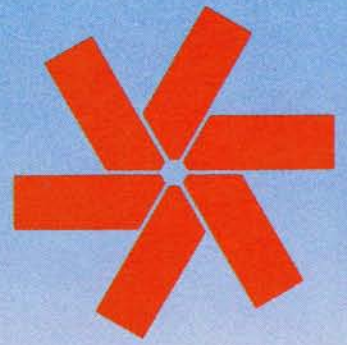


July 1990

REMark[®]



The Official Heath/Zenith Computer Users Magazine

*Don't forget our "Classified"
Section in every issue.*

Page 47

SUMMER



The VOTE'S In!

**TRY ME . . .
AT LEAST ONCE!**

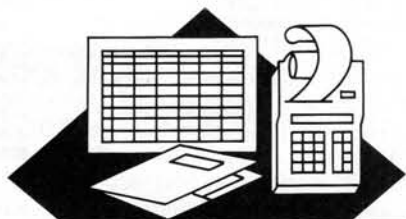
“. . . and I know you'll keep coming back! Why? Because I'm totally different now. I used to be amateurish and old-fashioned, but now I'm like, totally professional. There are practically no limits to what I can do! My measurements are 30,000, 33, 320. In BBS terms, that's messages, megahertz, and megabytes of mass storage, in that order! Yes, I still have the Bargain Centre, but now it's part of a new feature called, "The Keyboard Shopping Club." I'm still quite inexpensive too, so why not call my "HUG LINE" at least once? Just set your modem for 2400 baud or less (8N1), and call (616) 982-3956. I'm waiting to hear from you!

She's Out!

REMark®

Volume 11, Issue 7 • July 1990

| Reader Service No. | | Page No. |
|--------------------|---------------------------------|----------|
| 104 | FBE Research Co., Inc. | 8 |
| 175 | Key Computer Publications | 14 |
| 136 | Lindley Systems | 46 |
| 117 | Payload | 38 |
| 193 | QuikData, Inc. | 4 |
| 188 | Serendipity Associates | 14 |
| 149 | WS Electronics | 45 |



ACCOUNTING & TAX

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

PC Compatible

| | |
|---|-----------|
| dBASE III — Part 5 | 7 |
| <i>D. R. Cool</i> | 7 |
| Programming With VGA — Part 2 | 13 |
| <i>Mark Mangerson, Greg McDonald</i> | 13 |
| Powering Up — Volume 2 | 23 |
| <i>William M. Adney</i> | 23 |
| Accessing the System Bus on the Z-181 Laptop Computer Part 2 | 39 |
| <i>Dennis L. Myers</i> | 39 |
| Getting Started With . . . GDU — Part 2 | 43 |
| <i>Jan Axelson</i> | 43 |

H/Z-100 and PC Compatible

| | |
|-----------------------------------|-----------|
| Assembly Language — Part 6 | 20 |
| <i>Pat Swayne</i> | 20 |
| On the Leading Edge | 33 |
| <i>William M. Adney</i> | 33 |

General

| | |
|--|-----------|
| Graphics Printer or Epson FX — Part 7 | 28 |
| <i>John A. Day</i> | 28 |

Resources

| | |
|-----------------------------------|-----------|
| HUG Price List | 2 |
| Buggin' HUG | 5 |
| H/Z Related Products | 47 |
| Classified Ads | 47 |

PC Compatibles

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

HUG

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

Printer Imperial Printing
St. Joseph, MI

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

*U.S. Funds

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

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

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

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

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

REMark is a registered trademark of the Heath/Zenith Users' Group, St. Joseph, Michigan.

Copyright © 1990, Heath/Zenith Users' Group

| PRODUCT NAME | PART NUMBER | OPERATING SYSTEM | DESCRIPTION | PRICE |
|-------------------------------------|---------------|------------------|-------------------------|-------|
| H8 - H/Z-89/90 | | | | |
| ACCOUNTING SYSTEM | 885-8047-37 | CPM | BUSINESS | 20.00 |
| ACTION GAMES | 885-1220-[37] | CPM | GAME | 20.00 |
| ADVENTURE | 885-1010 | HDOS | GAME | 10.00 |
| ASCIRITY | 885-1238-[37] | CPM | AMATEUR RADIO | 20.00 |
| AUTOFILE (Z80 ONLY) | 885-1110 | HDOS | DBMS | 30.00 |
| BHBASIC SUPPORT PACKAGE | 885-1119-[37] | HDOS | UTILITY | 20.00 |
| CASTLE | 885-8032-[37] | HDOS | ENTERTAINMENT | 20.00 |
| CHEAPCALC | 885-1131-[37] | HDOS | SPREADSHEET | 20.00 |
| CHECKOFF | 885-8010 | HDOS | CHECKBOOK SOFTWARE | 25.00 |
| DEVICE DRIVERS | 885-1105 | HDOS | UTILITY | 20.00 |
| DISK UTILITIES | 885-1213-[37] | CPM | UTILITY | 20.00 |
| DUNGEONS & DRAGONS | 885-1093-[37] | HDOS | GAME | 20.00 |
| FLOATING POINT PACKAGE | 885-1063 | HDOS | UTILITY | 18.00 |
| GALACTIC WARRIORS | 885-8009-[37] | HDOS | GAME | 20.00 |
| GALACTIC WARRIORS | 885-8009-[37] | CPM | GAME | 20.00 |
| GAMES 1 | 885-1029-[37] | HDOS | GAMES | 18.00 |
| HARD SECTOR SUPPORT PACKAGE | 885-1121 | HDOS | UTILITY | 30.00 |
| HDOS PROGRAMMERS HELPER | 885-8017 | HDOS | UTILITY | 16.00 |
| HOME FINANCE | 885-1070 | HDOS | BUSINESS | 18.00 |
| HUG DISK DUPLICATION UTILITIES | 885-1217-[37] | CPM | UTILITY | 20.00 |
| HUG SOFTWARE CATALOG | 885-4500 | VARIOUS | PRODUCTS THRU 1982 | 9.75 |
| HUGMAN & MOVIE ANIMATION | 885-1124 | HDOS | ENTERTAINMENT | 20.00 |
| INFO. SYSTEM AND TEL. & MAIL SYSTEM | 885-1108-[37] | HDOS | DBMS | 30.00 |
| LOGBOOK | 885-1107-[37] | HDOS | AMATEUR RADIO | 30.00 |
| MAGBASE | 885-1249-[37] | CPM | MAGAZINE DATABASE | 25.00 |
| MAPLE | 885-8005 | HDOS | COMMUNICATION | 35.00 |
| MAPLE | 885-8012-[37] | CPM | COMMUNICATION | 35.00 |
| MISCELLANEOUS UTILITIES | 885-1089-[37] | HDOS | UTILITY | 20.00 |
| MORSE CODE TRANSCIEVER | 885-8016 | HDOS | AMATEUR RADIO | 20.00 |
| MORSE CODE TRANSCIEVER | 885-8031-[37] | CPM | AMATEUR RADIO | 20.00 |
| PAGE EDITOR | 885-1079-[37] | HDOS | UTILITY | 25.00 |
| PROGRAMS FOR PRINTERS | 885-1082 | HDOS | UTILITY | 20.00 |
| REMARK VOL 1 ISSUES 1-13 | 885-4001 | N/A | 1978 TO DECEMBER 1980 | 20.00 |
| RUNOFF | 885-1025 | HDOS | TEXT PROCESSOR | 35.00 |
| SCICALC | 885-8027 | HDOS | UTILITY | 20.00 |
| SMALL BUSINESS PACKAGE | 885-1071-[37] | HDOS | BUSINESS | 75.00 |
| SMALL-C COMPILER | 885-1134 | HDOS | LANGUAGE | 30.00 |
| SOFT SECTOR SUPPORT PACKAGE | 885-1127-[37] | HDOS | UTILITY | 20.00 |
| STUDENT'S STATISTICS PACKAGE | 885-8021 | HDOS | EDUCATION | 20.00 |
| SUBMIT (Z80 ONLY) | 885-8006 | HDOS | UTILITY | 20.00 |
| TERM & HTOC | 885-1207-[37] | CPM | COMMUNICATION & UTILITY | 20.00 |
| TINY BASIC COMPILER | 885-1132-[37] | HDOS | LANGUAGE | 25.00 |
| TINY PASCAL | 885-1086-[37] | HDOS | LANGUAGE | 20.00 |
| UDUMP | 885-8004 | HDOS | UTILITY | 35.00 |
| UTILITIES | 885-1212-[37] | CPM | UTILITY | 20.00 |
| UTILITIES BY PS | 885-1126 | HDOS | UTILITY | 20.00 |
| VARIETY PACKAGE | 885-1135-[37] | HDOS | UTILITY & GAMES | 20.00 |
| WHEW UTILITIES | 885-1120-[37] | HDOS | UTILITY | 20.00 |
| XMET ROBOT X-ASSEMBLER | 885-1229-[37] | CPM | UTILITY | 20.00 |
| Z80 ASSEMBLER | 885-1078-[37] | HDOS | UTILITY | 25.00 |
| Z80 DEBUGGING TOOL (ALDT) | 885-1116 | HDOS | UTILITY | 20.00 |

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

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

Price List

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

H/Z-100 (Not PC) Only

| | | | | |
|------------------------------|-------------|-------|--------------------------|-------|
| ACCOUNTING SYSTEM | 885-8048-37 | MSDOS | BUSINESS | 20.00 |
| CALC | 885-8043-37 | MSDOS | UTILITY | 20.00 |
| CARDCAT | 885-3021-37 | MSDOS | BUSINESS | 20.00 |
| CHEAPCALC | 885-3006-37 | MSDOS | SPREADSHEET | 20.00 |
| CHECKBOOK MANAGER | 885-3013-37 | MSDOS | BUSINESS | 20.00 |
| CP/EMULATOR | 885-3007-37 | MSDOS | CPM EMULATOR | 20.00 |
| DBZ | 885-8034-37 | MSDOS | DBMS | 25.00 |
| DUNGEONS & DRAGONS (ZBASIC) | 885-3009-37 | MSDOS | GAME | 20.00 |
| ETCHDUMP | 885-3005-37 | MSDOS | UTILITY | 20.00 |
| EZPLOT II | 885-3049-37 | MSDOS | 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 |
| SPELLS | 885-3035-37 | MSDOS | SPELLING CHECKER | 20.00 |
| SPREADSHEET CONTEST PACKAGE | 885-3018-37 | MSDOS | VARIOUS SPREADSHEETS | 25.00 |
| TREE-ID | 885-3036-37 | MSDOS | TREE IDENTIFIER | 20.00 |
| USEFUL PROGRAMS I | 885-3022-37 | MSDOS | UTILITIES | 30.00 |
| UTILITIES | 885-3008-37 | MSDOS | UTILITY | 20.00 |
| ZPC II | 885-3037-37 | MSDOS | PC EMULATOR | 60.00 |
| ZPC UPGRADE DISK | 885-3042-37 | MSDOS | UTILITY | 20.00 |

H/Z-100 and PC Compatibles

| | | | | |
|--------------------------------|----------|---------|---------------------|-------|
| ADVENTURE | 885-3016 | MSDOS | GAME | 10.00 |
| ASSEMBLY LANGUAGE UTILITIES | 885-8046 | MSDOS | UTILITY | 20.00 |
| BACKGROUND PRINT SPOOLER | 885-3029 | MSDOS | UTILITY | 20.00 |
| BOTH SIDES PRINTER UTILITY | 885-3048 | MSDOS | UTILITY | 20.00 |
| CXREF | 885-3051 | MSDOS | UTILITY | 17.00 |
| DEBUG SUPPORT UTILITIES | 885-3038 | MSDOS | UTILITY | 20.00 |
| DPATH | 885-8039 | MSDOS | UTILITY | 20.00 |
| HADES II | 885-3040 | MSDOS | UTILITY | 40.00 |
| HELP | 885-8040 | MSDOS | CAI | 25.00 |
| HEPCAT | 885-3045 | MSDOS | UTILITY | 35.00 |
| HUG EDITOR | 885-3012 | MSDOS | TEXT PROCESSOR | 20.00 |
| HUG MENU SYSTEM | 885-3020 | MSDOS | UTILITY | 20.00 |
| HUG SOFTWARE CATALOG UPDATE #1 | 885-4501 | VARIOUS | PROD 1983 THRU 1985 | 9.75 |
| HUGMCP | 885-3033 | MSDOS | COMMUNICATION | 40.00 |
| ICT 8080 TO 8088 TRANSLATOR | 885-3024 | MSDOS | UTILITY | 20.00 |
| MAGBASE | 885-3050 | VARIOUS | MAGAZINE DATABASE | 25.00 |
| MATT | 885-8045 | MSDOS | MATRIX UTILITY | 20.00 |
| MISCELLANEOUS UTILITIES | 885-3025 | MSDOS | UTILITIES | 20.00 |
| PS's PC & Z100 UTILITIES | 885-3052 | MSDOS | UTILITY | 20.00 |
| REMARK VOL 5 ISSUES 48-59 | 885-4005 | N/A | 1984 | 25.00 |
| REMARK VOL 7 ISSUES 72-83 | 885-4007 | N/A | 1986 | 25.00 |
| REMARK VOL 8 ISSUES 84-95 | 885-4008 | N/A | 1987 | 25.00 |
| REMARK VOL 9 ISSUES 96-107 | 885-4009 | N/A | 1988 | 25.00 |
| REMARK VOL 10 ISSUES 108-119 | 885-4010 | N/A | 1989 | 25.00 |
| SCREEN DUMP | 885-3043 | MSDOS | UTILITY | 30.00 |
| UTILITIES II | 885-3014 | MSDOS | UTILITY | 20.00 |
| Z100 WORDSTAR CONNECTION | 885-3047 | MSDOS | UTILITY | 20.00 |

PC Compatibles

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

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

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

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

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

ORDERING INFORMATION

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

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

QUIKDATA - 13 YEARS OF H/Z SUPPORT!

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

ACCELERATE YOUR PC!

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

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

EXP-12 - \$295 (NEW PRICING!)
EXP386 - \$395 Much faster 16Mhz 80386 SX version

MEMORY UPGRADES

Note: All memory upgrades come without memory chips. Call for current chip pricing. Memory chip prices are down. Example: 256K 150ns DRAM is \$2.39 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/Z150/160 series with up to 704K of main memory, and about 512K for RAMDRIVE memory. Includes documentation, software RAMDRIVE disk, PAL and jumper wire. For the full 1.2 megs total memory, 45 256K DRAM chips are required.

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.

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

WINCHESTER UPGRADE KITS

PCW20 - \$269 Complete winchester setup for a H/Z150, 148, 158, 159, 160, PC etc. Includes 21 meg formatted half-height Segate ST-225 65ms drive, WX1 controller, cable set, doc.

PCW30 - \$349 32 meg with 38ms Segate ST-138.
PCW40 - \$395 42 meg with 28ms Segate ST-251-1.
PCW80 - \$629 80 meg with Segate ST-4096 full size drive.

We also have the DTC controllers (\$69) and daughter board expansions (\$69) to place a hard drive in the **H/Z148** computers

BARE WINCHESTER DRIVES

ST-125 - \$249 21 meg 37ms autopark 3.5" drive in 5" frame.
ST-225 - \$219 21 meg 65ms HH 5" drive.
ST-138 - \$295 32 meg 38ms autopark 3.5" drive in 5" frame.
ST-251-1 - \$349 42 meg 28ms autopark HH drive.
ST-4096 - \$579 80 meg 28ms autopark FH drive
MIN3180 - \$949 **157MB ESDI** Miniscribe 3180 half-height 18ms hard drive (Type 100).

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 - \$33 (add \$2 for Z100 installation for spacers)

ANY DRIVE IN YOUR PC/XT/AT

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

CCARD2 - \$89
CCARD - \$109
CCARD4 - \$139

FLOPPY DRIVES

MF501 - \$79 5" 360K DS/DD drive
MF504 - \$89 96 TPI 1.2 meg AT/Z100 drive
MF353 - \$85 720K 3.5" drive in 5" frame
MF355 - \$95 1.4 meg 3.5" AT drive in 5" frame
SIEM-R - \$39 40 track SS refurb (H8/H89 type)
TM100-4R - \$75 90 trk DS refurb (H8/H89 type)

8-BIT/Z100

We carry a full line of replacement boards, parts and power supplies for the H/Z89/90 and Z100. We also have some H8 boards available. We continue to fully support and carry a full line of hardware and software products for the H8/H89/90 and Z100 computers. This includes diskettes, printers, modems, hard drives, etc. Here's a sample.

| | |
|--|-------|
| H37 SOFT SECTOR CONTROLLER | \$169 |
| H8 SOFT SECTOR/WINNIE CONTROLLER PACKAGE | \$195 |
| DISKPACK for Z100 (allows reading and writing of PC formats including 1.2 and 1.4 meg on Z100.) | \$39 |
| MSDOS 3.1 FOR Z100 | \$89 |

OTHER STUFF

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

H-SCOOP

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. FAX: (414) 452-4344.

QUIKDATA, INC.

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

BUGGIN' HUG

The Modem and the Mouse Port

Dear HUG:

I've got a slight problem with my Prodigy Service — or perhaps the problem lies in my computer hardware.

