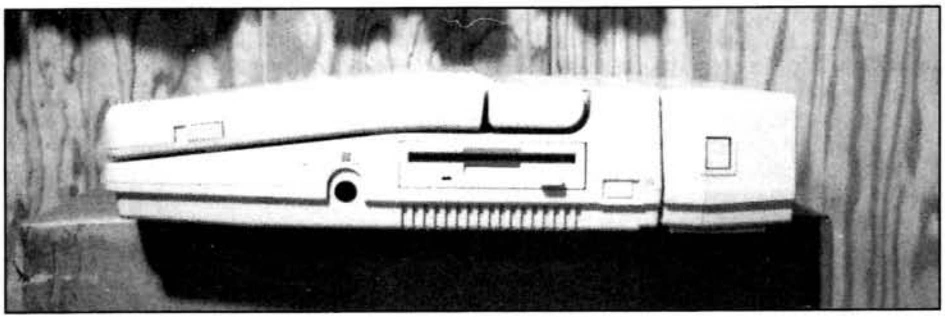


Photo 7
In this side view of the completed computer, you can see that the screen is thicker than the original SupersPort screen.

does the best it can by using character boxes 16 pixels high and 8 pixels wide, which means that 25 80-character lines will use up 640 × 400 pixels of screen space. (By the way, this is the same scheme used for text display on the original SupersPort 286, which has a 640 × 400 pixel screen.) That means that the normal text mode does not fill the screen, so when the stretch mode is on, extra lines are added. These extra lines can show up anywhere with reference to a particular character. The horizontal line in the middle of a lowercase "e", for example, can be fatter on one text line than on another. Like I said, the text mode looks lousy when stretching is on. I operate with it off except when I am going to run a graphics program that does not use the 640 × 480 mode, and I want the height to width ratio to be correct. Later I'll have to see if there is a way to change the stretch mode without having to run Setup.

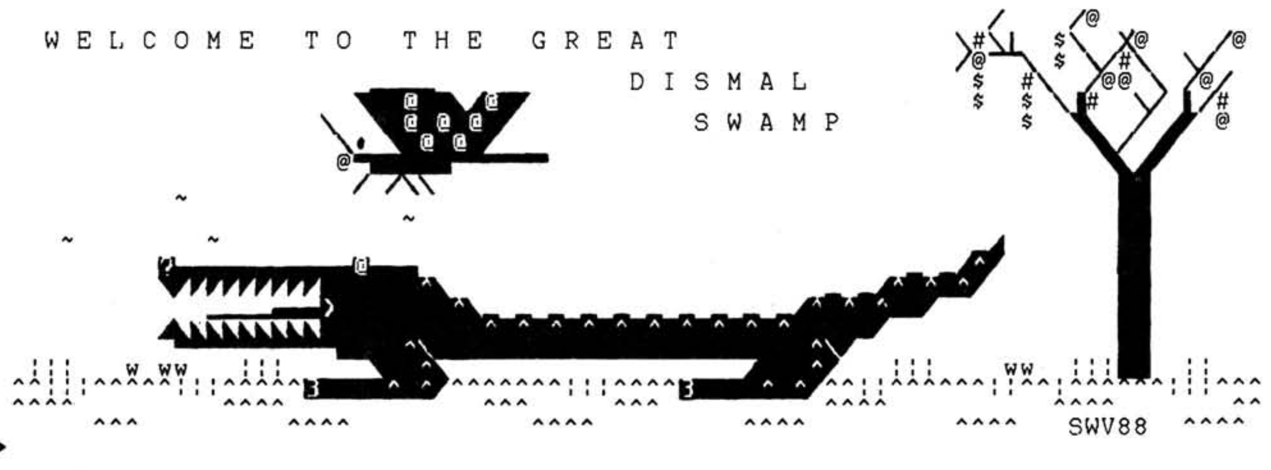
A VGA system can be programmed into text modes having more than 25 lines on the screen. For example, the NCC program that comes with the Norton Utilities™ can set up 40 or 50 lines on the screen. However, it still uses 400 lines on the screen for these modes. All that changes is the size of the character box. You'd think that the VGA designers would have made some text modes available that used all 480 lines.

Other than the screen changes (brighter, better graphic resolution, and sometimes not all of it used), the HS-2862 is like the HS-2860, and they both have the same "feel". One problem that my unit has (perhaps because it is a pre-production unit) is that the right side of the screen heats up more than the left side. After a period of use, the contrast becomes uneven across the screen. You may be able to tell from Photo 1 that the border around the screen is slightly wider on the right side, so my guess is that the screen driver circuitry is on that side, and that it is what generates the extra heat. The display (in fact, the entire top half of the computer) comes assembled, so I can only speculate as to what's in there. The heat sink for the power supply (in the main cabinet) is also on the right side, so it could be that heat rising from the power supply is partly or perhaps wholly the cause of the problem.

As I stated previously, the HS-2862 has software in its ROM that converts color images to gray scale when the screen is in the 320 × 200 256 color mode. This software does not seem to work under all conditions (but again, perhaps that is because this is a pre-production unit). For example, I have two programs that display GIF (Graphical Interchange Format™) images, such as the picture of the cheetah in Photo 1. One of these programs, called CSHOW, works properly on the HS-2862, but the other one, called VPIC, does not. Through experimentation, I found that the gray scale conversion works if images are placed on the screen by writing directly to video memory (which is how most programs would do it), but it does not work if images are placed using the BIOS (Basic Input/Output System) pixel writing routine. However, I doubt if that is why VPIC does not work, because it seems to be too fast to be using the BIOS routine.

I have also found that the LCD system is not 100 percent VGA compatible. A "real" VGA system can be programmed into other 256-color modes besides the 230 × 200 mode. For example, the CSHOW program can program it into 320 × 400 and 320 × 480 modes that display 256 colors. These modes do not work at all on the LCD screen. However, when an external color monitor is connected to the video connector on the back of the computer, all VGA modes work perfectly. I should mention that the video connector on the back supports only analog VGA-

Continued on Page 20



COLOR PAINT.ASM FOR THE H/Z-100!

We continue with a color version of "PAINT.ASM" for the H/Z-100!

As you recall, we discussed the first half of "PAINT100.ASM" in the December 1989 issue. This month, we pick up exactly where we left off. Reopen your program document in your favorite word processor and we'll get right to work.

You will notice that when we get into the RVIDC and GRAFC printer routines, we start discussing bit-image graphics. My article "PAINT.ASM Again?" in the December 1987 issue of "REMark" discusses this in depth. Here H/Z-89 owners should also take note.

As you may recall from that article, data at the end of that program contained the bit-image code for all the reverse video printer characters, the graphic printer characters, and the reverse video graphics printer characters. The RVIDC and GRAFC routines used the addresses RVCHAR, GCHAR, and RVGCHR and adjusted the number of the character to get the proper bit-image code, which was then sent to the printer.

Well, a close look at the graphics characters and the reverse video graphics characters shows the reverse video information is only the complement of the graphics character. For example, where the normal graphics character may be composed of '000011000', its reverse video counterpart is '111100111', or the one's complement of the initial data.

The 8080 instruction codes permit us to take the complement of the Accumulator thru the CMA instruction, giving us exactly what we need back in the Accumulator. Therefore, if we just received the bit-image code for the normal graphics character, complemented that code, then sent it to the printer, we would get the reverse video version of that character. PERFECT!

Now, some of you may be thinking that we could carry this one step further, by complementing any character we send to the printer to get its reverse video version. The kicker here is that the normal characters are part of the printer's normal character set and bit-image information is not used. To print a normal character, we send a byte of data to the printer, and the printer prints the character that is represented by that byte.

For example, the letter 'A' is represented by 41 Hex (decimal 65) in the ASCII conversion chart. Sending 41H to the printer will print the letter 'A'. Represented by binary '01000001', the complement would be '10111110' or BE Hex (decimal 235), which may represent some other character in the printer's character set — a script greater-than sign on my printer.

In short, it only works in the bit-image mode of the printer. If we defined the full normal character set, including the graphics symbols, in bit-image graphics, then we could easily get the reverse video of that entire set.

H/Z-89 owners can, therefore, modify these same routines and eliminate the RVGCHR character set, saving 2K of space.

LPTAB permits us to print our "PAINT" screen anywhere on a page — great for doing tasks, such as greeting cards, where the screen must be printed in the lower right corner of the page. It works equally well on the H/Z-89 version.

The routine accepts one or two numbers to describe the number of spaces that must be printed before a line of our "PAINT" screen is printed. Characters that would be printed beyond the right side of the paper are discarded. Note that the routine recognizes the print pitch (e.g., it will print 96 characters in Elite mode and 144 characters in the compressed mode). PRNTSP then prints the required number of spaces.

Disk operations haven't changed much from prior versions. My article in the March 1986 of "REMark" provides information on these routines. Changes have only been made to improve efficiency.

The RDSCN1 routine displays "PAINT100.SC1" as the opening screen and any time F9 or the "HELP" key is pressed.

PMSG and LPMSG routines permit us to print the messages (MSGs) on the screen or the printer. These MSGs provide an easy means to print line 25 information, ask questions, print status messages, and transmit ESCape codes to clear the screen, make the cursor visible or invisible, set or exit graphics or reverse video modes, etc. Information is sent for any number of lines, being terminated by a '$' symbol only.

The TDLY (time delay) routine slows a routine sufficiently to permit displaying screens on the monitor without missing characters. The DELAY routine permits the program to pause until any key is pressed. EXIT restores the CP/M flags and stack, returning us back to the operating system and A> prompt.

Following the MSGs, several flags are defined and memory space allotted to store their respective pieces of information. Space is also set aside for storing our mark positions, memory pointers (addresses) and a line input buffer. Our two file names containing our opening screen and menu screen are also defined here.

Steven W. Vagts
2409 Riddick Road
Elizabeth City, NC 27909

Finally, our bit-image printer codes are defined as RVCHAR and GCHAR. The codes are generally the same as those printed beginning on page 46 of the December 1987 issue of "REMark" and are not printed here to save space. Those characters that I've changed are listed.

Let's discuss these character codes a bit. One change to note right off the bat is that the graphics 'j' was changed as a marketing decision from a division sign in the H/Z-89 character set to a block character in the H/Z-100 character set. If you must have compatibility between the H/Z-89 and H/Z-100, this character can be changed back in the H/Z-100.

I preferred to change a few other characters, however, between the two character sets and I'll discuss these shortly.

The last few lines of the program establish the space for a buffer area and our stack. Now that you have the program typed in, what happens next?

Well, first save the program. Next, we need to construct our temporary opening screen file. Using your word processor again, open a file "PAINT100.SC1" and place a simple message in it: "This is PAINT100.SC1." Save this.

Assuming you use disk drives, place your program disk in Drive A. This should contain your assembly programs, ASM .COM and LOAD.COM. Place your data disk, with our "PAINT100.ASM" and "PAINT100.SC1" files, in Drive B.

Per the instructions in the CP/M manual, at the A> prompt, type "ASM PAINT100.BBZ", without the quotes. The program will assemble leaving "PAINT100 .HEX" on Drive B. Next, at the A> prompt, type "LOAD B:PAINT100", again without the quotes. This takes the .HEX file and produces "PAINT100.COM" on Drive B.

Using the PIP command, copy "PAINT100.COM" and "PAINT100.SC1" to Drive A and at the A> prompt, type "PAINT100" without the quotes. PAINT100 should load and display an opening screen saying "This is PAINT100.SC1."

Delete this message and type the information of Listing 2. Feel free to use reverse video, graphics, or color in making your version. On mine, I don't use graphics or reverse video, but use yellow on blue for the commands, yellow on black for the key words, and yellow on red for words indicating danger — such as ERASE. Let your imagination run wild.

To save your creation, use 'CTRL-W' (press "CTRL" key, followed by "W") and on line 25, following the question "WHAT IS THE FILE NAME?" type "B:PAINT100.SC1" without the quotes. This saves the file on Drive B. Save the file again, this time typing "PAINT100.SC1". This saves the file on Drive A.

Type "F0" and "HOME" and the screen will blank. Now do Listing 3 in the same manner as Listing 2. Save this file when done as "B:PAINT100.SC2" and "PAINT100.SC2". Clear the screen again and press "F5". The "PAINT100.SC2" screen should display. Pressing "Z" should erase it again.

That's all there is to it!

ALTCHAR.SYS

Several times, I stated that we could change the H/Z-100's character set — we could get it to talk Greek if we had to. If you recall the CP/M system disk when you received it, it had several "character" sets — "DANISH.CHR", "FRENCH.CHR", "SWEDISH.CHR", and "GRAPHICS.CHR" to name a few.

"ALTCHAR.SYS" contained the same data as the "GRAPHICS.CHR" file as the initial default. When stored in ".CHR" files, these fonts are inactive. However, when any of these files are copied to the file named "ALTCHAR.SYS", the font becomes active.

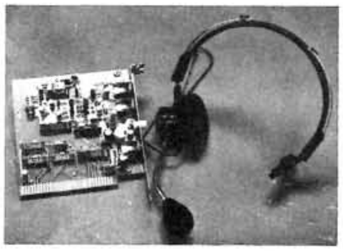
Appendix C of the CP/M manual provides some excellent material on this subject. Two previous articles in "REMark" also discusses these files, and making changes to them:

"A Z-100 Font Editor" by Marc Aagenas of Zenith Data Systems is published in the July 1983 edition. It presents

VOICE MASTER KEY®
VOICE RECOGNITION SYSTEM
FOR PC/COMPATIBLES & TANDY 1000 SERIES
A FULL FEATURED VOICE I/O SYSTEM

GIVE A NEW DIMENSION TO PERSONAL COMPUTING. . .The amazing **Voice Master Key System** adds recognition to just about any program or application. Voice command up to 256 keyboard macros from within CAD, desktop publishing, word processing, spread sheet, or game programs. Fully TSR and occupies less than 64K. Instant response time and high recognition accuracy. Voice recognition tool-box utilities are included. **A genuine productivity enhancer!**

SPEECH RECORDING SOFTWARE. . .Digitally record your own speech, sound, or music to put into your own software programs. Software provides sampling rate variations, graphics-based editing, and data compression utilities. Create software sound files you can add to macros for voice recognition verification response, **a complete, superior speech and sound development tool.**



SOFTWARE CONVERSION CODES. The **Voice Master Key System** operates a growing list of third party talking software titles using synthesized phonetics (text - to - speech) or digitized PCM, ADPCM, and CVSDM encoded sound files. **Voice Master Key system does it all!**

EVERYTHING INCLUDED. .**Voice Master Key System** consists of a plug-in card, durable lightweight microphone headset, software, and manual. Card fits any available slot. External ports consist of mic inputs and volume controlled output sockets. **High quality throughout, easy and fun to use.**

ONLY $149.95
COMPLETE

ONLY $89.95 FOR TANDY 1000 SL/TL MODELS SOFTWARE ONLY—REQUIRES TANDY BRAND ELECTRET MICROPHONE.

ORDER HOTLINE: (503) 342-1271 Monday-Friday, 8AM to 5 PM Pacific Time

Visa/MasterCard, company checks, money orders, CODs (with prior approval) accepted. Personal checks subject to 3 week shipping delay. Specify computer type and disk format (3½" or 5¼") when ordering. Add $5 shipping charge for delivery in USA and Canada. Foreign inquiries contact Covox for C & F quotes. 30 DAY MONEY BACK GUARANTEE IF NOT COMPLETELY SATISFIED. ONE YEAR WARRANTY ON HARDWARE.

CALL OR WRITE FOR **FREE** PRODUCT CATALOG

COVOX INC.
675 Conger Street, Eugene, OR 97402
TEL: 503-342-1271 • FAX 503-342-1283

Reader Service #196

FORM FILL-R &
FORM EDIT-R

Fill in pre-printed forms, or create your own and print on blank paper. Use a FORM FILL-R output file as input to your program. Or take your program's output files and print them on any form.

FORM EDIT-R creates form definitions. It handles:

- Placement of data entry fields on the data-entry screen and the printed form
- Attributes of each data field (including type — Text, Integer, Real, Dollars, etc.)
- Field formulas (for automatic calculations)
- Form characteristics (page length, margins, etc.)

Once you create the form definition, use the FORM FILL-R to fill in forms and print them, or save them to disk files for later recall and/or correction. A number of example form definitions are included.

FORM FILL-R and FORM EDIT-R run on any PC or compatible computer, and also on the Z-100. **$49.95** postpaid. Call or write for free catalog and information.

Lindley Systems **4257 Berwick Place, Woodbridge, VA 22192 (703) 590-8890**

Reader Service #136

a ZBASIC program "FONTED.BAS" that can create or modify "ALTCHAR.SYS" type files.

"Anatomy of Altchar.sys" by Steven Howard is published in the January 1986 issue and also discusses designing your own font.

I'm going to go through a brute force method briefly with you. Before we start, however, we need to know which characters we want changed and how. This is very similar to the procedures we discussed in my December 1987 article.

We set up a nine by eight matrix of zero's to represent the size of the character, then put ones where the character will be. Keep in mind that unlike our printer that prints the character a vertical column at a time, the computer prints each character a row at a time. We must put the character vertically, 9 rows vertical by 8 columns horizontal.

For example, in Appendix B of the Z-100 User's Manual the graphic 'c' and 'j' are:

```
'c' 0000 0000 = 00H  'j' 1111 0000 = F0H
    0000 0000 = 00H      1111 0000 = F0H
    0000 0000 = 00H      1111 0000 = F0H
    0000 0000 = 00H      1111 0000 = F0H
    1111 1000 = F8H      1111 1111 = FFH
    0001 1000 = 18H      0000 1111 = 0FH
    0001 1000 = 18H      0000 1111 = 0FH
    0001 1000 = 18H      0000 1111 = 0FH
    0001 1000 = 18H      0000 1111 = 0FH
```

I split the characters down the middle to show the pairing. The first hexadecimal numeral represents the first four binary digits and the second represents the second set of four binary digits.

So, to change a character, you just draw out a new one and calculate its series of hexadecimal equivalents. I've done this for four characters. I have little use for the graphics characters representing 'g', 'h', and 'k'. Instead, I wanted to move 'j's character to 'k', then make the following:

```
'g' 1111 1111 = FFH  'h' 0001 1000 = 18H
    1111 1111 = FFH      0001 1000 = 18H
    0001 1000 = 18H      0001 1000 = 18H
    0001 1000 = 18H      0001 1000 = 18H
    0001 1000 = 18H      0001 1000 = 18H
    0001 1000 = 18H      0001 1000 = 18H
    0001 1000 = 18H      0001 1000 = 18H
    0001 1000 = 18H      1111 1111 = FFH
    0001 1000 = 18H      1111 1111 = FFH

'j' 1010 1010 = AAH  'k' 1111 0000 = F0H
    0000 0000 = 00H      1111 0000 = F0H
    0101 0101 = 55H      1111 0000 = F0H
    0000 0000 = 00H      1111 0000 = F0H
    1010 1010 = AAH      1111 1111 = FFH
    0000 0000 = 00H      0000 1111 = 0FH
    0101 0101 = 55H      0000 1111 = 0FH
    0000 0000 = 00H      0000 1111 = 0FH
    1010 1010 = AAH      0000 1111 = 0FH
```

Knowing this, we can now modify our graphics character set. Since you may not want to modify "GRAPHICS.CHR" for some reason, we'll copy it to another file first:

```
A>PIP VAGTS.CHR=GRAPHICS.CHR
```

Now we use DDT to modify the file:

```
A>DDT VAGTS.CHR
```

Printing the file:

```
-D0100
```

Gives:

```
0100 FF FF 50 00 00 00 2C 32 32 2C 20 20 51 00 00 00
0110 1A 26 26 1A 02 02 59 00 00 00 22 22 26 1A 02 1C
0120 60 00 00 1C 3E 3E 3E 1C 00 00 61 FF 7F 3F 1F 0F
0130 07 03 01 00 62 18 18 18 18 18 18 18 18 18 63 00
0140 00 00 00 FF 00 00 00 00 64 18 18 18 18 FF 18 18
0150 18 18 65 00 00 00 00 F8 18 18 18 18 66 18 18 18
0160 18 F8 00 00 00 00 67 18 18 18 18 1F 00 00 00 00
0170 68 00 00 00 00 1F 18 18 18 18 69 00 08 3E 08 00
0180 3E 00 00 00 6A 00 00 08 04 3E 04 08 00 00 6B AA
0190 55 AA 55 AA 55 AA 55 AA 6C F0 F0 F0 F0 FF 0F 0F
01A0 0F 0F 6D 00 00 08 08 2A 1C 08 00 00 6E 00 00 00
01B0 00 0F 0F 0F 0F 0F 6F 00 00 00 00 F0 F0 F0 F0 F0
```

This is all we need, but type 'D' twice more to get the full file printed. The first column of the above block of data is the address of the first byte of data, then the rest of the row runs thru 16 addresses (e.g., the first line of data represents addresses from 0100 to 01FF Hex). Also, not shown on the right side of this data block is displayed the ASCII representation of these bytes of data. It will be on your printout.

The data above has a pattern. Starting at address 0120 you will find the number 60. Every 9 bytes you'll find numbers 61, 62, 63, etc., up to 80. Circle these numbers on your printout. These numbers identify the graphics characters in hexadecimal, except that the number is 2 more than the actual hexadecimal value — I don't know why.

Examining the data carefully, we can identify the old graphics value of 'g' following the value 69, at address 017A Hex. To change these values to the new ones, we use the 'S' command (Substitute). We then type in the new numbers following the old numbers (we type what is in quotes):

```
-S017A (we want to check we have the
                         right number)
017A 69 (CR - we have the right number)
017B 00 "FF"
017C 08 "FF"
017D 3E "18"
017E 08 "18"
017F 00 "18"
0180 3E "18"
0181 00 "18"
0182 00 "18"
0183 00 "18"
0184 6A (CR - etc)
```

We can continue right through all four characters or restart the substitution at any address. Do all the changes you want. When done substituting, type a '.' (decimal) and a carriage return, then at the '-' prompt, dump the data to the printer again:

```
-D0100
```

As before, follow with two more 'D's to print all the applicable data. The file runs from address location 0100 to 02FF Hex. To exit DDT, type a 'CTRL-C'.

At the A> prompt again, we need to save our work:

```
A>SAVE 2 VAGTS.CHR
```

Finally, copy our file to "ALTCHAR .SYS":

```
A>PIP ALTCHAR.SYS=VAGTS.CHR
```

Upon a new COLD BOOT, our new modified characters will be used in the "PAINT" program. Modify the "PAINT-100.ASM" data block to match the new H/Z-100 characters and the printer won't know the difference. Neat!

My only complaint is that there doesn't appear to be any way to activate the new character set without rebooting. If anyone has any suggestions, I'm all ears.

Closing

Well, that wraps up another article. I sure enjoy these discussions and hope you do too. I know I've said this before, but that should be it on "PAINT". I may try to convert the program to MS-DOS, but there are several excellent, though complex, graphics programs out there and this one satisfies my needs.

If anyone has any problems, or suggestions to improve this program, I'd be happy to hear from you. Please include a self-addressed, stamped envelope if you desire an answer — it helps my budget.

As last time, upon request, I would be happy to send anyone the source code for "PAINT89", "PAINT100", and "HELLO" for $8.00. I'll also send the finished products, ASM & COM files to all three programs, on disk for $15.00, if you include a preformatted disk with your request. Upgrades to prior customers will be $10.00. I now primarily use H/Z CP/M (H/Z-100) or CDR CP/M (H/Z-89) and double-sided, double-density extended, soft-sectored disks. However, I can still copy to the older 10 sector, single-sided disks. Please include a phone number in case I have problems.

Please feel free to distribute these programs to your friends as you see fit.

Here's a parting thought. Do you know that if there were more programmers, there would be fewer people on the streets at night. Think about it. 'Til I meet with you again, please take care.