The situation:

- I have two Zenith Data Systems' eaZy PCs — one in my home and the other at my summer cottage. Both have 640K RAM, 20 Meg. hard drives, 720K floppy drives, and are IBM compatible.
- My computer at home has a "regular" serial port (com1) and serial port labeled "mouse port" (com2). The regular serial port was added via a special ZDS plug-in module. Both serial ports are the 9-pin kind.
- My computer located at my summer home is identical except that it does not have a regular serial port (com1). It does, however, have a mouse port (com2).
- I have a 2400 baud Hayes type modem.
- For about a year, I have used Prodigy Service, utilizing my home computer, com2 (regular serial port), and a Hayes type modem. I also use several other information services using this setup and Procomm software.
- In an effort to see if I could access these info services using the mouse port (so I could use my computer at my summer cottage), I connected my modem to the mouse port, made the necessary software changes, and proceeded to access the information services via the telephone network.
- **This setup (mouse port) worked for all the info services (PCMAGNET, GENIE, and a number of others) used with Procomm Plus software.** However, it would not work with Prodigy accessed through Prodigy software.
- I then connected my modem to the mouse port of my computer at my cottage. Same result — all info services, but Prodigy can be accessed.
- It seems as though the modem just can't dial out when Prodigy is being used. No such problem with the other services, when the "mouse port" and Procomm software is used.
- If I dial the Prodigy number using Procomm software, I make the connection. However, as you know I can't use the service without the Prodigy software.

Question:

- It must be the Prodigy software causing

the problem. Are there any changes I can make?

- Any suggestions?
- Can you help me?
- I would be grateful for your thoughts.

Eugene R. Bernier
7630 Reuter
Dearborn, MI 48126

Sprint on the Z-100

Dear HUG:

I was quite interested in Mr. Rylander's article in the November issue. I had tried setting up Sprint to run on my Z-100 in its native mode some time ago, and had given up when I discovered that the Z-100 definition supported reverse video as the only screen attribute, and that I couldn't even call up the main menu using the Advanced Borland Interface. I quickly gave up and resigned myself to running Sprint under ZPC, which actually worked quite well, but somewhat slowly. After reading Mr. Rylander's article, I tried again, with considerably more success. And perhaps I can even pass along a little additional information for those trying to run Sprint on a Z-100.

The first thing I did was to change the set definition as suggested by Mr. Rylander. I even tried adding a few additional color combinations, and soon discovered why the definition was limited to eight combinations. While compiling the Z-100 definition, Sprint runs out of string space, presumably because of the long definitions for the up and down strings. Fortunately, the set string can be shortened.

Mr. Rylander's set string looks like this:

```
set %[m70%;^m07%;^m04%;^m40%;^m53%;^m35%;^m26%;^m62%;]
```

In each color combination, we have included the escape and 'm' characters. These can be moved out in front of the case structure, like this:

```
set ^[m%[70%;07%;04%;40%;53%;35%;26%;62%]
```

We now have room for an additional four or more color combinations before we run out of string space. Notice also that I've eliminated the final semicolon, defining this combination as the default for any attribute number higher than the number of cases (colors) defined. For my all-in-one Z-100, I use the following definition:

```
set ^[m%[40%;07%;73%;70%]
```

In looking through the Sprint documentation, and the .SPL file, it looks to me like Mr. Rylander's set 'Kludge', as he described it is the proper, and perhaps only way, to set up the color attributes for the Z-100 computer. The ANSI.SYS definition, I believe, refers to IBM specific attributes, although the Sprint manual doesn't come

right out and say it.

Even under ZPC I discovered that I couldn't use autospell because Sprint tried to beep using the IBM sound port. That problem is solved by entering the simple editor macro:

Bell :

```
rawout "^G"
```

Just enter this using Sprint, delete the ruler line by placing the cursor on the ruler line and pressing control-Y, and execute the editor macro by calling up the Utility menu, and executing Macro Run. After that the Z-100 will beep whenever Sprint rings the bell.

I also tried changing the Advanced Borland Interface to accept the function keys in their expansion disabled mode, but with no luck. It looks to me like Sprint strips the parity bit during input, effectively disabling input from any of the Z-100 function keys. Using keymap appears to be the simplest way to get around this problem. But if you should run Sprint without first running keymap, you don't have to reset the computer. You can call up the main menu by entering a control-@ (control-shift-2) followed by a control-J character. If you look at the keymap definitions, you can see that this is the key combination put out by the IBM F10 key. Now you can quit, and try it again with keymap installed.

I installed Sprint using SP-SETUP under ZPC with a hardware video port emulator similar to the one described by Pat Swayne some time ago, and also found that the installation would not run right. It would quit before copying over all the files. If I told SP-SETUP that I was using a hard disk, the installation would proceed correctly. I could only get away with that, however, because I have two eight inch drives.

As a final note, I am typing this letter with Sprint, without the help of keymap. I have modified the Advanced Borland Interface to recognize the function keys in their expansion mode. The stock interface has recognized several control-Q and control-K character combinations left over from WordStar. I have carried this over to a set of control-[(ESCAPE) character combinations representing the Z-100 function keys. The Borland file is copyrighted, but specifically grants permission for redistribution, provided that the copyright notice is intact, and that the redistribution is not for profit. When its finished, perhaps HUG will be able to distribute it to interested members.

A special thanks to Mr. Rylander for his article. Without it, I would still be talking to Sprint like an IBM using ZPC.

Sincerely,
Gary A. Appel
1318 Old Abbey Place
San Jose, CA 95132

Article on 'Installing the Intel Inboard 386/PC into the Heath/Zenith-150'

Dear Jim:

There are a few mistakes and omissions that I would like to correct via this letter.

First, the omissions. It is absolutely necessary to upgrade the ROM BIOS chips (2) in the H/Z-150/151/152 to the latest version. These chips are numbered 444-290-xx, and 444-260-xx where xx is the revision number (17 at this writing). If you fail to do this, a user would experience timing errors at boot up.

A 20 MHz upgrade to the INBOARD 386/PC is available from my company, Data Management Associates. Interested users can contact me there at P.O. Box 220, Litchfield, CT 06759 or (203) 567-5188.

Secondly, the errors. There may be some confusion about the piggyback boards mentioned in the article. These boards are **exTended** memory boards **not exPanded** boards as published. The Inboard 386/PC driver is called INBRDPC.SYS no INBRD.SYS, and the exPanded memory driver is called ILIM386.SYS not ILIM.SYS as published.

DESQVIEW/386 has been tested on the H/Z-150 and works, as well or better than WINDOWS/386 in some cases, especially the expanded memory manager QEMM386.SYS.

The cost of the laser printed manual for installation is \$25.00 not \$15.00 as published.

Thank you for publishing the article. I hope it meets with a lot enthusiasm from HUG members. I am working on more articles and will send them as they are ready.

Sincerely,
David M. Caranci
Data Management Associates
P.O. Box 220
Litchfield, CT 06759

Add Years to MAGBASE

Dear HUG:

I have recently purchased MAGBASE. The disk contains a REMark Index for one year only, 1987. I would like to add years to this Base, but I don't want to do it if someone else is doing it. There is just too much work involved.

Do you have plans to add more years? Or, do you know if anyone else plans to do it?

If not, would you please ask the members via Buggin' HUG as to whether anyone is or has. I would like to get started, but I am very hesitant to get involved in duplicating someone else's

work (assuming they are willing to share it).

Thank you,
Robert F. Hassard
3466 Tice Creek Drive, #4
Walnut Creek, CA 94595

Ed: HUG has no plans for future updates. How about it Huggies? . . . Here's your chance to make some extra coin!

MPI Printer Owners

Dear HUG:

This is my chance to give back to HUG members.

If you have an MPI printer, parts are available from:

Cyber-Force
3508 East T.C. Jester
Houston, TX 77018
(713) 682-0668

Thanks goes to a HUG member — Mr. Al Lingo — from Arkansas for answering my classified ad in the month of January.

Thank you,
Bert Desmarais
6532 N. Fremont Road
East Syracuse, NY 13057

Ultra-RTTY Software #885-6012

Dear HUG:

I purchased a copy of Ultra-RTTY last year for use with my Z-158 computer (MS-DOS version 3.21). It has a glitch in that if I use an RTTY.PRM file to set colors, etc., it will not look to any drive for the message files (including CQ.DAT). If I do not use an RTYY.PRM file, it does look for the message files! (But I'm stuck with mono video.)

Have there been any published solutions to my problem? Any other people with a similar situation?

Thanks,
John W. Lockhart
1035 W. California Avenue
St. Paul, MN 55117

X-10 Powerhouse CP290

Dear HUG:

I am responding to Frederick Jenke's letter published in the February issue to say that I have written software for the X10 Powerhouse CP290. My package, called XControl, allows scheduling of controlled devices, such as lamps, at either fixed times or at times depending on the local sunrise or sunset. For example, a front porch light controlled by an X10 module can be programmed to come on

30 minutes before sunset and go out at 11:42 PM. The current version has been tested on a Z-151, an IBM XT, an IBM AT, and a 386 clone (25 MHz).

XControl uses one data file to schedule events, up to the 128 maximum that the X10 Powerhouse controller can handle, and a second data file for the local sunrise and sunset times. The sunrise/sunset file is created by XControl's install program for any latitude between 20 and 60 degrees North. The schedule data file is created and modified by a screen editor, XCEdit which is supplied with the package.

XControl is not a memory resident program, so there is no danger of conflict with other TSRs. You must run the program periodically to update the event schedule if you use the sunrise/sunset feature. However, the command to do this is simply "XCONTROL L", making it easy to run from a batch file, a mouse menu, or other utility.

Anyone wishing a copy of this program may send \$15.00 to me for a 5.25 inch disk, or \$16.00 for a 3.5 inch disk.

Sincerely,
William L. Sexauer
15548 S. E. 175th Court
Renton, WA 98058

Added Note

Dear HUG:

This is just to note an addition to Figure 1 on page 10 in Bill Adney's January 1990 Powering Up article. The key combination Ctrl-Backspace on a PC generates a DEL character (ASCII character 127). This little-known fact can be useful when using your computer to talk to mainframe computers which expect this character to be used for certain operations.

Bill Hall
3665 Benton Street, #66
Santa Clara, CA 95051

Interest to '8/89/90 Users

Dear HUG:

Of possible interest to anyone using an H-8, H-Z-89 or Z-90 with a Packet TNC for amateur radio packet operations:

I've modified the modem controller 'PLINK' for compatibility with packet operations, to facilitate protocol modification and line printer interfacing. This was done as an act of desperation when I couldn't find any one of the available modem drivers under HDOS or CP/M that did everything I wanted of it. This latest version, which I am calling "PACKET89" to identify it, offers function key toggling of the line printer, storing of incoming files

Continued on Page 37

dBASE III Part 5

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

In this, the fifth of the dBASE III series, the PROJECTS and PROJXREF update program and the project validation program will be combined into a single integrated package. The PROJUPD program of Part 4 made use of format files for the editing of the PROJECTS and PROJXREF data bases. Although format files are a great improvement over the editing format generated by dBASE III, they have serious limitations:

1. They provide no data validation beyond what can be defined by picture templates and RANGE values.
2. They do not permit editing of two or more data bases on the same screen.
3. They provide no search capabilities.
4. Conditional processing is not possible.

Consider item 2. Since the PROJECTS and PROJXREF data bases are linked by project number, it would be convenient to be able to edit both data bases using a single data input/inquiry screen. Also, validating each project record at the time of entry would allow immediate correction of invalid fields, thereby eliminating the need to run a batch validation process at regular intervals. An integrated program can also prevent duplicate entries of the same drawing number and revision letter, a definite requirement for this particular system.

To summarize, the new PROJECTS update program should meet the following specifications:

1. Editing and display of both PROJECTS and PROJXREF data bases using a single integrated screen.
2. Validation of each record at time of entry.
3. Search capability by drawing number and revision letter.
4. Force user to create at least one device type record for every new project record.
5. All functions should be menu driven.
6. The system should generate prompts and appropriate error messages.

In addition to the above, the program must prevent the status of a project going from ACTIVE to COMPLETED unless an approval date is entered into the APP-DATE field.

The main menu of the program should provide the following functions:

1. Search for project by drawing number and revision letter.

2. Advance to the next record.
3. Return to previous record.
4. Update either the project record or device type record.
5. Exit to the dot prompt.

Item 4 must give the user the choice of updating the project record or device type record. Finally, a submenu for each update process must provide the following capabilities:

1. Edit current record.
2. Delete current record.
3. Recover current record.
4. Add a new record.
5. Exit back to the main menu.

With a program of this complexity, a flow diagram of some sort is almost a necessity. This can take the form of a full-fledged flow chart or an outline-type listing using pseudocode. Whatever method is used, designing the program in modular fashion is highly preferable to attempting to write the entire program line-by-line at the moment of conception. Generally speaking, most of the work of designing a computerized information system should be done before a single line of code is written. Once the structure of a well-designed program has been established, the actual code should require very little additional effort.

As an example of a program flow, the main module of the new projects update program could be outlined similar to the following:

```
Set environment
Open files
Paint screen
Do the following until user chooses to exit:
  Display main menu line
  Ask user for his selection
  Process selection:
    if selection is "X"
      Set default environment
      Close files
      Clear screen
      Return to calling program (or dot prompt)
    if selection is "S":
      search for specified drawing
      if drawing not in data base:
        Display error message
        Return to top of loop
      Display project data for specified drawing
      Display associated device type data
    if selection is "N":
      if not at end of file:
        Go to next record
        Display project data
        Display associated device type data
```

```
      if at end of file:
        Display error message
    if selection is "P"
      if not at beginning of file:
        Go to previous record
        Display project data
        Display associated device
          type data
      if at beginning of file
        Display error message
    if selection is "U"
      Ask user for update option
      if option = "P"
        Do project update
      if option = "C"
        Do crossref update
      otherwise
        Sound bell
    otherwise
      Sound bell
Go back and ask for another selection
```

Within this main program outline are two subprograms — the project update module and the cross reference update module. These two modules should also be outlined before writing any program code. The project update module should provide the following options:

1. Edit the current record
2. Delete the current record
3. Recover (undelete) the current record
4. Add a new record
5. Change the status of the current record
6. Exit to the main menu

Item 5 — change status — is not included as part of item 1 because of cer-

First Class H/Z Enhancements!

No Slot Clock/Calendar

FBE SmartWatch: Automatic date/time on bootup. Installs under BIOS/Monitor ROM. Ten year battery. Works with all Heath/Zenith MSDOS computers. For PC's \$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 for '157, '158, '159)

VCF-150: Eliminate video card. Install EGA or VGA card. All plug in. Includes circuit board, SRAM and RM-150. \$54.95

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 to COM3. Put internal MODEM at COM2. Don't lose serial port. With software. \$29.95

H/Z-100 Modifications

ZMF100A: Expand "old" motherboard (p/n 181-4917 or less) using 256K RAM chips (not included). No soldering. \$65.00

ZRAM-205: Put 256K RAM chips on your Z-205 board. Get 256K plus 768K RAM disk. Contact us for data sheet before ordering. Without RAM chips. \$39.00

Z-171 Memory Expansion

MegaRAM-171: Put 256K RAM chips (not included) on existing memory card. Get 640K plus 384K RAM disk. \$49.95

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

FBE

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

Reader Service #104

tain system requirements. First, the status of an active project must not be allowed to go to "completed" until an approval date is entered into the record. Secondly, once a project is completed, it is never to be re-opened. Therefore, the status of a completed project can never again be "ACTIVE". However, no restriction is imposed on changing an "ACTIVE" project to a "DISCONTINUED" project or visa versa. This is the main reason for item 5.

Based on these requirements, the outline for the project update module follows:

7. Exit to main menu

Options 1 and 2 apply only to project records that are associated with more than one device type record. In this context, "next" or "previous" means only device type records that belong to the project record. I will not outline this module, since it is very similar to the project module. Listing 1 is the main update program. This program calls five subprograms:

1. SCREEN.PRG (Listing 2) — Program to paint update screen
2. GET_PD.PRG (Listing 3) — Program to get project data

```
Do the following until user exits:
Display project update menu
Get user input
Process selection:
  if selection = "X"
    exit
  if selection = "C"
    Edit project data
    if approval date is not blank
      Change project status to COMPLETED
    Validate data
  if selection = "D"
    Ask user for confirmation to delete
    if confirmed
      Delete
      Display message
  if selection = "R"
    Recall project
    Display message
  if selection = "S"
    if STATUS = "C"
      Display message "Status of a completed project
        cannot be changed"
    if STATUS = "A"
      Change STATUS to "D"
      Display message
    if STATUS = "D"
      Change STATUS to "A"
      Display message
  if selection = "A"
    Get drawing number and revision letter from user
    If project already exists
      Display message "Project already exists"
    otherwise
      Add blank record
      Input data
      Validate date
      Save project number
    Do the following until user exits
      Append blank record to PROJXREF
      Replace project number
      Input data
      Ask user if another record is to be added
      if yes
        Return to start
      else
        exit
```

The sequence for selection equals "A" insures that at least one device type record will exist for every project record.

Finally, we consider the outline for the cross reference module. This module must provide the following options:

1. Advance to next record
2. Go to previous record
3. Edit current record
4. Delete current record
5. Recall current record
6. Add new record

3. DISP_ST.PRG (Listing 4) — Program to display project status

4. DISP_DEL.PRG (Listing 5) — Program to display a project record's "deleted" status

5. GET_DTD.PRG (Listing 6) — Program to get the device type data. Because of the length of these programs, particularly Listing 1, the analysis of the programs will have to be deferred until Part 6.

Listing 1

```

1 * PROJUPD2.PRG
2 * PROGRAM TO INQUIRE, EDIT, ADD, AND DELETE PROJECTS AND PROJXREF
3 * WRITTEN BY: D.COOL 01/13/90
4 * REVISED BY: D.COOL 02/05/90

```

```

5 * SET ENVIRONMENT:
6 set talk off
7 set bell off
8 set deleted off

```

```

9 * OPEN DATABASE FILES:
10 select 2
11 use PROJXREF index PROJXPJN
12 select 1
13 use PROJECTS index PROJDWV

```

```

14 store "Press any key to continue" to PRESS_MSG
15 store " " to PRUP_OPT
16 store " " to CRUP_OPT
17 ROW = 1

```

```

18 do SCREEN

```

```

19 skip -1

```

```

20 do while .T.    && MAIN PROGRAM LOOP

```

```

21 select PROJECTS
22 set intensity off
23 store " " to MAIN_OPT
24 @ 22, 0 clear
25 @ 22,35 say "MAIN MENU"
26 @ 23,19 say "(S)earch (N)ext (P)rev (U)pdate e(X)it";
27 get MAIN_OPT picture "!"
28 read
29 set intensity on

```

```

30 do case    && MAIN_OPT

```

```

31 case MAIN_OPT = "X"
32 set talk on
33 set bell on
34 set intensity on
35 close databases
36 clear
37 return

```

```

38 case MAIN_OPT = "S"
39 store space(10) to MDWGNR
40 store " " to MREV
41 @ 23, 0
42 @ 23,10 say "Enter drawing number and revision letter: ";
43 get MDWGNR picture "@! A"
44 read
45 if MDWGNR = " "
46 loop
47 endif
48 store recno() to RECNO

```

```

49 seek MDWGNR + MREV
50 if eof()
51 goto RECNO
52 ? chr(7)
53 @ 23, 0
54 @ 23,16 say "Project not found -- " + PRESS_MSG
55 wait ""
56 loop
57 endif
58 do GET_PD
59 clear gets
60 do DISP_ST
61 do DISP_DEL
62 store PROJNR to MPROJNR
63 select PROJXREF
64 seek MPROJNR
65 if .not. eof()
66 do GET_DTD
67 clear gets
68 else
69 ? chr(7)
70 @ 23, 0
71 @ 23,10 say "Device type data does not exist for this " +
"project -- " + PRESS_MSG
72 wait ""
73 endif

```

```

74 case MAIN_OPT = "N"
75 if .not. eof()
76 skip
77 do GET_PD
78 clear gets
79 do DISP_ST
80 do DISP_DEL
81 store PROJNR to MPROJNR
82 select PROJXREF
83 seek MPROJNR
84 do GET_DTD
85 clear gets
86 endif
87 if eof()
88 ? chr(7)
89 @ 23, 0
90 @ 23,13 say "This is the last project -- " + PRESS_MSG
91 wait ""
92 endif

```

```

93 case MAIN_OPT = "P"
94 if .not. bof()
95 skip -1
96 do GET_PD
97 clear gets
98 do DISP_ST
99 do DISP_DEL
100 store PROJNR to MPROJNR
101 select PROJXREF
102 seek MPROJNR
103 do GET_DTD
104 clear gets
105 endif

```

```

106 if bof()
107   ? chr(7)
108   @ 23, 0
109   @ 23,13 say "This is the first project -- " + PRESS_MSG
110   wait ""
111   endif
112 case MAIN_OPT = "U"   && UPDATE PROMPT
113   if eof().or. bof()
114     ? chr(7)
115     @ 23, 0
116     @ 23,15 say "You must search first -- " + PRESS_MSG
117     wait ""
118   else
119     set intensity off
120     store " " to UPDT_OPT
121     ? chr(7)
122     @ 23, 0
123     @ 23,25 say "Projects or Crossref? (P/C)";
124     get UPDT_OPT picture "i";
125     set intensity on
126   do case   && UPDT_OPT
127     case UPDT_OPT = "P"
128       do while .T.   && PROJECT UPDATE
129         @ 22, 0 clear
130         @ 22,33 say "PROJECT UPDATE"
131         @ 23,15 say "(C)hange (D)e1 (R)ecover (A)dd "+;
132           "(S)tatus e(X)it"
133         set intensity off
134         store " " to PRUP_OPT
135         @ 23,66 get PRUP_OPT picture "i";
136         read
137         set intensity on
138       do case   && PRUP_OPT
139         case PRUP_OPT = "X"
140           exit
141         * case PRUP_OPT = "C"
142           do GET_PD
143           read
144           if dtoc(APPDATE) <> " "
145             replace STATUS with "C"
146           endif
147           do VALIDATE
148           do DISP_ST
149         case PRUP_OPT = "D"
150           store " " to CONFIRM
151           do while .not. CONFIRM $ "YN"
152             ? chr(7)
153             @ 23, 0
154             @ 23,14 say "Are you sure you want to " + ;
155               "delete this project? (Y/N)";
156             get CONFIRM picture "i"
157             read
158           enddo
159           if CONFIRM = "Y"
160             delete
161             PRESS_MSG
162             wait ""
163             endif
164             case PRUP_OPT = "R"
165               if PRUP_OPT = "R"
166                 recall
167                 do DISP_DEL
168                 @ 23, 0
169                 @ 23,15 say "Project has been recovered -- " + ;
170                   PRESS_MSG
171                 wait ""
172                 endif
173               case PRUP_OPT = "S"
174                 do case && PRUP_OPT = "S"
175                   case STATUS = "C"
176                     ? chr(7)
177                     @ 23, 0
178                     @ 23, 2 say "Status of a completed " + ;
179                       "PROJECT cannot be changed -- " + ;
180                       PRESS_MSG
181                     wait ""
182                   case STATUS $ "AD"
183                     if STATUS = "A"
184                       store "D" to NEW_STAT
185                       store "DISCONTINUED" to NS_WORD
186                     else
187                       store "A" to NEW_STAT
188                       store "ACTIVE" to NS_WORD
189                     endif
190                     replace STATUS with NEW_STAT
191                     do DISP_ST
192                     ? chr(7)
193                     @ 23, 0
194                     @ 23, 6 say "Status has been changed to " + ;
195                       NS_WORD + " -- " + PRESS_MSG
196                     wait ""
197                   endcase
198                   PRUP_OPT = "S"
199                   case PRUP_OPT = "A"
200                     store space(10) to MDWGNR
201                     store " " to MREV
202                     @ 23, 0
203                     @ 23,10 say "Enter drawing number and " + ;
204                       "revision letter: ";
205                     get MDWGNR picture "9999-99999"
206                     @ 23,68 get MREV picture "e! A"
207                     read
208                     if MDWGNR = " "
209                       loop
210

```

```

202 endif
203 seek MDWGNR + MREV
204 if .not. eof()
205   ? chr(7)
206   @ 23, 0
207   @ 23.17 say "This project already exists -- "+;
      PRESS_MSG
208   wait ""
209   else
210     append blank
211     replace DWGNR with MDWGNR, REV with MREV;
      STATUS with "A"
212   * Blank out any previous device type data:
213   @ ROW+6,53 say space(2)
214   @ ROW+7, 53 say space(15)
215   do GET_PD
216   read
217   do VALIDATE
218   do DISP_ST
219   do DISP_DEL
220   store PROJNR to MPROJNR
221   select PROJXREF
222   do while .T.
223     append blank
224     replace PROJNR with MPROJNR
225     do GET_DTD
226     read
227     set intensity off
228     store " " to ANOTHER
229     ? chr(7)
230     @ 23, 0
231     @ 23.31 say "Add another? (Y/N) ";
      get ANOTHER picture "!"
232     read
233     set intensity on
234     if ANOTHER = "Y"
235       @ 23, 0
236       loop
237     else
238       @ 23, 0
239       exit
240     endif
241     enddo
242   endif
243   otherwise
244     ? chr(7)
245     PRUP_OPT
246   endcase
      PROJECT UPDATE
247 case UPDT_OPT = "C"   && CROSSREF UPDATE
      do while .T.   && CROSSREF UPDATE LOOP
248   select PROJXREF
249   @ 22, 0 clear
250   @ 22.32 say "CROSSREF UPDATE"
251   @ 23.11 say "(N)ext (P)rev (C)hange (D)el " +;
      "(R)ecover (A)dd e(X)it"
252   set intensity off
253   store " " to CRUP_OPT
254   @ 23.70 get CRUP_OPT picture "!"
255
256 read
257 set intensity on
258 do case   && CRUP_OPT
259 case CRUP_OPT = "N"
260   if .not. eof()
261     skip
262   endif
263   if PROJNR <> MPROJNR
264     skip -1
265     ? chr(7)
266     @ 23, 0
267     @ 23, 3 say "No further dev. type exists "+;
      "for this project -- " + PRESS_MSG
      PRESS_MSG
268     wait ""
269   else
270     do GET_DTD
271     clear gets
272   endif
273 case CRUP_OPT = "P"
274   if .not. bof()
275     skip -1
276   endif
277   if PROJNR <> MPROJNR
278     skip
279     ? chr(7)
280     @ 23, 0
281     @ 23, 3 say "This is the first dev. "+;
      "type for this project -- " +;
      PRESS_MSG
282     wait ""
283   else
284     do GET_DTD
285     clear gets
286   endif
287 case CRUP_OPT = "C"
288   do GET_DTD
289   read
290 case CRUP_OPT = "D"
291   store " " to CONFIRM
292   do while .not. CONFIRM $ "YN"
293     ? chr(7)
294     @ 23, 0
295     @ 23.19 say "Are you sure you want to "+;
      "delete this device? (Y/N) ";
      get CONFIRM picture "!"
296   read
297   enddo
298   if CONFIRM = "Y"
299     delete
300     @ ROW+8,48 say "** DELETED *"
301     ? chr(7)
302     @ 23, 0
303     @ 23.19 say "Record has been deleted -- " +;
      PRESS_MSG

```

```

304     wait ""
305     endif
306     case CRUP_OPT = "R"
307         recall
308         @ ROW+8,48 say space(11)
309         ? chr(7)
310         @ 23, 0
311         @ 23,13 say "Record has been recovered -- " + ;
312         PRESS_MSG
313         wait ""
314         case CRUP_OPT = "A"
315             append blank
316             replace PROJNR with MPROJNR
317             do GET_DTD
318             read
319         case CRUP_OPT = "X"
320             exit
321         otherwise
322             ? chr(7)
323         endcase
324     CRUP_OPT
325     enddo CROSSREF UPDATE LOOP
326     otherwise
327         ? chr(7)
328     endcase
329     UPDATE SELECTION
330     endif eof() or bof()
331     otherwise
332         ? chr(7)
333     endcase
334     MAIN_OPT
335     enddo
336     MAIN PROGRAM LOOP

```

Listing 2

```

* SCREEN.PRG
* PROGRAM TO DRAW UPDATE SCREEN FOR PROJECTS/CROSSREF
* WRITTEN BY: D.COOL 01/16/90

```

```

clear
text
place figure here
endtext
return

```

Listing 3

```

* GET_PD
* PROGRAM TO GET PROJECT DATA
* (CALLED BY PROJUPD2.PRG)

```

```

* WRITTEN BY: D.COOL 01/13/90
* REVISED BY: D.COOL 02/05/90
if MAIN_OPT $ "SNP"
@ ROW+ 3,20 get DWGNR picture "9999-99999"
@ ROW+ 3,38 get REV picture "@! A"
@ ROW+ 3,54 get PROJNR picture "@! 9999-A9999"
endif
if PRUP_OPT = "A"
@ ROW+ 3,20 say DWGNR
@ ROW+ 3,38 say REV
@ ROW+ 3,54 get PROJNR picture "@! 9999-A9999"
endif
if PRUP_OPT = "C"
@ ROW+ 3,20 say DWGNR
@ ROW+ 3,38 say REV
@ ROW+ 3,54 say PROJNR
endif
@ ROW+ 3,74 get ENGINEER picture "@! AAA"
@ ROW+ 5,23 get STARTDATE
@ ROW+ 6,23 get ESTDATE
@ ROW+ 7,23 get EDITIN
@ ROW+ 8,23 get EDITOUT
@ ROW+ 9,23 get APPDATE
@ ROW+10,23 get DOCDATE
@ ROW+12,14 get COMMENTS picture "@!"
return

```

Listing 4

```

* DISP_ST.PRG
* PROGRAM TO DISPLAY PROJECT STATUS
* (CALLED BY PROJUPD2.PRG)
* WRITTEN BY: D.COOL 01/13/90

```

```

do case
case STATUS = "A"
@ ROW+13,12 say "ACTIVE"
case STATUS = "C"
@ ROW+13,12 say "COMPLETED"
case STATUS = "D"
@ ROW+13,12 say "DISCONTINUED"
otherwise
@ ROW+13,12 say "UNKNOWN"
endcase
return

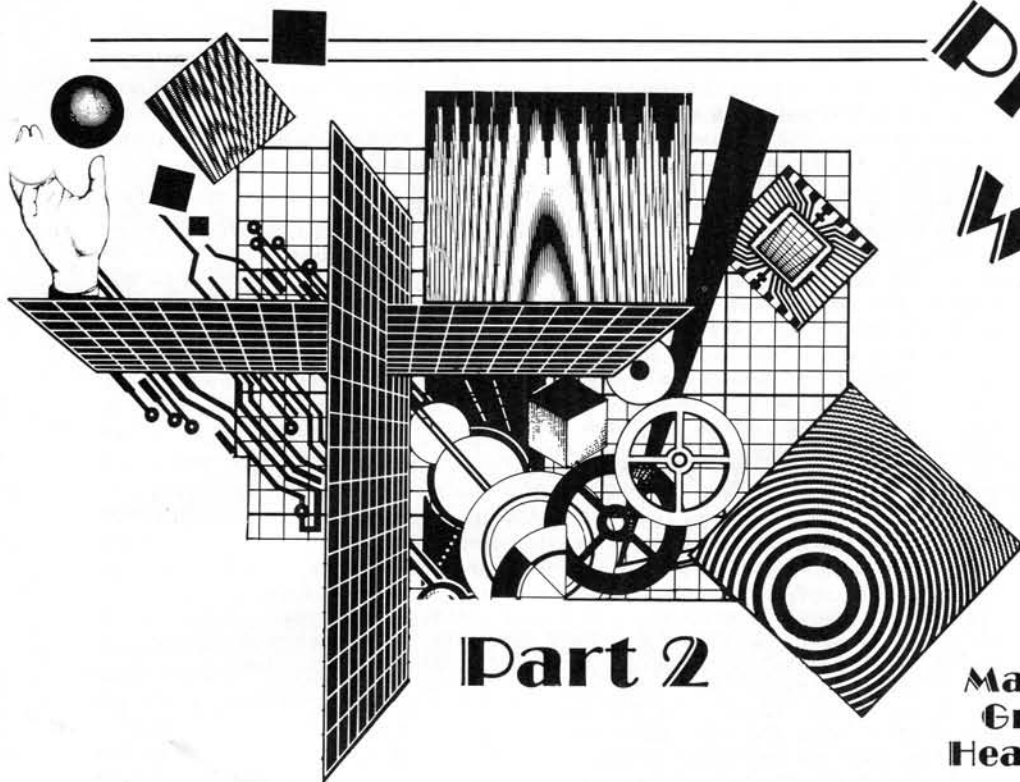
```

Listing 5

```

* DISP_DEL.PRG
* PROGRAM TO DISPLAY DELETED STATUS OF PROJECT RECORD
* [CALLED BY PROJUPD2.PRG]
* WRITTEN BY: D.COOL 02/05/90
if deleted()
@ ROW+13,40 say "** DELETED *"
else
@ ROW+13,40 say space(11)
endif
return

```



Programming With VGA

Part 2

Mark Mangerson
Greg McDonald
Heath Technicians

Mark Mangerson Heath Technician

In last month's article, we covered the basics of EGA and VGA programming. This month, we will show you some programming techniques by way of example. Listings 2 through 4 all display a VGA picture, but are customized to work well with different video displays. Listing 1 converts the standard .vga file format into a bsave format used by BASIC. When in screen mode 13 (used in Listing 2) the bload command will load a picture in the blink of an eye. Due to overhead from dithering, the other video modes are not nearly as fast as mode 13, but they will give you the best results obtainable with the hardware they were written for and livable performance on most machines. The program in Listing 2 will display a picture on a vga screen at 320×200 resolution in 256 colors. It also gives you the ability to do some limited image enhancements and save the resulting picture back to disk. Listing 3 will display the same 256 color picture on a 449 video card, but in black and white only. The 449 can only do 16 colors at one time, but if these 16 colors are programmed to 16 shades of gray and some dithering is used, you can get 31 effective gray levels at an effective resolution of 320×200 . Listing 4 is a program to display the picture on a Zenith Data Systems laptop equipped with VGA. This program displays the picture in the $640 \times 480 \times 16$ color mode on the LCD screen and used dithering to give 61 effective levels of gray. This program might run on the 449 card if a newer ver-

Listing 1

```

10 ON ERROR GOTO 370 : REM set error trap
20 N$=COMMAND$ : REM recieve file name from command line
30 OPEN "I",#1,N$+".VGA" : CLOSE #1 : REM check to see if file exists
40 DEFINT A-Z : REM define all variables as integers
50 SCREEN 13 : REM set screen to 320 x 200 x 256 color mode
60 DIM R(256),G(256),B(256),I(256)
70 REM define storage areas for palette information
80 OPEN "R",#1,N$+".VGA",1 : FIELD #1,1 AS A$
90 REM open file for random access
100 OUT &H3C8,0 : REM set palette write address to 0
110 FOR P=0 TO 255 : REM start loop to program palette
120   GET #1 : R(P)=ASC(A$) : OUT &H3C9,R(P)
130   REM get red value from disk and write it to palette table
140   GET #1 : G(P)=ASC(A$) : OUT &H3C9,G(P)
150   REM get green value from disk and write it to palette table
160   GET #1 : B(P)=ASC(A$) : OUT &H3C9,B(P)
170   REM get blue value from disk and write it to palette table
180   I(P)=(R(P)*.3+G(P)*.59+B(P)*.11)
190   REM calculate the intensity value of the palette location
200 NEXT P : REM end of palette programming loop
210 FOR Y=0 TO 199 : REM start of y axis scan loop
220   FOR X=0 TO 319 : REM start of x axis scan loop
230     GET #1 : C=ASC(A$) : REM get pixel value from disk
240     PSET (X,Y),C : REM write pixel value to screen
250   NEXT X : REM end of x axis loop
260 NEXT Y : K=1 : REM end of y axis loop
270 DEF SEG=&HA000 : REM set address pointer to video memory segment
280 FOR P=0 TO 255 : REM start of palette read loop
290   POKE 64000+P*3,R(P)
300   REM get red data from palette and place in memory
310   POKE 64000+P*3+1,G(P)
320   REM get green data from palette and place in memory
330   POKE 64000+P*3+2,B(P)
340   REM get blue data from palette and place in memory
350 NEXT P : REM end of palette read loop
360 BSAVE N$+".ART",0,64768 : REM save picture in basic bsave format
370 CLOSE #1:END : REM end of program

```

sion of the compiler is used. I was using version 4.0 of Quick BASIC to write and test these programs and mode 12 will not work on the 449 with this version of the compiler.