Listing 1
PAINT100.ASM Cont'd.

```
RVIDC   CALL    BITIMG  ;BIT IMAGE GRAPHICS SET
        LDA     LPGFFG
        ORA     A
        JNZ     GRAF1   ;RVID GRAPHICS CHAR
        LXI     H,RVCHAR
        MOV     A,E
        SUI     01FH    ;SUBTRACT 31
        MOV     C,A     ;C=NUMBER OF CHAR
        JMP     ADJ1

RVGC    LDA     LPRVFG  ;RVID?
        ORA     A
        RZ              ;NOT RV
        MOV     A,E
        CMA             ;COMPLEMENT A
        MOV     E,A
        RET

GRAFC   CALL    BITIMG
GRAF1   LXI     H,GCHAR
        MOV     A,E
        SUI     05DH    ;SUBTRACT 93
        MOV     C,A     ;C=NUMBER OF CHAR
ADJ1    DCR     C
        JZ      ADJ3    ;COUNT ADJUSTED
        MVI     B,18
ADJ2    INX     H
        DCR     B
        JNZ     ADJ2
        JMP     ADJ1

ADJ3    MVI     B,18
ADJ4    MOV     E,M     ;E=CHAR BYTE
        LDA     LPGFFG
        ORA     A
        CNZ     RVGC    ;RVID GRAPHICS CHAR
        CALL    PRNTCH
        INX     H
        DCR     B
        JNZ     ADJ4
        LDA     PICAFG
        ORA     A
        RZ              ;0=PICA
        CPI     010H    ;16=ELITE
        RNZ             ;NOT PICA OR ELITE
        DCX     H       ;SEND LAST
        DCX     H       ; 2 BYTES AGAIN
        MOV     E,M
        CALL    PRNTCH
        INX     H
        MOV     E,M
        CALL    PRNTCH
        RET

BITIMG  PUSH    D
        LDA     PICAFG
        ORA     A
        JNZ     IMG1
        LXI     H,MSG19   ;BIT IMAGE SET - PICA
        JMP     IMG2
IMG1    LXI     H,MSG20   ;BIT IMAGE SET - NOT PICA
IMG2    CALL    LPMSG
        POP     D
        RET

LPTLS   LXI     D,MSG33   ;GET LS NUMBER
        CALL    MSG50
        MVI     C,CONIN
        CALL    BDOS    ;A=CHAR
        CPI     00DH    ;CR?
        JZ      QUEST   ;NO
        CPI     031H
        JZ      LS10    ;SET 10LPI
        CPI     032H
        JZ      LS8     ;SET 8LPI (GRAPHICS)
        CPI     033H
        JZ      LS7     ;SET 7LPI
        CPI     034H
        JZ      LS6     ;SET 6LPI (STANDARD)
        CPI     035H
        JZ      LS4     ;SET 4LPI
        JMP     QUEST

LS10    LXI     H,MSG28
        JMP     SETMSG

LS8     LXI     H,MSG29
        JMP     SETMSG

LS7     LXI     H,MSG30
        JMP     SETMSG

LS6     LXI     H,MSG31
        JMP     SETMSG

LS4     LXI     H,MSG32
SETMSG  CALL    LPMSG
SETMSG1 LXI     D,MSG16   ;SET MSG
        CALL    PMSG
        CALL    DELAY1
        JMP     QUEST

LPTAB   LXI     D,MSG45   ;GET TAB NUMBER
        CALL    MSG50
        MVI     C,CONIN
        CALL    BDOS    ;A=CHAR
        CPI     00DH    ;CR?
        JZ      QUEST   ;NO
        CALL    NBR     ;NUMBER?
        ORA     A
        JZ      QUEST   ;NO
        SUI     030H
        STA     TABLEN  ;SAVE #
        MVI     C,CONIN
        CALL    BDOS
        CPI     00DH    ;CR?
        JZ      SETMSG1   ;1 DIGIT #
        CALL    NBR
        ORA     A
        JZ      SETMSG1   ;1 DIGIT #
        SUI     030H
        STA     NUMBR   ;SAVE SECOND DIGIT
        LDA     TABLEN
        MOV     B,A
        LDA     NUMBR
ADD10   ADI     10
        DCR     B
        JNZ     ADD10
        STA     TABLEN  ;SAVE TAB #
        JMP     SETMSG1

LPTABR  LDA     TABLEN  ;GET TAB
        CPI     000H    ;ZERO?
        RZ
        MOV     B,A
PRNTSP  MVI     E,020H  ;PRINT SPACE
        CALL    PRNTCH
        DCR     C
        DCR     B
        JNZ     PRNTSP
        RET

NBR     MVI     B,039H  ;A=CHAR
        MVI     C,10    ;SET COUNT
NBR1    CMP     B       ;A=B?
        RZ              ;YES
        DCR     B
        DCR     C
        JNZ     NBR1
        MVI     A,0
        RET
```

```
CLRFCB  LXI     D,DFCB+1  ;BEGIN OF FCB
        MVI     B,31
        MVI     A,' '
CLEAR   STAX    D         ;CLEAR FCB
        INX     D
        DCR     B
        JNZ     CLEAR
        XRA     A
        STA     DFCB    ;0 DRIVE CODE
        STA     DFCB+12   ;0 EXTENT
        STA     DFCB+13
        STA     DFCB+14
        STA     DFCB+15   ;0 RECORD COUNT
        STA     DFCB+32   ;0 CURRENT RECORD
        RET

FILNM   CALL    CLRFCB
        LXI     D,MSG13   ;REQUEST FNAME
        CALL    MSG50
        LXI     D,LINEB   ;POINT TO LINE BUFR
        MVI     C,LINPUT
        CALL    BDOS    ;INPUT THE LINE
        LXI     H,LINEB+1 ;POINT TO # OF CHARS
        MOV     C,M     ;GET COUNT
        INX     H
        MVI     B,0
        DAD     B       ;GET END OF USED LINEB
        MVI     M,' '   ;STORE BLANK AS MARKER
        MVI     B,14
CKL     DCR     C
        JM      LN0
        JZ      LNGTH   ;LENGTH OK
        DCR     B
        JZ      LNERR   ;LENGTH TOO LONG
        JMP     CKL

LNERR   LXI     D,MSG9  ;FNAME TOO LONG
        JMP     LN1

LN0     LXI     D,MSG10   ;NO FNAME MSG
LN1     CALL    MSG50
        CALL    DELAY
        MVI     A,0
        ORA     A
        RET

LNGTH   MVI     B,8
        LXI     D,DFCB  ;POINT TO BEGIN OF FCB
        LXI     H,LINEB+1 ;POINT TO BEGIN OF LINEB
FILIT   INX     D
        INX     H
        MOV     A,M     ;GET LETTER OF NAME
        CPI     03AH    ;COLON?
        JZ      COLON   ;DRIVE # GIVEN
        CPI     02EH    ;DECIMAL?
        JZ      BLNK
        CPI     ' '     ;BLANK?
        JZ      EONM    ;END OF NAME
        CALL    MUC     ;MAP TO UPPER CASE
        STAX    D       ;STORE IN FCB
        DCR     B       ;CHAR WAS STORED
        JNZ     FILIT   ;LOAD ANOTHER
        INX     H
        MOV     A,M
        CPI     02EH    ;LAST CHECK FOR DECIMAL
        JZ      FILTYP
        CPI     ' '     ;BLANK?
        JZ      EONM    ;END OF NAME
        JMP     LNERR   ;TOO LONG

EONM    LDA     DFCB+1
        CPI     ' '     ;CHK FOR BLANK
        JZ      LN1
        RET

FILTYP  MVI     B,3     ;3 CHARS FOR FILE TYPE
        LXI     D,DFCB+9  ;POINT TO FCB TYPE
```

```
TYPLP   INX     H
        MOV     A,M
        CPI     ' '     ;CHECK FOR BLANK
        JZ      TYPDN
        CALL    MUC     ;MAP TO UPPER CASE
        STAX    D
        INX     D
        DCR     B
        JNZ     TYPLP
TYPDN   CPI     A       ;FORCE Z=1
        RET

MUC     CPI     'a'     ;MAP TO UPPER CASE
        RC              ;LESS THAN 'a'
        CPI     'z'+1
        RNC             ;MORE THAN 'z'
        SUI     32
        RET

COLON   LXI     D,DFCB+1  ;POINT TO BEGIN OF FNAME
        LDAX    D       ;GET LETTER OF DRIVE
        SUI     040H    ;SUBTRACT 040H TO GET DRIVE #
        DCX     D       ;GO TO BEGIN OF DFCB
        STAX    D       ;STORE DRIVE #
        MVI     B,8     ;RESET B
        LXI     H,LINEB+3 ;POINT TO BEGIN OF FNAME
        INR     C
        INR     C
        JMP     FILIT

BLNK    MVI     A,' '
BLNK1   STAX    D       ;STORE BLANK
        INX     D
        DCR     B
        JNZ     BLNK1
        JMP     FILTYP

OPNFIL  MVI     C,OPEN
        LXI     D,DFCB
        CALL    BDOS    ;TRY TO OPEN FILE
        RET             ;A=255 IF ERROR

WDISK   CALL    ESCJ
        LXI     D,MSG43   ;SET DEFAULT COLOR
        CALL    PMSG
        CALL    FILNM
        JZ      EMSG
        CALL    OPNFIL
        CPI     255
        JNZ     CONT    ;OPEN OK
        LXI     D,DFCB
        CALL    MKFIL
        JZ      EMSG
CONT    LXI     D,MSG6  ;SAVING TO DISK
        CALL    MSG50
        CALL    SAVE
        LXI     H,DMA
        SHLD    DMAPTR
        LXI     H,BUFR
        SHLD    BPTRL
NEXTLN  MVI     B,80    ;# CHARS IN A LINE
NEXTCH  LHLD    BPTRL
        MOV     A,M
        CPI     15Q     ;CR?
        JZ      ENDLP   ;IF CR WE ARE DONE - CLEANUP
        INX     H       ;INC BUFR
        SHLD    BPTRL
        PUSH    PSW     ;SAVE CHAR
        CPI     ESCAPE
        CZ      ESCCH
        POP     PSW
        CALL    DISKOUT
        JNZ     DKERR
        DCR     B
        JNZ     NEXTCH
        MVI     A,00DH  ;CR
        CALL    DISKOUT
        JNZ     DKERR
        MVI     A,00AH   ;LF
        CALL    DISKOUT
        JNZ     DKERR
        JMP     NEXTLN

MKFIL   MVI     C,MAKFIL
        CALL    BDOS     ;CREATE FILE
        CPI     255      ;A=255 IF ERROR
        RNZ
        LXI     D,MSG11    ;NO ROOM IN DIR
        JMP     LN1

DISKOUT LHLD    DMAPTR
        MOV     M,A      ;A TO DMA
        INX     H        ;INC DMA
        SHLD    DMAPTR
        XRA     A
        CMP     L
        JZ      TODISK   ;END OF DMA
        MVI     A,0
        ORA     A
        RET

TODISK  LXI     H,DMA
        SHLD    DMAPTR   ;RESET TO BEGIN
        PUSH    B        ;LINE CHAR COUNT
        MVI     C,WRTFIL
        LXI     D,DFCB
        CALL    BDOS
        POP     B
        ORA     A        ;0?
        RET

ENDLP   LHLD    DMAPTR
        DCX     H
        MVI     A,1AH    ;CTL-Z MEANING EOF
        MOV     M,A
        INX     H
        XRA     A
        CMP     L
        JZ      ENDFL
FILLZ   MOV     M,A      ;FILL BUFR W/ZEROS
        INX     H
        CMP     L
        JNZ     FILLZ
ENDFL   CALL    TODISK   ;WRITE OUT PADDED RECORD
        JMP     DKDONE

RDISK   CALL    FILNM
        JZ      EMSG
        CALL    OPNFIL
        CPI     255      ;A=255 IF ERROR
        JNZ     CONT1    ;OPEN OK
        LXI     D,MSG14    ;NO FILE IN DIR
        CALL    MSG50
        CALL    DELAY
        JMP     EMSG

CONT1   LXI     D,MSG7   ;RECALLING FROM DISK
        CALL    MSG50
RDDSK   CALL    RLOOP
        JNZ     UNSUC2
        JMP     DKDONE

RLOOP   LXI     D,DFCB   ;FCB ADDRESS
        MVI     C,RDFIL
        CALL    BDOS     ;READ A RECORD
        ORA     A        ;READ OK?
        RNZ
        LXI     H,DMA    ;POINT TO DMA
        MVI     B,128    ;PRINT 128 CHARS
PLOOP   MOV     A,M      ;GET CHAR
        CPI     'Z'-40H    ;CTRL-Z?
        RZ
        CALL    CHOUT
        INX     H
        DCR     B
        JNZ     PLOOP    ;LOOP UNTIL 128 CHARS PRINTED
        JMP     RLOOP

RDSCN1  LXI     D,MSG1   ;CLEAR SCREEN
        CALL    PMSG
        CALL    CLRFCB
        MVI     B,11
        LXI     D,DFCB+1  ;POINT TO BEGIN OF FCB
        LXI     H,PSCRN1  ;FILE NAME ADDR
        CALL    RDNAME
        JNZ     RDDSK    ;OPEN OK
        LXI     D,MSG21    ;PAINT100.SC1 NOT ON DISK
        CALL    PMSG
        JMP     EXIT

RDNAME  MOV     A,M
        STAX    D
        INX     D
        INX     H
        DCR     B
        JNZ     RDNAME
        CALL    OPNFIL
        CPI     255      ;A=255 IF ERROR
        RET

DKERR   LXI     D,MSG12
        CALL    MSG50    ;NO MORE ROOM
        CALL    DELAY
DKDONE  CALL    CLOSDK
        JMP     EMSG

NOREAD  LXI     D,MSG15    ;UNSUCCESSFUL READ
        CALL    MSG50
CLOSDK  MVI     C,CLOSE
        LXI     D,DFCB
        CALL    BDOS     ;CLOSE FILE
        RET

LPDONE  CALL    PCRLF
        JMP     EMSG

PCRLF   PUSH    H
        PUSH    D
        MVI     E,00DH   ;CR
        CALL    PRNTCH
        MVI     E,00AH   ;LF
        CALL    PRNTCH
        POP     D
        POP     H
        RET

CRLF    MVI     E,00DH
        MVI     C,CONOUT
        CALL    BDOS
LF      MVI     E,00AH
        MVI     C,CONOUT
        CALL     BDOS
        RET

UNSUC1  CALL    NOREAD
        MVI     A,0
        RET

UNSUC2  CALL    NOREAD
        CALL    DELAY
EMSG    LXI     D,MSG
        CALL    PMSG
        JMP     INPUT    ;RETURN TO PAINT MODE

PMSG    PUSH    PSW
        PUSH    H
        MVI     C,PRINT
        CALL    BDOS
        POP     H
        POP     PSW
        RET
```

```
LPMSG   MOV     A,M      ;GET MSG CHAR
        CPI     '$'      ;END OF MSG?
        RZ
        MOV     E,A
        CALL    PRNTCH
        INX     H
        JMP     LPMSG

TDLY    PUSH    H
        PUSH    B        ;16 BIT TIME DELAY
        MVI     B,45     ;ADJUST AS NECESSARY
TDLY1   MVI     C,255
TDLY2   NOP
        NOP
        DCR     C
        JNZ     TDLY2
        DCR     B
        JNZ     TDLY1
        POP     B
        POP     H
        RET

DELAY   LXI     D,MSG17   ;TYPE ANY KEY
        CALL    PMSG
DELAY1  MVI     C,CONIN
        CALL    BDOS     ;ABSORB CHAR
        RET

EXIT    CALL    DELAY
EXIT1   LXI     D,MSG8
        CALL    PMSG
        POP     PSW      ;GET CP/M FLAGS
        POP     H        ;GET CP/M STACK
        SPHL             ;SET IT
        RET              ;RETURN TO CP/M

MSG     DB  27,121,49,27,120,49,27,121,53,27,89,56,32
  DB  'F0',27,'pERASE',27,'q F1',27,'pRVID',27,'q F2',27,'pNVID',27,'q F3'
  DB  27,'pGRAF',27,'q F4',27,'pNORM',27,'q F5',27,'pMENU',27,'q F6'
  DB  27,'pMARK',27,'q F7',27,'pMOVE',27,'q F8',27,'pCOPY',27,'q F9'
  DB  27,'pHELP',27,'q F10',27,'pCOLOR',27,'q',27,72,27,107,'$'
MSG1    DB  27,79,27,72,27,121,49,27,69,27,106,27,120,53,'$'
MSG2    DB  27,121,49,27,120,49,27,89,56,43,27,120,53,27,79,27,113,27,71,'$'
MSG3    DB  'DUMPING SCREEN TO MEMORY.',27,35,'$'
MSG4    DB  'RECALLING SCREEN FROM MEMORY.',27,72,'$'
MSG5    DB  'PRINTING SCREEN.',27,35,'$'
MSG6    DB  'SAVING TO DISK.',27,35,'$'
MSG7    DB  'RECALLING FROM DISK.',27,72,'$'
MSG8    DB  27,79,27,72,27,121,49,27,121,53,27,109,55,48,27,113,27,71,27,69,'$'
MSG9    DB  'FILE NAME TOO LONG!$'
MSG10   DB  'NO FILE SPECIFIED ON COMMAND LINE!$'
MSG11   DB  'NO ROOM ON DIRECTORY TO CREATE FILE!$'
MSG12   DB  'NO MORE ROOM ON DISK!$'
MSG13   DB  27,121,53,'WHAT IS THE FILE NAME?'$'
MSG14   DB  13,10,'NO FILE IN DIRECTORY!$'
MSG15   DB  'UNSUCCESSFUL READ OPERATION!$'
MSG16   DB  27,89,56,95
MSG17   DB  '  TYPE ANY KEY:$'
MSG18   DB  27,113,27,71,27,109,55,48,'$'
MSG19   DB  27,94,6,9,0,'$' ;9-PIN BIT IMAGE MODE (6)
```

```
MSG20   DB  27,94,1,10,0,'$' ;9-PIN BIT IMAGE MODE (1)
MSG21   DB  'PAINT100.SC1 WAS NOT FOUND$'
MSG22   DB  'PAINT100.SC2 WAS NOT FOUND$'
MSG23   DB  27,72,'Normal: $'
MSG24   DB  'Graphics: ',27,70,'$'
MSG25   DB  27,80,'$' ;PICA; 80CPL
        DB  27,77,'$' ;ELITE; 96CPL
MSG26   DB  27,69,'$' ;SET EMPHASIS
        DB  27,70,'$' ;REL EMPHASIS
MSG27   DB  27,71,'$' ;SET DOUBLE
        DB  27,72,'$' ;REL DOUBLE
MSG28   DB  27,49,'$' ;10 LPI
MSG29   DB  27,48,'$' ;8 LPI(GFX)
MSG30   DB  27,65,10,27,50,'$' ;7 LPI
MSG31   DB 27,65,12,27,50,'$' ;6 LPI(STD)
MSG32   DB 27,65,18,27,50,'$' ;4 LPI
MSG33   DB  27,121,53,'What is the number of your selection? $'
MSG34   DB  27,121,53,'What is the number or letter of your selection? $'
MSG36   DB  'MOVE TO PLACE MARK2, TYPE "F6".  CTRL-B RESETS.',27,121,53,'$'
MSG37   DB  'GO TO NEW LOCATION: F7=MOVE, F8=COPY, CTRL-B=RESET.',27,121,53,'$'
MSG38   DB  13,10,'BLOCK INCORRECTLY DEFINED.$'
MSG39   DB  27,89,56,32,27,121,53,27,109,55,48,'0BLK.',27,109,55,49,'1BLU.'
        DB  27,109,55,50,'2RED.',27,109,55,51,'3MAG.',27,109,48,52,'4GRN.'
        DB  27,109,48,53,'5CYN.',27,109,48,54,'6YLW.',27,109,48,55,'7WHT.'
        DB  27,109,55,48,' Foreground<0-7>? $'
MSG40   DB  ' Background<0-7>? $'
MSG41   DB  27,78
MSG42   DB  27,109,55,48,'$' ;VARIABLE COLOR
MSG43   DB  27,109,55,48,'$' ;DEFAULT, WHITE ON BLACK
MSG45   DB  27,121,53,'Set printer left margin<default=0>: $'

SPCL    DB     0
RSCNT   DB     0        ;START BLOCK COUNT - ROW
CSCNT   DB     0        ;START BLOCK COUNT - COL
PCPL    DB     0        ;PRINTER S CHAR/LINE
SCPL    DB     0        ;SCREEN S CHAR/LINE
RCOUNT  DB     0        ;ROW COUNT
CCOUNT  DB     0        ;COL COUNT
PICAFG  DB     0        ;0=PICA, 16=ELITE, 64=COMPRESSED
LPRVFG  DB     0        ;0=OFF, 1=ON
LPGFFG  DB     0        ;0=OFF, 1=ON
LSFLAG  DB     12       ;12 IS DEFAULT LINE SPACING
INSFG   DB     0        ;1 IF INSERTING CHARS
MOVFG   DB     0        ;0 IF MOVE; 1 IF COPY
MARKFG  DB     0        ;0=MARK1; 1=MARK2
MARKS   DS     12       ;MARKED CURSOR POSITIONS
LINEB   DS     20       ;LINE INPUT BUFFER
BPTRL   DS     2        ;BUFR POINTER
DMAPTR  DS     2        ;DMA POINTER
TABLEN  DB     0        ;LPT TAB DEFAULT
NUMBR   DB     0        ;LPT TAB TMP
PSCRN1  DB      'PAINT100SC1'
PSCRN2  DB      'PAINT100SC2'
;
; Following codes are for reverse video printer characters:
RVCHAR
R1 thru R95 per page 46 of December 1987 issue of "REMark", except change the
following lines for H/Z-89 and H/Z-100:
R7  DB  255,128,241,128,0,128,14,128,110,128,4,128,145,128,228,128,255,128
R13 DB  255,128,255,128,255,128,249,0,248,0,248,128,255,128,255,128,255,128
```

```
;
; Following codes are for graphic printer characters:
GCHAR
G1 thru G33 per page 84 of December 1987 issue of "REMark", except change the
following lines for H/Z-100 only:
G10 DB  192,0,192,0,192,0,255,128,255,128,192,0,192,0,192,0,192,0
G11 DB  1,128,1,128,1,128,255,128,255,128,1,128,1,128,1,128,1,128
G12 DB  36,128,73,0,146,0,36,128,73,0,146,0,36,128,73,0,146,0
G13 DB  170,128,0,0,85,0,0,0,170,128,0,0,85,0,0,0,170,128
G14 DB  240,0,240,0,240,0,240,0,15,128,15,128,15,128,15,128,15,128
Also change the following line for H/Z-89 and H/Z-100:
G30 DB  1,128,1,128,1,128,1,128,1,128,1,128,1,128,1,128,1,128
;
;
BUFR    DS      6000
        DS      32      ;SPACE FOR STACK
STACK   EQU     $       ;PUT STACK HERE
        END     START
```

Listing 2
PAINT100.SC1

```
                          WELCOME TO PAINT

 Paint a picture you desire using the arrows to move the cursor.
 Special function keys are defined as:
    F0 & <ARROW> or "HOME" key - ERASES portions of screen.
    F1    - SETS REVERSE VIDEO;           F2     - EXITS REVERSE VIDEO
    F3   - SETS GRAPHICS CHARACTERS;    F4     - EXITS GRAPHICS CHARACTERS
    F5    - MENU for other functions.
    F6    - SETS BLOCK MARKER location;  CTRL-B - WILL RESET.
    F7    - MOVES BLOCK to new cursor location.
    F8    - COPIES BLOCK to new cursor location.
    F9 or "HELP" key -  DISPLAYS THIS SCREEN after saving work.
    F10   - CHANGES FORE/BACKGROUND COLOR.

    ESC-E  - Also ERASES entire screen (EVERYTHING IS LOST!)

    CTRL-D  - DUMP the painting.         CTRL-L  - LOAD  the painting.
    CTRL-W  - WRITE to disk.              CTRL-R  - READ  from disk.
    CTRL-P  - PRINT to printer.           CTRL-E  - ENDS  the program.
    CTRL-G  - DISPLAY GRAPHICS on top 2 lines.
```

Listing 3
PAINT100.SC2

```
                             PAINT MENU

    SELECTION OF PRINTER CHARACTERISTICS:
      1  =  PICA Pitch (80 cpl).             2  =  ELITE Pitch (96 cpl).
      3  =  Set EMPHASIS MODE.               4  =  Release EMPHASIS MODE.
      5  =  Set BOLDFACE MODE.               6  =  Release BOLDFACE MODE.

    OTHER SELECTIONS:
      L  =  PRINTER LINE SPACING
              1  =  10 LPI (lines per inch)
              2  =   8 LPI (set for GRAPHICS)
              3  =   7 LPI
              4  =   6 LPI (DEFAULT setting)
              5  =   4 LPI
      T  =  Set TAB for Printer Output.
      Z  =  No Change/Done.
```
*

Continued from Page 6

Any ideas as to where my problem is nested? I have no ideas as to where to look further. Thought someone out there could help sort out my difficulty.

Gratefully,
Dan Setters
Edwin K. Williams & Co.
309 Windsor Road
Champaign, IL 61820 *

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

A-One H/Z Enhancements!

Clock Uses No Slot
FBE SmartWatch: Automatic date/time on bootup. Installs under BIOS/Monitor ROM. Ten year battery. Works with all Heath/Zenith MSDOS computers. For PC's $35.00, Z-100 $36.50 Module: $27.50

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

H/Z-148 Expansions
ZEX-148: Adds one full-size and one half-size expansion card slot. $79.95
ZP-148: Replacement PAL chip expands existing 640K memory to 704K. $19.95

H/Z-150 Stuff (Not '157, '158, '159)
VCE-150: Eliminate video card. Install EGA or VGA card. All plug in. $39.95, SRAM Chip (Required) $20.00
RM-150: PROM used in removing video card. With detailed instructions. $9.95
ZP640 PLUS: Expand standard memory card to 640/704K with 2 banks of 256K RAM chips (not included). $19.95
LIM150: Get 640K RAM plus 512K of simulated Lotus/Intel/Microsoft EMS v3.2 expanded memory. Installs on standard memory card. No soldering. Must have 45 256K RAM chips (not included). $39.95
MegaRAM-150: Get 640/704K plus 512K RAM disk on standard memory card. No soldering. Without RAM chips. $39.95
COM3: Change existing COM2 address. Put internal MODEM at COM2. Don't lose serial port. Includes software. $29.95

Maximize Your Z-100
ZMF100A: Expand "old" motherboard (p/n 181-4917 or less) using 256K RAM chips (not included). No soldering. $65.00
ZRAM-205: Put 256K RAM chips on your Z-205 board. Get 256K plus 768K RAM disk. Contact us for data sheet before ordering. Without RAM chips. $49.00

Z-171 Memory Expansion
MegaRAM-171: Put 256K RAM chips (not included) on existing memory card. Get 640K plus 384K RAM disk. $59.95

H/Z-89 Corner
H89PIP: Parallel printer 2 port interface card. With software. $50.00 Cable $24.00
SLOT4: Add fourth expansion slot to right-side accessory bus. $39.95

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

FBE

FBE Research Company, Inc.
P.O. Box 68234, Seattle, WA 98168
206-246-9815 Voice/FAX TouchTone Selectable

Reader Service #104

Making Your *Hard* Disk *Easy*

Mark Haverstock
6835 Colleen Drive
Youngstown, OH 44512

One of the most useful upgrades to your computer system is a hard drive. The speed improvements in disk I/O and storage are well worth the investment. However, many hard drives end up resembling cluttered closets or overstuffed attics. One of the pitfalls of having multi-megabytes of storage space is that files can easily become disorganized and hard to find.

DOS has an excellent system for organizing files. In addition, there are several programs available that will make file access even easier. The purpose of this article is to help you plan how you'll set up your hard disk, and how to use the DOS tree structure to your best advantage.

Preparation

Before you begin to add programs and files, a hard disk must be installed and formatted. Follow the disk manufacturer's installation instructions. If you partition your disk, be sure to specify the default boot partition as 1 (the first partition). The next step is to do a high-level format and transfer the DOS system files to your hard disk. To do this, you'll need to put your DOS system disk in drive A and boot your computer. At the A>, type:

```
FORMAT C: /S
```

After the format is complete, get the directory for drive C by typing at the C>:

```
DIR C:
```

If the file COMMAND.COM appears, your hard disk is ready to test.

Open the door on drive A: to be sure the system doesn't boot from drive A. Restart the system by pressing CTRL-ALT-DEL (hold down CTRL and ALT, press DEL). If you're using a Zenith hard disk controller, use CTRL-ALT-INS to access the monitor ROM. Type BW0, followed by a return, to boot the system. If all went well, the C> should appear, showing that the current drive is the hard disk.

If the system doesn't work the first time, retry. If DOS still doesn't restart, put the DOS system disk into drive A and repeat the procedure from the FORMAT C: /S.

Setting Down Roots

The DOS file system is hierarchical in nature. In diagram form, it looks much like a corporate organizational chart or a family tree. It starts with a *root directory*. From this root, you can build *subdirectories*. These subdirectories can contain files or more subdirectories, and so on. Figure 1 illustrates an example of the tree structure.

In Figure 1, there are six subdirectories or branches from the root. Each of the subdirectories contains the operating files for a particular program, such as a word processor. The word processor and data base also have subdirectories for their own documents.

How many directories can we make? In recent versions of DOS, a hard disk can support up to 512 directories that branch from the root directory. Branching from a subdirectory, we can make as many subdirectories as disk space will support. However, there are practical limits. Your "tree" will look more like a maze if you make too many subdirectories.

There is no "correct" number of subdirectories to use for program files. It depends on how many programs you decide to put on your hard disk. Some planning is necessary — you can't just copy everything to the root directory. Organize your hard disk with the same care you'd organize a good filing system. Here are a few things to consider before creating directories:

1. *The subdirectories of frequently used programs should reside in the root directory.* This is simply for convenience. Limiting the root directory to a few necessary items will also help you better keep track of the hundreds of files that will eventually become part of your hard disk.
2. *Your root directory should contain only subdirectories.* You should avoid keeping stand-alone programs or files in the root directory. Notable exceptions are AUTOEXEC.BAT, COMMAND.COM, CONFIG.SYS.
3. *Attach data files to their program subdirectories.* Data files are usually accessed through their applications programs. Create a subdirectory within the program directory. See Figure 1 for the relationship between the word processor and its data files.
4. *Limit yourself to no more than two levels of subdirectories, if possible.* Examine Figure 1 again. Using the root directory as a starting point, level one includes the word processor, DOS, utility, data base, games and menu subdirectories. The second level of subdirectories include the word processing and data base files, as well as the Tools, Xtree, Wheel and Chess programs.

Now it's time to get a pencil and paper. Make a list of the programs you plan to install on your hard drive. Consider the

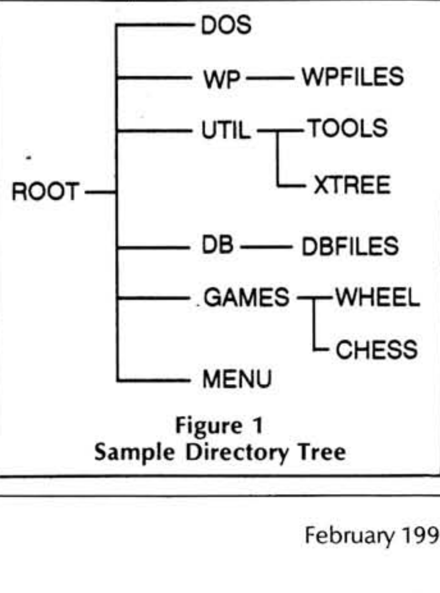


Figure 1
Sample Directory Tree

programs and how you use your computer. Then sketch a directory structure that best matches your needs without becoming overly complex.