The picture files discussed in this arti-

cle are stored in a format of our own design. The first 768 bytes of the file contain the palette data and the next 64000 bytes contain the picture data. Since the data is not in a compressed form, it takes up much more room than pictures stored in

TAKING COMMAND
OF
ENABLE



Includes 2.0 and O.A.

#1
Seller

**Taking Command of
Enable.....\$28.95 ppd.**
(Mention this ad and receive \$4.00 off.)

To order call 1-800-752-6083.
MC, VISA, AMEX

Send for a **Free Catalog**
from the Nation's Largest
Distributor of Enable
Products.

Key Computer Publications
3221-A Ruckriegel Pkwy.
Louisville, KY 40299

Reader Service #175

SAVE!
New Lower Prices
**A FRESH
NEW LIFE FOR
YOUR Z-171**

THE Z-171 HARD DISK!

LapDrive

| | |
|---------------------|-------|
| 21M 80ms Economy | \$585 |
| 21M 68ms Miniscribe | \$675 |
| 33M 68ms Miniscribe | \$725 |

**SERENDIPITY
ASSOCIATES**

5036 W Bantff Lane • Glendale, AZ 85306

(602) 938-3729

Reader Service #188

Listing 2

```

10 ON ERROR GOTO 500 : REM set error trap
20 N$=COMMAND$ : REM recieve file name from command line
30 OPEN "I",#1,N$+".ART" : CLOSE #1 : REM check to see if file exists
40 DEFINT A-Z : REM define all variables as integers
50 SCREEN 13 : REM set screen to 320 x 200 x 256 color mode
60 DIM R(256),G(256),B(256),I(256) : REM define areas to store palette info
70 OUT &H3C6,0 : REM turn screen off
80 DEF SEG=&HA000 : BLOAD N$+".ART",0
90 OUT &H3C8,0 : REM set palette write address to 0
100 FOR P=0 TO 255 : REM start loop to program palette
110 R(P)=PEEK(64000!+P*3) : OUT &H3C9,R(P)
120 REM get red value from disk and write it to palette table
130 G(P)=PEEK(64000!+P*3+1) : OUT &H3C9,G(P)
140 REM get green value from disk and write it to palette table
150 B(P)=PEEK(64000!+P*3+2) : OUT &H3C9,B(P)
160 REM get blue value from disk and write it to palette table
170 I(P)=(R(P)*.3+G(P)*.59+B(P)*.11)
180 REM calculate the intensity value of the palette location
190 NEXT P : REM end of palette programming loop
200 OUT &H3C6,255 : K=1 : REM turn screen on and set color on
210 K$=INKEY$ : REM trap keyboard input
220 IF K$="C" OR K$="c" THEN K=1 : GOSUB 290 : REM turn color on
230 IF K$="M" OR K$="m" THEN K=0 : GOSUB 290 : REM turn monochrome on
240 IF K$="R" OR K$="r" THEN IC=0 : GOSUB 290 : REM reset brightness
250 IF K$="S" OR K$="s" THEN 430 : REM save picture
260 IF K$="+" THEN IC=IC+1 : GOSUB 290 : REM brighten picture
270 IF K$="-" THEN IC=IC-1 : GOSUB 290 : REM darken picture
280 IF K$=CHR$(13) THEN END ELSE 210 : REM exit to system
290 OUT &H3C8,0 : REM set palette write address to 0
300 FOR P=0 TO 255 : REM start palette program loop
310 IF K=1 THEN RR=R(P)+IC : GG=G(P)+IC : BB=B(P)+IC
320 IF K=0 THEN RR=I(P)+IC : GG=I(P)+IC : BB=I(P)+IC
330 IF RR>63 THEN RR=63 : REM don t go above intensity 63
340 IF RR<0 THEN RR=0 : REM don t go below intensity 0
350 OUT &H3C9,RR : REM output red intensity to palette
360 IF GG>63 THEN GG=63
370 IF GG<0 THEN GG=0
380 OUT &H3C9,GG : REM output green intensity to palette
390 IF BB>63 THEN BB=63
400 IF BB<0 THEN BB=0
410 OUT &H3C9,BB : REM output blue intensity to palette
420 NEXT P : RETURN : REM end of palette programming loop
430 OUT &H3C7,0 : REM set palette read address to 0
440 FOR P=0 TO 255 : REM start palette read loop
450 POKE 64000!+P*3,INP(&H3C9) : REM write red palette value to memory
460 POKE 64000!+P*3+1,INP(&H3C9) : REM write green palette value to memory
470 POKE 64000!+P*3+2,INP(&H3C9) : REM write blue palette value to memory
480 NEXT P : REM end of palette read loop
490 BSAVE N$+".NEW",0,64768! : REM save altered picture to disk
500 END : REM end program
    
```

other formats, such as .gif, but it is much easier to understand and the speed of loading makes programs that display other picture formats to look like they're standing still.

This program enables the user to do the following:

1. Convert color pictures to 64 gray level pictures.
2. Increase and decrease the intensity of the color or gray scale image.
3. Save the image back to disk under the same name, but with a .new extention.

The user can toggle between color and gray images by typing "c" for color and "m" for mono. The picture can be lightened or darkened when in either color or black and white by typing the "+" and "-" keys. If you want to reset the picture to its original brightness you type "r".

The picture can be saved by typing the "s" key. After the picture is saved, the program will end. If you want to exit the program without saving the picture type "enter". Some of you may want to know why you would want to convert a color picture into a black and white picture. There are two reasons for this.

1. Most desktop publishing programs use black and white images and this program will allow you to view and adjust images to be imported into your desktop publishing applications.
2. You can combine pictures or parts of pictures together more easily and with better results when both pictures have the same palette. These pictures will also take up less room when arced, zipped or gifed than their color counterparts.

Listing 3

```
10 ON ERROR GOTO 300 : rem set error trapping on
20 N$=COMMAND$: rem get file name from command line
30 OPEN "I",#1,N$+".VGA" : CLOSE #1 : rem check to see if file exists
40 DEFINT A-Z : rem define all variables as integers
50 SCREEN 8 : rem set screen to 640 x 200 x 16 color mode
60 DIM I(256),C1(31),C2(31)
65 rem define area to store intensities of all colors and dither patterns
70 FOR L=0 TO 31 : rem start loop to read in dither array
80 READ C1(L),C2(L) : rem read pixel data for dither patterns
90 NEXT L : rem end of dither array loop
100 OPEN "R",#1,N$+".VGA".1 : FIELD #1,1 AS A$
110 rem open file for random access
120 OUT &H3C8,0 : rem set palette write address to 0
130 FOR P=0 TO 255 : rem start loop to read palette info from disk
140 GET #1 : R=ASC(A$) : rem read red value from disk
150 GET #1 : G=ASC(A$) : rem read green value from disk
160 GET #1 : B=ASC(A$) : rem read blue value from disk
170 I(P)=(R*.3+G*.59+B*.11)/2
175 rem calculate intensity value for color
180 NEXT P : rem end of palette read loop
190 FOR C=0 TO 15 : rem start loop to program palette registers
200 IF C=8 THEN OUT &H3C9,16 : rem skip 8 palette locations
210 OUT &H3C9,C*4 : OUT &H3C9,C*4
211 rem program red, blue and green registers to intensity value
220 NEXT C : rem end of palette programming loop
230 FOR Y=0 TO 199 : rem start of y axis loop
240 FOR X=0 TO 638 STEP 2 : rem start of x axis loop
250 GET #1 : C=I(ASC(A$)) : rem get pixel data from disk
260 PSET (X,Y),C1(C) : PSET (X+1,Y),C2(C)
265 rem write dither pattern to screen
270 NEXT X : rem end of x axis loop
280 NEXT Y : rem end of y axis loop
290 K$=INKEY$: IF K$<>"X" AND K$<>"x" THEN 290 : rem scan keys to end program
300 CLOSE #1 : END : rem end of program
310 DATA 0,0,1,1,1,1,2,2,2,3,3,3,4 : rem data for dither patterns
320 DATA 4,4,4,5,5,5,6,6,6,7,7,7,8
330 DATA 8,8,9,9,9,10,10,10,10,11,11,11,11,11
340 DATA 12,12,12,13,13,13,14,14,14,15,15,15,15,15
```

Listing 4

```
10 DEFINT A-Z : rem define all variables as integers
20 SCREEN 12 : rem set screen to mode 12
30 N$=COMMAND$: rem get file name from command line
40 CLS : rem clear screen
50 OPEN "R",#1,N$+".VGA".1 : rem open file for random access
60 FIELD #1,1 AS A$ : rem define string as one byte
70 OUT &H3C8,0 : rem set palette write address to 0
80 FOR L=0 TO 15 : rem start of loop to program palette
90 OUT &H3C9,L*4 : rem output red value to palette
100 OUT &H3C9,L*4 : rem output green value to palette
110 OUT &H3C9,L*4 : rem output blue value to palette
120 NEXT L : rem end of palette programming loop
130 DIM R(256),G(256),B(256) : rem define storage area for palette data
140 DIM C1(64),C2(64),C3(64),C4(64)
145 rem define storage area for dither patterns
150 FOR N=0 TO 63 : rem start of loop to read dither data
```

```
160 READ C1(N),C2(N),C3(N),C4(N) : rem read 4 pixel dither data
170 NEXT N : rem end of dither read loop
180 FOR I=0 TO 255 : rem start of palette read loop
190 GET #1 : rem get red palette data
200 R(I)=ASC(A$) : rem convert string to numeric variable
210 GET #1 : rem get green palette data
220 G(I)=ASC(A$) : rem convert string to numeric variable
230 GET #1 : rem get blue palette data
240 B(I)=ASC(A$) : rem convert string to numeric variable
250 NEXT I : rem end of palette read loop
260 FOR Y=40 TO 438 STEP 2 : rem start of y axis loop
270 FOR X=0 TO 638 STEP 2 : rem start of x axis loop
280 GET #1 : rem get pixel data
290 C=ASC(A$) : rem convert string to numeric variable
300 I=INT(G(C)*.59+B(C)*.11+R(C)*.3)
305 rem calculate intensity of pixel
310 PSET(X,Y),C1(I) : rem write first pixel of dither pattern
320 PSET(X+1,Y),C2(I) : rem write second pixel of dither pattern
330 PSET(X,Y+1),C3(I) : rem write third pixel of dither pattern
340 PSET(X+1,Y+1),C4(I) : rem write fourth pixel of dither pattern
350 NEXT X : rem end of x axis loop
360 NEXT Y : rem end of y axis loop
370 CLOSE #1 : rem close file
380 A$=INKEY$: rem scan keyboard
390 IF A$="X" OR A$="x" THEN END ELSE 380 : rem if x is hit end program
395 rem the following data is for the dither patterns
400 DATA 0,0,0,0,0,1,1,0,1,1,0,1,1,1,1,1,1,1,1,1,1,2,2,1,2,1,2,2
410 DATA 2,2,2,2,2,3,3,2,3,3,2,3,3,3,3,3,3,3,3,3,3,4,4,3,4,4,3,4,4
420 DATA 4,4,4,4,4,5,5,4,5,5,4,5,5,5,5,5,5,5,5,5,5,6,6,5,6,5,6,6
430 DATA 6,6,6,6,6,7,7,6,7,7,6,7,7,7,7,7,7,7,7,7,7,8,8,7,8,7,8,8
440 DATA 8,8,8,8,8,9,9,8,9,9,8,9,9,9,9,9,9,9,9,9,9,10,10,9,10,9,10
450 DATA 10,10,9,10,10,10,10,10,10,10,10,10,10,10,11,11,10,10,11,11
460 DATA 10,11,11,11,11,11,11,11,11,11,12,12,11,11,12,12,11,12,12,12,12
470 DATA 12,12,12,12,13,13,12,12,13,13,13,13,13,13,13,13,13,13,13,13,13
480 DATA 14,14,13,13,14,14,13,14,14,14,14,14,14,14,14,14,14,14,14,14,14,15
490 DATA 15,14,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15
```

The dither pattern for Listing 3 uses 2 pixels. If you want to make intensity 15, you place gray level 8 in the first location and level 7 in the second location and the result is the 15th dithered gray level. The dither pattern for Listing 4 uses 4 pixels. These pixels are arranged in a square pattern. The following is an example of the dithering technique.

```
00 10 10 11 11 21 21 22 22
00 00 01 01 11 11 12 12 22 this is the dither pattern for gray levels 0-8
```

Gregory McDonald Heath Technician

This concludes the section on BASIC graphics programming. The remainder of the article will be concerned with EGA and VGA programming in C.

ShowVGA

ShowVGA displays a file with an extension ".VGA", and of the following format:

768 bytes of palette information.
64000 bytes of bitmap data.

The source C code for ShowVGA is shown in Listing 5. First, ShowVGA checks if the correct number of arguments have been entered.

```
if(argc!=2)
{
    puts("\n\r      Syntax [d:][pathname]SHOWVGA
          [pathname]filename[.VGA]");
    exit(-1);
}
```

The first argument is the program name, and the second argument is the VGA display file. Therefore, there must be two arguments not one. If an incorrect number of arguments is entered, the program displays the proper syntax and exits with an error code of -1.

Since the extension ".VGA" is optional, ShowVGA must add it if it is not present.

```
if((sp=strrchr(argv[1],'.')==NULL)
    strcat(argv[1],".VGA");
else
{
    if(strompi(sp,".VGA"))
    {
        puts("\n\r  File must be .VGA");
        exit(-1);
    }
}
```

The file name, argv[1], is searched using strchr for the character '.'. If there is no '.' present in the file name, then sp is assigned the value NULL, and the extension ".VGA" is appended the file name. Just adding the ".VGA" extension works in DOS because, at most, the extension will overwrite part of ShowVGA's copy of the environment, and ShowVGA does not use the environment. If there is a '.', sp is assigned a pointer to it, and strcmpi is used to verify the file has ".VGA" for an extension.

Now ShowVGA can attempt to open the file.

```
if((fh=open(argv[1], O_RDONLY | O_BINARY)) < 0)
{
    perror(NULL);
    exit (-1);
}
```

For maximum speed, ShowVGA uses the low-level file library functions. If the file cannot be opened, the DOS error is displayed and the program exits. Other-

wise, the file handle is assigned to fh.

Using the __setvideomode library function, the program enters the 320 X 200 256 color graphics mode.

```
if(!__setvideomode(_MRES256COLOR))
{
    puts("\n\r      VGA display required");
    exit(-1);
}
```

__setvideomode returns a zero if it cannot enter the requested video mode.

ShowVGA cannot read and close the file.

```
read(fh, screen, (unsigned)65024;
close(fh);
```

If a file error occurs, read returns a -1. However, seeing how the data has been corrupted may be helpful in resolving the problem.

After the palette and bitmap is in memory, the picture can be displayed.

```
outp(0x3c8,0);
for(c=0;c<768;c++);
    outp(0x3c9,*(screen+c));

memcpy((void *)0xa0000000,(screen+768),(size_t)64000);
```

First, a zero is outputted to port 3C8HEX. Causing the palette chip to accept data at its first location. 768 bytes are then outputted to port 3C9HEX. This writes a red, green, or blue value to every location. The 64000 byte bitmap is copied to the screen memory using memcpy.

Now the program waits until the user presses a key.

```
getch();
__setvideomode(_DEFAULTMODE);
puts("\n\r      ShowVGA V1.0 \r");
puts("      By Gregory McDonald");
exit(0);
```

The getch library function blocks the program until a printing key is pressed, and the character is not echoed to the screen. Then __setvideomode resets the video mode to the previous mode. The exit message is displayed and the program terminates.

ShowVGA can be compiled with Microsoft (MS) Quick C or MS C 5.0 or higher. If you have MS C use:

```
cl /AC /Ox showvga.c /link /E
```

If you have Quick C use:

```
gcl /AC /Ox showvga.c /link /E
```

These command lines specify the compact memory model, maximum opti-

mization and the linker packs the exe file. If you have Quick C 2.0, you may select the memory model and compile this program within the Quick C environment.