Making Directories

Once you've decided what directories you want to create, the next step is to make them using the DOS *Make Directory* command, abbreviated MD. At the prompt, you'd type:

```
MD\<path>
```

Path stands for the name and location of the directory you wish to create. The backslash (\) tells DOS to start this path at the root directory. No backslash means that the path will start in the current directory.

Let's assume that you wanted to create a directory tree as shown in Figure 1. You're at the root directory, C:\. The first step would be to make a directory for the rest of the DOS files. To create a directory for DOS on drive C:

```
MD\DOS
```

Before you copy files, you need to move from the root directory to the newly created DOS directory. To do this, use the DOS command, *Change Directory*, or *CD*. Type at the C>:

```
CD\WP
```

Next, copy the DOS files from drive A to the DOS directory on drive C. Type:

```
COPY A:*.* C:
```

The name of each file will be displayed on the screen as it is copied. When all the files have been copied to the DOS directory, the C> will reappear. If you're using a 3.5" disk, you're done copying DOS to the hard drive. 5-1/4" disk users will have a second DOS disk to copy. Insert the second disk into drive A and repeat the copy command (COPY A:*.* C:) to add the remainder of the files to your DOS directory. Next, return to the root directory by typing:

```
CD\
```

Once DOS is installed, it's time to start making other subdirectories. To create another subdirectory for the word processor, type:

```
MD\WP
```

WP stands for the name of your word processing program. You could use any name here, but it's a good idea to keep these names short. You'll save time later by only having to type a few characters to enter the subdirectory. Your directory tree now looks like Figure 2.

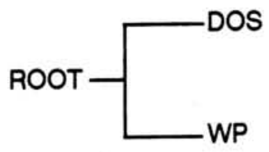


Figure 2
Subdirectories

You've created the directory, so it's time to install the word processing program — almost. You're still in the root directory and need to change to the word processor subdirectory WP. Type:

```
CD\WP
```

Now copy the word processor program files to the hard disk. To do this, place the word processor program disk into drive A and type at the C>:

```
COPY A:*.* C:
```

Remember, *.* is a "wildcard" designation in DOS. You're copying all the files from drive A to the newly created WP subdirectory on drive C.

Creating a subdirectory for your word processing files is done as in the previous examples. Next you'll create a file subdirectory named WPFILES. You're currently in the WP subdirectory, so you'll begin with:

```
MD\WPFILES
```

WPFILES is a subdirectory within WP. You can save any word processing files by guiding them to this directory. The path for this directory is C:\WP\WPFILES. Figure 3 shows the newly updated directory tree.

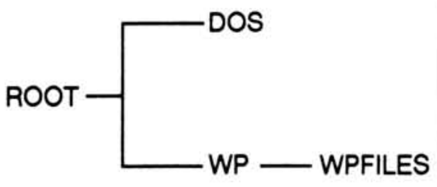


Figure 3
Word Processor File Subdirectory

If you're ready, return to the root directory and continue to make other subdirectories and install your programs. You may find that some programs have their own automatic install routines, but you'll still be asked to provide a directory name or path. Be sure these don't duplicate names you've already used.

Grouping Programs

If you have a large number of programs, you may want to try different a strategy for hard disk organization. As mentioned before, you'll want to make your most used programs branch from the root directory. However, if you have over twenty items in this directory, it's time to think about consolidating them. You can create a category name and have each program become a related subdirectory. Let's use UTIL as a sample directory. Starting from the root directory, type:

```
MD\UTIL
```

Next, move into this directory by typing CD\UTIL. Now that you're in this directory, you can create as many subdirectories as needed to hold your utility programs. For example, to create a subdirectory for Xtree, you'd type:

```
MD\XTREE
```

Move to this directory by typing its path, CD\UTIL\XTREE. Then copy your program files into the new subdirectory using the wildcard copy command. Put your program disk into drive A and type:

```
COPY A:*.* C:
```

Finally, return to the UTIL directory by typing CD\. Create another directory for the next utility and so on. Figure 4 traces these steps on the directory tree.

Note the path designations or pathnames in the diagram. To get to Tools,

Menu Software

Menu programs are available from HUG, as shareware, and from software manufacturers. These are some of the programs available to Heath/Zenith and other PC compatible users:

HUG Menu System Part #885-3020 Heath User's Group P.O. Box 217 Benton Harbor, MI 49022-0217 (616) 982-3463	$20.00
Automenu Magee Enterprises P.O. Box 1587 Norcross, GA 30091 800-662-4330	$50.00*
HDM III Jim Haas P.O. Box 447 Richfield, OH 44286	$25.00*
HD Menu VAI Financial Software P.O. Box 37034 Honolulu, HI 96837	$39.00*
TheEmcee Command Software Systems 28990 Pacific Coast Highway Suite 208 Malibu, CA 90265 (213) 457-1780	$89.95
Direct Access Delta Technology International, Inc. 1621 Westgate Rd. Eau Claire, WI 54703 (715) 832-7575	$89.95
Take Charge! Departmental Technologies, Inc. 13 Kilroy Road Newton, NJ 07860 (201) 786-6878	$139.95

* Demo versions of these programs are available on many BBS systems and from shareware distributors for a nominal cost. The prices listed are for registered versions with documentation.

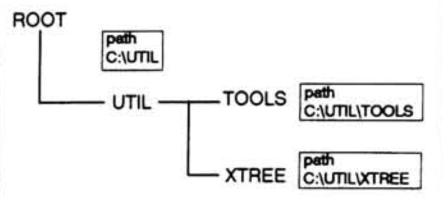


Figure 4
Pathnames

you follow a path from the Drive C root directory to UTIL, then to *Tools*. The path is:

```
C:\UTIL\Tools
```

Menu Programs

Menu programs are one more way to help you organize your hard disk. If you have them loaded in a batch file, you'll never need to deal with a DOS prompt. Most menu programs list the names of the programs on the screen and allow you to run programs or call other menus with a single keystroke. They save time and typing, and look less intimidating to beginning users than a C> with a blank screen.

However, these programs don't magically organize your hard disk. Menu programs require that you set up your own directories. They will ask you for pathnames so the menu can locate programs on your hard disk easily. A basic knowledge of DOS tree structure is still essential to configure them properly.

Several of these programs, such as *The Emcee, Direct Access* and *Take Charge!* include additional file utilities and programs that you might find helpful.

Summary

The key to setting up your hard disk is organization. Gather your programs and draw up a sample "tree" on paper. Try to keep this tree as simple as possible, limiting it to two levels if possible. Make your subdirectories and copy the programs into them, tracing the path as you go. If you have trouble keeping track of your location on the directory tree, enter the DOS Roadmap batch file in Listing 1. ✱

A DOS Roadmap

Hansel and Gretel left bread crumbs on the path to help them find their way back through the woods. DOS has a neater and more reliable way to pinpoint its location. A path in the directory tree is the route we follow to trace our way from the root directory to a particular subdirectory and program. The description of the path is called the *pathname*. This pathname can be displayed on the screen by invoking the *PROMPT* command.

Here is a simple, but useful batch file to add to your hard drive. There will be no question where you're at on the DOS tree at any given time, and DOS functions will be available without changing directories. Be sure you're in the root directory and type the following:

```
COPY CON AUTOEXEC.BAT
ECHO OFF
CLS
PROMPT Drive and subdirectory
   location are: $p$_$n$g
Path C:\DOS
^Z
```

(The ^Z is created by pressing F6 on the keyboard and signals the end of the batch file)

Listing 1
Roadmap Batch File

To activate this batch file, reboot your computer. The message that appears on the screen will now look something like this:

```
Drive and Subdirectory location are:
   C:\myprog\myfile
C>
```

If you currently have an AUTOEXEC.BAT file, you can add the prompt line using EDLIN or any other ASCII text editor. It should proceed any path statements that currently appear in your AUTOEXEC.BAT file.

Continued from Page 9

type monitors. The connector on the back of the old HS-2860 only supplied CGA-type video, so the built-in video is a vast improvement on the new machines.

Because the computer must do extra work in order to convert a 256 color picture into gray scale before putting it on the LCD screen, it takes it longer to load displays in this mode. Action games or animation programs cannot be used while you are using the LCD display if they run in the 256 color mode. The color conversion software is turned off in other graphic video modes, so displays load as fast as on a CRT screen. The conversion of colors to gray scale in these modes is determined by which palette the LCD system is using, and you can change palettes by using the Fn-F8 and Fn-F9 keys (as on the HS-2860), or with the MACHINE utility that comes with MS-DOS 3.3+. However, if you run Microsoft Windows™, you will find that the Fn-F8 and Fn-F9 keys do not work while Windows has control, so you should set up the palette you wish to use for Windows before you start it.

Aside from the video problems, the HS-2862 has performed flawlessly, and it turns in impressive times with speed testing programs. The hard disk speed is especially impressive, with a Norton Disk Index of 5.5. I have a TurbosPort 386 in my office (the original version) that has a disk index of 5.1, and my desktop Z-386/16 only does 3.7.

A Battery Extension Cord

One of the complaints about the SupersPort 286 is that it weighs too much, and the improvements made to the new version have not helped that any. Mine weighs in at 12 pounds (5.4 kg) on my old Heathkit digital scale, and the original is advertised at "under eleven pounds", so it seems to have gained a pound. These weights are without the battery pack, which not only adds weight to the unit, but also length. I have not tried mine on an airplane yet, but users report that with the battery pack attached, it is too long to be used comfortably in the "second class" seats. However, it is possible to construct an extension cord that will let you put your battery pack on the floor while you hold the computer in your lap.

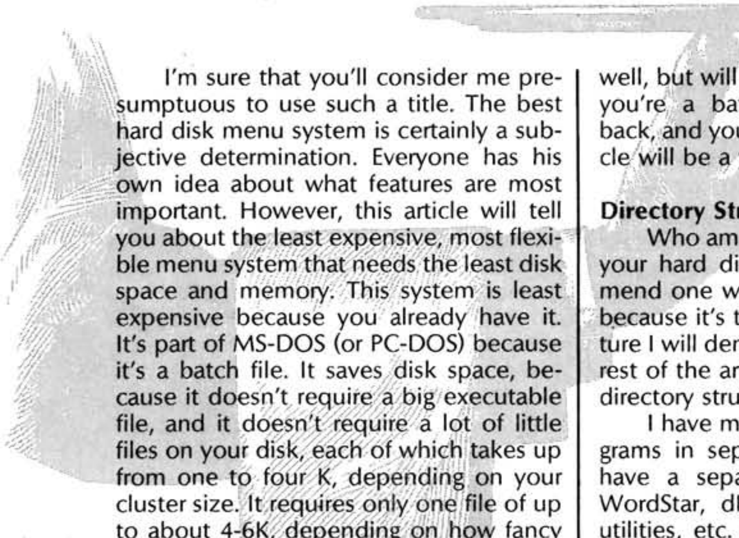
The battery pack has a coaxial power plug on it that fits into a coaxial power jack on the back of the computer. This type of power connector is in common use, but they are available in a variety of sizes. The size used on SupersPort computers is 5.5 mm outside diameter, and 2.1 mm inside diameter. To make my battery pack extension cord, I used a Radio Shack (Archer) catalog no. 274-1563 Coaxial Power Jack, and a Radio Shack catalog no. 274-1567A Coaxial Power Plug. I soldered these to each end of a 4 foot (1.2 m) length of speaker wire (Radio Shack catalog no. 42-2449, but any comparable wire will do). The plug I used is actually a 5.0 mm OD by 2.1 mm ID model, but it works. If you can find a 5.5 by 2.1 plug at your electronics store, use it instead.

The battery pack extension cord not only allows you to operate the computer on batteries without having to bear the weight of the battery pack, but it also allows you to use a printer or other peripheral while you are operating on batteries. Someone could probably make a few bucks by fabricating these things and selling them for about 10 dollars apiece. Mine cost 4.97 (plus tax) to build, and I had an extra plug and several feet of speaker wire left over.

By the way, if you are wondering about the title of this article, it is not just because of the cheetah in Photo 1. It is because Zenith Data Systems makes up a code name for each new model of computer, apparently used while it is in development, and the code name for this model is LYNX. The old SupersPort is a BADGER. ✱

The Best Hard Disk Menu System

Peter E. Walberg
390 Loma Drive
Forsyth, IL 62535-0057



I'm sure that you'll consider me presumptuous to use such a title. The best hard disk menu system is certainly a subjective determination. Everyone has his own idea about what features are most important. However, this article will tell you about the least expensive, most flexible menu system that needs the least disk space and memory. This system is least expensive because you already have it. It's part of MS-DOS (or PC-DOS) because it's a batch file. It saves disk space, because it doesn't require a big executable file, and it doesn't require a lot of little files on your disk, each of which takes up from one to four K, depending on your cluster size. It requires only one file of up to about 4-6K, depending on how fancy you make it. It's most flexible, because you can use almost any combination of DOS commands, such as COPY, CD, RENAME, APPEND, SUBST, PATH, SET, etc. You can use command line arguments (replaceable parameters, in batch file terms), so you can set up your menu to do just about anything. Since you are always working from the DOS level, there is no additional step when you want to execute a command that is not on your menu. Finally, there is no big memory-resident menu program taking up your RAM while you are running your programs.

Introduction

This article is intended for intermediate skill level computer users. If you know how to use subdirectories and what a batch file is and how to make any simple one, you're ready for this article. As a matter of fact, if you read Mark Haverstock's article on page 19 of the June 1989 issue of REMark, you're ready. It's written on the basis of PC-DOS 3.30, but is applicable to PC- or MS-DOS versions 2.XX and up. Some of the commands I will demonstrate may not be applicable if your version of DOS doesn't recognize them, but the concepts for this menu system will still apply. This article also assumes that you have a hard disk on your computer, and that you have several application programs and data directories that you use. The concepts will apply to floppy disks, as well, but will be of limited utility there. If you're a batch programmer from way back, and you know all the tricks, this article will be a waste of your time.

Directory Structure

Who am I to tell you how to organize your hard disk? However, I will recommend one way and explain why I like it, because it's the basis for the menu structure I will demonstrate. You can apply the rest of the article no matter what kind of directory structure you like, though.

I have my application and utility programs in separate directories. That is, I have a separate directory for LOTUS, WordStar, dBase, BASIC, miscellaneous utilities, etc. I also have a few separate data directories. I have one for home, one for my work with the church youth group, one for my daughter, etc. Then I keep all the data files for each subject in that directory. I have the documents, spreadsheets, databases, BASIC programs, graphics, etc. all in that directory. Then I can back up those data directories quite easily on one or two floppies, using just the XCOPY command. There is no need to back up the whole disk often, because the rest of the disk hardly ever changes. (Don't forget to do it when you make changes, however.) This makes backups of your data easy and fast (and they need to be, or you won't do it).

Menu Structure

I'm sure many of you have seen batch file menu structures that print a menu and ask you to make a selection which calls for another batch file. Using that method requires not only the main batch file, but the menu text file, and then individual batch files for each menu selection. Each batch file, even if it contains only 11 bytes, takes up a whole cluster on your disk, up to four K-bytes! What a waste! However, you can put everything in one file. That's what I'll show you how to do.

In my example, there are five main sections of the batch file: first, the selection section; second, the command string section; third, the menu text; fourth, an error message section; and finally, a restoration and end section. The example at the end of the article shows the structure of the file and the order of the sections.

Selection Section. The selection section of the menu uses the IF batch file statement. Here is a short example of this section. Notice particularly three things. First, the %1 in each line. If you are not familiar with batch files with replaceable parameters, that's what this is. When you execute a batch file, you normally just type its name and hit a RETURN. However, if you type in other things after the name of the batch file, but before you hit RETURN, the batch file will employ these values as variables. For example, if the name of this batch file is DO.BAT, then typing DO SS RETURN will cause the batch file to substitute the SS every place there is a %1 in the file. Secondly, notice the logical comparison. In batch files, this is done with the double equal sign (==). The second thing to notice is the lowercase "a" in each line. If you try to compare a "nothing" using the logical comparison, you'll get a syntax error. In other words, if you type just the name of the batch file, with no replaceable parameter, then %1 will be blank, and each of the IF 1% == statements will give a syntax error, but with the "a"s, it will work smoothly. The third thing to notice is that each statement is repeated twice, once with uppercase and once with lowercase comparison values. This will allow you to type in either upper- or lowercase, and get the same result.

Example selection section:

```
IF %1a == a GOTO ERROR
if %1a == WPa GOTO WP
if %1a == wpa GOTO WP
if %1a == SSa GOTO SS
if %1a == ssa GOTO SS
 .
 .
 .
 .
GOTO error
```

Command String Section. The IF statements in the selection section all contained a GOTO command. The parameter after the GOTO must be a label in another part of the batch file. These labels (except for ERROR) will be in the command string section of the file. Here is a short piece of such a section. Note that there must be a label for each part of it. A label is a word (or group of letters) with a colon in the first column. The labels must correspond to the parameters in the GOTO statements in the selection section. The end of each part of this section must end in a GOTO that jumps past the rest of the commands toward the end of the file, otherwise, all the rest of the commands will also be executed each time you run the batch file. Between the label and the GOTO menu statement you can put almost any DOS command. If you don't understand all the statements in this short example, don't worry, because I'll explain them all later in the article.

Example command string section

```
:SS
CD\123
copy \HOME\HOME123.cnf 123.cnf
COPY \HOME\HOMEGR.CNF GRAPH.CNF
KEY-FAKE 0 13 0 13 0 13 0 13 0 13 "/FR"
LOTUS
copy 123.CNF \HOME\HOME123.CNF
copy GRAPH.cnf \HOME\HOMEGR.CNF
CD\HOME
GOTO menu
:WP
path=c:\utility;c:\dos;c:\;c:\words
append c:\words
CD\home
mark
wf c:
ws %2
PATH=C:\UTILITY;C:\DOS;C:\
GOTO menu
  .
  .
  .
```

Menu Text Section. This is the user friendly part of your menu system. This is where you put the list of options and their descriptions so you can remember what your batch file does and how to get it to do it. It also has instructions, such as what to type. You can put in as many options as you can get on the screen. You can use two columns if you want. Actually, you can use more than one screen, or you can use submenus, but it's up to you to develop those ideas. Note the label (:menu) for this section. The batch file will jump to this label after it completes any other part of the file.

Example menu text portion:

```
:menu
cd\
echo              MAIN DO MENU
echo .        ======================
echo    Chose one of the following options
:
echo .
echo  SS  Spreadsheet
echo  WP  Word Processing [file]
  .
  .
  .
echo .
echo   Type DO [OPTION][S]..and hit RETURN.
echo .====================================
GOTO end
```

Error Message Section. This section is another attempt at user-friendliness. Instead of just ending when you make a mistake, it will tell you why the batch file is ending and show you the menu again for another chance.

Example error message section:

```
:error
echo Not a valid option.  Try again.
GOTO menu
```

Closure and End. This section provides a place to jump to in order to restore any path or directory that you want to use as a default and, also, to provide a jumping off place from the menu.

```
:end
cd\
path=c:\utility;c:\dos;c:\
```

Well, that's it. That's the whole file structure. You can build from that as much as you want. Now I'm going to tell you about some of the commands that you can have in your file, in the command line section.

Batch File Commands

I'm not going to tell you about every possible command you can use in a batch file. This is an intermediate level article, remember? But I'll tell you about the ones I use for my menus, and that's most of them. I'll tell you about them and give an example of each. Of course, besides the ones I'll demonstrate, you can use BACKUP, CHKDSK, CLS, DEL, DIR, DISKCOMP, DISKCOPY, FIND, FORMAT, etc.

ECHO — I'm going to start out by not doing what I said I would do. The examples for ECHO are in the menu text and error message section of the article (above). Echo will print on the screen whatever you type after it, providing you leave one space for a delimiter. Actually, ECHO is used for two more purposes. If you type "ECHO OFF" as one of the first lines in your batch file, each of the following lines in the batch file will not show on the screen as the batch file runs. Then you can type "ECHO ON" at the end of the file to turn this function back on. Typing just "ECHO" by itself will show whether the ECHO function is on or off.

CD (Change Directory) — CD is used to change the active directory (I'm sure you know this), but I use it to change to the appropriate data directory for the files I want to work on. In the following example, the data directory is "home".

```
:db
cd\home
path=c:\utility;c:\dos;c:\;c:\dbase
dbase %2
cd\
GOTO menu
```

PATH — PATH is used for running a program in a different directory. It's really a major part of making my disk directory structure work. I can use the CD command to change to the appropriate data directory, then use the path command to find the program I want to run which is in a different directory (called dBase, in this case). In the example above, the path statement is used to find the dBase program to work on files in the home directory.

APPEND — APPEND is used for programs that have auxiliary files, such as overlays or configuration files that it needs in order to run. For example, my version of WordStar has several overlays that it must find. The PATH command will find WS.COM (or an .EXE or .BAT file) (note the path to "words" in the example), but it won't find the overlay files that way, and there is no way to configure WordStar to find them, either. But the DOS command APPEND will do for the auxiliary files what the path command will do for executable commands. It doesn't work for all programs, but most. Some later examples will show other ways around this problem.

```
:WP
path=c:\utility;c:\dos;c:\;c:\words
append c:\words
cd\home
wf c:
ws %2
PATH=C:\UTILITY;C:\DOS;C:\
GOTO menu
```

Note that this part of the batch file also loads the Word Finder Thesaurus (wf) before it starts WordStar (ws).

COPY — I'm sure you know what the COPY command does. Here's an example applicable to menus. You may know that LOTUS 1-2-3 must be configured to find data files. You have to tell it where to look. Then it keeps this information in a .CNF file. LOTUS is also one of those programs that can't find its auxiliary files, even with append. So there's no option but to go to the directory it's in in order to start it. No problem, because the configurations file will tell it where to put the data. But how about if you want to have spreadsheet files in different directories for different applications? One way you can do this is to re-configure it every time you want to change data directories. But there's a better way. For each data directory in which you want to have LOTUS files, make up a separate configuration file. You'll need one for spreadsheets (123.CNF) and one for PrintGraph (GRAPH.CNF). Store the appropriate ones in the applicable directories. Then in the part of the batch file for working on spreadsheets, copy the appropriate configuration files to the LOTUS directory. CD to the LOTUS directory and start LOTUS. When you're done, be sure to copy the configuration files back to the data directories, so that any changes you make to your default configuration will be there next time you start LOTUS.

```
:SS
CD\123
copy \HOME\HOME123.cnf 123.cnf
COPY \HOME\HOMEGR.CNF GRAPH.CNF
KEY-FAKE 0 13 0 13 0 13 0 13 0 13 "/FR"
LOTUS
copy 123.CNF \HOME\HOME123.CNF
copy GRAPH.cnf \HOME\HOMEGR.CNF
CD\HOME
GOTO menu
```

KEYFAKE — You may have noticed the line in the above example that starts with KEY-FAKE. No, you won't find it in your DOS manual. This is a nifty little utility that will get you past all the RETURNs you have to type just to start 123. If you use LOTUS release 1A, as I do, this line will get you right to the place where you can select the file you want to work on. Or if you always want to default to a given spreadsheet, you can still use the KEY-FAKE line, but name your file AUTO123.WKS, and it will load automatically. Back to KEY-FAKE. It's one of a batch of utilities from *PC Magazine* that you can find on many bulletin boards as PCMAG.ARC. The documentation is in the same file. What it does, is to fool a program into thinking it's getting input from the keyboard, while it is really getting them from the KEY-FAKE line in the batch file.

SET — SET is another way that a program in one directory can use data files in another. Certain programs check the DOS "environment" for instructions. The environment is a small memory buffer that keeps various configuration commands for DOS. For example, it is where the PATH information is stored. Besides PATH, SET also affects what is stored there. For those programs that check the environment then, you can put configuration commands into the environment with SET. In the following example, DOODLER is a program that checks the environment to find out where to locate its PBK, CHR, and DVR files. The executable program is still found with the PATH statement, but when the executable program starts, it reads the environment and finds out that the files it needs are located in the \doodler directory, and it finds them and is satisfied. Note that I reset the environment when I'm done running Doodler in order to recover the space that the SET statements used, because there is limited space in the environment for storage. (Actually, this space can be expanded, but this is an intermediate level article, remember?)

```
:HG
path = c:\utility;c:\dos;c:\;c:\doodler
set pbk=c:\doodler
set chr=c:\doodler
set dvr=c:\doodler
COPY \DOODLER\%2.DVR \DOODLER\DOODLER.DVR
doodler psetup
path = c:\utility;\c:\dos;c:\
set pbk=
set chr=
set dvr=
GOTO menu
```

SUBST — There is one final way to deal with programs that can't access different subdirectories. That is with the SUBST command. The SUBST command will make DOS think you have another disk drive. You can SUBSTitute a drive name for a subdirectory path. In the following example, drive f: is substituted for the path c:\home. So in order to access that path, you just refer to it as drive F:. For example, you can get a directory of F:, copy to F:, delete everything on F:, etc. Outline, actually The Classical Classifier, is a program that must be started from its own directory, but it can access different drives, although it knows nothing about paths. Note the subst f: /d line that cancels the SUBSTitution. Note also the pause statement. This is the first time we've used that. It just stops the batch file until you push a key, so you can read what's on the screen.

```
:HO
SUBST f: C:\home
DIR f:*.Otl
echo
echo NOTE: Outlines are on drive F:
pause
CD\OUTLINE
OUTLINE
cd\
subst f: /d
GOTO MENU
```

COMMAND LINE PARAMETERS — You may have noticed something in the examples for CD, APPEND, and SET that I didn't explain in those sections. In the command lines that started the application programs there was a %2 (dbase %2 and ws %2). Well, for those programs that accept command line parameters, you can put them in with this menu system. Earlier, when I discussed batch file replaceable parameters, I didn't tell you that you could have up to nine of them (actually more, but that's beyond the scope of this article; nine should be plenty for a menu). So if you know the name of the file you want to edit, and you remember the menu selection for word processing, you can type, for example, "do wp myfile.doc" RETURN, and you'll be there. Go back and look at the menu text part of the article, and you'll see that for the line for selecting word processing there is an entry [file]. This is to remind you that that capability is there. The square brackets indicate that entering the file name is optional. In the case of DOODLER, I also use another parameter to copy the printer file corresponding to the printer I want to use for this Doodler session. I have one utility program that requires an option, a file name, and switches, and this can all be handled with replaceable parameters. Here's another copy of the lines I've just discussed, so you don't have to look back, if you don't want to.

```
dbase %2
ws %2
COPY \DOODLER\%2.DVR \DOODLER\DOODLER.DVR
echo  GR  Doodler Graphics [printer]
echo  WP  Word Processing [file] .
```

COMBINATIONS — One really nice thing about this kind of menu system is that you can string together as many commands as you want. I have one situation in which I need to do some processing in dBase, then take the output from that and process it with BASIC, and then transfer it by serial interface to my other computer. I can do this all by just typing the menu selection.

Here is an example:

```
cd\workdir
path=c:\utility;c:\dos;c:\;c:\dbase
dbase PREPTRAN
path=c:\utility;c:\dos;c:\;c:\basic
basica CONVRTDB
path=c:\utility;c:\dos;c:\
APPEND \COMMUNIC
\COMMUNIC\MITE HDOSXFER G
GOTO menu
```

Of course it's not completely automatic; I still have to set up the other computer. But it sure saves a lot of typing, and even more trying to remember, "Now what was that program name?" or "What was the name of that intermediate file?", etc.

Enhancements

Now, you might be saying to yourself, "Yeh, that's all very well, but I like the colors on my other menu system." Take heart. We can dress this one up, too. You can have a nice easy yellow on blue for the menu text. You can have white on red for the error message. You can even have instructions flashing. And something I haven't seen any other menu system handle, you can print the menu selections in double wide characters, if you like, so you can read them without your specs. There are two steps to adding these enhancements.

The first step is to make sure that the file "ANSI.SYS" (It comes with DOS) is in your root directory. Then include a line like the following in your "CONFIG.SYS" file:

```
DEVICE = ANSI.SYS
```

Then reboot your computer.

The second step is to include the desired codes into your menu system. Explaining all the ANSI codes is beyond the scope of this article, but I've included several in the following example, so you can see how to use them. Each of the lines that has a bunch of funny symbols like ^[[34h is an ANSI escape code. It tells what video mode should be used for the following characters until a new escape code is encountered. The ^[represents the escape character (ASCII 27). Depending on the editor that you use to write your batch files, you may have to do some tricks to get these into your file. With WordStar or Sidekick, you first type a CTRL-P, then the ESC key. With others, you may have to hold down the ALT key while you type

027 or 155 (128+27) on the number pad and then let up the ALT key. If worse comes to worse, what you may have to do is write a BASIC program that will put the ANSI escape codes into a file, and then import them into your batch file. That's how I had to start out, because none of the editors I had could handle the job (or at least I didn't know how to get them to do it). There's one thing to watch out for — you can change colors or video attributes any place in your file, but if you change between 40 and 80 column mode, the screen will clear, and you won't be able to read what went before, so you need to carefully consider where you want to do that. You may also notice, in the example, that there are periods at the end of each line of the menu text. The video attributes will only be effective where there is something written to the screen, so without those, the screen will be black after the last character in each line, giving a very unprofessional appearance.

```
echo off
echo ^[[3h
REM SELECTION SECTION
if %1a == HGa GOTO HG
  .
  .
  .
REM COMMAND STRING SECTION
:HD
  .
  .
  other labels and commands
  .
  .
:menu
REM EXAMPLE MENU TEXT PORTION
cd\
echo ^[[44m^[[33m^[[=1h
echo             MAIN DO MENU               .
echo .      ======================          .
echo    Chose one of the following options:.
echo .                                      .
echo HD Data Base Work [program]
  .
  .
  other menu text lines
  .
echo .
echo ^[[7m^[[5m.
echo Type DO [OPTION][S]..and hit RETURN. .
echo .=====================================
echo ^[[0m
GOTO end
:error
REM ERROR MESSAGE SECTION
echo ^[[5m^[[41m^[[1m^[[1h.
echo Not a valid option. Try again.       .^[[0m
GOTO menu
:end
REM CLOSURE AND END
cd\
path=c:\utility;c:\dos;c:\
```

Acknowledgments

Here is a list of programs that have been mentioned in this article along with their publishers/sources:

MS-DOS — Microsoft Corporation, 10700 Northup Way, Box 97200, Bellevue, Washington 98009
dBase — Ashton-Tate, 10150 W. Jefferson Boulevard, Culver City, California 90230
WordStar — MicroPro, P.O. Box 7079, San Rafael, California 94901-0079
Doodler — Paul F. Herman, 3620 Amazon Drive, New Port Richey, FL 34655
Lotus 1-2-3 — Lotus Development Corporation, 161 First Street, Cambridge, Massachusetts 02142
Key-Fake, *PC Magazine*, Volume 4, Number 26
The Classical Classifier, David E. Toliver, 6100 Greene Street, Philadelphia, Pennsylvania 19144

REMOVABLE HARD DISK STORAGE
FOR ZENITH PC'S

HARD DISK CARTRIDGE SYSTEMS

- Random Access
- Unlimited Capacities
- 20MB to 50MB Capacities Available
- Proven Winchester Technology
- Maximum Data Security
- Image Back-up
- 100% Media Interchangeability
- 24 Month Warranty

Model 181ES 50MB Removable Hard Disk System $1,345.00

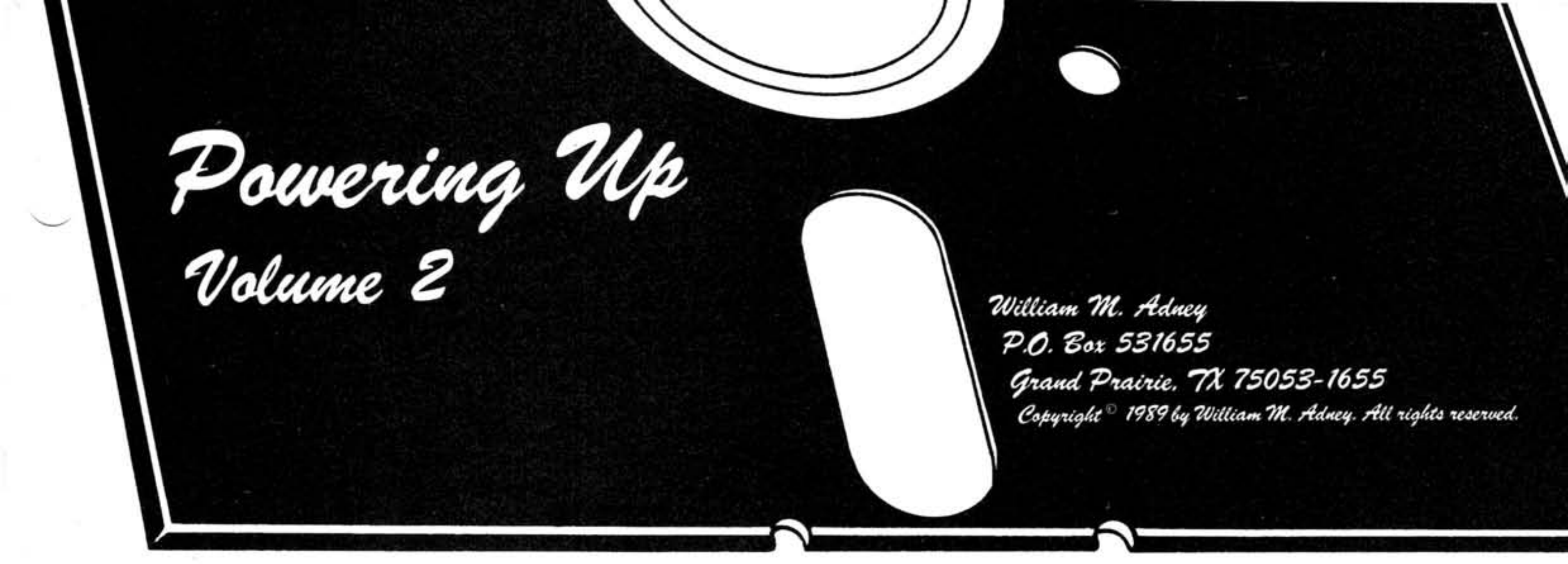
See your local Zenith dealer or call 1-805-964-0733

DMA Technologies, Inc.
601 Pine Ave., Goleta, CA 93117

Call 805-964-0733 for Information
Fax Inquiries: 805-964-0734

Reader Service #197

EXPLORE NEW WORLDS WITH HUG GAME SOFTWARE



Powering Up

Volume 2

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

How Disks and DOS Work Together

Understanding how the DOS works with disk information can sometimes be essential when you are working with a computer system, especially when something goes wrong. Some people will tell you that you don't need to know any of this technical mumbo-jumbo. Unfortunately, these people don't realize that many HUG members do not have immediate access to a knowledgeable friend or a computer store that can provide help when things go wrong. But if you understand some of these fundamentals, it can help you avoid potential problems, as well as figure out how to recover when something happens.

Like all of the POWERING UP articles, this one is intended for the non-technical user who wants to know a little more about computer systems, but does not want to know all of the gory details about exactly how the system works. In this article, we will examine how disks and DOS work together to perform the general functions of starting the operating system and generally accessing a disk drive. And you will generally see how the relationship between the system ROM really works from a user perspective in terms of the various "pieces" of DOS. Let's begin by looking at how DOS is generally constructed.

The Four Parts of DOS

As mentioned in the original *Powering Up* book (Chapter 2), the primary function of an operating system is to allow users and programmers to conveniently access a computer system's resources. Resources, in this context, simply means the keyboard, memory, disk drives, CRT or monitor, printers, and so on. As an example, pressing the *A* key on the keyboard results in the display of an "A" on the CRT. The computer hardware and the operating systems take care of interpreting the electrical signals from the keyboard and translating them into video display form. Operating systems also provide special features which aid the programmer who develops application software. In short, operating systems, such as DOS, take care of the details of resource management and use of a microcomputer system.

As you might expect, all operating systems have some kind of basic organization, and DOS is no exception. Like many operating systems, DOS is constructed in modules, which are nothing more than special "programs" that are required to help you use your computer system. Each of these modules has a specific function or purpose, which is part of the design of DOS.

DOS has four major parts or components. These components are the: Boot Loader, Basic Input/Output System (BIOS), the System Kernel, and the Command Interpreter. All of these components are separate, identifiable "programs" that work together. In order to understand how DOS works, it is important to recognize what each component does, and in general, how it does it. Each one of these components is located somewhere on a disk, and we will look at them in that order.

The Boot Loader

The Boot Loader (technically called the bootstrap loader) is a very short and simple program that provides the beginning instructions to load the DOS into memory from a disk. You may remember from a previous article that its real name is the bootstrap loader since the program was designed to help the system "pull itself up by its bootstraps". The name has been shortened to Boot Loader by custom and usage.

The Boot Loader is always located in the very first part (called sector zero) of a floppy disk or hard disk partition. In addition to the Boot Loader, this first sector also contains information on the specific type of disk format, such as whether a disk has information on one side or two. Unlike every other program on a disk, the Boot Loader does not have a file name (or a directory entry), and you won't see it unless you use a special command like DEBUG to look at the first sector of a formatted disk.

Because the Boot Loader is hardware-specific, it is generally written by each manufacturer of computer hardware, like Zenith or IBM. Although all Boot Loaders do about the same thing, there are slight differences in each manufacturer's "program" because of minor hardware differences. And because the Boot Loader *is* hardware dependent, the discontinued Z-100's (which is *not* PC compatible) Boot Loader is quite different from the one used on all Zenith PC-compatible computers.

Although we'll take a look at the subject in more detail in the next article, it is important to note that the Boot Loader (actually part of the Boot Sector) is *always* written on every floppy disk and hard disk partition when the FORMAT command is executed. In addition, the FORMAT command also records certain information about the disk in the Boot Sector, but that will be discussed in the next article.

The Basic Input/Output System (BIOS)

The Basic Input/Output System, called the BIOS, is also a hardware dependent component (or program) of DOS. It is also written by each manufacturer of microcomputer hardware like the Boot Loader. But before we take a look at what some of the details of the BIOS does, let's take a quick look at why the BIOS is a separate part of DOS.

By design, the BIOS is a separate and distinct DOS component because that makes it very easy for manufacturers to use different computer hardware to run the basic MS-DOS. Since a BIOS is typically written by a computer manufacturer, it is one of the two DOS components (the other is the Boot Loader) that must be "fixed" for each kind of computer hardware. Fixing these two components is obviously much easier than changing every single program that is provided with DOS, and that is the advantage. For example, MS-DOS can be easily modified and customized to run on a specific PC compatible, such as the Z-386, or on a Z-100, even though the internal hardware of each computer is quite different. That particular feature — the separate BIOS — is one reason that many different brands of microcomputers can use MS-DOS.

The BIOS is really just a special program that has a standard file name. A standard file name is required so that the Boot Loader will not have to look for other variations of the name. In a generic MS-DOS system, it may be called something like IO.SYS. In most PC-compatible systems, the BIOS in virtually all current DOS versions (including Zenith) usually has a file name of IBMBIO.COM to ensure compatibility. Older versions of Zenith MS-DOS for both the Z-100 and PC compatibles used the IO.SYS file name.

From a user viewpoint, the BIOS is really a simple program because it acts as a "translator" between the computer hardware and all programs that run on that computer, including the DOS. For example, the BIOS contains a table of all required values to translate requests for access to any kind of valid disk drive. Because this "translation table" defines the valid types of disk drives for that DOS version, it needs to be changed as new kinds of disk drives are available. This explains why the version 2 releases of DOS do NOT support 3.5 inch disk drives — there is no table for them. For Zenith MS-DOS, 3.5 inch, double-density drives (720 K) were supported in MS-DOS 3.10 for PC compatibles (not for the Z-100), but the high-density 1.44 MB drives were not supported until a later version. In other words, you could add the appropriate 3.5 inch drive hardware to a computer using some version 2 release of DOS, but you could not get it to work because there was no table for it, although you could write a device driver for the drive if you knew how.

In technical terms, the primary function of the BIOS is Input/Output (I/O) handling. The BIOS receives input from an application program or DOS (the System Kernel to be specific), translates it to work correctly on the computer hardware, and then outputs (i.e., sends) the request to the appropriate hardware such as a disk drive.

In summary, the BIOS is a hardware-dependent component of DOS. It is developed and customized by each computer manufacturer depending on the specific type of internal hardware that are part of each computer. The BIOS acts as a translator between a DOS or a program and the computer hardware. Because of differences in internal hardware and the associated BIOS translation tables, programs developed for one specific type of computer hardware will not run on another. This explains why programs that are developed for a PC compatible computer, such as a Z-286, will *not* run on a Z-100 system — the hardware (and the associated BIOS) is different.

The System Kernel

The third major component of the DOS is the System Kernel. Because the System Kernel is *not* hardware dependent (unlike the Boot Loader and BIOS), there is no reason for a computer manufacturer to change this "program" because it is independent of the hardware.

The System Kernel essentially contains the system interface used by most programs. It includes the system management functions for file handling and memory, as well as the standard system calls.

Like the BIOS, the System Kernel is also a special type of program which has a specific file name. In a generic MS-DOS system, it may be called something like MSDOS.SYS. For most current PC-compatible systems, it is usually called IBMDOS.COM for compatibility.

The System Kernel performs several major functions that are critical to the correct operation of the DOS and the computer system. These functions are file management, memory management, and Input/Output (I/O) handling.

The file management function is responsible for access (i.e., reading and writing) to disk files. As a file is read from or written to disk, the file management function groups the data into "blocks" so that disk access is more efficient. The file management function is also responsible for allocating the disk space required to store a file and determining the amount of free space on a disk. File management keeps track of the files on a disk by maintaining the disk directory, which includes such information as the beginning physical location and the size of the file. Disk space allocation, both usage and free space, is determined through the use of the File Allocation Table (FAT) that is maintained by the file management function of the System Kernel. We'll take a look at how the FAT performs its functions in the next article.

The memory management function controls the use of Random Access Memory (RAM) which contains programs and/or data. This function allocates (i.e., assigns) a specific amount of memory based on a request for memory space from an application program. When that memory space is no longer needed, memory management can recover memory space by loading new programs and/or data into the same memory area. This recovery process actually overwrites the old programs and/or data with the new.

The I/O handling tasks are usually "shared" by the System Kernel and the BIOS. This occurs when a program issues a special type of "call request" for service from the DOS. The "call request" may be called a DOS system call, a function request, or a service routine. Regardless of the name, standard system calls or functions are generally provided by all operating systems. An example of a standard system call is to "print a character string on the CRT". In assembly language, a certain register (AH) is set to the value 9 to indicate this function, and the string to be printed is stored in a defined place in memory. Then an instruction is sent to the system to print the string. The System Kernel identifies the request, goes through the BIOS (for translation to the hardware), and prints the string on the CRT.

If you are considering doing any programming, it is good programming practice to use the DOS system calls whenever possible. This allows for the maximum amount of hardware compatibility and also provides some other support features (e.g., I/O redirection) that are not available when the BIOS is used directly.

In summary, the System Kernel is a hardware independent component of DOS. It performs three functions: file management, memory management, and I/O handling. The I/O handling function is usually shared with the BIOS. The system calls usually remain stable so that upward compatibility between DOS releases can be maintained, although new system calls are usually added in each successive DOS version. In some cases, application programs may not work correctly or at all if the correct DOS version is not used.

The Command Interpreter

The last major component of DOS is the Command Interpreter. It is the only component of DOS that has a file name that you can see with the DIR command, and its file name is COMMAND.COM. Even though both the BIOS and the System Kernel have file names, the names are "hidden" from the DIR command which cannot display them.

The primary function of the Command Interpreter is to act as the interface between the computer user and the system. It is responsible for reading the commands entered at the system prompt (e.g., A>), finding a program in memory (an internal command) or on disk (an external command), copying that program into system RAM (if required), and beginning program execution. Because of the size and complexity of the Command Interpreter, it contains three major sections of program code — resident code, transient code, and initialization code.

The resident portion of the Command Interpreter allows the DOS to load programs into memory and handle device errors. As the name implies, this part of the code is loaded into, and remains resident in, memory (i.e., RAM) during normal computer operation. It also checks the transient portion of the Command Interpreter, and reloads the transient part of COMMAND.COM if it has been overwritten by another program.

The transient portion of the Command Interpreter contains all of the MS-DOS internal commands, as well as the processor for the batch files. This portion of the program contains the search order for the MS-DOS commands — internal, COM, EXE, and BAT files in that order.

The initialization portion of the Command Interpreter is only resident during the initial boot of the system. It contains the code necessary for the setup of the AUTOEXEC.BAT file execution and also determines the appropriate memory address for program loading. After performing its function, this part of the Command Interpreter is overwritten by the first program that COMMAND.COM loads after system startup.

The DOS Components

DOS has four components: the Boot Loader, the BIOS, the System Kernel, and the Command Interpreter. The Boot Loader is a small, hardware-dependent program called by the ROM monitor to help load the rest of DOS. The BIOS is a hardware-dependent program that acts as a translator between the computer hardware and all software, including DOS, that runs on the computer. The System Kernel is a hardware-independent DOS component, and it provides file and memory management functions, as well as sharing the I/O handling function with the BIOS. The Command Interpreter is the primary interface between the user and DOS. It contains the internal DOS commands and processes all command line entries, including batch files. The Command Interpreter consists of three types of program code: resident, transient, and initialization. The resident portion loads programs into memory and handles device errors, as well as reloading the transient portion when required. The transient portion contains all of the internal DOS commands, as well as the batch file processor. And the initialization portion sets up the system and runs the AUTOEXEC.BAT file if one exists.

DOS Error Messages

By now you may be wondering why knowing a little about these DOS components is important, especially with such a long introduction. Well, have you ever seen a message like: "Insert disk with COMMAND.COM in drive A: and press any key to continue"? If you know that the Command Interpreter is attempting to reload the transient portion of COMMAND.COM and can't find it, then the reason for the error message is obvious. Of course, you can always just blindly follow the instructions to get things working, but you can also prevent these kinds of situations with a little knowledge of what's possible.

In short, a working knowledge of what is going on in DOS can help avoid all kinds of problems, including the "Abort, Retry" messages that are the subject of a separate article in this series. For now, let's continue with the original subject to see how DOS and disks work together.

Disks and DOS

As previously mentioned, the FORMAT command always writes the Boot Loader on the very first sector (called the Boot Sector) on a floppy disk or hard drive partition. And in order to boot a DOS-based computer, all four components — the Boot Loader (in the Boot Sector), the BIOS, the System Kernel, and the Command Interpreter (COMMAND.COM) — must be on the disk in the boot drive. Now let's do a quick review of what happens when you boot the computer, either by powering it on or with CTRL-ALT-DEL.

As discussed in the article on "Using the Zenith ROM Monitor" (*Powering Up — Volume* 2), you will recall that one ROM function is to run its bootstrap loader program. If the ROM Monitor can read the Boot Sector, then it passes control to the Boot Loader that is physically on the disk (always created by the FORMAT command). Then, the Boot Loader attempts to load the second component: the BIOS, which must be the very first file (e.g., IBMBIO.COM) on a disk for most DOS versions. The BIOS must be loaded so that DOS knows how to communicate with the computer hardware. If the BIOS is loaded successfully, the System Kernel (IBMDOS.COM) is then read from the disk and loaded into memory. The System Kernel provides file management, memory management, and I/O handling functions. If that is successful, the Command Interpreter is loaded so that we can talk to DOS. The complete program contains the resident, transient, and initialization code. Among other things, the initialization portion of the Command Interpreter looks for a file name of AUTOEXEC.BAT and runs the batch file if it is found in the root directory of the boot drive. Although all of that sounds great, let's see if we can figure out a way to verify any of this.

Using CHKDSK

The CHKDSK program should be one of your most-often-used DOS programs because it can tell you so many things. In fact, you can use CHKDSK to see if a disk is bootable, although many people don't know that. To check this out, take a new disk and FORMAT it (e.g., FORMAT B:). After FORMAT is complete, use the CHKDSK command (e.g., CHKDSK B:) to see what is on the disk. If you FORMAT a 3.5-inch double-density floppy (720 K), you should see a report something like Figure 1.

```
730112 bytes total disk space
730112 bytes available on disk
655360 bytes total memory
487712 bytes free
```

Figure 1
CHKDSK Report with No System Files

If there are no bad sectors, then you will have a total of just over 730,000 bytes of disk space as shown in Figure 1.

Now use the SYS command (e.g., SYS B:) to transfer two of the system files to the disk, and run CHKDSK again. You should see a report like Figure 2.

```
730112 bytes total disk space
 55296 bytes in 2 hidden files
674816 bytes available on disk
655360 bytes total memory
487712 bytes free
```

Figure 2
CHKDSK Report with Two System Files

Notice that CHKDSK now reports 2 hidden files that use 55,296 bytes of disk space. Actually, these two files are nothing more than the BIOS (IBMBIO.COM) and the System Kernel (IBMDOS.COM) which you cannot see with the DIR command (try it!) because they are hidden files. But the disk is not bootable (try it!) because it does not contain all four DOS components — the Command Interpreter is missing because the SYS command does NOT copy it to a disk. To make the disk bootable, you must use the COPY command to copy the Command Interpreter (e.g., COPY COMMAND.COM B:) to the disk. Now run the CHKDSK command again, and you will see a display like Figure 3.

```
730112 bytes total disk space
 55296 bytes in 2 hidden files
 25600 bytes in 1 user files
649216 bytes available on disk
655360 bytes total memory
487712 bytes free
```

Figure 3
CHKDSK Report with Three System Files

Notice that a new line is added to the report: "25600 bytes in 1 user files", and that file is COMMAND.COM which you can see with the DIR command (check it!). Now that the disk contains all four DOS components, it can be used as a system disk to boot the system (try it!).

Continued on Page 30

Graphics Printer or Epson FX Part 6

Downloading and Graphics

John A. Day
5 rue Sauer
77500 Chelles, FRANCES

Downloading fonts are only available using the 'F' command set. They give you an extra set of characters:
• The standard draft set,
• The standard NLQ set,
• Your own draft set.

You can design whatever characters you want within the standard 11×8 matrix, load them into the printer memory, and use them in your documents. I use this option two ways:

• The FX emulation on my printer doesn't give me IBM characters; so I download a full set on power-up from a batch file. This lets me use my printer with the same characters as my XW-148 standard screen, but with the extra flexibility of the full 'F' command set.

• For publishing technical documents, there are a lot of useful symbols which aren't in any standard set, such as double negatives for boolean algebra, symbols for logical 'OR' or logical 'AND', useful arrows and so on. I have two or three standard assortments, where these characters replace other symbols which we consider less useful. For each document, I download the proper character set just before printing it. The double dots used for drawing double-strike and NLQ pin patterns in this series of articles were defined in pretty much this way, although the actual printing was done in graphics mode.

A full draft font takes up about 3K of memory. The manufacturer-supplied fonts are in ROM (read-only memory). Every printer has a fair amount of RAM (random-access memory) for buffering print lines. A DIP switch setting lets you allocate part (or all) of this memory for an extra font instead of buffer space. If all the RAM memory is switched over to a download font, you may slow down your printer slightly. To see if this is so, take an ASCII file of about 20 or 30 lines — a source program will do fine as long as it's not in BASIC binary format — and send it to the printer with

```
type program >prn
```

If the system prompt comes back well before the printer stops, then it's still buffering, and you will lose a few milliseconds at the start of each line on data transmission from the computer.

Usual download practice is to copy the existing ROM draft font into the RAM area as a starter, then send any modified characters from the computer to replace the previous values. This lets you use all unmodified characters as usual. You can either keep the printer working on the downloaded font only, or swap between both to get two complete fonts. You could also change download characters while printing, but I think this would be going too far.

I have tried two ways of switching fonts, and both work fine. For the technical documents, I customized my word processor ribbon colors so that "RED" switched me to the download font with the extra characters, and "BLACK" brought me back to the standard font. With WordStar 3.3, 3.4, 4.0, this is the ^Y toggle. For my IBM characters on an FX emulation, the italics select sequence is customized to switch to the ROM font and set italics, and the deselect sequence switches back to the download RAM font and clears italics. You would need at least WordStar 4.0 to do this, because the control sequences become too long for 3.3 or 3.4. I didn't define any italic download characters because there was no room for them among the IBM characters.

The font select sequences are easy. When you switch the printer on, the RAM is always clear and the machine uses the ROM. *ESC ":" 0 0 0* copies all of the ROM character generator into RAM, giving you identical characters for draft ROM and download RAM. The three zeros are for compatibility with bigger systems, in which you may have more than one download font at a time. The second '0' indicates which ROM to copy, and the third which RAM to copy it to. The FX only has one of each, but it uses the compatible form of the command. *ESC "%" 1 0* switches the printer to the downloaded RAM characters, and *ESC "%" 0 0* switches back to the standard ROM. Here again, the last zero can be used on bigger systems to choose between several RAMs.

Downloading is very easy — once you've got the numeric definition for the character. This takes 12 bytes: 11 for the 11 columns, and one more byte for miscellaneous information.

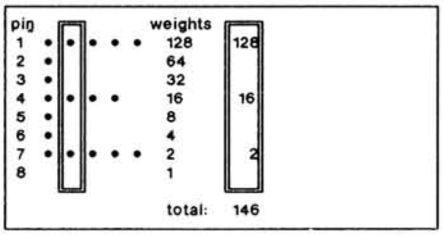


Figure 6-1
Bytes for a Download Character Definition

The weights for each pin are shown

on the above diagram. For each column, you "just" add up the weights of the dots used, and you have the value to put in the download definition; for the capital 'E', 254 0 146 0 146 0 146 0 130 0 0.

The numerous zeros shouldn't surprise you too much. The trailing zeros correspond to the three-column space (the twelfth position is always zero, so it isn't included) before the following letter. After firing a pin, you *must* give it 1/60" to recover, so after each non-zero bit in a byte, the next byte will have a zero in the corresponding position. '254' corresponds to all pins except number 8, so the next byte could only be '0' or '1'. Alternate zero and non-zero bytes correspond to the squarish characters on 1/60" centers, whereas characters with staggered pins will have a series of non-zero bytes. But if you use the same weight in two consecutive bytes, you risk damaging your print head. You can try and figure out the byte values for the letter 'A' given in the first of this series of articles; you should find *30 32 72 128 8 128 72 32 30 0 0*.

I said "just" add up the values in quotes, because when you've figured out three or four characters by hand, you'll be looking for an easier way to do it. Once I found that downloading was what I needed, I wrote a BASIC program that would let me maintain font files on disk, bring characters up on the screen, and draw or modify them using the cursor keys. A full program for this can run to several thousand structured lines, covering multiple file and printer formats, but I have stripped one down to one format and minimum error checking, and will give it, with comments, in the final article.