Improving ShowVGA

Using the in-line assembler available with Quick C 2.0 and MS C 6.0, it is possible to significantly reduce the .exe file size of ShowVGA. The library function __setvideomode is designed to operate with all standard video adaptors, and includes code to check for each type of adaptor. However, ShowVGA will only work with a VGA or MCGA adaptor, making it unnecessary to check for any other type of adaptor.

The modified version of ShowVGA, SHOWVGA1.C, is shown in Listing 6. Listings 5 and 6 are nearly the same except __setvideomode has been replaced with a new function setvgamode. First, setvgamode must determine if a VGA or MCGA adaptor is present.

```
mov ax, 1a00h
int 10h
cmp al, 1ah
jne error
```

```
cmp bl, 07h
jb error
```

INT 10HEX function 1AHEX, video display combination, is only available in PS/2's or PC's with VGA. If successful, the interrupt returns with the value 1AHEX in the AL register. This indicates a VGA or MCGA is installed in the machine. This interrupt also places an active display code in the BL register. Values above 7 specify a VGA or MCGA active display. If either of these conditions are false, the routine jumps to the error label.

Next, setvgamode determines if it should switch back to the previous mode.

```
mov al, mode
cmp al, -01h
je setdefault
```

If the variable mode is equal to -1, the setvgamode jumps to the label setdefault. Also notice, the in-line assembler can recognize C variables.

Using INT 10HEX function 0FHEX, the variable defaultmode is set equal to the current mode.

```
mov ax, 0f00h
int 10h
mov defaultmode, al
```

```
mov al, mode
mov ah, 00h
int 10h
mov ax, 0001h
jmp done
```

The graphics mode is set by invoking INT 10HEX function 0. The value 1 is then

Listing 5

```
# include <conio.h>
# include <fcntl.h>
# include <graph.h>
# include <io.h>
# include <stdio.h>
# include <string.h>
# include <stdlib.h>

main(int argc, char *argv[])
{
    char *sp;          /* string pointer */
    int c;            /* counter */
    int fh;          /* file handle */
    static unsigned char screen[65024]; /* screen file buffer */

    if(argc!=2) /* if there is an incorrect number of arguments */
    {
        puts("\n\r Syntax [d:][pathname]SHOWVGA [pathname]filename[.VGA]");
        exit(-1);
    }

    if((sp=strchr(argv[1],.))==NULL) /* check file name for extension */
        strcat(argv[1],".VGA"); /* if the file has no extension add ".VGA" */
    else
    {
        if(strempi(sp,".VGA")) /* if the file has an extension make shure */
            puts("\n\r File must be .VGA");
    }

    if((fh=open(argv[1], O_RDONLY | O_BINARY) < 0) /* open file to read screen */
        perror(NULL);
        exit (-1);
    }

    if(!_setvideomode(_MRES256COLOR)) /*Switch to 320 X 200 and clear screen */
        puts("\n\r VGA display required");
        exit(-1);

    read(fh, screen, (unsigned)65024); /* read and close file */
    close(fh);

    outp(0x3c8,0); /* setup palette */
    for(c=0;c<768;c++)
        outp(0x3c9,*(screen+c));

    memcpy((void *)0xa0000000,(screen+768),(size_t)64000); /* show screen */

    getch(); /* Wait for keyboard hit and reset screen */
    _setvideomode(_DEFAULTMODE);

    puts("\n\r ShowVGA V1.0 \r\n"); /* display exit message and exit */
    puts(" By Gregory McDonald");
    exit(0);
}
```

Listing 6

```
# include <conio.h>
# include <fcntl.h>
# include <io.h>
# include <stdio.h>
# include <string.h>
# include <stdlib.h>

char setvgamode(char mode);

main(int argc, char *argv[])
{
    char *sp;          /* string pointer */
    int c;            /* counter */
    int fh;          /* file handle */
    static unsigned char screen[65024]; /* screen file buffer */

    if(argc!=2) /* if there is an incorrect number of arguments */
    {
        puts("\n\r Syntax [d:][pathname]SHOWVGA [pathname]filename[.VGA]");
        exit(-1);
    }

    if((sp=strchr(argv[1],.))==NULL) /* check file name for extension */
        strcat(argv[1],".VGA"); /* if the file has no extension add ".VGA" */
    else
    {
        if(strempi(sp,".VGA")) /* if the file has an extension make shure */
            puts("\n\r File must be .VGA");
    }

    if((fh=open(argv[1], O_RDONLY | O_BINARY) < 0) /* open file to read screen */
        perror(NULL);
        exit (-1);
    }

    if(!_setvgamode(0x13)) /* Switch to 320 X 200 and clear screen */
        puts("\n\r VGA display required");
        exit(-1);

    read(fh, screen, (unsigned)65024); /* read and close file */
    close(fh);

    outp(0x3c8,0); /* setup palette */
    for(c=0;c<768;c++)
        outp(0x3c9,*(screen+c));

    memcpy((void *)0xa0000000,(screen+768),(size_t)64000); /* show screen */

    getch(); /* Wait for keyboard hit and reset screen */
    setvgamode(-0x1);

    puts("\n\r ShowVGA V1.0 \r\n"); /* display exit message and exit */
    puts(" By Gregory McDonald");
}
```

```

exit(0);
}

char setvgamode(char mode)
{
static char defaultmode=-1;                /* previous video mode */

_asm \
{
    mov ax, 1a00h                          /* test for VGA or MCGA */
    int 10h
    cmp al, 1ah
    jne error
    cmp bl, 07h
    jb error

    mov al, mode                            /* test for default mode */
    cmp al, -01h
    je setdefault

    mov ax, 0f00h                          /* find default mode */
    int 10h
    mov defaultmode, al

    mov al, mode                            /* set mode */
    mov ah, 00h
    int 10h
    mov ax, 0001h
    jmp done

setdefault:
    mov al, defaultmode                    /* test for a previous mode */
    cmp al, -01h
    je error

    mov ah, 00h                            /* set default mode */
    int 10h
    mov ax, 0001h
    jmp done

error:
    mov ax, 0000h
done:
}
}

```

Listing 7

```

#include <conio.h>                          /* compile with /Gt28001 */
#include <fcntl.h>
#include <io.h>
#include <stdio.h>
#include <string.h>
#include <dos.h>
#include <stdlib.h>

main(int argc, char *argv[])
{
char *sp;                                  /* string pointer */
char c;                                    /* counter */
int fh;                                    /* file pointer */
static unsigned char redplane[28000];     /* red plane */
static unsigned char greenplane[28001];   /* green plane */
static unsigned char blueplane[28016];    /* blue plane */
static unsigned char intensityplane[28000]; /* intensity plane */
union REGS regs;                          /* cpu registers */
char defaultmode;                         /* previous video mode */

if(argc!=2)                               /* if there is an incorrect number of arguments */
{
    puts("\n\r Syntax [d:][pathname]SHOWEGA [pathname]filename[.EGA]");
    exit(-1);
}

if((sp=strrchr(argv[1], .))==NULL)        /* check file name for extension */
    strcat(argv[1], ".EGA"); /* if the file has no extension add ".EGA" */

```

placed in the AX register, this will be returned to the calling function. The routine jumps to the done label and exits.

If the calling function requested the default video mode, setvgamode checks if a previous mode has been saved in the defaultmode variable.

setdefault:

```

mov al, defaultmode
cmp al, -01h
je error

```

```

mov ah, 00h
int 10h
mov ax, 0001h
jmp done

```

Then the video mode is set to the defaultmode value.

Errors are signified by returning a zero. The done label simply allows the inline assembler routine to fall through.

error:

```

mov ax, 0000h

```

done:

SHOWVGA1.C is compiled the same way as SHOWVGA.C. However, SHOWVGA1.EXE is only 7766 bytes long. Compare this to 20342 bytes for SHOWVGA.EXE.

ShowEGA

ShowEGA displays a file with an extension ".EGA", and of the following format:

16 bytes of palette information.
28000 bytes for each of 4 bit planes.

The source C code for ShowEGA is shown in Listing 7. Since ShowEGA opens files in the same way ShowVGA does, I will start the program description at the EGA test.

```

regs.h.ah=0x12;
regs.h.bl=0x10;
int86(0x10, &regs, &regs);
if(regs.h.bl==0x10)
{
    puts("\n\r
        EGA display required");
    exit(-1);
}
if(regs.h.bl==0x0)
{
    puts("\n\r
        128K Min. video RAM required");
    exit(-1);
}
if(regs.h.bh)
{
    puts("\n\r
        EGA color display required");
    exit(-1);
}

```

Using the int86 library function, ShowEGA invokes INT 10_{HEX} function 12_{HEX}. When called with the value 10_{HEX} in the BL register, this function returns the EGA video configuration. If there is no EGA or VGA installed, the value 10_{HEX} will remain in the BL register. Otherwise, BL will contain the amount of EGA video RAM (0=64KB, 1=128KB, 2=192KB, and 3=256KB). The BH register

```

else
{
    if(strempi(sp, ".EGA")) /* if the file has an extension make shure */
    {
        puts("\n\r      File must be .EGA");
        exit(-1);
    }
}

if((fh=open(argv[1], O_RDONLY | O_BINARY)) < 0) /* open file to read screen */
{
    perror(NULL);
    exit (-1);
}

regs.h.ah=0x12; /* test for ega */
regs.h.bl=0x10;
int86(0x10, &regs, &regs);
if(regs.h.bl==0x10)
{
    puts("\n\r      EGA display required");
    exit(-1);
}
if(regs.h.bl==0x0)
{
    puts("\n\r      128K Min. video RAM required");
    exit(-1);
}
if(regs.h.bh)
{
    puts("\n\r      EGA color display required");
    exit(-1);
}

regs.x.ax=0x0f00; /* find current video mode */
int86(0x10, &regs, &regs);
defaultmode=regs.h.al;

regs.h.ah=0x0; /* switch to 640 X 350 and clear screen */
regs.h.al=0x10;
int86(0x10, &regs, &regs);

read(fh, blueplane, (unsigned)28016);
read(fh, greenplane, (unsigned)28000);
read(fh, redplane, (unsigned)28000);
read(fh, intensityplane, (unsigned)28000);
close(fh);

outpw(0x3c4, 0x802); /* copy planes to the screen */
memcpy((void *)0xa0000000, intensityplane, (size_t)28000);

outpw(0x3c4, 0x402);
memcpy((void *)0xa0000000, redplane, (size_t)28000);

outpw(0x3c4, 0x202);
memcpy((void *)0xa0000000, greenplane, (size_t)28000);

outpw(0x3c4, 0x102);
memcpy((void *)0xa0000000, (blueplane+16), (size_t)28000);

for(c=0; c<16; c++) /* set palette */
{
    regs.h.ah=0x10;
    regs.h.al=0x0;
    regs.h.bl=c;
    regs.h.bh=(blueplane+c);
    int86(0x10, &regs, &regs);
}

getch(); /* Wait for keyboard hit and reset screen */
regs.h.ah=0x0;
regs.h.al=defaultmode;
int86(0x10, &regs, &regs);

puts("\n      ShowEGA V1.0 \r"); /* display exit message and exit */
puts("      By Gregory McDonald");
exit(0);
}

```

will equal 0 for color systems or 1 for monochrome.

After verifying the correct video hardware is installed, ShowEGA can get the current video mode number and enter the 640 x 350 16 color graphics mode.

```

regs.x.ax=0x0f00;
int86(0x10, &regs, &regs);
defaultmode=regs.h.al;

```

```

regs.h.ah=0x0;
regs.h.al=0x10;
int86(0x10, &regs, &regs);

```

Like ShowVGA1, ShowEGA invokes INT 10HEX function 0FHEX to get the current video mode. Then INT 10HEX function 0 is used to enter the graphics mode.

Next the program reads and closes the EGA file.

```

read(fh, blueplane, (unsigned)28016);
read(fh, greenplane, (unsigned)28000);
read(fh, redplane, (unsigned)28000);
read(fh, intensityplane, (unsigned)28000);
close(fh);

```

The names of the plane variables are appropriate for the default palette. Also the palette is contained in the first 16 bytes of the variable blueplane.

Now the EGA bitplanes can be loaded with the bitmap.

```

outpw(0x3c4, 0x802);
memcpy((void *)0xa0000000, intensityplane,
        (size_t)28000);
outpw(0x3c4, 0x402);
memcpy((void *)0xa0000000, redplane,
        (size_t)28000);
outpw(0x3c4, 0x202);
memcpy((void *)0xa0000000, greenplane,
        (size_t)28000);
outpw(0x3c4, 0x102);
memcpy((void *)0xa0000000, (blueplane+16),
        (size_t)28000);

```

Notice 16 bytes are added to the blueplane address to skip over the palette.

ShowEGA now writes the EGA palette.

```

for(c=0; c<16; c++)
{
    regs.h.ah=0x10;
    regs.h.al=0x0;
    regs.h.bl=c;
    regs.h.bh=(blueplane+c);
    int86(0x10, &regs, &regs);
}

```

INT 10HEX function 10HEX subfunction 0 updates one EGA palette register. Before invoking the interrupt, place the palette location in the BL register and the desired palette value in the BH register.

After ShowEGA has the picture on the screen, it waits for the user to press a key, then switches back to the previous video mode and exits.

```

getch();
regs.h.ah=0x0;
regs.h.al=defaultmode;
int86(0x10, &regs, &regs);
puts("\n      ShowEGA V1.0 \r");
puts("      By Gregory McDonald");
exit(0);

```

ShowEGA can be compiled with Microsoft (MS) Quick C or MS C 5.0 or

Continued on Page 46

Assembly Language

Part 6

The Instruction Set (Part 4)

Pat Swayne
HUG Software Engineer

This is part of a continuing series on Assembly Language. In this installment, I will attempt to complete my discussion of the instruction set.

Branch Instructions

Normally, when the Central Processing Unit in your computer executes instructions, it does so in a sequential manner. When it "fetches" an instruction from memory, it automatically points the IP (Instruction Pointer) register to the next instruction in memory, skipping over any data that may have followed the last instruction. A branch instruction alters this process, because it actually loads a new value into the instruction pointer. Branch instructions are also called control transfer instructions.

There are three types of branch instructions: jump, call, and software interrupt. A jump instruction works like GOTO in BASIC. It transfers control to a different place in the program, and sequential instruction processing resumes at that point. A call instruction works like GOSUB. It transfers control to a different place in memory, but saves the old place so that control can be returned there later. A software interrupt is a special kind of call instruction.

Jump Instructions

There are two types of jump instructions: conditional and non-conditional. A non-conditional jump simply transfers control to a new address. A conditional jump transfers control if a specified condition is met, or it just skips to the next sequential instruction if the condition is not met.

Non-conditional jumps are further classified according to how far the jump

is, such as short, near, or far jumps. A short jump always uses the relative addressing mode, and the destination can be only up to +127 bytes or -128 bytes from the current instruction pointer position. A near jump can use absolute or relative addressing. If the addressing is absolute, the destination can be anywhere within a 64k segment. If the addressing is relative, the destination can be to +32767 bytes or -32768 bytes from the current position. When you consider that a 64k segment contains 65536 bytes, you may wonder how you can jump to the high end of the segment (using relative addressing) if the instruction pointer is at the low end. It is possible because a jump past one end of a segment will wrap to the other end. So, if the instruction pointer is at the "low" end of a segment, and the destination for a jump is at the "high" end, a jmp in the negative direction can be used.

Relative addressing is the type of addressing most commonly used for short and near jumps, but you work with these jumps in Assembly Language as if they used absolute addressing, specified with the immediate addressing mode (see Part 3 of this series). Consider this example:

```
START: JMP OVER ;JUMP OVER DATA
DATA DB 'ABCD'
OVER: MOV DX,OFFSET DATA
```

When the assembler assembles this code, it calculates the distance from the jump instruction to the label OVER and uses that value as the argument to the jump instruction. You do not have to be concerned with calculating a distance yourself. However, if you are working with short jumps, you need to make sure that the destination is not beyond the rather limited range of this type of jump. The actual starting point for measuring the dis-

tance between a jump instruction and its destination is the address of the first byte following the jump instruction. In the case of the example above, the distance would be measured from the label DATA to the label OVER.

Some assemblers, including the Microsoft assembler, use the mnemonic JMP for all non-conditional jump instructions, and the argument is used to determine what kind of jump instruction the assembler uses. In the example above, the distance is short enough for a short jump. However, the destination address is not known when the assembler first arrives at the JMP instruction as it processes the code. This situation is called a "forward reference". The Microsoft assembler, and some other assemblers, will handle this situation by inserting a near jump instruction into the output code followed by a couple of dummy bytes. Then, when the assembler has processed enough code to know the destination address, it will replace the dummy bytes with the actual offset to the destination. If the offset is 127 or less (a short jump), the assembler will replace the near jump instruction with a short jump, put in the offset, and a NOP (no operation) code to fill in the left over byte space. This wastes one byte for each short jump processed this way. To avoid this waste, you can specify a short jump in each case where you have a forward reference, and you know that the distance is 127 bytes or less. To specify a short jump, just use the word SHORT in the argument, as in:

```
START: JMP SHORT OVER
```

A few assemblers use the mnemonic JMP only for near jumps. A short jump is specified with JMPS, and a far jump with JMPF.