There is one more byte to code, called the attribute byte, which indicates whether to use pins 1 thru 8, or 2 thru 9. It also carries the information on proportional spacing which will allow the printer to squash the character to under 1/10". The high-order bit is usually a 1 (weight 128); if it is zero, the character is shifted down on to pins 2 thru 9. The three following bits give the first byte to print in proportional spacing, and the last four give the last byte to print, including the white margin. The weight to use for the left margin is 16. A normal full-width character prints 12 bytes from 0 to 11 (the last three being 0) on pins 1 thru 8. This gives the value 128 + (0 × 16) + 11, or 139, which gives no difference between fixed or proportional spacing. The attribute must specify a minimum width of 5 bytes, the last byte cannot be greater than 11, and you must allow at least one zero byte to the right. Look at the letter "i" in Figure 4-2 of the previous article, for instance. The data bytes are *0 0 34 0 190 0 2 0 0 0 0* which means that in proportional spacing, we will want to print from byte 2 to byte 9 (three bytes past the last print position, the usual inter-letter gap). The attribute byte will be 128 + (2 × 16) + 9, or 169. The character width is (9 + 1 − 2), or 8, which is the value you will find in the FX printer handbook.

To download the character k, you send the sequence *ESC "&" 0 "kk" attr b0 b1 b2 ... b10*, where "attr" is the attribute byte, and "b0" ... "b10" are the eleven character data bytes. To send the character data for "i" to its normal position, for example (this doesn't serve any useful purpose, because copying ROM to RAM means the data was there already, but I need an example . . .): *ESC "&" 0 "ii" 169 0 0 34 0 190 0 2 0 0 0 0*.

In BASIC, use *CHR$(..)* for everything that's not in quotes. You can use the fourth and fifth bytes on an FX and on *most* emulations to send several consecutive characters with one command sequence; give the first and last positions to load, then the right number of 12-byte data sequences. For example, to load data for *"i"*, *"j"* and *"k"*, send *ESC "&" 0 "ik"* ... and 36 bytes of data for the three values. I never bother with this possibility, as it would make the disk structure of my download fonts too complicated; by repeating the Escape sequence for each character, I slow down loading by less than 40%. Since downloading takes two or three seconds and I don't do it all the time, the extra second or so doesn't bother me. For big printers with several download fonts, the third byte indicates the font to use; for FX emulations, only font 0 is available.

NLQ fonts take up much more space, and can only be downloaded if you have extra RAM. The Brother LQ200 PCB add-on that I use on my printer has two NLQ ROM fonts and 16K for downloading. If the printer is in draft mode when it receives download instructions, it uses the standard RAM, whereas if it's in NLQ mode, it automatically addresses the extension RAM on the PCB. NLQ download sequences start with *ESC "&" n "ik"*... like draft downloading; but the value *"n"* will be used to indicate print density, and the data bytes have a more complicated format, with a count byte giving the number of data columns following, and three bytes per print column to give the 18 pin firing values from top to bottom. NLQ can use all 9 pins at the same time. I'll come back onto this one for the last article, but if you want to use this option, you'll certainly have to read through your printer handbook.

Printer Graphics

You can also send values directly from your computer to design special characters, without bothering with downloading. In this case, the printer doesn't use its own character generators; it takes the pin patterns exactly as you send them. You can define a symbol to any convenient width, and if you advance the platen 1/216" (one motor step) with *ESC "J" 1* and overprint, you can have NLQ symbols. All you have to do is figure out the byte values and, as with downloading, if you have a lot of symbols to design, you'd best start off by designing a program to draw your symbols on the screen. As a first example, we'll see how we can design a curly arrow to point to something on the line below it:

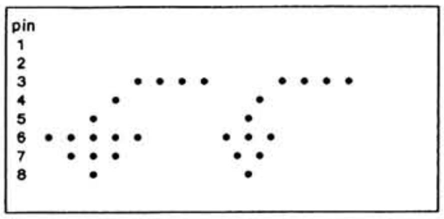


Figure 6-2
Bytes for a Graphics Symbol

The left-hand sketch fires pins on a 1/60" horizontal pitch. The bytes are quickly calculated using the same weights given for downloading:
4 6 15 22 36 32 32 32

The right-hand sketch uses pins staggered on 1/120" centers, so we need one byte for every 1/120":
4 2 13 18 4 32 0 32 0 32 0 32 0
not forgetting a final zero to be sure that pin 3 will have enough time to recover before the next character. The right-hand symbol is narrower than the one on the left, but takes nearly twice as many bytes to describe it. However, using 1/120" centers gives much neater designs. Both these symbols will be plotted at full head speed, but using *ESC "k"* for the first (one byte of data per 1/60"), and *ESC "Y"* for the second (one byte of data per 1/120"):
ESC "K" 8 0 4 6 15 22 36 32 32 32 ESC "Y" 13 0 4 2 13 18 4 32 0 32 0 32 0 32 0

ESC "Y" corresponds to the print mode used for draft text. All the graphics sequences use the same coding to indicate the number of data bytes: *ESC "." n1 n2* ..., where the number of bytes is equal to n1 + (256 × n2), and n1 and n2 are both less than 256. Also, the total number of bytes must fit on your printer line; it depends on the platen width. In the first example, there are 8 bytes; in the second, 13. Don't forget that you need special handling to print CHR$(13) in BASIC; see article 4 in this series.

Both these symbols can be inserted in the middle of a line of text, but they will throw alignment off to the right. The first is 8/60" wide, or 2/15", and the second is 13/120" or just over 1/10". If you want to keep your 1/10" pitch across the page, you must pad them out to a multiple of 6 or 12 characters. Conversely, you can slip in a dummy graphics sequence, with all data bytes 0, to pad part of a line in increments of 1/60" or 1/120". *ESC "K" 3 0 0 0 0*, for instance, will insert a pad one-half space wide in a text. Some word processors use this as an option to micro-justify text. I tried it with WordStar 4.0. I got tired before the end of the first page; at each space, the head stopped and backed up as the printer switched to graphics mode, then stopped and backed up again as the printer reverted to text mode.

Two other graphics commands print the same way, but at half speed. *ESC "L"* can print at 1/120" pitch. *ESC "Z"*, the one I usually use corresponds to printing in NLQ: 1/120" between successive strikes on a pin, but staggered pins at 1/240". The following sketch shows a 1/20" block in each print mode, with the maximum number of pins:

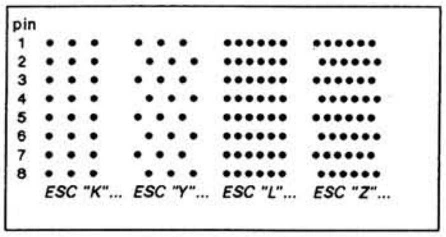


Figure 6-3
The Four Basic Graphics Modes

The four command sequences for these blocks would be:
ESC "K" 3 0 255 255 255
ESC "Y" 6 0 170 85 170 85 170 85
ESC "L" 6 0 255 255 255 255 255 255
ESC "Z" 12 0 170 85 170 85 170 85 170 85 170 85 170 85
(not including any spaces to the right — be careful of the following character for *ESC "Y" ..., ESC "Z" ...!*)

ESC "Z" ... can be used to obtain very high quality graphics:

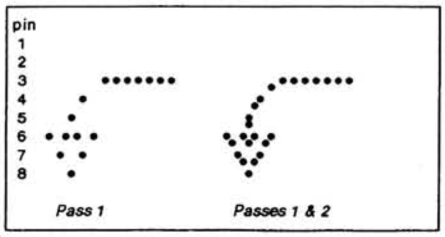


Figure 6-4
Bytes for High-Resolution Graphics

ESC "Z" 24 0 4 0 2 4 9 4 18 0 4 0 32 0 32 0 32 0 32 0 32 0 32 0 32 0 ESC "J" 1 CR ESC "Z" 24 0 0 4 0 2 8 18 0 4 32 0 0 0 0 0 0 0 0 0 0 0 0 0 0

The first *ESC "Z"* sequence sends 24 bytes of data to draw half of the curly arrow. *ESC "J" 1* rolls the platen up one motor step, that is less than half the distance between two pins, and the CR brings the head back to the left. The second *ESC "Z"* sequence fills in the gaps. Don't use *ESC "j" 1* to bring the platen down again to where it was; because of backlash, it won't move. Finish everything you have to print on the line before you add in the second pass, then advance 35/216" to make up the 1/6" line feed: *ESC "J" 35*. Note that to print the same width, *ESC "K"* needs half as many bytes as *ESC "Y"*, and *ESC "L"* takes half as many bytes as *ESC "Y"*, and *ESC "L"* takes half as many *ESC "Z"; ESC "Y"* and *ESC "Z"* need a lot of zeros, corresponding to positions where a pin that's just fired can't fire a second time.

Now that's fine, but when you look at your printer handbook, you will find four more bit image modes: *ESC "*"* n, where *n* is 4, 5, 6, or 7. What are these for? The answer lies in aspect ratios. You can easily fill the whole page with graphics; use *ESC "K"* ... to print a whole line (480 bytes, or more with a wide platen), then *ESC "J" 24* to advance 1/9" and continue. But if you try to draw a square 100 points wide by 100 points high, it won't be. The horizontal pitch is 1/60" and the vertical pitch is 1/72" between points, so your "square" will be 1.67" wide and 1.39" high, an aspect ratio (width to height) of 1.2.

ESC "" 5* ... and *ESC "*" 7* give you graphics with horizontal pitches of 1/72" and 1/144", respectively. With these, you can set up simple graphics with an aspect ratio of 1, which is easier. The 100 × 100 point square with *ESC "*" 5...*, for example, will come out 1.39" × 1.39", which is what we want. But things get even worse when you look at your CGA display screen. In high-resolution, black-and-white graphics mode (640 × 200 points), a pixel on a 13" monitor is about .015" wide and .034" high, which gives an aspect ratio of 0.44. This is not any convenient mutiple of the printer aspect ratio. So *ESC "*" 4...* and *ESC "*" 6...* give you two pitches which, hopefully, may correspond to your screen. *ESC "*" 4...* is pitched at 1/80", and *ESC "*" 6...* at 1/90". This gives aspect ratios of 0.9 and 0.8, respectively. 0.9 ties in quite well with the 0.44 of a CGA screen. By printing each screen pixel as two dots on two adjacent pins, a 640 × 200 screen will print as a 640 × 200 dot copy measuring about 7.1" × 5.5". This means that you can get a circle to copy as a circle, whereas with other pitches you will end up with an ellipse.

For color graphics, 320 × 200 points with each pixel .030" wide and .034" high, you can print dot for pixel, which will give you a 320 × 200 dot copy measuring about 3.5" × 2.7"; or you can print 2 × 2 dots to each pixel, to get the same size on paper as for the black-and-white screen.

You can also get the standard aspect ratio of 1.2 by using a block of screen pixels, 8 wide by 3 high. This gives a blob on the screen .12" wide and .1" high, about 7.5 times the size of a basic printer dot. I often use a 16 × 6 blob for designing graphics symbols, working from a x15 enlargement on transparent film, as the horizontal size can be divided into 4 parts for high-resolution graphics, and the vertical into 2 or 3 for two- or three-pass printing.

The last article in this series will give programming hints for designing download characters and graphics printouts. I shall give two simple programs, which you can develop yourself to cover your particular requirements. ✱

Continued from Page 27

As a part of this testing, take another new disk and FORMAT it using the /S (System) switch (e.g., FORMAT B:/S). Run CHKDSK on this disk, and you should see the identical report that was displayed before, such as shown in Figure 3.

There are several important things to note about this testing. First, the number of bytes displayed by CHKDSK will depend on what disk size, format, and DOS version you are using. These examples were created using MS-DOS 3.3 Plus, and earlier DOS versions typically have smaller file sizes because they do not have all of the features of 3.3 Plus. Don't be surprised if the numbers displayed by CHKDSK do not exactly match these examples because they probably won't unless you are using the same DOS version I am. Of course, different sizes and types of disks have different capacities, so those numbers will be different too.

I hope you have also noticed that there are two ways to create a system disk — one using the SYS and COPY commands, and one using the FORMAT command. It is always much easier to use the FORMAT command on a new disk, but you can always add the system files to a new disk so long as no other files have been added. Some earlier DOS versions are quite fussy about where the BIOS and System Kernel files must be, and a few versions will not allow you to SYS a disk even when all files have been deleted.

One other interesting note about the BIOS and System Kernel files. I mentioned that they are "hidden files" which means that you cannot see them with the DIR command. Actually, they are hidden from nearly all DOS commands. For example, you cannot delete them with the DEL command and you cannot COPY them either, even though you know the file names. If you want to recover the space used by the BIOS and System Kernel files, the most common way to do that is simply to run FORMAT again without the /S switch, after first backing up any other files that may be on the disk of course.

Powering Down

Understanding how disks and DOS work together is not really too complicated, but there is another aspect of this. In the next article, we will take a look at "What a Disk Really Contains", and you will see how all of the information presented in this article actually fits on a disk. ✱

dBASE III Part 2

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

In my last article, we created two data bases: PROJECTS and PROJXREF, the two being related by the common field PROJNR (project number). In this article, I will discuss the EDIT, BROWSE and APPEND commands and show how to delete and recall records. But first we need actual data.

STATUS	DWGNR	REV	PROJNR	ENGINEER	STARTDATE	ESTDATE
	5962-88472	N	E1201	ABC	09/15/89	03/15/90
	5962-88654	N	E1189	JAD	09/13/89	03/31/90
	5962-88470	N	E1190	ABC	09/19/89	04/19/90
	5962-88466	N	E1191	ABC	09/19/89	03/19/90
	5962-88465	N	E1194	JZB	09/19/89	03/19/90
	5962-88867	N	E1198	JZB	09/20/89	03/31/90
	5962-88763	N	E1192	JZB	09/20/89	03/31/90
	5962-88462	N	E1195	ABC	09/21/89	03/21/90
	5962-87811	A	E1197	ABC	09/21/89	03/21/90
	5962-88461	N	E1196	ABC	09/21/89	04/21/90
	5962-88463	N	E1202	ABC	09/21/89	05/21/90
	5962-88464	N	E1206	ABC	09/21/89	03/21/90
	5962-88449	N	E1205	ABC	09/21/89	03/21/90
	5962-88454	N	E1203	ABC	09/21/89	03/21/90
	5962-88985	N	E1204	JAD	09/21/89	04/21/90
	5962-88984	N	E1207	JAD	09/28/89	03/28/90
	5962-88983	N	E1208	JAD	09/28/89	03/28/90
	5962-88979	N	E1212	JAD	09/28/89	03/28/90
	5962-88972	N	E1209	JAD	09/28/89	03/28/90
	5962-88500	N	E1210	JZB	09/28/89	03/28/90
	5962-88478	N	E1211	JAD	09/29/89	09/29/90
	5962-88455	N	E1213	TTT	09/29/89	03/29/90
	5962-88477	N	E1214	TTT	09/29/89	03/29/90
	5962-88452	N	E1215	TTT	09/29/89	03/29/90

Figure 1
PROJECTS Data Base

Figure 1 shows the data for 24 records for the PROJECTS data base. Before you can enter data, however, you must open the data base with the USE command. Also, since this data base has an index PROJDWRV, this index must be included in the USE command statement; otherwise, any records appended to the data base will not have their keys included in the index. At the dot prompt then, type

```
USE PROJECTS INDEX PROJDWRV
```

Now you can begin entering data into the data base with the APPEND command. For the present, leave the STATUS field blank. Note that dates are entered in the form mm/dd/yy. If you enter an "impossible" date such as 13/01/89, you will see the error message "Invalid date. (press SPACE)" on the right side of the top line. Dates can be entered in five other formats other than American; for example, they can be entered in British format, which is dd/mm/yy. To change to another format other than American, you must use the SET DATE command. To enter data, you use the APPEND command, which you can type at the dot prompt or simply press F9.

You should now see a menu at the top of the screen and a series of labeled blank fields below the menu. Each field will have the field name assigned to it during the CREATE process. The cursor should be sitting on the first space of the first field. You can now enter the data from Figure 1. To correct mistakes in data, use the left cursor, backspace or delete keys. The default mode is with INSERT off, that is, you simply type over any incorrect character. If you press the INSERT key, you will see the word INSERT in the right side of the top line. In this mode, all characters to the right of the cursor will move over one space for each character inserted. To advance to the next field, simply press ENTER. Pressing the right cursor a sufficient number of times will also advance to the next field. Note that when you press ENTER for the last field, dBASE automatically brings up another blank record. To avoid this when on the last record, after entering data in the last non-blank field, press CTRL-END instead of ENTER. This causes the last record to be written to disk and terminates the APPEND process.

After you have entered the data for PROJECTS, enter the data from Figure 2 into the PROJXREF data base. Open the data base and its index with the following command:

```
USE PROJXREF INDEX PROJXPJN
```

PROJNR	DEVTYPE	VENDPN
E1201	01	SG117R
E1201	02	LM117HVH
E1201	03	LM137H
E1201	04	LM137HVH
E1201	05	SG117AT
E1201	06	SG137AT
E1201	07	LT117HVH
E1201	08	LT137AHVK
E1189	01	LM124
E1190	01	CD4078BF/3A
E1190	02	4078B
E1191	01	DG508
E1191	02	IH6108
E1191	03	IH5108
E1194	01	201
E1194	02	201A
E1198	01	10516
E1192	01	DG129AGD
E1195	01	AM687
E1195	02	AM6687
E1195	03	AD96687
E1197	01	26LS31
E1196	01	55183
E1202	01	4052B
E1206	01	ADLH0032
E1205	01	2420
E1203	01	7831
E1203	02	7832
E1204	01	6N134
E1207	01	68000-6
E1207	02	68000-12
E1207	03	68000-10
E1207	04	68000-8
E1208	01	NSC800
E1208	02	NSC805
E1212	01	54HC02
E1209	01	54HC86
E1210	01	80C86
E1210	02	80C86-2
E1211	01	SMJ32010
E1213	01	54HC74
E1214	01	54HC374
E1215	01	54HC373

Figure 2
PROJXREF Data Base

This command automatically closes the PROJECTS data base and its index and opens the project cross reference data base with its corresponding index. Once again, use the APPEND command (F9) and enter the data from Figure 2. Note that this data includes at least one record for every record in PROJECTS as shown by the common field PROJNR. Note also that some projects have more than one associated record in PROJXREF, which is the primary reason for having two data bases instead of one. As stated in the previous article, some projects could conceivably cover as many as 30 different device types. To include all relevent fields for each project duplicated thirty times would be an enormous waste of disk space, especially when the number of projects reaches into the hundreds.

To check your work for errors, one way would be to compare a printed list of the data base against the original. For this you can use the LIST command. Open the projects data base with the command USE PROJECTS. (The index will not be necessary for this exercise.) Now type LIST plus ENTER and you will see all fields of all 24 records scroll up the screen without pause. Since the number of characters for each record exceeds the 80-column width of the screen, wrap-around occurs. Not a very satisfactory display and besides, we wanted a printed list. To get the listing to go to the printer, you must add the option "TO PRINT" to the list command. If you do this though, unless you have a printer with a very wide carriage and very wide paper, you will run off the page when printing the last part of the record, unless, of course, you switch to condensed printing. If you have a printer that can switch to condensed pitch (around 16-17 cpi) with external control, you can try this now. However, note that in our PROJECTS data base as it currently exists, certain fields are blank in all 24 records, so why print out blank fields? The LIST command can therefore be modified to include only those fields that actually have data:

```
LIST DWGNR,REV,PROJNR,ENGINEER,
   STARTDATE,ESTDATE TO PRINT
```

Notice in the syntax of this command that the fields to be printed are all separated by a comma. (A space is not necessary between the comma and the next field.) Now look at the printout. You will notice that the first column is the record number, which is usually of no interest to the user. To eliminate the record number from a printout (or screen display), you must use the word OFF in the command statement:

```
LIST OFF .................... TO PRINT
```

(The dots indicate the fields to print.) Fields can be entered in any order.

After you have checked PROJECTS for accuracy, you can do the same for the PROJXREF data base. Again, use the command USE PROJXREF to close PROJECTS and open PROJXREF. Notice that in both listings, since the indices were not used when opening the data bases, the records are in the exact order in which they were entered. To see the effect of turning an index on, enter the command

```
USE PROJECTS INDEX PROJDWRV
```

Now type the same command to print the project records as you did before. Note now that the listing is now in drawing number sequence. This is why the SORT command in dBASE is rarely used, since indexing accomplishes the same result without creating a second data base. (If you use the SORT command, it will create a sorted version of the original data base, but with a different name, and leave the original data base as is. This would be fine if you never added or deleted records.)

By the way, did you make any typing errors when entering that long LIST command and have to re-enter it? This can really get frustrating if you end up typing a very long command line several times. (A command line can contain up to 254 characters.) For a long time, I wondered why the first few lines at the top of the screen went into reverse video whenever I pressed the ESC key at the dot prompt. Finally I discovered that this activates a feature that is documented only under the command SET HISTORY, which I obviously never bothered to read. In this mode, you can recall previous commands up to the last 20 issued. To illustrate this feature, press ESC at the dot prompt. The first three lines of the screen should go into reverse video. Now press the up cursor. The previous command will be displayed. The up and down cursors can be used to scroll through previous commands. Any of these commands can be edited and executed by pressing ENTER. This is a real timesaver if you ever make an error in typing out a long command line.

The DISPLAY command is virtually identical to the LIST command with two exceptions: (1) DISPLAY ALL will pause every 20 lines with the prompt "Press any key to continue . . ." and (2) DISPLAY by itself will display only the current record. As with the LIST command, DISPLAY can be modified as to scope, fields and conditional expressions such as FOR and WHILE. Adding the modifying expression "TO PRINT" causes the display to echo to the printer. As with LIST, if a record exceeds 80 characters, DISPLAY will wrap around to the next line and may break up a field into two or more lines.

Whereas DISPLAY does not permit editing of a record, the BROWSE command does. To see how BROWSE works, open the PROJECTS data base and then type BROWSE at the dot prompt. You should now see a menu followed by 11 records showing the first 8 fields. Unlike LIST and DISPLAY, BROWSE shows only those fields that will fit within the 80-column screen. To view the remaining fields, use CTRL-RIGHT Cursor to scroll right, CTRL-LEFT Cursor to scroll left. PgUp and PgDn scroll the data base backward and forward, respectively. An options menu can be accessed by pressing CTRL-HOME. The options are:

Bottom — Goes to the last record.
Top — Goes to the first record.
Lock — Defines the number of fields at the left side of the screen that remain stationary during panning.
Record # — Goes to a specified record number.
Freeze — Allows editing of a single field; that is, the other fields are frozen out of the edit process. The chosen field remains in reverse video. This feature is useful if you need to edit a single field and you don't want to take the chance of changing any other fields.
Seek — Appears only when the file is indexed and the index is turned on; it will then locate the first record that matches a specified key expression.

The main feature of the BROWSE command is that it allows editing of existing records and appending of new records. To demonstrate this, we are going to add some information to the PROJECTS data base. If you haven't done so already, open the PROJECTS data base with USE PROJECTS. Leave the index out of the USE command so that the records appear in the order they were entered. Now freeze the EDITIN field by pressing CTRL-HOME and then moving the cursor to highlight the word FREEZE. Press ENTER, then answer the prompt with "EDITIN". You should now see only the EDITIN field in reverse video with the cursor on the first record. Now add the dates from Figure 3 to the corresponding record. Notice that you do not have to press ENTER after entering each date. After you have entered all the dates, press CTRL-END to exit BROWSE. (If you press ESC, changes made to the current record will not be saved. This feature, by the way, is very useful if you are making changes to a record, then discover you are changing the wrong record and have forgotten what was in a particular field before you changed it. Just press ESC and the record remains as it was before you began editing it.)

Sooner or later you will have a need to eliminate one or more records from a data base. This is done with the DELETE command, either at the dot prompt or by pressing CTRL-U when in edit mode. (CTRL-U is actually a toggle; if a record is already marked for deletion, CTRL-U will unmark it.) The nice thing about DELETE is that you can retrieve a deleted record with the RECALL command. This is because the record is only *marked* for deletion, not actually deleted. As I stated in my first article (Oct. 1989), every record of

DWGNR	EDITIN
5962-88472	10/21/89
5962-88654	09/30/89
5962-88470	10/01/89
5962-88466	10/09/89
5962-88465	10/21/89
5962-88867	10/30/89
5962-88763	10/29/89
5962-88462	10/05/89
5962-87811	10/05/89
5962-88461	10/05/89
5962-88463	10/21/89
5962-88464	10/28/89
5962-88449	10/15/89
5962-88454	10/15/89
5962-88985	10/15/89
5962-88984	10/30/89
5962-88983	10/30/89
5962-88979	11/02/89
5962-88972	11/02/89
5962-88500	11/03/89
5962-88478	11/05/89
5962-88455	11/03/89
5962-88477	11/03/89
5962-88452	11/03/89

Figure 3
Additions to PROJECTS

any data base includes a single character "field" not defined by the structure. If a record is marked for deletion, this "field" (or byte) will contain an asterisk; otherwise, it will contain a space. You can see this asterisk if you delete one or more records and then LIST the data base. The deleted records will show an asterisk prior to the first field displayed. Also, in the BROWSE mode, you will see "*DEL*" directly above the menu when the cursor is sitting on a deleted record.

To permanently delete a record, you must use the PACK command, which rewrites the entire data base, leaving out any record marked for deletion. Once this is done, only special file recovery programs outside the realm of dBASE can recover these records. Obviously, you should use PACK only with discretion, possibly reviewing all records marked for deletion first.

Assuming a data base contains deleted records, when do you see these records and when don't you? The answer to this question depends on the command used and the state of the SET DELETED command. SET DELETED is one of the class of SET commands — commands which determine such things as screen attributes, function key assignments, default drive, etc. (For a more detailed explanation, see the SET command in the dBASE III manual.) If SET DELETED is OFF (the default state), the deleted records will be included with all commands. With SET DELETED ON, most commands act as if the deleted record does not exist. For example, LOCATE will not locate deleted records. Certain commands, however, ignore the deletions: DISPLAY, DISPLAY RECORD, GO and GOTO. Also, no records can be RECALLed with SET DELETED ON. The RECALL command is the command used to undelete a deleted record, but SET DELETED must be OFF for it to work. Like most commands, RECALL can be limited by scope and condition. If you want all deleted records recalled, you would use the command RECALL ALL; whereas if you wanted only the current record recalled, you would say simply RECALL.

In typing in the data for projects, the STATUS field was left blank. Since all the projects in the data base are to be considered as active projects, the STATUS field can be edited easily with the REPLACE command. Open the data base with USE PROJECTS, then at the dot prompt type REPLACE ALL STATUS WITH "A". dBASE should respond with "24 records replaced". Note the word "ALL" after REPLACE. Had "ALL" been omitted, only the current record would have been changed. The word "ALL" is an optional part of many commands that is designated by dBASE III as the *scope* of a command; scope defines the amount of the data base to which the command applies. Besides "ALL", scope may be:

RECORD N — A single record in which N is the record number.

NEXT N — The next N records, beginning with the current record.

REST — From the current record to the end of the file.

As with many other commands, REPLACE may be modified by a FOR/WHILE clause. For example, suppose I wanted to discontinue all projects for which "ABC" is the engineer. I could do this very easily with the command

```
REPLACE STATUS WITH 'D' FOR ENGINEER =
   'ABC'
```

If you enter this command, you will see "10 records replaced". Notice that the word "ALL" is not necessary, since a FOR or WHILE condition automatically sets the scope to "ALL". To see the difference between FOR and WHILE, use the REPLACE command to restore the STATUS fields to "A" for all records, then enter the command GO TOP, which sets the record pointer to the "top" or first record of the data base. Now enter the command

```
REPLACE STATUS WITH 'D' WHILE ENGINEER
   = 'ABC'
```

This time only 1 record is replaced. Why? If you look at the first two records (use BROWSE), you will see ENGINEER for record #1 is "ABC" and record #2 is "JAD". The status is replaced with "D" for record #1 because ENGINEER = 'ABC', which meets the WHILE condition. But then dBASE advances the pointer to the next record, which is record #2 for which ENGINEER = "JAD". Thus, the WHILE condition fails and the command terminates.

The REPLACE command can also be used to replace the contents of more than one field within a single command line. For example, I could say:

```
REPLACE STATUS WITH 'D', COMMENTS WITH
   'DISCONTINUED' FOR ENGINEER = 'ABC'
```

Note the comma between the field statements. After you are through experimenting, use the REPLACE command to restore the STATUS field to "A" for all records.

The REPLACE statement can be useful for making mass changes to a data base. For example, suppose you have decided every project number should start with "5962-". This could be a lot of work if you had to do it manually for each record. The REPLACE command can do it automatically:

```
REPLACE ALL PROJNR WITH '5962-' + PROJNR
```

This may seem like strange syntax until you think about what it actually says. What it means is — take the current contents of PROJNR, append it to the string "5962-" and enter this new value into the PROJNR field. Execute this command and then LIST or DISPLAY the data base and you will see that this is exactly what has occurred. The REPLACE command is one of the more useful commands in manipulating a data base, especially when used with string manipulation functions.

In my next article, I will discuss simple programming and creating custom data entry screens. *

Classified Ads

INTERESTED IN YOUR FAMILY TREE? F-TREE is a user-friendly genealogical data base management system for IBM PC compatible DOS 2.+ users. For free details, write MIN DAR CO., Dept. BR, Box 254, Elk River, MN 55330.

FOR SALE: 2 Dual Floppy Disk Drives, H-25 Printer, 8 Ribbons, Excess of 200 disks. Call (601) 384-5886.

Z-100 LOW PROFILE with EASY/PC, new style 8 MHz Motherboard, 768k and Floppy $475; Hayes Smartmodem 1200 $150; 386 AT Z-449 Video Card $175; For Z-100: Z-217 Winchester Controller w/10 MB $250; UCI Memory Card w/2 MB $350; UCI 8 MHz Speed Module $25; Dual 1.2 MB 5-1/4" Drives with Cabinet, Supply, and Adaptor $175. (708) 246-7856.

PAYLOAD * PAYLOAD * PAYLOAD * PAYLOAD

Seagate HARD DRIVES

MODEL	CAPACITY/FORMAT/SPEED/SIZE	DRIVE ONLY	XT KIT
* ST-125	21.4 MEG / MFM / 40 MS / 3.5"	$228.00	$278.00
* ST-138	32.1 MEG / MFM / 40 MS / 3.5"	$271.00	$321.00
* ST-151	42.5 MEG / MFM / 24 MS / 3.5"	$380.00	$430.00
* ST-138R	32.7 MEG / RLL / 40 MS / 3.5"	$263.00	$323.00
* ST-157R	49.1 MEG / RLL / 40 MS / 3.5"	$312.00	$372.00
* ST-225	21.4 MEG / MFM / 65 MS / 5.25"	$199.00	$249.00
* ST-251-1	42.8 MEG / MFM / 28 MS / 5.25"	$324.00	$374.00
* ST-4096	80.2 MEG / MFM / 28 MS / 5.25" FH	$548.00	$598.00
* ST-238R	32.7 MEG / RLL / 65 MS / 5.25"	$216.00	$266.00
* ST-277R-1	65.5 MEG / RLL / 28 MS / 5.25"	$366.00	$426.00
* ST4144R	122.7 MEG / RLL / 28 MS / 5.25"	$628.00	$688.00

* All 3.5" drives are shipped with a 5.25" frame kit for mounting.

* XT KITS include drive, controller, cables screws and manuals.

*** 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. $38.00

Z-150 SERIES HARD DISK DRIVE KIT

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

* ST-125/Z150 Kit	21 Meg, 40 MS,	$282.00
* ST-138/Z150 Kit	32 Meg, 40 MS,	$325.00
* ST-151/Z150 Kit	42 Meg, 24 MS,	$434.00

Z-148 HARD DISK DRIVE KIT

⇒Includes the hard disk drive and a Z-148 compitable 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 to the following price if you would like us to include a SmartWatch.

* ST-125/Z148 Kit	21 Meg, 40 MS,	$358.00
* ST-138/Z148 Kit	32 Meg, 40 MS,	$399.00
* ST-151/Z148 Kit	42 Meg, 24 MS,	$499.00

Z-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

Z-150 MEMORY UPGRADE

* A replacement PAL chip for your Z-150 or Z-160 computer. With this PAL chip installed, two banks of 64K memory chips can be replaced with 256K chips bringing your memory up to 704K. To complete this kit, order 18 pieces of the 256K RAM chips. Order PAL Chip ZP640+ $18.00

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

FLOPPY DISK DRIVES

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

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

PAYLOAD CUSTOM ASSEMBLED COMPUTERS

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

VIDEO MONITORS

⇒ZCM-1490	ZENITH Color Flat Screen VGA	$ 679.00
⇒MA2565	SAMSUNG Amber TTL 720x350	$ 89.00
⇒CW4644	SAMSUNG Color RGB 640x200	$ 274.00
⇒CM4531	SAMSUNG Color EGA 640x350	$ 389.00
⇒CN4551	SAMSUNG Multi-sync VGA 800x560	$ 489.00

VIDEO CARDS

⇒EGA350	PARADISE AUTOSWITCH 640x350	$ 124.00
⇒EGA480	PARADISE AUTOSWITCH 640x480	$ 139.00
⇒VGAPLUS	PARADISE AUTOSWITCH 800x600	$ 172.00
⇒VGAPLUS16	PARADISE AUTO 16 BIT 800x600	$ 214.00
⇒AUTOEGA STB SYSTEMS 640x350		$ 118.00
⇒AUTO VGA STB SYSTEMS 800x600		$ 154.00

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	150 ns	$ 2.95
⇒41256	256x1	120 ns	$ 3.10
⇒41256	256x1	100 ns	$ 3.95
⇒1Mbit	1Mx1	80 ns	$ 10.75
⇒1Mbit	1Mx1	100 ns	$ 10.15

Z-100 SERIES COMPUTER UPGRADES

High Density 1.2 Meg Drives

⇒External floppy drive set-up complete with drive, power supply, case and cables.
⇒Ready to connect to your 8" floppy controller. Single Drive Unit $217.00
⇒Dual Drive Unit $309.00 Bare drive and cable for internal mount $127.00

ZMF100A by FBE Research

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

UCI Memory Upgrade Pal Chip Set

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

UCI EasyWin HARD DRIVE SYSTEM

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

CDR Z-100 SPEED MODULE

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

Z-100 SERIES SOFTWARE

⇒PART NUMBERDESCRIPTION	LIST PRICE	SALE PRICE
⇒MS-463-1Z-Basic (16 bit)	$175.00	$10.00
⇒MS-253-1Basic-80 (8-bit)	$175.00	$10.00

⇒Please Mail, Phone or FAX your order. All hardware carries the manufacturers warranty plus the PAYLOAD 90 day guarantee. No surcharge on credit card orders. COD shipments on request. Add $5.00 to all prepaid orders for handling and shipping in the Continental USA, we pay the balance. Actual shipping costs for foreign, overseas and net billing orders. We accept purchase orders from schools, government and approved accounts. Mail or phone your order for prompt friendly service. Texas residents please add 8.0% state sales tax.

PAYLOAD

15718 SYLVAN LAKE * HOUSTON TX 77062

VISA DISCOVER MasterCard

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

PAYLOAD * PAYLOAD * PAYLOAD * PAYLOAD

Reader Service #117

LOTUS 1-2-3 VERSION 3.0 PROBLEM, MEMORY MAPS, MEMORY USAGE, MS-DOS 3.3 PLUS GENERAL DISK UTILITIES (GDU & GDUTSR)

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

Compatibility is always a major concern for all users, and there is one current issue that could be a significant problem for Zenith and Heath computer users. Simply stated, Lotus 1-2-3 version 3.0 will not run (i.e., the program will not load into memory) on Zenith and Heath computers that only have the standard 1 MB of memory. As of the time of this writing, that includes all current desktop and laptop computers made by Zenith and Heath with older ROM versions. For those systems, the only cure is to add at least 1 MB of memory to the system because the problem does not occur on any computer that has at least 2 MB of memory. Unfortunately, that is an expensive fix to get software to run on a system that should be compatible, so I have done some research on the problem.

The Lotus 1-2-3 Version 3 Problem

Those of you who have read this column for a while know that I have never liked Lotus 1-2-3, and I will tell you that before we begin. Part of that may be because I originally learned SuperCalc back in the 8-bit CP/M days. But there were really three major reasons why I never liked or used Lotus 1-2-3 much. First, the early PC versions of 1-2-3 were copy protected, and if you have read this column for a while, you know that I do not like copy-protected software, I do not use it, and I do not review it in this column. The second reason is that 1-2-3 always seemed very slow after working with SuperCalc. And the third reason is that I found 1-2-3 clumsy and difficult after working with SuperCalc. For example, I have never cared for the Lotus convention of using the at-sign (@) to "signal" a function like SUM, not to mention having to precede formulas (e.g., C3+D4) with a sign (i.e., +C3+D4) to prevent 1-2-3 from assuming that I wanted text mode based on the first character. It seems to me that a lot of extra keystrokes are required for 1-2-3 that are not required for SuperCalc. I have also never cared for the comparatively high charge that seems to be associated with 1-2-3 upgrades. And to complicate matters further, it seems that this new Lotus version 3 will NOT run on any system that has less than 1 MB of memory because that is the minimum requirement. No matter how many new features this new version has, it seems to me that this minimum memory requirement is quite unnecessary given that most other spreadsheets (e.g., Quattro, SuperCalc, and Excel) can still run on a system with 640 K, although they can use more memory if you have it. What this requirement really means is that Lotus version 3 will not run on millions of older systems because they do not have 1 MB of RAM. Now that you know I have a definite bias against 1-2-3 (and why), I will share some information that I have learned about this problem.

My research indicates there were really two parts of the problem. The first problem was that there was apparently a bug in the way the Lotus 3 memory manager worked with 1 MB systems. From what I can tell, the memory manager did not reset "something" it should have, which caused the problem. To be fair to Lotus, this is one of those obscure kinds of bugs that it is nearly impossible to find, unless you know what to look for. However, this does not cause a problem in many other brands of systems because they don't have the performance characteristics of Zenith and Heath computers. And that's the second part of the problem.

For performance reasons, there are some minor differences in the Zenith ROM code that are not found in other manufacturer's ROMs. That's why you'll find that Zenith systems frequently outperform other computers, especially with memory-intensive processing. A pessimist might say that a Zenith computer is not compatible because of that, but I disagree. However, it turns out that the same "something" was also not reset by Zenith code (for performance reasons), and since the Lotus memory manager also did not reset it, version 3 would not run on a system with less than 1 MB of RAM.

As of the time of this writing, I hear rumblings that Zenith is working on a ROM fix for this problem, and it will probably be available by the time you read this. Since I originally began researching this problem as a result of reading about it in the August 14, 1989 *InfoWorld* (page 6), any fix to any version of a Zenith ROM for the 1 MB systems would obviously have been made after that date and probably would have been released near the end of 1989. That is strictly a guess, so don't start buying ROMs under the assumption that the current one will fix the problem. My suggestion is to check with Heath Technical Assistance at (616) 982-3309 (8:00 am to 4:30 pm EST during the week only) to see if there is an updated (and available) ROM for your system to correct the Lotus version 3 problem. If you call, be SURE to have the hardware characteristics of your system written down, including the computer model number and your current ROM version number (use CTRL-ALT-INS to find it). If you need to order new ROMs to run Lotus version 3, you can call (616) 982-3571 for Heath Parts Department listed at the end of the article, and the cost should be less than $30. By the way, do NOT expect the Heath Parts Department to answer questions about this problem — their job is ONLY to help you order parts, not provide technical assistance. If your system is less than one year old, the ROM replacement should be covered by the warranty, and you should contact the dealer from whom you purchased the system.

As a matter of information, there are some technical errors in that *InfoWorld* article, so don't believe everything you read in it. Perhaps the most glaring error is that the article states (apparently written by Lotus) that "... the RAM-shadowing technique makes 384K of memory unavailable." That is simply not true. The fact is that the IBM standard defined the address space from 640 K to 1 MB as "reserved", and you will see how that works later in this article. It is also interesting to note that Zenith was not the only computer manufacturer to have this problem — a similar problem also occurs with AST computers.

In summary, this particular problem occurs when attempting to run Lotus 1-2-3 version 3 on Zenith and Heath 80286 and 80386 systems which have a standard 1 MB of memory (NOT the Z-241 or Z-248). If you already have 2 MB of memory installed, the problem will not occur, and there is no reason to order or install a new ROM set ("If it ain't broke, don't fix it"). Now, let's take a more detailed look at some of the facts related to how memory is really used in a computer system.

PC Memory

Understanding how memory is used is not difficult so long as a lot of the boring technical details are ignored. The purpose of this discussion is to present a non-technical description of how memory is generally used in a computer, and while this information is technically correct as presented, it is far from adequate for any programming purposes. In most cases, I will discuss something as a fact, without any explanation as to how one can figure out that information. For example, we will begin by considering a basic 8088-based system, and it is a fact that the 8088 Micro Processor Unit (MPU) can talk to (or address) an address space of no larger than one megabyte.

Because IBM set the standard for how the address space was used in the original PC, it is important to take a look at the way the entire 1 MB of "memory" was used. The easiest way to see this is to look at a "memory map" as shown in Figure 1.

The entire box shown in Figure 1 represents all of the address space that can be accessed by the 8088 MPU. The area at the bottom of the box (up to 640 K) represents Conventional System Memory which can be ignored in this discussion. Memory at the bottom of the box is usually referred to as "low memory" because it begins at zero. Memory closer to the 1 MB limit (usually above 640 K) is called "high memory" which is represented intuitively in this memory map as toward the top.

```
/-----------------------------------\     1024 K  (1 MB)
|        Reserved (ROM) -- 64 K       |
|-----------------------------------|       960 K
| Reserved (PC Jr. Cart.) -- 64 K   |
|-----------------------------------|       896 K
| Reserved (PC Jr. Cart.) -- 64 K   |
|-----------------------------------|       832 K
| Miscellaneous Expansion -- 64 K   |
|   (e.g., HD ROM, EGA ROM, etc.)   |
|-----------------------------------|       768 K
|          CGA Video (32 K)         |
|      ------------------------     |
|          MDA Video (32 K)         |
|-----------------------------------|       704 K
|        EGA Video RAM (64 K)       |
|-----------------------------------|       640 K
|                                   |
|            Conventional           |
|               System              |
|               Memory              |
|                                   |
\-----------------------------------/         0 K
```

Figure 1
1 MB Memory Map for IBM PC

The "boundaries" are represented as horizontal lines showing how the entire 1 MB address space is organized. For technical reasons, these boundaries are organized in blocks of 64 K as shown for high memory, but the Conventional System Memory is shown as a total of 640 K rather than 10 blocks of 64 K.

In case you are unfamiliar with this notation, it is essential to know that one kilobyte (usually abbreviated as *K* or *KB*) represents 1,024 bytes, not 1,000. In other words, a system that has 640 K of conventional memory has 655,360 bytes of memory, not 640,000. If you know your system has 640 K of conventional memory installed, you can easily prove this to yourself by running CHKDSK which will display a message like "655,360 bytes total memory." Even if your system was originally equipped with 1 MB of memory (e.g., any current 80286 or 80386 system), CHKDSK will *still* display 655,360 bytes of conventional memory because that is the maximum for a PC compatible system, regardless of what MPU the system has.

By definition, a megabyte (usually abbreviated as *MB*) is 1,024 kilobytes or 1,048,576 bytes (1024 times 1024). Even though the definitions of kilobyte and megabyte are technically accurate, you will still find that some people and documentation define kilobyte as 1,000 bytes and megabyte as 1,000,000 bytes due to sloppiness or misunderstanding.

In any case, 640 K are used for conventional memory, but how is the rest of the one megabyte address space used? Well, IBM defined a general standard, or map, of the organization of the remaining 384 K of memory. How this memory is organized for a specific computer, such as the IBM PC or any compatible, is actually defined in the hardware by a special computer "chip" called a PAL (Programmable Array Logic). Sometimes you will hear people refer to 640 K as the "DOS limit" for conventional memory, and that is not technically accurate. The *real* limit and definition (i.e., memory map) for conventional memory is contained in a PAL, which explains why the discontinued Z-100 could have up to 768 K of conventional memory and a PC only has 640 K.

Although I will mention it again, the most important point about the remaining 384 K of memory was that it was defined by IBM for a specific purpose and it was RESERVED for that purpose. In this context, RESERVED simply means that this memory was not available to an application program or hardware, other than to be used for its defined purpose. For example, I seem to remember that a PAL was available (from a non-Zenith source) for the Z-151 which allowed the use of memory between 640 K and 704 K to be used as a virtual disk (VDISK), even though that 64 K block of memory was reserved. Although it seemed like a good idea at the time, it is obviously not a good idea today because that block of memory is specifically reserved for EGA video RAM. I would not try to predict what might happen on a Z-151 if a user added an EGA video system and attempted to use a VDISK permitted by that PAL, but the results would probably be at least interesting.

The 64 K block of memory between 704 K and 768 K is reserved in two 32 K parts. The lowest 32 K block from 704 K to 736 K is reserved for the IBM Monochrome Display Adapter (MDA), even though only the first 4 K is actually used (28 K unused). My research indicates that non-IBM monochrome cards may use a larger portion of this 32 K block which explains why some software provides special video drivers for these cards, such as Hercules. The block from 736 K to 768 K is reserved for the Color/Graphics Adapter (CGA), but only the first 16 K is actually used (16 K unused).

The block of memory between 768 K and 832 K was reserved for miscellaneous expansion, and the first use of an address within this block was for a hard drive ROM (on a hard drive controller card) at about 800 K. The code for the EGA ROM (not RAM) is also located within this block.

In the original definition, memory between 832 K and 960 K was reserved for system use with no real designated purpose, but it was eventually used on the IBM PC Jr. for ROM cartridges that contained various application programs, such as Lotus 1-2-3. Some manufacturers, such as Zenith, even added an additional ROM socket to some computers so that a system could use part of this reserved address space, but I never heard of any commercially available ROMs that were developed for this purpose.

The last (top) 64 K block is reserved for the system ROM. This includes the ROM Monitor, ROM-BIOS, and ROM-BASIC for the IBM computers. For most Zenith systems, this area includes the special debugging, test, and other features of the Zenith ROM as described in the October and December 1989 "Powering Up — Volume 2" articles. These features generally "replace" the ROM-BASIC feature on IBM computers. That is no great loss because the kindest thing I can say about ROM-BASIC is that it is extremely limited.

For a computer to be PC compatible, it must generally use this kind of memory map. The fact that the Z-100 has a different memory map is one reason that it is not PC compatible, not to mention the other hardware and ROM differences. So much for a "standard" PC-compatible definition of how a 1 MB address space is generally used.

Zenith 8088-Based Systems

As you might expect, there is no significant difference in the basic memory map for all Zenith 8088-based computers which are PC compatible. Some of the implementation details are different, depending on which system you are discussing, but the basic memory map is not. For example, the eaZy PC has the same memory map as shown in Figure 1, but it does not have any of the Zenith-unique commands and ROM features like the more expensive systems.

Zenith 80286-Based Systems

For purposes of this discussion, there are really two different groups of 80286-based systems: those with a standard 512 K of memory and those with a standard 1 MB of memory. The Z-241 and Z-248 systems that came equipped with a standard 512 K of memory use the same basic memory map shown in Figure 1, with the exception that the block of memory from 896 K to 1024 K (128 K) is reserved for the system ROM. This includes all of the standard Zenith ROM features in addition to the ROM-based SETUP command. The use of the 64 K block of memory from 896 K to 960 K does not cause a compatibility problem because the IBM AT series computers did not use it anyway. Besides, I think the Zenith ROM-based SETUP command is much more convenient than the disk-based SETUP command that IBM provided with the AT series computers. And it is interesting to note that many other manufacturers have since implemented a SETUP command in their ROMs because it really does not make all that much difference where the actual program is located — either ROM or disk. The biggest advantage is that a ROM implementation is more efficient because you don't have to go hunting for the SETUP disk if you want to change something. (Where's that stupid SETUP disk?) Of course, it is probably not reasonable to suppose that computer manufacturers were completely altruistic in this because it is quite likely that a ROM-based SETUP command is cheaper to implement too.

The second group of 80286-based systems include all which came equipped with a standard 1 MB of memory. All of these systems were released after the one megabit RAM chips became available at reasonable cost in production quantities. These high-performance systems have been optimized to run at clock speeds of 12 MHz or so, such as the Z-286 and SupersPort 286. And Zenith developed an interesting technique for improving performance of these current 80286 and all 80386 systems.

Snow Inside Your Computer?

Not really. But if you listen carefully to one of these current systems, you might hear something "slushing" around inside. It's not exactly hardware, it really isn't firmware, and it is not altogether software . . . it's SLUSHWARE! Terrific. Now, what is slushware?

Given the usual nature of the weather in most parts of the country in February, it is appropriate to take a quick look at what slushware is. If you ignore all of the gory technical details, the concept of slushware is pretty easy to describe. Actually, SLUSHWARE is program code (software) that is copied from the system ROM (firmware) into the high memory space up to the 1 MB RAM (hardware) installed in the computer. The whole reason for copying the ROM program code into RAM is to improve system performance because it is much faster to access information in RAM than ROM. In general, you will find that's the reason that Zenith computers are faster than many other brands of computers in some tests, all other things being equal. For example, you will find that a Zenith computer with an EGA video card will have a "faster" video display as compared with another computer that does not use the slushware concept, assuming the same video card is tested in both systems of course. And any test that involves significant access to any of the ROM code, such as the ROM-BIOS, will also be faster than a similar system that does not use the slushware concept.

Like many things associated with computers, the whole idea of slushware is not really new. In fact, the basic concept of slushware was developed and implemented by some Heath Company computer experts in the late 1970s (around 1979 I believe) on a computer kit called the H-8. For the H-8, the implementation was called "ORG 0 RAM", but the idea is the same. Of course, this general concept was developed long before anyone ever thought of Zenith Data Systems, but some of those same people became part of ZDS, and the concept was never forgotten. Today, ZDS calls this implementation slushware; other manufacturers of high-performance systems call it "shadow RAM" as a descriptive name indicating that ROM programs are copied (i.e., as a "shadow") into RAM. As previously mentioned, this feature is *not* available in the Z-241 and Z-248 systems.

How the 1 MB of RAM is Used

When the IBM ATs and the Zenith Z-241/248s were released, there was a lot of criticism that these systems did not have the 640 K conventional memory. The standard configuration was 512 K, even though there were a number of ways to increase conventional memory to the 640 K maximum. On the surface, it seemed kind of dumb to have a superpowerful computer (at that time, anyway) that only had 512 K of conventional memory. The question was: "Why do these systems only have 512 K of conventional memory?" Actually, there were two answers.

First, there has never been a RAM chip developed with a 640 kilobit capacity, and there are technical reasons for that. Most people are familiar with RAM chips that have a capacity of 64, 128 (not very popular), 256, 512, and 1024 kilobits. The point is that it was easy to use the 512 kilobit chips to install 512 kilobyte RAM, but any capacity less than one MB required a second bank of nine chips (sometimes eight, if memory parity was not used). That leads to the second answer.

Space is always at a premium inside a

computer, and a second bank of nine RAM chips either requires more space on an existing board or the use of a second board. That leads to a choice of making a computer "bigger" (and more expensive) or including a second "standard" memory board that uses up an expansion slot (also more expensive).

In short, space and cost were the major reason that 512 K of memory was provided with these earlier systems. That particular problem was eliminated with the production use of one megabit memory chips, but how this memory was used has not been clearly understood, so let's take a look at how this one megaBYTE of RAM is used in the newer systems.

During this discussion, it is important to keep in mind that there is a major difference in how memory is really used in a 1 MB system. The design of the 8088 MPU limits it to a total of one megabyte of address space, including the ROM and all other items shown in Figure 1. Systems with an 80286 or 80386 MPU can actually talk to far more than one megabyte of *extended memory* (above the 1 MB limit), and this limit is normally 16 megabytes on many production systems today. Of course, you can always add expanded memory to any system, regardless of the MPU, but the real key is how the system uses memory.

To illustrate the difference between how memory is used for the 8088-based systems and the current 1 MB systems, consider the memory map shown in Figure 2.

This memory map looks about the same as Figure 1, which indeed it must to maintain compatibility; however, this is a memory map of the total one megabyte of RAM, not just the one megabyte address space for the 8088 shown in Figure 1. This is a very significant technical distinction that is really not too important for this discussion.

Figure 2 shows that the top 128 K of the RAM (from 896 K to 1024 K) is reserved for slushware (or shadow RAM). When this concept is used, the top 64 K contains the Monitor ROM code, the next 48 K contains the System ROM code, and the remaining 16 K is used to slush EGA video for faster displays.

The 64 K block between 832 K and 896 K is used for EMS paging. If you read the article in last month's issue about "How to Use EMM.SYS", you will recall that somehow you can actually use 256 K for expanded memory, but there is only a 64 K block of RAM actually defined for that. I will not try to describe how this works because it involves page frames and other gory details. The fact that you really can use up to 256 K is an observable fact as described in that article, so it is not too important to understand how it works.

From 832 K down to the bottom of memory, the rest of the memory map in Figure 2 is pretty much standard compared to Figure 1. The real difference is in how the top 192 K of memory is used. There really is not a conflict because that address space (or RAM in these systems) was really "reserved" in the IBM PC hardware anyway. In many other computers, much of this 192 K of RAM is wasted because it is not used for anything other than the ROM, although it is generally a standard in these current systems to define EMS support in the block from 832 K to 896 K, and the Zenith expanded memory device driver (EMM.SYS) is not unique in that respect.

```
/-----------------------------------\         1024 K  (1 MB)
|  Slushware (Monitor ROM) -- 64 K  |
|-----------------------------------|          960 K
|  Slushware (System ROM) -- 48 K   |
|     ------------------------      |
|  Slushware (EGA Video) -- 16 K    |
|-----------------------------------|          896 K
|          EMS Page -- 64 K         |
|-----------------------------------|          832 K
|        Add-On Cards -- 48 K       |
|     ------------------------      |
|        EGA Video ROM -- 16 K      |
|-----------------------------------|          768 K
|          CGA Video (32 K)         |
|     ------------------------      |
|          MDA Video (32 K)         |
|-----------------------------------|          704 K
|        EGA Video RAM (64 K)       |
|-----------------------------------|          640 K
|                                   |
|            Conventional           |
|               System              |
|               Memory              |
|                                   |
\-----------------------------------/            0 K
```

Figure 2
1 MB Memory Map for 80286 and Later Systems

Lest you think the Lotus problem is restricted to Zenith and AST computers, it isn't, and even IBM has a problem with it. In particular, Lotus 3 uses the Virtual Control Program Interface (VCPI), and the expanded memory device driver (like Zenith's EMM.SYS) must support the VCPI standard in order to run version 3. Interestingly enough, the expanded memory device driver included in IBM's PC-DOS version 4.0 does NOT support VCPI, and some IBM users are also having problems too because Lotus will not run on their IBM systems with that DOS version.