The address for a near jump can be specified using the direct, index, and register indirect addressing modes, an addition to the relative mode. Those of you who know Assembly Language for the old 8080 8-bit processor and are using this series to update yourselves will recall that the JMP instruction only used the immediate addressing mode to specify an absolute address. The immediate addressing mode is not even available with the 8086 processor family, but you can emulate it using the register indirect addressing mode, as in this example:

```
MOV  DX, OFFSET ADDRESS
JMP  DX
```

A far jump can be to anywhere in the 1-megabyte address space available to 8086 family processors. The destination can be specified using the immediate, direct, or index addressing modes. Since the Microsoft assembler and similar assemblers use JMP for both near and far jumps, and since the near jump can also use the direct and index addressing modes, the argument must specify the type of jump. The expression "WORD PTR" is used to specify a near jump, and the expression "DWORD PTR" is used to specify a far jump, as in these examples:

```
JMP  WORD PTR [SI] ;THIS IS A NEAR JUMP
JMP  DWORD PTR [BX] ;THIS IS A FAR JUMP
```

If the direct addressing mode is used, and if the location for the jump address is defined before the jump instruction, the expressions WORD PTR or DWORD PTR may not be necessary.

```
DEST DD 0
JMP  DEST ;THIS IS A FAR JUMP
```

In the above example, the assembler knows to use a far jump instruction, because the label DEST is defined using the DD (Define Double word) directive. If an address is defined before a jump, but it is not the right type (word or double word), you can override its type, as in this example:

```
DEST DW 0,0
JMP  DWORD PTR DEST ;THIS IS A FAR JUMP
```

By the way, the type of construction I used in the above two examples is normally used in cases where the address for a jump is calculated and filled in when the program is run. The zeros used in the DD and DW directives are just place holders.

The conditional jump instructions are all short jumps, and use relative addressing only. Here is a description of them.

JE/JZ (Jump if Equal/Jump on Zero) — The jump is made if the Zero flag is set. Since the CMP (Compare) instruction actually does a non-destructive subtraction, you can see that the Zero flag will be set if two numbers compared are equal.

JNE/JNZ (Jump if Not Equal/Jump

on Not Zero) — The jump is made if the Zero flag is not set.

Now that I have presented the first two conditional jump instructions, this would be a good place to answer the obvious question of what to do if you need to have a conditional jump to a destination too far away for a short jump. The answer is to use a conditional jump of the opposite condition around a non-conditional jump to the destination. For example, suppose you need a jump on zero to an address too far away. Here is how it could be done:

```
JNZ  AROUND
JMP  FARAWAY
AROUND:
```

Control will be transferred to the address FARAWAY if the Zero flag is set, or to AROUND if it is not set.

JB/JNAE/JC (Jump if Below/Jump if Not Above or Equal/Jump on Carry) — The jump is made if the carry flag is set. The carry flag will be set whenever a larger number is subtracted from a smaller one or when a larger number is compared with a smaller one. As you can see, there are often more ways than one to express a particular condition.

JBE/JNA (Jump if Below or Equal/

Jump if Not Above) — The jump is made if either the Carry or Zero flags are set. In other words, if a condition tests below or equal, the jump is made.

JNBE/JA (Jump if Not Below or

Equal/Jump if Above) — The jump is made if both the Carry and Zero flags are clear.

JNB/JAE/JNC (Jump if Not Below/Jump if Above or Equal/Jump on Not Carry) — The jump is made if the Carry flag is clear.

JL/JNGE (Jump if Less/Jump if Not

Greater or Equal) — The jump is made if either the Sign flag or the Over flag is set, but not if both are set. The difference between JB and JL is that JB is for comparisons of unsigned numbers, and JL is for comparisons of signed numbers. I have seen programs where JL was used, and JB should have been used. The programs worked fine as long as the numbers being compared were within the range +/-127 for bytes or +/- 32767 for words, but they "crashed" or worked incorrectly if larger numbers were compared. If in doubt, use JB instead of JL.

JLE/JNG (Jump if Less or Equal/Jump if Not Greater) — The jump is

made if either the Sign flag or the Overflow flag is set (but not both) or if the Zero flag is set.

JNLE/JG (Jump if Not Less or Equal/Jump if Greater) — The jump is made if the Sign flag and Overflow flag are both in the same state (both set or both cleared), or if the Zero flag is set.

JNL/JGE (Jump if Not Less/Jump if Greater or Equal) — The jump is made if the Sign flag and Overflow flag are both in the same state.

JO (Jump on Overflow) — The jump is made if the Overflow flag is set.

JNO (Jump on Not Overflow) — The jump is made if the Overflow flag is clear.

JP/JPE (Jump on Parity/Jump if Parity Equal) — The jump is made if the Parity flag is set.

JNP/JPO (Jump on Not Parity/Jump if Parity Odd) — The jump is made if the Parity flag is clear.

JS (Jump on Sign) — The jump is made if the Sign flag is set.

JNS (Jump on Not Sign) — The jump is made if the Sign flag is clear.

JCXZ (Jump if CX Zero) — The jump is made if the CX register contains a zero. This jump is useful in bypassing loops (explained below).

Loop Instructions

Loop instructions are a special kind of conditional jump instruction that decrements and then tests the CX register rather than, or along with, a flag. The JCXZ instruction sort of fits into this group, although it does not decrement the CX register.

LOOP — This instruction decrements the CX register, and makes the jump if CX is not zero. This instruction can replace:

```
DEC  CX
JNZ  LABEL
```

It is not really an exact replacement for the above, because it does not affect any flags. However, this usually does not matter unless you are concerned with preserving the state of the flags through the execution of a loop.

LOOPE/LOOPZ (Loop while Equal/Loop on Zero) — This instruction decrements the CX register, and makes the jump if CX is not zero and the Zero flag is set. It will not make the loop if the Zero flag is clear regardless of the value in CX.

LOOPNE/LOOPNZ (Loop while Not Equal/Loop on Not Zero) — This instruction decrements the CX register, and makes the jump if CX is not zero and the Zero flag is clear.

The CALL Instruction

The CALL instruction works much like a non-conditional jump. There are near and far calls, and they use the same addressing modes as near and far jumps. The difference is that CALL saves the current location in the program on the stack before it transfers control to the new ad-

dress. A near call saves the IP (Instruction Pointer) register on the stack, and a far call saves the IP register and the CS (Code Segment) register. There is also a RET instruction that restores the saved values to the appropriate registers, so that program execution can resume at the location following the CALL. The Microsoft assembler uses the mnemonic CALL for both near and far calls, and the argument can determine the type of call. There is also an assembler directive PROC that can be used to specify the type of call. Here is an example of it.

```

SUB1  PROC    NEAR
      ***** ;SUBROUTINE GOES HERE
      RET
SUB1  ENDP
CALL  SUB1    ;THIS IS A NEAR CALL

```

The directive PROC determines the start of a subroutine, and the directive ENDP determines the end of it. The "type declaration" following PROC can be NEAR to specify a near call, and FAR to specify a far call. The type of the RET instruction is also determined by the PROC directive. Some assemblers will also allow you to specify a far call with CALLF, and a far return with RETF.

If you do not use the PROC directive, and if there is no argument to define the type of CALL and RET instructions, then the assembler assumes that they are all of the near type. Therefore, if you are writing a program that will be a .COM file after it is assembled (rather than an .EXE file), you do not have to bother with PROC or other ways to declare the type of CALLs and RETs.

The RET instruction can take a numerical argument that will cause the stack pointer to be incremented by the indicated amount after the program location is restored. This has the effect of discarding saved values on the stack. I will show you a good use for this kind of RET after I explain interrupts.

Interrupts

An interrupt is a special kind of subroutine call that can be executed by the computer hardware (this kind is called a hardware interrupt), as well as by a computer instruction (a software interrupt). Hardware interrupts allow your computer more than one job at a time. To illustrate this, copy a file from one floppy disk to another, or from your hard disk to a floppy, and while the computer is busy copying the file, type DIR and press Return. When the computer has finished copying the file, it will execute the DIR command. What happened is that each time you pressed a key while the computer was copying, the keyboard circuitry generated an interrupt. The interrupt caused the currently running program (which in this case was the file copying routine of MS-DOS) to be stopped briefly ("interrupted"), and

then it called a subroutine that scanned the keyboard, and placed the code for the key that was pressed into a special memory area, called a "type-ahead buffer". After the file was copied, the MS-DOS command processor checked the type-ahead buffer, found some characters in it, and processed them.

When a hardware interrupt is executed, it is as if the processor found a way to stuff a far call instruction between the last instruction executed, and the next one about to be executed. An interrupt is similar to a far call except the flag register

is saved on the stack in addition to the IP register and the CS register. The address of the interrupt subroutine is stored in a special table that occupies the first 1024 bytes (1 k) of memory. Each entry in this table is 4 bytes long, and can contain the offset and the segment of a subroutine. Each interrupt has a "type" number associated with it, which is multiplied by 4 to get the location of that interrupt's subroutine address in the table.

There is a machine instruction, INT, which can be used to call an interrupt subroutine within a program. The INT instruction takes a number as its argument, which is the type number of the interrupt subroutine you wish to call. When you use INT to execute an interrupt subroutine, it is called a software interrupt. Actually, the INT instruction is also used to execute hardware interrupts. When a hardware interrupt occurs, the interrupt controlling circuitry causes the processor to stop fetching instructions for the currently running program from memory, and places the binary code for an INT instruction and a type number on the data lines. The processor "sees" this code as if it had been fetched from memory, and executes it in almost the same way. About the only difference is that it does not increment the instruction pointer, as it would after fetching a normal instruction.

In an IBM compatible computer, software interrupts are used to call subroutines in the BIOS (Basic Input/Output System). You can read the keyboard or display characters on the screen, among other things, using BIOS subroutines. When you hear someone talking about the "video interrupt", or the "keyboard interrupt", he may be talking about BIOS subroutines rather than hardware interrupts associated with the display or keyboard. Here is an example of how to make your computer beep using the BIOS.

```

MOV  AX,0E07H ;CODE TO BEEP
INT  10H      ;CALL VIDEO ROUTINE

```

The INT instruction normally uses two bytes: one for the instruction itself, and one for the type number. However, there are two special INT instructions that have the type number implied in the instruction. One is the type 3 interrupt, which is designated as the "breakpoint" interrupt. When you run DEBUG and execute some code with a breakpoint set, DEBUG replaces the byte at the breakpoint address with this special interrupt instruction. When the program gets to the breakpoint, the INT 3 subroutine is executed, which causes DEBUG to regain control, and replaces the original instruction in the program. The breakpoint interrupt is only one byte long so that it can be used to replace any other instruction, including single byte instructions. If you use INT 3 in an Assembly Language program, the assembler will use the special single byte instruction for the type 3 interrupt rather than using the general instruction followed by a 3.

The other single byte interrupt instruction is called INTO. It generates a type 4 interrupt if the overflow flag is set. If the overflow flag is not set, the processor just skips over the instruction.

The instruction used to return from an interrupt subroutine is called IRET. It works like a far RET instruction except that it also restores the flag register in addition to the IP and CS registers. Therefore, an interrupt saves the flags when it is executed, and restores them when it is done. This is desirable in the case of hardware interrupts, because you do not want the flags to be destroyed every time you do something that causes a hardware interrupt while a program is running. However, if a routine is called via a software interrupt, you may want to have a condition passed to the caller using the flags. You can do this by using the far RET instruction with a numerical argument, as mentioned previously. If the argument is 2, the instruction will restore the IP and CS registers, and discard the saved flag register. Control will return to the caller with the flags set according to what was done in the interrupt subroutine.

One thing that I should mention now is that whenever an interrupt is executed, whether it is a software or hardware interrupt, other hardware interrupts are disabled. In other words, the Interrupt flag (which is normally set) is cleared. This flag is restored by the IRET instruction, but if you use RET 2 to return from an interrupt subroutine, you must restore the Interrupt flag yourself. That can be done using one of the machine control instructions, which I have not covered yet. Since this article is long enough for one month, I will cover them next month. *

Powering Up Volume 2

William M. Adney
P.O. Box 531655
Grand Prairie, TX 75053-1655
Copyright © 1990 by William M. Adney.
All rights reserved.

Connecting a Modem to Your Computer

There are any number of ways to drive yourself crazy when working with a computer, but there is one preferred way above all others — try to get a serial device (e.g., printer or modem) working for the first time. It all sounds so simple when you are in a store or reading a magazine. Power off the computer, plug in the cable to the computer, plug in the cable to the peripheral, and power up everything. Unfortunately, this simple approach usually does not work in the real world, and that is the reason for this article. Sometimes you will find that the process is more difficult than you imagined, usually because of a lack of information. The information you need to know to solve problems is really pretty easy, but it is occasionally difficult to find, even for a simple task like connecting a modem to your computer. This article is an introduction to the RS-232C serial interface in terms of how it works with your computer and modem. The next article will include a discussion of connecting a serial printer which is a more complex problem, and there will also be a brief discussion on how to connect a parallel (using the Centronics interface) printer.

Because of the technical details in this article, I have assumed you know some of the basic definitions that are used, which were discussed in Chapter 6 (Connecting Peripherals to Your Computer) of the original *Powering Up* book. To be specific, this article deals with connecting devices to a serial port, usually called COM1. There are, of course, two ways to connect a serial device to your computer.

The Easy Way

As I mentioned in the chapter on "Connecting Peripherals to Your Computer", I strongly recommend that you buy a ready-made cable when you buy a peripheral, such as a modem or a printer.

You will generally find that this approach will save an incredible amount of time, especially if you are not sure exactly what cable you need. More importantly, do not assume that any old cable will work because it probably will not. A cable that works fine with a printer almost certainly will not work with a modem, and vice-versa. In somewhat technical terms, you will want a "regular" cable to connect your computer to an external modem, and you will want a null-modem cable to connect your computer to a printer. You will see what the real difference is in the construction of both kinds of cables later in this article.

Also, do not mix cables between different types of computers. For example, a cable used to connect an Apple computer to a printer will probably not work on a PC compatible system, such as a Heath/Zenith computer. If you have different types of systems, it is even more important to keep everything organized so you can find it, if nothing else. When you originally connect something and get it working properly, it is easy to see what that exact setup is. If you have a lot of cables for one reason or another, it is important to be able to identify them. And I will digress a moment to discuss a trick I use for identifying all kinds of cables — both power cables and cables to my peripheral devices.

Cable Labels

If you followed one of my suggestions in Chapter 1 of the original *Powering Up* book about using a power strip to make it easier to power up your system, you may have discovered at least three different items plug into that unit: the System Unit, the CRT, a printer, and perhaps an external modem. Which plug is which? Depending on the installation, you can have all kinds of interesting experiences trying to trace which power cord goes to

what device. That is one problem.

The other problem is what cable goes to which device, especially if you have a lot of cables like I do. For example, I have my serial port connected to an A-B Box so that I can use either a mouse or a modem on that port. I have two printers on the parallel port — an old DTC StyleWriter daisy wheel printer and an Epson FX-850. To complicate matters, I also have my old Z-248 in my study which is also connected to the two printers through a switching device called a crossover switch. That allows me to use either printer with either computer. But what about all those cables?

You can find various items called "cable tags", but those tend to get expensive. Instead, I simply use labels that I originally bought for file folders. First, I write the appropriate information on the label (e.g., Z-386, modem, FTM Monitor or whatever), remove it from the backing, and wrap it around the cable, taking care to have the two ends match up and stick together. If I want to record more information than will fit on a single label, then I just keep adding labels. Be sure to add a label to the power cord (so you will know which power cord goes with what hardware), as well as the connecting cable to any peripheral. This is a very inexpensive way to keep all cables organized, and you probably already have some of these labels like I did. If you really want to get fancy, you can get different colors for color coding too, but I normally use the plain white ones. Now let's look at the hard way to connect a serial device.

The Hard Way

If you do not get a cable with a peripheral that you buy, then you are probably looking at the hard way of connecting that device to your computer. Even so, sometimes you will run into some installation problems, some of which are really

tricky to find. At this point, you are into the troubleshooting mode, and it pays to be systematic. I have found that a good first item to check is the cable, but it really helps to have some idea before you jump into that whole process. A cable to connect a modem is quite simple, so let's begin by looking at some general information about the serial port and the RS-232C standard, usually referred to as just RS-232.

DTE and DCE

The RS-232 standard was established some years ago by the Electronic Industries Association (EIA). The standard specifies the requirements for the asynchronous serial interface: voltages, impedances, and physical connectors. In general, the RS-232 standard establishes two categories for equipment. The first is DTE or Data Terminal Equipment. The second category is DCE or Data Communications Equipment, although I have seen some IBM references that define the term DCE as Data Circuit-terminating Equipment, which is probably a more accurate definition for this discussion as you will see.

One of the critical points about the original RS-232 definition is that it was originally developed for data communications (i.e., between computers and modems) and had nothing whatever to do with the common serial devices available for computers today. There is no definition that was designed to support a serial printer, let alone something like a serial mouse. What actually happened is that manufacturers decided to ADAPT the RS-232 standard to support other devices, such as a serial printer. If you look at the original definition of the standard, for example, you will find that a computer by itself is actually defined as a DCE device and a "data terminal" is a DTE device. This is easier to see in a mainframe computer system where a data terminal, such as an IBM 3278, is physically separated from the computer while today's microcomputers consists of both the data terminal and the computer inside a single box. This distinction has become somewhat blurred because of the way one normally considers a microcomputer when you plug in a peripheral. For that reason, we must use a different definition.