This whole problem with Lotus emphasizes a point that I have made before in this column. Unless a software developer has made an extensive effort to ensure compatibility of new programs with existing systems, DOS versions, and other applications, these kinds of problems will inevitably occur, particularly when significant enhancements are added. The fact that Lotus apparently did not know beforehand that version 3 had problems with IBM PC-DOS 4.0, as well as Zenith and AST computers, shows how difficult it can be to verify how new software will work in different systems. And the point of all this is that each user must take care to verify that a new program will work on a given system configuration, even IBM. There are no guarantees that "compatible" software will even run correctly (or at all) on a totally "True Blue" (IBM) system, so don't believe everything someone may try to tell you about Zenith/Heath computers and/or MS-DOS "compatibility problems" — even IBM has them!

Zenith General Disk Utilities

For those of you who read my December column, I mentioned that I would discuss some observations about the Zenith General Disk Utilities in last month's column; however, I decided to include that information in this issue because the last article was so long.

One of the more interesting features of Zenith MS-DOS 3.3 Plus is the set of General Disk Utilities that is supplied as two programs: GDU and GDUTSR. As the names suggest, the basic GDU program is an external command, and GDUTSR (also external) is a memory-resident (TSR — Terminate and Stay Resident) program. These are Zenith-specific programs and are not part of the usual DOS. From my research, it appears that Zenith was the only DOS vendor who recognized that the technical changes in MS-DOS 3.3 Plus made virtually all of the current disk utilities useless because they would not work with this latest DOS version. And so, Zenith developed the GDU and GDUTSR programs to help ease the transition to this latest version.

The whole purpose of the memory-resident program, GDUTSR, is to record deleted file information (e.g., clusters for the FAT) in a special file so that it can be recovered later, if necessary. In concept, this program is similar to the Mace Utilities RXBAK program that makes a file copy of the directory and FAT, but RXBAK is *not* memory resident. As most of you know by now, I am not a real fan of memory-resident programs, and my testing indicates that both Microsoft Word 5.0 and Word Perfect 5.0 do NOT get along with GDUTSR. Apparently, both those programs do something non-standard when they write files to disk, but I have not figured out exactly what the problem is. I have known for years that Word occasionally does some non-standard things, even though it was developed by Microsoft who should follow their own standards, but the Word Perfect problem surprised me. Perhaps Zenith will be able to fix that problem, but I do not consider it important in my system because I do not use GDUTSR for several reasons, although I have already received several letters from version 5 users of both Word and Word Perfect about the problem.

First, I think it is really kind of silly to use GDUTSR with any programs that make automatic backups anyway. Many of the latest programs have an automatic timed backup feature that allows the user to specify how often the program should back up a file to disk. I prefer to not use that feature because it always seems to save a file when I want to do something. My usual practice is to save a file about every 15 minutes or so, or when I answer the phone, or when I get a cup of coffee, or whatever. Besides, it's kind of tricky to recover a file in that way when you remember that a backup file will have a name like MYFILE.BAK — you would have to RENAME the current BAK file to something else before you could recover the old version (also called MYFILE.BAK) anyway. As far as I'm concerned, it is really a moot point as to whether or not GDUTSR works with these applications or not, and it would probably be pointless for Zenith to try to fix it. If you really need to have a backup beyond the existing two files (the original and the backup) on a hard drive, you should exit an application and back up that file to a floppy anyway.

Another reason I don't use GDUTSR is because it is memory resident, and most of the programs I use need all the memory they can get. And I have mentioned before that I don't like to use many memory-resident programs because they may not always work correctly with my applications or other memory-resident programs that I do need to use. Okay, so when is a good time to use this program?

My suggestion is to only use GDUTSR when you are doing maintenance on a hard drive — specifically when you are deleting old files. You never know when the dreaded DEL *.* command may somehow delete the "wrong" files because you forgot to change to the correct subdirectory. That's the only time I have ever found GDUTSR to be very useful, and it is the only time I load it. And I disagree with one of the popular practices of loading all kinds of different TSR programs as part of the AUTOEXEC.BAT file processing. It's much better and cleaner to use several "STARTxxx.BAT" files to start various applications, and you will not have to remember memory-resident programs work with which applications. If you decide to use GDUTSR with any software, be sure to check it out for program conflicts (like a specific application or any other memory-resident program) before you use it in a "normal" configuration. I recommend using this program ONLY when you are doing some kind of disk maintenance.

The General Disk Utilities program, GDU, is an extremely thoughtful addition to MS-DOS 3.3 Plus by Zenith. It has a remarkably wide range of features, including recovering deleted files, although that feature requires the use of GDUTSR to ensure successful file recovery. And remember that "old" versions of Mace Utilities, Norton Utilities, HADES (from HUG), and other similar programs will NOT work with Zenith MS-DOS 3.3 Plus because of technical changes made by Microsoft as discussed in my June 1989 column. In this respect, Zenith MS-DOS 3.3 Plus is technically equivalent to IBM PC-DOS 4.0. It is fortunate that the technical changes cause most of these older programs to simply not work because of the possibility of really destroying data on a hard drive.

GDU is one of those programs that can be real dangerous if you do not know what you are doing. It has many of the features of other disk utilities, such as displaying drive parameters, editing a file, undeleting a file, and looking at or editing a FAT. All of these features may be useful to a user, but I think that both Mace and Norton generally do a better job. And if you really want to fool around with disks and formats with a low-level editor, I think that HADES II (from HUG) is the very best disk editor you can find. I think that HADES II is much easier to use than MUSE (Mace Utilities Sector Editor) or GDU. If you really don't need that kind of capability on a regular basis, GDU is certainly more than adequate for the job, and it is a nice addition to MS-DOS 3.3 Plus.

Powering Down

It will be interesting to see what happens to microcomputers in this decade. In particular, I am waiting to see what happens to the new operating systems, especially OS/2 and Unix. What will happen to DOS? It's difficult to believe that Microsoft will "abandon" DOS because there are at least 15 million 8088-based computers that simply will not run OS/2. Although OS/2 admittedly has some neat features, it is extraordinarily expensive to set up, and there still is not a lot of software that can use its features. Even if you already have a newer system with at least an 80286 MPU, OS/2 itself is expensive to buy ($300 to $1,000). OS/2 requires a lot of memory (expensive) and a hard drive, not to mention that a lot of popular programs (especially TSRs) will not work with OS/2 if they are not especially written for it. And based on some of the software that is currently available for OS/2, these new programs are even more expensive than their DOS equivalents are.

From a user perspective, I would like to know what you think about OS/2 in terms of cost and functionality. Would you spend at least $3,000 to upgrade your system to run OS/2? What do you think about the cost of existing DOS software and the general observation that OS/2-unique software will be even more expensive? I have neglected much discussion about OS/2 simply because I rarely get any letters about it, and there are many more important developments in the DOS world anyway, at least today. I have doubts that OS/2 can be cost justified by most users, and I would like to know what you think. This information will be used to help me plan articles on various subjects for at least the rest of this year. If I don't get many letters suggesting that OS/2 is important to most HUG members, this column will continue to focus on basic DOS-based software and Zenith hardware developments.

As announced in the December 1989 issue, I have a number of Zenith-specific FlipFast books (MS-DOS and GW-BASIC) available as a set of one each for $9.00 (plus $3.00 shipping). Sets cannot be broken into two of the same book, and quantity discounts are available for more than five sets. Be sure to tell me about how many you need for a specific quote because most of the savings is in shipping costs. This offer is good until March 31, 1990 or until I run out of books. If I am out of books when I receive your order, I will return your check or money order to you. Ordering information is listed at the end of this article.

For help in solving specific computer problems, be sure to include the exact model number of your system (from the back of the unit or series from the Owner's Manual), the ROM version you are using (use CTRL-ALT-INS to find it), the DOS version you are using (including both ver-

Continued on Page 46

W S Electronics

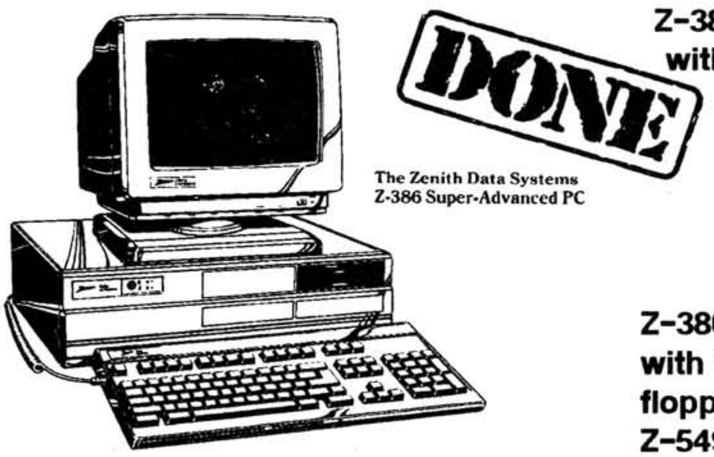
(513) 376-4348 ** Since 1975 **** (513) 427-0287**
1106 State Route 380, Xenia, Ohio 45385

W S Does it Again!!

Z-386/16 Truck Load Sale

We have a limited quantity of Z-386's .

You Want Fast?



Quantities are limited, monitors not included.

System # 1
Z-386/16 40 Zenith 80386 16 mhz with 40 meg hard drive, 1.2 meg floppy drive, 1 meg ram and Z-449 EGA+ Video.
Sale : $2399.

System # 2
Z-386/16 40 Zenith 80386 16 mhz. with 40 meg hard drive, 1.2 meg. floppy drive, 1 meg ram, and Z-549 VGA Video.
Sale: $2499

enable™ 2.0
Sale Price $99.
Retail Price $695.00

with purchase of a Z-386/16
Enable 2.0 $50.
*P2000G Alps printer $350.
Second 1 meg of Ram $250.
ZCM-1490 FTM VGA Monitor $500.
*P2000G are used grade B stock.

ZENITH | data systems
THE QUALITY GOES IN BEFORE THE NAME GOES ON®

Pat Swayne
HUG Software Engineer

Getting Started With . . .

Assembly Language Part 3

Flags, and the Instruction Set (Part 1)

This is the third part in a series in which I am attempting to teach basic assembly language program skills to persons familiar with BASIC. In the last installment, I introduced you to the CPU registers, and in this installment, I will introduce the instruction set. Before I do that, however, I must explain an additional register that I did not cover last time.

The Flag Register

The flag register is a special register that is not used directly by any instruction. The individual bits of this register, which are considered to be separate entities, are called "flag bits", or just flags. The flags are used to indicate certain conditions that occur as a result of arithmetic or logical instructions. For example, if the CPU executes an instruction that adds two 16-bit numbers together, and the result is a number larger than 16-bits, a particular bit in the flag register called the Carry flag is set to 1. Here is a chart listing the individual flags.

Flag Name	Bit No.
Carry	0
Parity	2
Auxiliary Carry	4
Zero	6
Sign	7
Trap	8
Interrupt	9
Direction	10
Overflow	11

Bits 1, 3, 12, 13, 14, and 15 are not used, at least for our purposes. The flags can be classified into two groups: flags that are set as a result of a mathematical or logical instruction, and flags that control the operation of the CPU. The Trap, Interrupt, and Direction flags are control flags, and the others are result flags. Here is an explanation of the flags.

Carry — This flag is set (the bit is set to 1) if the result of an addition is too large for the intended destination (whether 8- or 16-bits), or if a number is subtracted from another number smaller than itself (a borrow). If an addition does not result in a carry, or a subtraction does not result in a borrow, the flag is reset (the bit is set to 0).

Parity — This flag is set if the result of an operation contains an even number of bits, and it is reset otherwise.

Auxiliary Carry — This flag is set if an instruction causes a carry out of bit 3 into bit 4, and is reset otherwise.

Zero — This flag is set if the result of an operation is zero, and is reset otherwise.

Sign — This flag is set if the most significant bit in the result of an operation is 1, and is reset otherwise. When the CPU works with signed numbers, the most significant bit of a number is used as a sign bit. The number is considered negative if the sign bit is set.

Trap — When this flag is set, the CPU executes a special interrupt (called the Trap interrupt) for each instruction. An interrupt is like a subroutine call that is caused by the computer hardware and not by a program. For example, each time you press a key on your keyboard, an interrupt is generated. The subroutine called by this interrupt reads the keyboard and places the code read into a buffer for later use by whatever program you are running.

The Trap interrupt can be used by debugging programs to monitor each instruction executed by a program under scrutiny. We will not be using the Trap flag in this series.

Interrupt — This flag determines whether the CPU will recognize interrupts or not. It is normally set, and there are CPU instructions that can directly set or reset this flag. If you were to write a program that cleared this flag and left it cleared, your computer would not recognize the keyboard interrupt, among others, and therefore, would not accept anything you typed.

Direction — There are special string manipulation instructions that automatically increment or decrement the SI and/or DI registers in preparation for the next instruction. The direction flag determines whether the registers are incremented or decremented. If it is clear, the registers are incremented, and if it is set, they are decremented.

Overflow — This register is set if the result of a mathematical operation is too large for the destination. It is also set when an 8-bit multiplication results in a 16-bit product, or a 16-bit multiplication results in a 32-bit product, even though the number may not be too large.

The Instruction Set

The 8086 family of processors can execute approximately 90 basic instructions. At the machine level, each instruction consists of a one or two byte hexadecimal number, and many of them require "arguments", which are just additional numbers. These arguments may be data required by the instruction, or they may be pointers to other locations in memory (other than the location of the instruction itself). As I said in part one of this series, assembly language is just a way of expressing the processor's instructions in words that represent what the instruction does. These words are called "mnemonics". The instructions themselves, whether expressed as words or numbers, are sometimes called "opcodes" (for "operation codes"), and the arguments are sometimes called "operands". In assembly language, operands do not always represent data bytes, but often they just complete the mnemonic description of a specific instruction.

The program that translates mnemonics into machine code numbers is called an assembler. In addition to using the words that stand for the computer's instruction set, the assembler allows you to make up words of your own to represent addresses in memory or just numbers. A word that you make up that stands for an address in memory is called a "label", and a word that represents a number is called an "equate", a "value", or an "absolute". Collectively, these user supplied words are called "symbols".

The assembler recognizes another set of instructions besides the computer's instruction set. These instructions tell the assembler what to do in certain situations, and are called "assembler directives". If you have worked with a high level language compiler, then you are probably familiar with "compiler directives". An assembler, however, usually has many more directive instructions than a compiler. I will introduce assembler directives as they are needed in this series.

This is probably a good time to give you an idea of what assembly code actually looks like. You can think of each line of assembly code as being divided into "fields", as in this example:

```
Label   Opcode  Operand Comment
field   field   field   field

        ORG     100H
START:  MOV     AL,BL   ;MOVE BL CONTENTS TO AL
```

The first line in this example contains an assembler directive, ORG, which stands for origin. It tells the assembler that the completed program is designed to be placed in memory starting at 100 (hex) bytes from the start of its code segment. The next line of code starts out with the definition of a label. You can define a label in assembly language just by placing it in the label field. You must put a colon (:) after the label if it is by itself on a line, or if it is followed by a mnemonic. If it is followed by an assembler directive, you do not need the colon.

Each label that you define is assigned an address value which is called an "offset", because it indicates how far the label's position is offset from the bottom of a segment. The offset of the label START is 100 (hex), because there are no data or code bytes between it and the ORG directive.

Following the label START is a mnemonic, MOV, which stands for "move". This is the most common processor instruction, and it is used to move data from one place to another. Actually, it is misnamed, because it does not move data in the sense of moving something from one place to another, as when I move a pencil from one side of my desk to the other side. Instead it copies data. After the instruction is executed, the source of the data still contains what it had before, and the destination contains a copy of the data.

The operand in our example is AL,BL, which specifies which of the processor's many MOV instructions we want to use. (When I said that there were 90 instructions, I was counting MOV as one instruction. If you counted each specific implementation of each instruction, you would have hundreds of instructions.) This MOV instruction places a copy of the data from the BL register into the AL register. Notice that the destination of the "move" is listed first followed by the source. This is the convention used in all instructions that specify a destination and a source.

Following the operand field in the example above is the comment field. Actually, comments can be placed anywhere on a line, as long as they are the last thing on the line. An entire line can be a comment, or you can have a comment immediately following a label, etc. The comment is indicated by a semicolon (;), which tells the assembler to ignore everything to the right of the semicolon.

The computer's instruction set can be divided into the groups I discussed in part one of this series. You may recall that I stated in that article that much of the work of a computer consists of transferring data from one place to another. Because of this, I will discuss data transfer instructions first.

Data Transfer Instructions

There are only 10 basic instructions in the data transfer group, but there are many different ways of using some of these instructions. That is because there are different ways of pointing to a particular location in memory. These different ways are called "addressing modes", and this is a good time to discuss them. Here are the commonly used addressing modes.

Direct — The operand specifies the address of the data. You must usually place brackets ([]) around the operand to tell the assembler that you are using the direct mode.

Immediate — The operand is itself the data. If you use a label name as the operand, you must add the word OFFSET before it to indicate to the label that you mean the value (offset) of the label and not the data that it may point to.

Register — The operand is the name of one of the processor's registers.

Indexed — The operand is one of the index registers, and may also include an offset which is added to the address contained in the index register. It may also include the base index register (BX) along with one of the other index registers. In this case, the value in the BX register is added to the value in the other index register along with any specified offset to arrive at the address of the data.

Register Indirect — The operand is the name of a register other than an index register, which contains an address.

Relative — The operand is an offset which is added to the current value of the instruction pointer (IP register) to arrive at an address. Relative addressing is used only in data transfer instructions. However, all addressing modes, except those which specify both a segment value and offset value, are relative in a sense, because the addresses are relative to a particular segment.

Implied — The instruction itself implies the address of the data.

Data transfer instructions nearly always specify two addressing modes — one for the source and one for the destination. In the example above (MOV AL,BL), the register mode is used for both. Here are the data transfer instructions.

MOV — Transfers a copy of the data in one location to another. The source location can be specified using the direct, immediate, register, or indexed addressing modes. The destination can use the direct, register, or indexed addressing modes. The source and destination addressing modes cannot be the same unless both are the register mode. If either the source or destination is a segment register, then the other half of the transfer can only be a non-segment register or a memory location (the direct addressing mode).

PUSH — This instruction transfers a copy of the data in the source to the stack. The source addressing mode can be register or direct. The destination addressing mode is implied, and is the address contained in the stack pointer register, with the segment being in the stack segment register. A special variation of this instruction is PUSHF, which transfers the contents of the flag register to the stack. Some texts list PUSHF as a separate instruction. The stack pointer register is decremented by 2 after the transfer.

POP — Transfers a copy of the data at the top of the stack to the destination, which can be specified using the register or memory addressing modes. The source is the implied address in the stack pointer register. The stack pointer register is incremented by 2 after the transfer. A special variation of this instruction is POPF, which transfers the data at the top of the stack to the flags register.

XCHG — This instruction exchanges the contents of the source and destination. The addressing modes can be register or direct, but only one of either the source or destination can be specified using the direct mode.

XLAT — This instruction uses the BX register as a pointer to a table in memory. The value in the AL register added to the BX register to arrive at a specific location in the table, and then the byte at that location is transferred to the AL register. This instruction, therefore, implies both the source and destination.

LEA — The mnemonic for this instruction stands for "Load Effective Address". It calculates the offset contained in the source and transfers it to the destination. The destination addressing mode is always register. The source addressing mode can be a base or index register, an immediate value, or both, because two source arguments can be supplied. If two sources are supplied, they are added together to get an offset. The assembler often requires the source to be placed in brackets even though the immediate addressing mode is used, and not the direct mode. To help you understand this instruction, here are some examples.

```
LEA     BX,[BP+SI]
```

In this example, the contents of the BP and SI register are added together and the result is placed in BX. If two source registers are specified, one must be a base register (BX or BP), and one must be an index register (SI or DI).

```
LEA     DX,[BX+START]
```

In this example, the offset of the label START is added to the number in the BX register, and the result is placed in DX.

```
LEA     AX,[BX]
LEA     BX,[START]
```

Because the LEA instruction can take just one register or just a number as the source, it is sometimes used in cases where it would be better to use the MOV instruction. You should be aware that the MOV instruction is one byte shorter than the LEA instruction and takes less time to execute. Any time you see the LEA instruction used with only one source specified, it can be replaced with a MOV instruction. The last two examples could be replaced with

```
MOV     AX,BX
MOV     BX,OFFSET START
```

LDS — This instruction transfers 32 bits from the memory location pointed to by the source argument to a specified register and the DS register. The source can be a base or index register, a direct address, or both. The lower 16 bits of the destination can be any register, but the higher 16 bits are always transferred to the DS register. The source is specified using the same rules as for the LEA instruction, except that the direct addressing mode is used in place of the immediate mode. In other words, if you have

```
LDS     BX,[START]
```

then the data pointed to by START is transferred to the destination, and not the value of START.

LES — This instruction is identical to LDS except that the ES register receives the higher 16 bits of the transfer.

LAHF — This instruction transfers a copy of the lower 8 bits of the flag register to the AH register. It is used in cases where a program must be compatible with the old 8080 processor. It can also be used instead of PUSHF when you do not need to save the Trap, Direction, Interrupt, or Overflow flags, and you do not need the contents of the AH register. Depending on your processor type, a PUSH instruction can take up to nearly 4 times longer to execute than the LAHF instruction. Both instructions may take only microseconds to execute, but when you are writing for speed, little time savings can add up.

SAHF — This instruction is the opposite of LAHF. It transfers the contents of AH register to the lower 8 bits of the flag register.

Input/Output Instructions

At the assembly level, the input/output instructions are not the direct equivalent of input/output instructions in higher languages, such as PRINT. You may never have to use one of them, because there are subroutines built into the operating system and your computer's BIOS (Basic Input/Output System) that you can use to handle such chores as printing. The input/output instructions at the assembly level are a specialized form of data transfer instructions that use a "port" rather than a memory location for either the source or destination. A port is a connection between the processor and the outside world. Ports have addresses similar to memory addresses, except that there are only 65536 of them, so they can be specified without the use of a segment register. There are two input/output instructions, which are called, as you might expect, IN and OUT.

IN — This instruction transfers data from the specified port to the AL or AX register. The port can be specified directly (direct addressing mode), or it can be contained in the DX register (register indirect addressing mode). If the port is specified directly, it must be a byte. Therefore, only the first 256 ports can be accessed via the direct addressing mode. If the destination is the AL register, one byte is "read" from the port and transferred to AL. If the destination is AX, one byte is read from the specified port and transferred to AL (the lower half of AX), and another byte is read from the port address + 1 and transferred to the AH register (the upper half of AX).

OUT — This instruction transfers data from the AL or AX register to the specified port. As with the IN instruction, the port can be either a directly specified byte, or it can be contained in the DX register. If the source is AL, one byte is transferred to the port. If the source is AX, one byte is transferred from AL to the port, and one byte from AX to the port + 1.

Perhaps I should explain why it is better to use built-in subroutines for printing, etc., rather than using the IN or OUT instructions. The "port" on the back of your computer where you plug your printer is not connected directly to the processor inside. Instead, it is connected to some intermediate circuitry that can detect such things as whether your printer is "on line" or not. This intermediate circuitry is connected to two or more of the processor's actual ports, so that a port can be used for transferring data to the printer, and another one or more ports can be used to monitor the condition of the printer. A printer cannot receive data as fast as a computer can send it, so it must have a way of telling the computer to wait when it is not ready. The built-in subroutines handle the jobs of making sure that the printer is ready and sending it characters when it is.

This concludes this part of this series. Next time, I will present more of the computer's instruction set. ✱

SAVE!
New Lower Prices

A FRESH NEW LIFE FOR YOUR Z-171

THE Z-171 HARD DISK!

LapDrive

$50 Price Reduction

21M 80ms Economy ~~$585~~ $535

21M 68ms Miniscribe ~~$675~~ $625

33M 68ms Miniscribe ~~$725~~ $675

SERENDIPITY ASSOCIATES
5036 W Banff Lane • Glendale, AZ 85306
(602) 938-3729

Reader Service #188

DOS And Unix Part III

Tom Bing
2755 Carolyn Drive
Smyrna, GA 30080

Stalking The Great 'awk'

Instant gratification; it's the reason many of us were attracted to PCs in the first place. Maybe we had worked on a big mainframe that we shared with 100 other people, and it was great to have a whole machine to yourself, even a small one. With a BASIC interpreter, we could test a programming idea in a matter of seconds. In a spreadsheet model of a mortgage repayment schedule, we could see the effects of changing interest rates just as quickly.

People who like this aspect of PCs will feel right at home with 'awk', one of the Unix utilities in the MKS Toolkit. Of course, before we got to the "instant gratification" part of using BASIC or a spreadsheet, we had to invest some time to learn it. You may ask, "Why learn awk? Is it worth the effort and time?". As to "why", I have found that awk will enable you to do jobs on the PC that you wouldn't have even started before. For instance, maybe you've got a mailing list in a computer file that you'd like to convert to a data base input file for dBase. With an awk "script" or program, you can split the list into last name, first name, street address, and so on, and store the output in a fixed-record-length file, or even a comma-separated file. You can even get the script to take into account that some addresses won't have middle names, some won't begin with "Mr." or "Ms.", and some will have "Jr." after them. Can you imagine even trying to program that task in BASIC? I did it in awk with a 73-line script. You'll read about other unusual awk applications later in this article. The point is that awk opens up new vistas of what's possible on the computer, and the time awk has saved me has amply repaid the time I spent learning it.

For this article, we will look at the basics of awk, with simple examples. Awk can be thought of as a translator, a way to transform a file based on its contents and using just about any logical rules we want to apply. For instance, we might want to read a file consisting of columns of numbers, and if there are more than four columns on a line, print the last column. The awk script that does this is simply:

```
NF > 4 { print $NF }
```

and we can run it by typing:

```
awk 'NF > 4 { print $NF }' columns.in >
   lastcol.out
```

In this case, the "columns.in" file contains our input, and the output is written to "lastcol.out". The only "assumptions" awk makes about the input are that each line of input is one "record" and that one or more spaces or tabs separate each column (or "field") on a line. If you want to use different record and field separators, it's easy to tell awk that the defaults do not apply; more about this later.

The part of the awk script in curly braces — { print $NF } — is called an *action*; the *pattern* (in this case, NF > 4) is the test condition which, if true, causes the action to be performed. The pattern-action statement above could be given in English as "if there are more than four fields in a record, print the last field". If an action has no pattern in front of it, it will be performed for all input records. Awk examines each input record (a line of a file or a line typed at the keyboard) and processes it using each pattern-action pair in the awk script. From here on, a pattern-action statement will be called a "command".

Let's explore the pattern-action concept a little further. To get an idea of the way it works, let's imagine a kindergarten teacher using flash cards to teach a child addition. The teacher holds up a card with an addition problem on it (for example, 3 + 5) and the child gives the answer (8). This is the way awk works: when presented with a pattern, it responds with an action. For instance, we could write an awk program to simulate the responses of a child who can add two numbers, but can't do other math problems. It would look like this:

```
$2 == "+" { print "THE ANSWER IS "
   $1 + $3 }
$2 != "+" { print "I DON'T KNOW." }
```

If the above instructions were stored in a file named *adder*, and we type:

```
awk -f adder <CR>
```

then the program will take each line we type at the terminal as a "pattern" and it will respond with an "action". It will continue this until we type Control-C on a line (or DELETE if we're running awk on a Unix system). Let's look at a run of the above program.

```
c:/>awk'-f'adder
3 + 5
THE ANSWER IS 8.
2 * 3
I DON T KNOW.
5 + 5
THE ANSWER IS 10.
^C (Control-C)
c:/>
```

We would have gotten the same three lines of output if we had created a file named "problems.3" containing the three problems and typed:

```
awk -f adder problems.3
```

The records are assumed to have one or more spaces separating the numbers from the "+" or "*" sign. If we just entered "2+3" as input (no spaces), awk would read this as a line with a single field, and the value of the second field ($2) would be "" instead of "+". As a result, it would print "I DON'T KNOW".