By definition then, a PC compatible computer is generally defined as Data Terminal Equipment or DTE. That is probably a result of the fact that the original IBM PC did not have a built-in serial interface, although one could be added using the optional Asynchronous Communication Adapter card to add the COM1 serial port. Although the original IBM PC did include a parallel port (LPT1) that was essentially "reserved" for a printer, it is apparent that IBM thought that a serial port would essentially be used only if one had a modem.

By any definition you care to cite, a modem is defined as Data Communications Equipment (or Data Circuit-terminating Equipment, if you prefer) or DCE. In essence, DCE is simply the hardware that performs the signal conversion between the DTE (i.e., the computer) and a TELEPHONE line. This signal conversion is required because your computer works with a digital signal, and a telephone line was designed to carry an analog signal, like your voice. The transmitting modem converts the digital signal to an analog signal, and a receiving modem converts the analog signal back to digital form. As it turns out, the word MODEM is a simple acronym from the words MOdulator-DEModulator, which is the technical name for this conversion process.

For DTE and DCE, the theory says that the general rule is: Connect DTE to DCE (or vice-versa), and NEVER connect two like types of equipment together (i.e., DTE to DTE or DCE to DCE). From a theoretical standpoint, you can connect a modem (DCE) to your computer (DTE), and that works fine. But if you know that a LOT of serial printers are defined as DTE, then the general rule says that you cannot connect it to your computer, which is also DTE. Fortunately, we can use a trick, which is usually called a NULL-MODEM, to make a DTE printer LOOK like it is a DCE device.

To carry this thought one step further, it is important to note that you cannot use a cable for a DCE device (modem) to connect a DTE device, such as a printer, to

your computer because that violates the rule. In other words, the cable for the modem is physically different from the cable used to connect a DTE printer to the computer because of the null-modem "feature." You will see the wiring differences in the next article.

The 25-Pin Connection

For the sake of completeness, it is probably appropriate to list the descriptions, by pin number, of all 25 lines that are defined in the RS-232 standard. These descriptions are shown as Figure 1.

Figure 1 represents a list of all the RS-232 signals and their associated pin or line number. The Name of each signal is a common abbreviation that may be used in documentation. The *Originates From* columns contain an "X" which indicates whether the signal originates from the computer or the modem. And the *Description* is the complete description of the acronym used for the Name. A *Description* listed in capital letters indicates there is no signal flow on that line (or at least there should not be), either because it is a ground (i.e., pins 1 and 7) or because it is not defined by the standard (e.g., pins 18 and 25).

One special note about Figure 1 — I have shown Pin 8 as Carrier Detect (CD), which is the most common description you will find in documentation. However, some documentation also calls this same signal "Data Carrier Detect" or DCD, and you should know that these two terms are used interchangeably in this context.

| Pin | Name | Originates From . . . | | Description |
|-----|--------|-----------------------|-------|-------------------------------|
| | | Computer | Modem | |
| 1 | CG | | | CHASSIS GROUND |
| 2 | TD | X | | Transmitted Data |
| 3 | RD | | X | Received Data |
| 4 | RTS | X | | Request To Send |
| 5 | CTS | | X | Clear To Send |
| 6 | DSR | | X | Data Set Ready |
| 7 | SG | | | SIGNAL GROUND |
| 8 | CD | | X | Carrier Detect |
| 9 | +V | | X | Positive DC Voltage |
| 10 | -V | | X | Negative DC Voltage |
| 11 | QM | | X | Equalizer Mode |
| 12 | (S)DCD | | X | Secondary Data Carrier Detect |
| 13 | (S)CTS | | X | Secondary Clear To Send |
| 14 | (S)TD | X | | Secondary Transmitted Data |
| 15 | TC | | X | Transmitter Clock |
| 16 | (S)RD | | X | Secondary Received Data |
| 17 | RC | | X | Receiver Clock |
| 18 | | | - | NOT DEFINED |
| 19 | (S)RTS | X | | Secondary Request To Send |
| 20 | DTR | X | | Data Terminal Ready |
| 21 | SQ | | X | Signal Quality Detect |
| 22 | RI | | X | Ring Indicator |
| 23 | DRS | X | | Data Rate Selector |
| 24 | TC | X | | Transmitter Clock (External) |
| 25 | | | | NOT DEFINED |

Note: A blank under Originates From means NOT APPLICABLE.

Figure 1
25-Line RS-232 Standard

Most Heath/Zenith manuals and other documentation use the CD terminology, so I will consistently use that in the rest of this article.

Not all signals are used in microcomputers for many reasons; some of them obvious, some of them not. One of the major reasons that not all of these signals are used is because the standard was originally developed for data communications as mentioned earlier. I have included this list for the sake of completeness, but we will only consider the actual lines used in the microcomputer serial interface for PC compatible systems. As it turns out, there are a maximum of 9 of these signals that are used in today's computers, even though many computers, such as the Z-150, still used a 25-pin connector for the serial port.

The 25-Pin Connector

Since there are only a maximum of 9 of these signals actually used, the first thing to do is to define them as shown in Figure 2.

| Pin | Name | Originates From . . . | | Description |
|--------|------|-----------------------|-------|---------------------|
| | | Computer | Modem | |
| 2 | TD | X | | Transmitted Data |
| 3 | RD | | X | Received Data |
| 4 | RTS | X | | Request To Send |
| 5 | CTS | | X | Clear To Send |
| 6 | DSR | | X | Data Set Ready |
| 7 | SG | | | SIGNAL GROUND |
| 8 | CD | | X | Carrier Detect |
| 20 | DTR | X | | Data Terminal Ready |
| 22 | RI | | X | Ring Indicator |
| Shield | CG | | | CHASSIS GROUND |

Note: A blank under *Originates From* means NOT APPLICABLE.

Figure 2
9-Line Serial Connection

Before we look at the signals listed in Figure 2, it is important to note that the Pin-1 signal, CHASSIS GROUND (sometimes called Frame Ground), is not shown on Pin 1 because it is usually connected to the cable shield (on the DB-25 case) on ONE end of the DB-25 connector. It is intended to have the CG line connected at only one end because it is possible that two pieces of equipment will have DIFFERENT ground potentials, which is just a result of being connected to different grounding points. Connecting the CHASSIS GROUND (shield) at both ends can cause a problem.

Because of the potential difference, current will actually flow in the CHASSIS GROUND (CG) line (or shield) if it is connected at both ends, which defeats the purpose of the shield in terms of reducing interference and may actually increase it. This is a point that you should know because if you have checked everything else and have found no problem, you may be able to fix an *interference* problem by simply cutting the ground (from the

shield) at one end of the cable because a long cable acts like an antenna. Be SURE that the cable is actually grounded at both ends before you try this. For example, you may find that you hear or see interference on a television or radio when you are printing, and you cannot find any other reason for it. I do NOT recommend doing this to all cables as a just-in-case interference-protection measure, but keep it in mind as a possible solution to an interference problem. If everything works fine the way it is, leave it alone.

As you might have noticed from the Description of these signals, most of them are related as pairs, and we will consider them as such. And since Pin 7 is the SIGNAL GROUND (SG), let's continue with the discussion of grounding.

Pin 7 — the SIGNAL GROUND (SG) — provides a common reference point for all signals (e.g., data and timing signals), and it *must* be connected at both ends of the cable for the interface to work properly. Like the previous CHASSIS GROUND (CG), a difference in the "reference"

ground of the hardware can also cause a problem here, especially if only one end of the CG is connected. What happens is that the voltage difference now causes a current flow in the SIGNAL GROUND LINE, which can also cause data transmission problems, such as a printer failing to print data correctly.

As you can see, this is kind of a "Catch 22" situation. If you have the CHASSIS GROUND (CG) connected to only one end of the cable, and the equipment is at different ground potentials, it can cause a problem in the CHASSIS GROUND (CG) which will garble the data. And if you fix that by grounding both ends of the SIGNAL GROUND, then you may have an interference problem to a television or radio. Some days are like that — you fix one problem only to cause another. If you have that problem, one easy solution is to plug in both the computer and printer (or modem) to the same electrical socket (or circuit), assuming the socket is the common 3-prong type which accepts the equipment plugs. That should elimi-

nate any difference in the CG potential and not cause any data transmission problems.

When looking at the *Name* and *Description* of each signal, it is important to know that they are identified as seen from the (i.e., computer) DTE point of view. This applies to all lines listed in Figure 1, as well as Figure 2. Let's see how that works with the first pair of lines in Figure 2.

Pin 2 — Transmitted Data — and Pin 3 — Received Data — are described from the DTE (i.e., computer) point of view. That is, the computer transmits (i.e., originates) data on Pin 2 and receives data on Pin 3. However, the modem does the reverse: it transmits data on Pin 3 and receives data on Pin 2.

Pin 4 — Request To Send — and Pin 5 — Clear To Send — are also described from the computer's point of view. First, the modem transmits a CTS signal to indicate that the computer may transmit. Then, the computer usually transmits an RTS signal, which frequently activates the modem's carrier signal.

Pin 6 — Data Set Ready — and Pin 20 — Data Terminal Ready — generally indicate whether the hardware is ready. In this context, it is important to know that the term "Data Set" is just another name for a modem, so the DSR signal actually indicates "modem ready". The DSR signal is used to indicate that the modem is powered on, among other things. If the modem has an auto-answer mode, the DTR signal is frequently used (and may be controlled by software) to tell the modem to answer a call in response to the Ring Indicator signal originated by the modem.

Pin 22 — Ring Indicator — is originated by the modem to tell the computer that the phone is ringing. Technically, the modem recognizes the voltage used to ring the phone's bell, which activates the Ring Indicator signal. Then, the computer tells the modem to answer the phone using the DTR signal discussed in the previous paragraph.

Pin 8 — Carrier Detect — is originated by the modem when it receives a valid carrier signal on the phone line. Some documentation I have seen, including that for most Heath/Zenith computers, identifies this signal as Carrier Detect or CD. Some computers (not PC compatibles) require this signal before they will transmit or receive data. For those computers, Pin 8 is connected to Pin 20 (DTR), which is sometimes required for a null-modem cable.

This brief summary of the signals used in a PC compatible computer and their peripherals is intended to give you a general idea of what the purpose of each signal is. Each signal is discussed in relation to its standard definition for use with a modem, and of course not all signals are used in each serial device that is connected to your computer. It is ridiculous,

for example, to say that the Ring Indicator signal has any application to a serial printer or mouse, but all of the signals mentioned are generally defined for most serial interfaces used in most computers. There is at least one exception.

The eaZy PC

Although the eaZy PC has a built-in "serial port", its use is *strictly* limited to a mouse, and it is sometimes called a "mouse port" for that reason. In other words, you cannot use the built-in port for a serial printer or modem because it simply will not work. If you want a true serial port for the eaZy PC, you must buy the separate plug-in module that includes it. As many of you know, the eaZy PC was discontinued some time ago, and that particular part is extremely difficult to find. If you do not already have that particular part for the eaZy PC, I do not have any suggestions as to where you might find one.

Although the eaZy PC is the only Heath/Zenith computer I know of that does not have a standard serial port, be sure to check your Owner's Manual and other documentation if you have any questions about how the serial interface is actually implemented on your computer. With the eaZy PC as the exception, let's see how to connect a modem to other Heath/Zenith computers.

Connecting a Modem

Connecting a modem is quite easy, and it uses the kind of "straight-through" cable that you might expect. Your computer will require one of two possible cable configurations, depending on exactly what kind of computer you have. Virtually all of the older 8088-based computers, such as the Z-150, use a 25-pin connector. Computers that have an 80286 or 80386 normally have a 9-pin connector. Because most current external modems have a 25-pin connector, there are two possible cable configurations. Figure 3 shows the cable configuration for a 25-pin computer port to a 25-pin modem.

As you can see, this cable is simple and straightforward because it is just a

matter of connecting the pin number on one end of the cable to the same pin on the other end. For newer computers that have a 9-pin connector, the cable configuration is shown in Figure 4.

type of parity (odd, even or none), and number of stop bits (usually 1). For example, you can call the HUG Bulletin Board at (616) 982-3956 at 300, 1200 or 2400 baud with parameter settings of 8 data

| Computer Pin # | Signal Direction | Modem Pin # | Description |
|----------------|------------------|-------------|---------------------------|
| 1 | <-- | 8 | Carrier Detect (CD) |
| 2 | <-- | 3 | Received Data (RD) |
| 3 | --> | 2 | Transmitted Data (TD) |
| 4 | --> | 20 | Data Terminal Ready (DTR) |
| 5 | <--> | 7 | SIGNAL GROUND (SG) |
| 6 | <-- | 6 | Data Set Ready (DSR) |
| 7 | --> | 4 | Request To Send (RTS) |
| 8 | <-- | 5 | Clear To Send (CTS) |
| 9 | <-- | 22 | Ring Indicator (RI) |
| Shield | | | CHASSIS GROUND (CG) |

Figure 4
9-Pin Computer to 25-Pin Modem

Figure 4 shows how to connect all of the computer's 9-pins to the corresponding line on a 25-pin connector to a modem. Some modems and communications software may have different requirements, and you should be sure to check both manuals for any changes. If you find any differences, be sure to follow the directions in the manuals. Both of the cable configurations shown in Figures 3 and 4 will also work as extension cables for other peripherals, such as a mouse.

Getting a Modem to Work

Once you have the proper cable, there are two other things you must do in order to use the modem. First, you must set the modem's switches (or jumpers) as appropriate for your system. This information is generally contained in most modem manuals (and some communications software manuals), and proper switch settings are critical. Otherwise, your modem and/or your software will not be able to communicate correctly and may not work at all.

The second thing you must do is find out the parameters of the system, such as a bulletin board, that you want to communicate with. These parameters specifically include a telephone number, baud rate, number of data bits (usually 7 or 8),

bits, no parity, and 1 stop bit. Most good communications software provides the capability to store these parameters in a "directory" so you do not have to fool around with them more than once.

Once you have connected a modem, set its switches, and determined the calling parameters of the system you want to talk to, it is a simple matter of using the communications software to access that system. There are lots of good modems available, and I always recommend a Hayes-compatible modem. I recommend an external modem (except for laptops) for the simple reason that it can be used with nearly any computer with the proper cable. For example, my Prometheus Pro-Modem has been used with my Z-89, Z-100, Z-200, and Z-386/16 over the years, and you will never find an internal modem that works with all of those different systems. Although an external modem costs slightly more, I have saved considerably more than the original price difference because I could use it with all of those systems. Of course, you would probably not want an external modem with a laptop because of the size and weight considerations, so an internal modem is the only reasonable choice for these systems.

There is a lot of good communications software available. If you are just beginning to explore data communications, one of the best, and least expensive, software packages available is the HUGMCP (Modem Control Program) that has all of the basic features you are likely to need.

If you are looking for what may be the most powerful communications software available, then I highly recommend HyperACCESS/5 that is available from Hilgraeve. HyperACCESS/5 is very sophisticated with an incredible variety of features, including a "programming" language called HyperPilot, but with all its power and sophistication, it is remarkably easy to use even for a beginner. Hyper-

| Computer Pin # | Signal Direction | Modem Pin # | Description |
|----------------|------------------|-------------|---------------------------|
| 2 | --> | 2 | Transmitted Data (TD) |
| 3 | <-- | 3 | Received Data (RD) |
| 4 | --> | 4 | Request To Send (RTS) |
| 5 | <-- | 5 | Clear To Send (CTS) |
| 6 | <-- | 6 | Data Set Ready (DSR) |
| 7 | | 7 | SIGNAL GROUND (SG) |
| 8 | <-- | 8 | Carrier Detect (CD) |
| 20 | --> | 20 | Data Terminal Ready (DTR) |
| 22 | <-- | 22 | Ring Indicator (RI) |
| Shield | | | CHASSIS GROUND (CG) |

Figure 3
25-Pin Computer to 25-Pin Modem

ACCESS/5 is highly recommended for both the beginner, as well as the most demanding power-user.

Connecting and using a modem is generally easy, but take your time during the setup and be sure to read the manuals. Be sure you have the right cable, and set the modem switches as recommended in the manual. Then be sure that you have the correct calling parameters for the system you want to communicate with, and take some time to read the manual with your communications software so you will understand how to use it. Now let's look at some simple troubleshooting ideas that can help you get a modem (or other peripheral) to work with your computer.

Troubleshooting

Using the wrong cable seems to be the most common problem when a peripheral does not work correctly or at all, and that is why I always recommend checking the cable first. Checking the cable can save a lot of time, and it is really easy (and inexpensive) to do. The whole idea is to make *SURE* that the pins are actually connected the way they are supposed to be, which is easily verified with something called a continuity tester. One way is to use a Digital Volt Meter (DVM), Digital Multi-Meter (DMM) or Volt-Ohm Meter (VOM). If you need a DMM for other things, such as checking voltages in a house or car, one of the best deals around is the small SM-2300-A DMM that is available from Heath Company for \$19.95. It has a continuity tester that beeps when the circuit is complete, which makes it extremely useful for this kind of application.

You can also buy a simple, but effective, continuity tester at most auto parts stores for under ten dollars. This kind of continuity tester typically looks like a penlight (flashlight) with a "probe" and a wire with an alligator clip. To check continuity, you connect the alligator clip to the pin on one connector and insert the probe into the appropriate pin on the other connector. If the light turns on for the appropriate pins (Figure 3 or 4), you have verified that that line is correctly connected. Of course, you want to be sure you have a good connection, and it is good practice to check the "non-matching" pins, just in case there is a short circuit in the cable or one of the connectors.