Awk also has a formatted print function ("printf", just like C) but we're going to keep it simple for this article.

We can "jazz up" this script to make it accept input lines that don't have spaces around the "+" sign. Here's the revised script:

```
BEGIN { FS = "+" }
NF == 2 { print "THE ANSWER
   IS " $1 + $2 }
NF != 2 { print "I DON'T KNOW." }
```

The first command above introduces BEGIN, which is a special pattern which is only true at the beginning of processing; that is, before any lines have been read. The action, which sets FS to "+", is thus performed once before any input lines are read. FS is a variable which has a special meaning to awk; it defines the input field separator. Before, we used one or more tabs or spaces as the FS value; this is the default, so we didn't have to tell awk about it in the original script. Since the "+" is now the field separator, the numbers before and after it are now field 1 ($1) and field 2 ($2). The number of fields on an input line is computed and stored in another special variable, NF. Thus, if there are two fields on a line, NF will be 2. Also, $NF and $2 will refer to the same second (and last) field. The second command above means, "If there are exactly two fields on a line, print the answer as the sum of field 1 and field 2". The command below has exactly the same result:

```
NF == 2 { print "THE ANSWER IS "
   $1 + $NF }
```

The third command is shown again below:

```
NF != 2 { print "I DON'T KNOW." }
```

This command means, "If there are more or less than two fields on the line, print the words I DON'T KNOW". This command would be executed if there was no "+" sign, or more than one "+" sign, on the input line. The "==" and "!=" symbols mean "is equal to" and "is not equal to", respectively. To sum up: the action in the first command is performed only once. Then, each line of input is processed using the second and third commands, and each line of input will result in either an addition or an "I DON'T KNOW" message, depending on which pattern was true.

Awk can process more than just numbers, but we need to make sure we handle number and character input intelligently. Fortunately, awk provides some impressive tools for just this purpose. If I type in "2 + 7" to the above awk script, the result will be "THE ANSWER IS 9". If I type "A + 7", I get "THE ANSWER IS 7". Why? The answer will horrify you Pascal programmers out there. If a field is being used in a mathematical formula, the result will be computed by ignoring the part of the field beyond the last numeric character. Thus, in the "$1 + $2" addition, the "A" in the "A + 7" gets handled as "" + 7, and the "" is interpreted as a 0. If I had typed "4A + 6", awk would have come back with "THE ANSWER IS 10". In other words, awk handles field values in a way which is at least roughly appropriate to what's being done with them, such as addition or subtraction. Since we are adding, which is what you do with numbers, awk will handle $1 and $2, as much as possible, as numbers. If the second command had read:

```
NF == 2 { print "THE ANSWER IS " $1
   " PLUS " $2 }
```

we could type in "4A + 7" as input and get:

```
THE ANSWER IS 4A PLUS 7
```

Because awk is not being asked to do any math in the print statement above, it just prints the fields back out as it read them, as character strings.

If we want awk to handle addition problems of more than two numbers separated by "+" signs, we have to introduce a "for" loop, similar to ones used in the C language, into the script. Our script would be:

```
BEGIN { FS = "+" }
NF < 2 { print "NOT AN ADDITION
    PROBLEM"; next }
{
sum = 0
for (i = 1; i <= NF; i++ )
      {
      sum = sum + $i
      }
print "THE ANSWER IS " sum
}
```

Notice a few things here. First, there are only three commands. The first is the BEGIN command. The second is the "NF < 2" command. If there aren't two fields on a line, there aren't at least two numbers to add. The script prints a message, and the "next" means "read the next input record and start the script over again". If there *had* been two or more fields, the script would have continued to the third and last command, which spans several lines. The third command begins with a "{", that is, no pattern is specified, so the action will be performed on all input lines which got this far. The "sum" variable is set to zero, and the variable "i" is stepped from 1 through the number of fields (NF) by the "for" loop. On the first pass through the loop, "sum" is increased from zero to $1, on the second it is incremented by $2 (the value of $i when i equals 2), and so forth, through "sum = sum + $NF". It's like C language, except that we don't put a semicolon at the end of each statement. Semicolons are only used between two or more statements on one line (as with "next" above). Also, we didn't have to tell awk that sum and i are numbers; it figures that out by what we do with them.

This is a lot of words to describe a primitive calculator, but here's the point: you can start with a simple awk script and test it with a few lines of input in minutes. As your tests reveal problems, you just edit the script file and test again. We never have to compile the script.

Awk, like C, also has functions, both built-in ones (about 20) and the "roll-your-own" variety. Let's use one of each kind to refine our script even more. The awk "gsub" function is used for global substitution. If we had an input line and we wanted to replace every occurrence of the word "this" with "that", we would use the "gsub" function like this:

```
gsub(/this/,"that",$0)
```

The "$0" is awk notation for "the entire current line of input". In our addition script, if we want to get rid of all blank spaces in the input, we would use:

```
gsub(/ /,"",$0)
```

Now, we're going to create our own function, "isnumber(fld)". It will return a 1 if the character string "fld" can be read as a number, or a zero otherwise. A "regular expression" (see the Part II article) can be used to perform this test. The whole function is given below, and the regular expression is inside the slashes (//) in the "if" statement. The "if" statement condition is true if the "fld" character string *matches* the regular expression. The match operator is the tilde (~). The expression "a ~ b" is true if variable "a" contains the *entire* value of "b". If it matches b, then "a ~ b" evaluates to 1 (1 is the truth value; zero is false).

```
function isnumber(fld) {
if ( fld ~ /^(\+|-)?[0-9]+\.?[0-9]*$/ )
      {
      return 1
      }
else
      {
      return 0
      }
}
```

Let's break the above regular expression down into parts:

`^(\+|-)?`	This means that there can be at most one plus or minus sign, and it has to be the first character, if it is present.
`[0-9]+`	This means that there has to be at least one digit in the number.
`\.?`	This means that a literal decimal point can occur either zero times or one time; i.e., at most once.
`[0-9]*$`	This means that the number ends with zero or more digits after the decimal.

The regular expression above is taken from page 30 of *The Awk Programming Language* by Aho, Kernighan, and Weinberger. This book also gives a good explanation of regular expressions. The square brackets in a regular expression indicate a range of allowable characters, such as [A-Z] for any capital letter. The backslashes (\) above force the "." and "+" to be taken literally. The "^" and "$" refer to the beginning and end of a string, respectively. Also, the *order* of the parts of the regular expression is important; the sign *has* to precede any digit in a number for a match to occur in the example above. There are three special characters which indicate how often a character or string must be repeated:

?	Zero or one times
*	Zero or more times
+	One or more times

It takes a little getting used to, but it has its advantages. An old BASIC routine I wrote to do the same thing is 50 lines long.

Regular expressions in awk scripts can be placed inside slashes or quotation marks. The following two expressions are equivalent in the Unix version of awk:

```
fld ~ /^(\+¦-)?[0-9]+\.?[0-9]*$/
fld ~ "^(\\+¦-)?[0-9]+\\.?[0-9]*$"
```

The second expression is slightly longer; notice the two extra backslashes. When awk interprets a quoted string, any single backslashes are removed; any double ones (\\) become single. It's a little strange, but no worse than the hassle of balancing parentheses in BASIC or Pascal. The second form above is the ONLY one that works with the MKS Toolkit version of awk; the first one produces an error message. This is about the only area I know of where the Toolkit and Unix versions don't work alike. The Unix version of awk handles either form correctly.

At this point, the whole script looks like this:

```
BEGIN { FS = "+" }
NF < 2 { print "NOT AN ADDITION PROBLEM";
                                    next }
{
gsub(/ /,"",$0)
sum = 0
for (i = 1; i <= NF; i++ )
     {
     if ( isnumber($i) )
          {
          sum = sum + $i
          }
     else
          {
          print "IN " $0
          print $i " IS NOT A NUMBER."
          next
          }
     }
print "THE ANSWER IS " sum
}
function isnumber(fld) {
if ( fld ~ "^(\\+¦-)?[0-9]+\\.?[0-9]*$" )
     {
     return 1
     }
else
     {
     return 0
     }
}
```

Normally, awk commands are executed in the order they occur in the script. The function definition can be placed anywhere a command can. The function is not actually called or executed until it is used in a command.

The awk language is not "strongly typed", as programmers put it. In a strongly typed language, variables have to be declared as integer, character, etc., before they are used. In contrast, awk is not typed at all. The fields ($1, $2) and variables (sum, fld) we have used are treated as character strings or numbers, depending on the expression in which they are used. Awk also supports arrays, with the novel feature that the array subscript is a character string. The following assignments are perfectly legal in awk:

```
europe["France"] = "yes"
europe["Spain"] = "yes"
```

This way, you can read a country name from an input and test it:

```
country = $1
if ( country in europe )
...
```

The computer science people call these things "associative arrays". Like ordinary awk variables, you don't declare them before you use them. If the array subscripts represent integers ("1", "2", "3", etc.), we can use a "for" loop like the one described above to process each array element:

```
for( i=1; i<=10; i++ )
     {
     total = total + amount[i]
     }
```

This article only scratches the surface. I've used awk for jobs like these:

- document conversion between different word processors
- analyzing function calls and menu options in source code
- turning ASCII files into VP-Planner .WKS spreadsheet format
- spotting data errors (short records, missing fields, bad dates) in data base load files
- printing the line number and contents of duplicate lines in reports

In the last year and a half, I've written over 70 awk scripts that were valuable enough to keep, and about three-fourths of these were less than 30 lines long; one is about 300 lines. What's more, I've written dozens of little two-liners that I needed once, and they worked fine, but I threw them away. Why? Because it takes less time to write them again than to remember their directory and file name. Awk is great for automating those jobs that we know the computer can do best, but writing and testing a C or BASIC program for the job is just too much trouble. MKS has done a real service by making this great utility available to PC users.

Continued from Page 39

sion and BIOS numbers from the VER command), and a list of ALL hardware add-ons (including brand and model number) installed in your computer. The list of hardware add-ons should specifically include memory capacity (either added to an existing board or on any add-on board), all other internal add-on boards (e.g., modems, bus mouse or video cards), the brand and model of the CRT monitor you have, and the brand and model of the printer with the type of interface (i.e., serial or parallel) you are using. Also be sure to include a listing of the contents of the AUTOEXEC.BAT and CONFIG.SYS files unless you have thoroughly checked them out for potential problems (e.g. TSR conflicts). If the problem involves any application software, be sure to include the name and version number of the program you are running when the problem appears.

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

Products Discussed
HUG SOFTWARE

HADES II (885-3040)	$40.00
Upgrade for original owners	15.00
Powering Up (885-4604)	12.00

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

SOFTWARE

Flipfast Guides (includes shipping)	$12.00

FlipFast Guide to MS-DOS
FlipFast Guide to GW-BASIC
William M. Adney
P.O. Box 531655
Grand Prairie, TX 75053-1655

MS-DOS 3.3 Plus (OS-51-3)

List Price	$149.00
Mail Order w/Update Card Only	49.00

Heath/Zenith Computer Centers
Heath Company Parts Department
Hilltop Road
St. Joseph, MI 49085
(616) 982-3571 (Parts Only)
(800) 253-0570
(Heath Catalog Orders Only)

Mace 5 Utilities	$99.00
Mace Gold Utilities	149.00

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

Diagnostics for the Heath/Zenith Personal Computer

Robert C. Brenner
Brenner Information Group
13223 Black Mountain Road #430
San Diego, CA 92129

As HUG computer users, there are a number of things we can do to maintain our systems in tip-top condition. There are also a number of things we can do when failures occur. Besides periodic cleaning and other forms of preventive maintenance, we can, if necessary, perform some adjustments (disk drive speed and read/write head cleaning) and possibly replace components (although we usually replace an entire malfunctioning board).

If we're technically oriented, we can do all the above plus perform disk drive radial head alignment, and Track 00 switch, index sensor, and head azimuth adjustment. Usually these operations require sophisticated test equipment. However, there are special diagnostics products now on the market that even let novice technicians perform complex tests without additional equipment other than the computer being tested.

When I proposed writing an article on diagnostics testing for Heath/Zenith PCs, I didn't realize the challenge I would face.

I contacted 28 manufacturers who developed and sold PC diagnostics packages. In my letters, I had been very careful to say that I was seeking diagnostics products that would work with the Heath/Zenith PCs. Of the 28 requests for information, I received five product packages (with disks) and nine information-only packages. I also received a dozen phone calls from sales reps, but upon questioning them, it didn't take long for my questions to exceed their limited knowledge level. I was usually referred to the caller's company president who was, like myself, technically oriented.

Given the types of diagnostics products for the microcomputer, I decided to partition these test products into four categories: 1) software, 2) firmware, 3) hardware, and 4) some combination of the first three.

Software programs are used to conduct performance tests on major system components (e.g., CPU, ROM, RAM, controllers, and disk drives). These programs include CheckIt (TouchStone), Disk Drive Analyzer (Verbatim), Disk Technician Advanced (Prime Solutions), DR-Chips (The Software Link), Drive Diagnostics (ASKY), HELPME (California Software Products), PC-Technician (Windsor Technologies), Service Diagnostics (SuperSoft), and Test Drive (Microsystems Development).

Most commercial diagnostics software products are for disk drive testing and alignment. An operating disk drive is required. Of the programs that test disk drives, some conduct both speed and alignment tests. One vendor reported that almost 50 percent of all new floppy disk drives sold to the public fail to meet the manufacturer's alignment specifications. And re-manufactured drives are sometimes sold after years of field operation although not all components are upgraded causing these drives to operate narrowly close to out-of-tolerance alignment specification.

Representatives from computer repair training company National Advancement Corporation insist that alignment with a digital disk is not suitable. They claim it's impossible to get the necessary resolution from a digital disk and some tests just can't be done digitally (azimuth, read amplitude, write amplitude, track 00 switch, and resolution). They feel the only way to be sure about alignment is to use an analog disk and some analog measuring device such as an oscilloscope. They suggest that less sophisticated disk test programs that simply report an out-of-alignment condition are actually better.

Several companies sell ROM firmware that can be used to check and validate a system without using a hard disk. In all the cases that I encountered, the ROM POST was more extensive than that installed in our systems.

Hardware adapter cards or complete test systems with their own PC can also be used to test a Heath/Zenith microcomputer. These include POSTcard from Award Software and R.A.C.E.R. from Ultra-X Inc. The R.A.C.E.R. folks claim it can find component faults on dead system boards and cards in backplane designed computers. It checks the CPU, keyboard, timer, memory, DMA channel, interrupt controllers, coprocessor, clock generator, serial and parallel ports and RAM memory from 0 to 640 K.

After reviewing the number of diagnostics products available, I decided the best approach is to describe each vendor's product in a separate article. Each article will take you step-by-step from removing the product from its package through installation and operation on a Heath/Zenith 386. In the analysis, I'll describe the product, the price, the extent of testing performed, and a candid evaluation of its usefulness and user-friendliness.

The first product to meet my critical eye is CheckIt, a set of diagnostics programs from TouchStone Software Corporation. This $149 menu-driven diagnostics package checks and then reports the exact configuration of your computer system, including the DOS device assignments. It also conducts diagnostics tests on the main system board, base, expanded and extended memory, hard and floppy disks, the video sub-system, the communication ports, and the keyboard, printer, mouse and joystick peripherals. A third part of the package performs sophisticated dhrystone, whetstone, and characters-per-second benchmark tests on the processor, coprocessor, and video display.

From the moment I received CheckIt, I knew it had, at its least, class in the design of its packaging. A colorful black, white, and red plastic VCR-cassette-like holder contained an easy-to-read 160-page instruction book, release notes for version 2.1, a registration card, and an envelope containing two 360K disks. The quality of Touchstone product design was evident in each component in the package.

The directions were easy to follow. First, I created a sub-directory called CHECKIT. Then, I copied the files on each of the two CheckIt disks into the directory, and simply typed in CHECKIT and pushed Enter to begin. The first time the program is run, it asks you to enter your name and company. From then on, it displays this information on the CheckIt opening screen. This information also appears at the top of any report you output to the printer. Pressing any key replaces the opening screen with a loading screen while the program determines the system configuration. This "investigation" screen keeps you informed as it checks your system's configuration. As each step is checked off in its own box, you see if the computer is configured for remote operation, the type of BIOS installed, whether the check of the system components and RAM is complete and if you have a math co-processor or mouse installed.

Once complete, pressing any key switches the display to a Global Menu that gives you access to drop-down (pull-down) windows for System (information), Tests, Benchmarks, Tools, Setup, and Exit for terminating the program and returning to the DOS prompt.

When I selected System, a sub-menu offered Configuration, Interrupts, CMOS Table, and Device Drivers options. I selected Configuration and pressed Enter. A new screen display told me I had an 80386 AT Machine with a Zenith ROM BIOS, DOS 3.30, 640K base memory, no expanded or extended RAM, a 44 Mb hard disk drive, a 1.2 Mb 5.25-inch floppy drive, a 1.44 Mb 3.5-inch microfloppy disk drive, information about the serial and parallel ports, and more. This screen was quite useful.

The Interrupts option listed the assignments for IRQ0 through IRQ15 and DMA0 through DMA7. The CMOS Table gave me an updating date and time display and described the floppy disk drive, base memory, display, and hard disk configuration. It was here that I noted my hard disk was a type 42, with 1024 cylinders, five heads, 17 sectors per track cylinder and contained 44,564,480 bytes of storage capability. The Device Drivers option provided information about the block and character devices in the system.

Diagnostics Packages Requested for Evaluation

AID/88
Interrogator
Award 386 Modular BIOS
PC-Technician
Blue Magic
POSTcard
CheckIt
R.A.C.E.R.
D.A.D.S.
RACET
Disk Drive Analyzer
ReadiScope
Disk Technician Advanced
S.T.A.T.
DR-Chips
Service Diagnostics
Dymek Alignment Disk
Test Drive
HDtest
WindsorPOST
HELPME
Your PC Analyst
IDEA

While this was already more information than I expected, the power of CheckIt became quite evident once I selected Tests. Here I could check almost every component in the system/memory, hard disk, floppy disk, system board, real-time clock, serial and parallel ports, printer, video, input devices, and even one option to check "everything". Selecting the Memory Test option, I answered the "Run Memory Test?" question with Y, Enter. While the test checks base, extended, and expanded memory, my system contains just 640K of base memory, so I noted a check of the test program buffers and then the program stepped a pseudo-random test pattern through every chip in the 640K map area. This test took about a minute to perform. Check-marks appear in a status window as the program steps through the memory segments. An optional "thorough" test will write, read, and verify eight additional test patterns including walking bit, inverse walking bit, checkerboard, and inverse checkerboard patterns. No bad chips were found (thank goodness), but if one or more were, the program draws a map of the board showing where the bad chip is located.

The hard disk test option is non-destructive and quickly checked the controller card. Then it performs three specific read operations on each track: linear read, a butterfly read, and a random read. During the last two tests, you hear the drive read/write heads really get exercised. The butterfly read is the words case test for the hard disk seeking mechanism. The random read realistically simulates actual hard disk activity. No errors occurred, but the program will report errors marked by the low level format, marked by DOS, minor soft errors, a bad track outside the bounds for a partition, an error within a DOS partition, a bad spot on a volume not allocated by DOS, an error in a file, and major errors such as partition table and bad sectors on critical areas of the disk. This test sequence took just over 15 minutes to perform.

Random read and random write tests can be performed on a floppy disk drive. This test catches compare, disk changed, drive not ready, seek, read, write, and write protect errors.

The System Board test quickly checked the CPU, and DMA and interrupt controllers. It took just 10 seconds for this to complete.

The video test is also extensive and produces a variety of displays on the screen verifying the capability and suitability of the video RAM and display circuitry. From bar matrices to changing color boxes, the test is fast and accurate.

When I selected the Main System Benchmark test, I was amazed to see it quickly analyze and report the CPU speed (3489 Dhrystones at 16.01 MHz), video speed (5316 characters per second), and math calculation speed (65.8K Whetstones without a math co-processor).

The Hard Disk Benchmark was also fast and measured the average seek time (30.5 ms), average track-to-track seek (3.8 ms), and average data transfer speed (264.1K per second) of a 43K block of data. This test is useful with disk caching software to determine how various caching settings will affect hard disk performance.

A Tools menu lets you locate the RAM chips, set the real-time clock, and perform a destructive low-level format of the hard disk.

A Setup menu lets you toggle the color, send an activity log to the printer or a disk file, and generate a custom RAM chip layout so CheckIt can locate bad RAMs as they occur and are found by diagnostics testing.

Conclusion: At $149, CheckIt is an outstanding program for any HUG member's toolbox. A few minutes using this program can give you welcome reassurance, as well as save considerable time when things start to go wrong. CheckIt is comprehensive and user-friendly. My recommendation: Buy it.

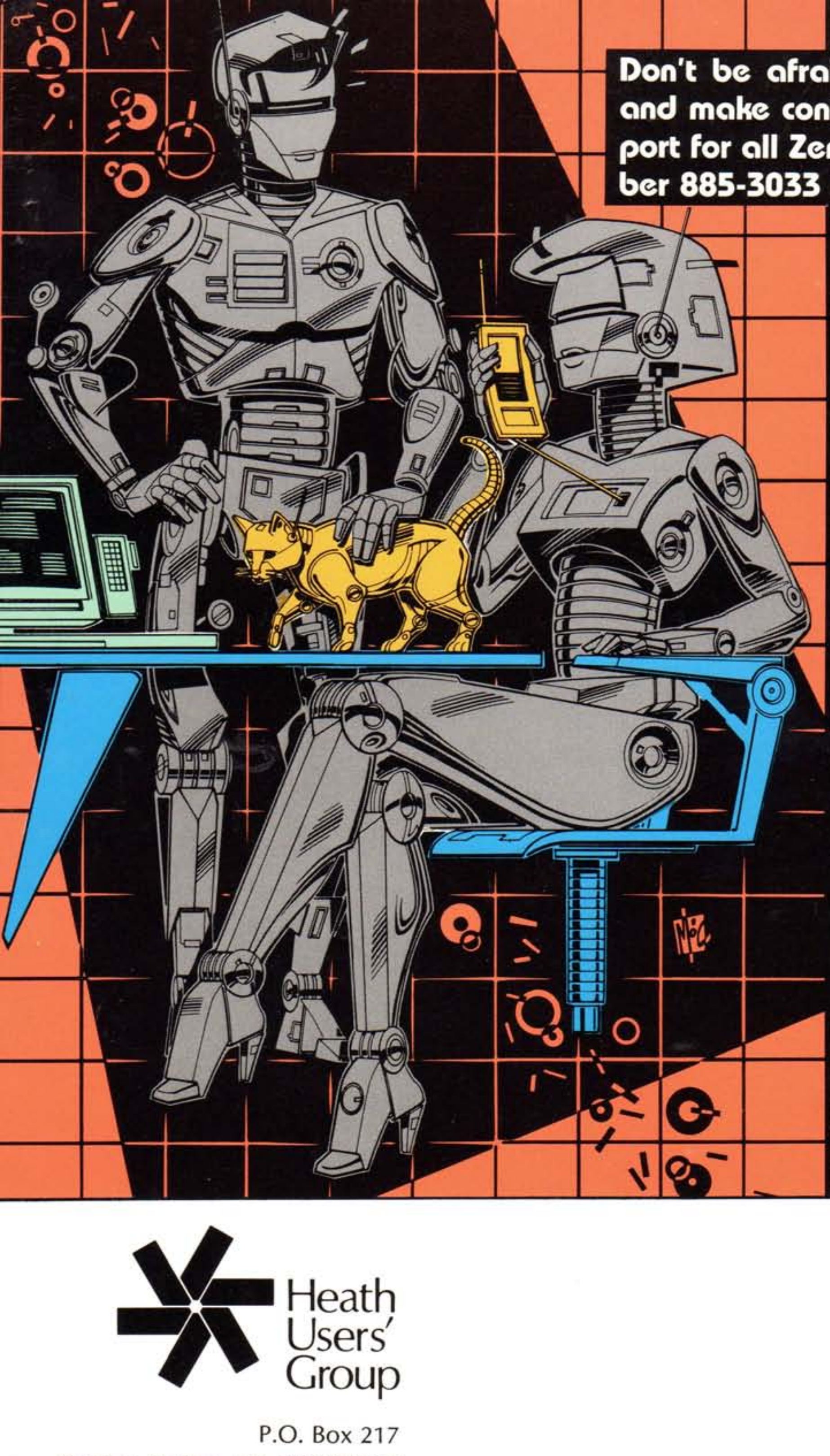
In the next article, I'll evaluate Disk Technician Advanced from Prime Solutions.

HADES II

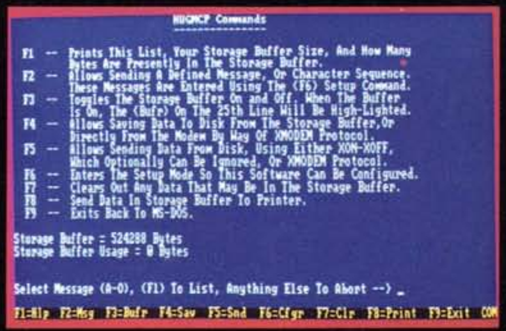
It's HOTTER than ever! Jam-packed with new features, HADES II still remains the easiest-to-use disk editor ever! Just look at some of the features:

- 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

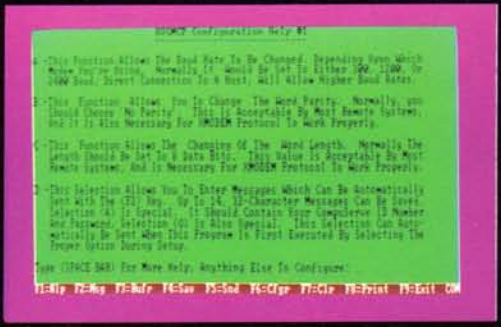
HADES II is still only $40, and original HADES owners can upgrade their distribution disk for only $15. Call HUG today at: (616) 982-3463.



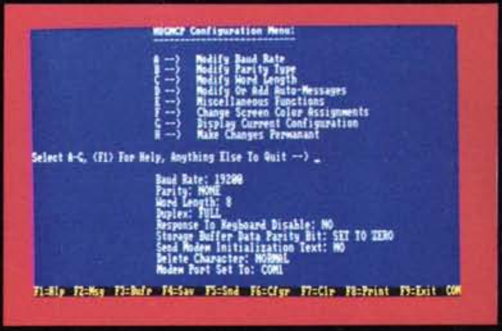
Don't be afraid to communicate! Get HUGMCP. and make contact the easy way. Now with support for all Zenith Laptops, order HUG Part number 885-3033 today.



```
                           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 14, 32-Character Messages Can Be Saved.
   Selection (A) Is Special. It Should Contain Your CompuServe ID Number
   And Password. Selection (O) Is Also Special. This Selection Can Auto-
   matically Be Sent When This Program Is First Executed By Selecting The
   Proper Option During Setup.

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



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

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

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

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

Heath Users' Group

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

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

POSTMASTER: If undeliverable, please do not return.

$2.50
P/N 885-2121