After checking the cable, next verify that all switches (or jumpers) on the modem are set correctly. Most of today's modems (and communications software) are Hayes compatible, and appropriate switch settings should be documented in the manual provided with the modem. Note that some switch settings may be unique to a PC compatible or Apple computer, so you want to be sure that you are using the right settings for your system. In most modem manuals, you will find a de-

tailed explanation of each switch setting which can help you figure out a problem in that area. And if you can find no other problems, don't be afraid to change a few of the settings that seem to make sense based on the documentation — just be sure to make a note of the current settings in case you want to return to the original configuration. As you can see, I suggest checking the hardware — the cable and modem switches — before anything else because that is the most common type of problem. And if you find something that is not correct in the hardware, be sure to test the modem with your communications program after fixing that problem.

The third thing to check is the calling parameters that you are using. This includes making sure that your communications software is set up to address the appropriate serial port which is usually COM1 (the default). Although an incorrect hardware configuration (i.e., cable or modem switches) can cause data to be garbled in some cases, the most frequent cause is the use of the wrong calling parameters. Different bulletin boards and services (e.g., CompuServe) use different parameters. As I mentioned earlier, the HUG Bulletin Board requires parameters of 8 data bits, no parity, and one stop bit; but CompuServe requires 7 data bits, even parity, and one stop bit. Not all bulletin boards are the same, and you must be sure to have the correct calling parameters. Most bulletin boards support 1200 baud, and many also support 300 and 2400 baud. Because of these calling parameter differences, a "directory" containing this parameter setup information is included in most good communications software, such as HUGMCP and HyperACCESS/5. Also, be sure to save the parameters in your communications software's directory.

A Note About CONFIGUR

Most Heath/Zenith computer users know about the ZDS CONFIGUR program which provides an easy way to configure the serial and parallel ports. However, its use is NOT usually required when you are setting up data communications with a modem because most communications software takes care of any "configuration" changes, such as baud rate. In other words, do not fool around with CONFIGUR when trying to get communications software to work with a modem because it is not necessary. By its nature, communications software must work directly with the computer's operating system and hardware, and it bypasses the CONFIGUR settings because it sets them directly — such things as baud rate, number of data bits, stop bits, and type of parity. In short, do not try using CONFIGUR because it will not help solve a problem with modem communications.

The same statement applies to a

mouse. Mouse-driver software also sets the required parameters directly, and CONFIGUR will not help if you cannot get a mouse to work properly.

Getting a Mouse to Work

As a final note, I should mention that many users seem to have problems getting a serial mouse to work correctly in their system. In most cases, that is a result of failing to read and follow the installation instructions provided with the mouse. If you need an extension for the mouse, be sure to use a straight-through cable as listed in this article for a modem. Aside from the obvious plug-in installation of a mouse to COM1, some mouse software *MUST* be installed with the manufacturer's INSTALL program. I am seeing more and more software that is "compressed" on a disk to save space, and it must be "uncompressed" before it is usable, which is one function of the INSTALL program.

Next, you must install a mouse driver, and most manufacturers supply two programs. One is a MOUSE.COM program that can be installed as desired from the command line (or in the AUTOEXEC.BAT file). The other is a device driver like MOUSE.SYS that can be "permanently" installed in the CONFIG.SYS file. For some reason, many users seem to think that both must be "installed", but that is simply not true for most popular brands of mice (mouses?). Only one or the other is usually required, and which one you use is a matter of preference. I personally prefer to install a mouse with a COM program so I can specifically control how memory is used. If you always use a mouse, you may prefer to use the device driver instead. Installing both generally does not cause a problem, but it does waste valuable memory. Again, be sure to read the mouse documentation for specific details on the mouse you have. Sometimes you will also find a "control panel" program that allows you to adjust some of the mouse parameters (e.g., sensitivity), and it is usually optional.

The final step is to check each application to see if you must "install" the mouse. You would not believe the number of letters I received some time ago when I wrote about the Logitech Mouse (my personal preference). Most of these letters noted that the Logitech Mouse would not work on various Heath/Zenith computers with the GEM Presentation Team (and other software), which is a graphics package that I use quite often and was mentioned in that article. In the specific case of that version of the GEM Presentation Team, I know that you must run the install program (INSTALL.APP) to inform the software that you are using a mouse and what kind of mouse it is. If you don't, well, the mouse will simply not work.

Continued on Page 46

Graphics Printer or Epson FX Part 7

John A. Day
5 Rue Sauer
77500 Chelles
France

Programs for Downloading and Graphics on your Printer

This final article gives two programs for extending the possibilities of your 9-pin matrix printer. Both of them are written in BASIC, which has a good set of instructions for bit operations. It also happens to be the language I usually use. GW-BASIC is slow, particularly for high-resolution print graphics, but with a BASIC compiler (such as Quick BASIC) speed is acceptable. The program logic is quite simple, and can easily be converted into any other language capable of ANDs, ORs and NOTs. The first program is stripped down from a much longer version which I use routinely for designing characters in several different formats, but which would entirely fill a copy of REMark. The short version is limited to one format only, the standard Epson draft download. The second program was written especially to show how you can combine graphics and your own characters to build up charts. It is not intended as a serious working program; it is there rather to show you how a fairly short program can give good results. Once you've seen how it's done, you will have no difficulty using the same design principles for your own graphics.

The programs were written with publication in view, so as to fit on a (short) printed page. They can be speeded up if you want to modify the code, adding or lengthening a few lines where necessary. Since these are not production programs, a few errors may have slipped by. Please bear with me, and correct my bugs.

We shall be needing some instructions which are not often used in BASIC, so it will be best to go through them first. We are going to set up bytes which contain the positions of pins to fire for each column of our graphics, with bit 0 corresponding to pin 1, the top pin, up to bit 7 for pin 8. Pin 9 is not used in graphics printing. There are BASIC instructions which make this fast and easy:

OR — To set one or more bits without changing the others.

AND — To clear one or more bits without changing the others.

XOR — To toggle bits (replace a 0 with a 1 and vice-versa).

***** — Multiplication to shift bits left.

**** — Integer division to shift bits right.

None of these instructions work directly on a single byte. They are all designed for two-byte BASIC integers, but the CHR\$ function can work on the low-order byte and ignore the other. A DEFINT A-Z instruction at the beginning of your program will default all numeric variables to two-byte integers, which is what you need for most simple graphics.

We will be using masks quite a lot. These are bytes with certain bits set only. The weights for the low-order byte are:

bit 0: 128 bit 1: 64 bit 2: 32 bit 3: 16
bit 4: 8 bit 5: 4 bit 6: 2 bit 7: 1

So, if you want to mask just bits 1, 2 and 3, the mask value will be 64+32+16, or 112. Now, if BYTE is a two-byte integer where you are working in the low-order byte only, you can set these three bits with:

```
BYTE = BYTE OR 112  
and toggle them with:
```

```
BYTE = BYTE XOR 112
```

Clearing bits is a bit more complicated, because the AND operation clears all bits which do not correspond to the mask. $BYTE = BYTE AND 112$ would clear bits 0, then 4 thru 7 — that is, everything except bits 1, 2 and 3. Luckily, the inverse mask is easy to declare: NOT 112. So,

```
BYTE = BYTE AND NOT 112  
will clear bits 1, 2 and 3 and leave the others untouched. If your mask corresponds to a single bit, you will set, toggle or clear one bit only without touching the others.
```

Shifting is done by multiplying and dividing by powers of two:

```
2 shifts 1 4 shifts 2 8 shifts 3 16 shifts 4  
32 shifts 5 64 shifts 6 128 shifts 7
```

You must be careful not to shift too far left. We are working as though we had an unsigned integer occupying two bytes, while BASIC is manipulating signed integers with the leftmost bit indicating sign, and seven data bits only in the high-order byte. As long as we keep clear of the sign bit, everything is all right; and although NOT 112 is a negative value, it works properly because BASIC works bitwise to create it. But if we ever shift 8 bits left, we may try and shift into the sign bit, which will generate an overflow error. Shifting our value right by, say, 3 bits, is done by an integer divide:

```
BYTE = BYTE \ 8.
```

Shifting left by the same amount will work, but we may not want to move bits into the high-order byte, where they will stop the CHR\$ function from working — it bombs out for any value greater than 255. We can always mask off all bits not in the low-order byte by using its mask value, 255 (128 + 64 + ... + 2 + 1); remember that the AND operation clears everything not in the mask. So, we write:

```
BYTE = (BYTE * 8) AND 255
```

to get a shifted value confined to the low-order byte.

For all these operations, you can use hexadecimal values if you are used to them. This would give:

```
BYTE = BYTE OR &H70
```

```
BYTE = BYTE XOR &H70
```

```
BYTE = BYTE AND NOT &H70 or BYTE =
```

```
BYTE AND &HFF8F
```

```
BYTE = BYTE \ &H08
```

```
BYTE = (BYTE * &H08) AND &HFF
```

BASIC makes no difference in usage between these two ways of describing a value. If you have to use a mask in which the sign bit is set, like FF8F (-113 in decimal, which is impossible to guess), you will want to use hexadecimal notation. Otherwise, use whichever you prefer.

In the first sample program, lines 130 thru 170 set up a table of masks, *MSK*. *MSK(0)*, 128, corresponds to bit 0, and so on to *MSK(7)*, 1, which is the mask for bit 7. So, in line 1730 *T(C,J,L) AND MSK(M)* is bit *M* in the byte stored at *(C,J,L)* in Table T. In the same way, the line:

```
2480 T(C,XC,YCA) = T(C,XC,YCA) OR MSK(YCB)
```

sets bit *YCB* in the indicated byte. The easily-made mistake is to write *AND* to mean 'both', when in fact you want one OR the other OR both.

DEF SEG = &H40 makes a comeback. You will remember from Part 5 that this is the segment where BIOS remembers things like port addresses. We are now interested in byte 23, in which the bits correspond to shift, super-shift and lock keys. Bit 5 is the Ctrl key, Bit 6 is the left shift key, and bit 7 is the right shift key. So, *PEEK(23) AND 2* (line 2280) is the left-shift bit, and *PEEK(23) AND 1* (line 2290) is the right-shift bit; they are non-zero (respectively, 2 and 1) while the key is pressed, and flip back to "0" as soon as the key is released. They are used to control plotting. The cursor keys move the cursor; if left shift is held down, they plot and move, and if right shift is held down, they erase and move.

The expression *IF variable THEN ...*, such as:

```
1350 IF TD(I) THEN GOSUB 1400
```

may surprise you, together with *-1* as a 'true' value (line 1160). *variable* is 'true' if it is non-zero, that is, if any bit is set, and 'false' if it is zero, with all bits clear. *NOT variable* only works as you would expect for values 0 and -1. This is because of the internal microprocessor structure. Circuits for addition are smallest and most compact when using "2's complement" notation, in which a value is made negative by flipping all the bits and then adding 1. Since 8086 series processors are small and compact, this is the notation they use. *NOT*, however, is a logical operation, with no arithmetic, which just flips every bit. '1' is coded 0000 0000 0000 0001 over two bytes, so -1 is 1111 1111 1111 1110 + 1: i.e., 1111 1111 1111 1111. If you flip every bit, you get 0. If you take 0 and flip every bit, you get -1. If you don't believe me, call up BASIC and type *PRINT NOT 0*, then *PRINT NOT -1*. So, anything non-zero is 'true', but the only 'true' value that gives 'false' after a *NOT* is -1.

The first program maintains its fonts in MASM format, *DB 27,'&','0',n,a,b0,b1,...b10*. As an example, 'A' is:

```
DB 27,38,0,65,65,139,30,32,72,128,8,128,72,32,30,0,0
```

To assemble your font, use:

```
masm font1;          to assemble,
link font1;          to link,
exe2bin font1       to convert to a binary file
and copy font1.bin prn /b to download.
```

font1.asm should contain, besides title and page data:

```
font SEGMENT public 'DATA'
DB 27,'6' ; expand printable area
DB 27,'I',1 ; print codes from 0 to 31
DB 27,':',0,0,0 ; copy ROM to RAM
INCLUDE font2.asm
DB 27,'%'.1,0 ; select RAM
font ENDS
END
```

font2.asm is the output file from the font program. The 'copy ROM to RAM' instruction is necessary if you define less than a full 256-character font, and the 'select RAM' line lets you use the download characters immediately. It's better to *INCLUDE* the font data file, as my BASIC program can't update a file containing other Esc functions besides the download descriptions. You might need the */i* option on the MASM call to give an 'include' search path.

If you haven't got an assembler, you can easily write a Basic program to copy the font values to the printer (beware of any CR (0D) or ^Z (1A) bytes which could be trapped); it will just run slower than a binary copy. You can also maintain a binary file on diskette with *PUT* and *GET* instructions. The following sequence will open a file and write the preamble (*DB 27,6; DB 27,1,1; DB 27,:0,0,0*):

```
DATA 27,54,27,73,1,27,58,0,0,0
OPEN "R",#1,file.1
FIELD# 1,1 AS BYT$
FOR I = 1 TO 10
  READ N
  LSET BYT$ = CHR$(N)
  PUT# 1
NEXT I
```

For those of you not familiar with working in relative files, the '1' at the end of the *OPEN* means we are working byte-by-byte, and the *FIELD#* instruction links *BYT\$* to the file — notice that the variable name is not given in the *PUT#* instruction. Variables named in *FIELD#*s can be used freely to the right of an equals sign, but can only be used to the left of an equals in *LSET* and *RSET* instructions. Writing *LET BYT\$ = CHR\$(N)*, for example, gives no error messages but breaks the link between *BYT\$* and the file buffer. *GET# 1,n* will read byte *n* into *BYT\$*, and *GET# 1* will give sequential reads from the next byte on. Use the *DEBUG* program to check your output.

You can't modify the binary file with a word processor as you could an Assembler source, and if you overwrite an existing file, you can lengthen it, but not shorten it. To suppress one or two characters,

as easy, but you can't test for end-of-file. The function *LOF(n)*, or length-of-file, will give you the total number of bytes in file *n*, and you must stop when the program has read enough.

The font program would be easy if it weren't for nitty-bitty considerations. The basic principle is to have a 9x11 box on the screen, corresponding to the 9 print pins and 11 usable columns, and draw blobs wherever a pin fires. However, you must prevent the same pin from firing twice in rapid succession, so code is included which

- Limits blobs to alternate columns while tracing horizontal lines (variable *DBL*).
- Rubs out anything immediately left and right of a blob (lines 2690 thru 2750).

Also, you can only use 8 pins out of 9, so code at lines 2570 thru 2670 prohibits using pin 1 if pin 9 is in use, and vice-versa. Finally, you must provide proportional spacing data. A good guesstimate is to fix the width from the first non-blank column thru three blank columns to the right, to give 3/120" between letters. However, the beginning column must lie between 0 and 7, because only three bits are reserved for it in the attribute byte, and the maximum for the end column is 12, as 12/120" = 1/10". I've added a keystroke 'F' (or 'f') to fix any character at a full 1/10". I use this with semi-graphics characters which I want to mix in with proportionally-spaced text, without the graphics getting squashed up. If a font character is given as full width even though its real size is smaller, the program supposes it has been fixed at 1/10" — line 1830 sees whether a starting column of 0 (bits 1 to 3) is non-blank, and the following line checks whether the end column, bits 4 thru 7 (weight 15) figures with the last used column plus 3. When finishing off the character, line 2170 forces the attribute byte to 11 or 139 (0000 1011 or 1000 1011) if *TF*, fixed-width, is 'true'.

There are several character strings in the program which are used as easy tables. *CMV\$* is a list of characters which can't be shown on the screen, like 'line feed' or 'cursor up', because they will actually move the cursor instead of displaying. *INSTR(CMV\$,C\$)* (line 690) is a simple go/nogo test; if *C\$* is in the list of 'difficult' bytes, the function returns non-zero or 'true', whereas if *C\$* is ok, the function returns zero or 'false'. *S1\$* and *S2\$*, lines 100 thru 120, are more subtle. They give nu-

you would have to erase and rewrite the whole file. Reading the binary file is nearly

meric codes for the cursor keys, shifted and non-shifted, in order: 1-3 (top line), 4 and 5 (middle line), 6-8 (bottom line). Non-shifted gives two-byte keystroke codes, *S2\$*, whereas shifted gives one-byte codes, *S1\$* which includes also *Ff* to Fix a character width, and CR and Esc to end entry. The two tables are set up so that cursor keys have the same positions whether they are shifted or not. *INSTR* (*S1\$,Z\$*) gives the position of a shifted character in the string; *INSTR*(*S2\$,RIGHT\$(Z\$,1)*) gives the same value for a two-byte cursor character, knowing that in this case, the first byte is always zero. This can then be used (line 2350) to select cursor up (top line), left or right (middle line) or down (bottom line). For diagonal moves, line 2360 adds a left or right move after up or down.

To use the program, give your old data file name at the "Input file?" prompt, or hit *Enter* if you are creating a new file. You will receive the prompt "Character:". Give the character you want to redefine, either by hitting the corresponding key or by using *Alt* and the numeric keypad; the character and its ASCII code appear. The cursor, a small spot, will appear in the top-left corner of the box. Move it around with the cursor keys. Plot a point by pressing left-shift before you move. Erase the point at the cursor location by pressing right-shift before moving off. If you try to put two spots adjacent to each other, the first one is rubbed out. A "beep" means you are trying to use all 9 pins, which is not possible when coding on one byte. Toggle the "Fix" indicator with key *F*; when the indicator is not visible, proportional width is fixed automatically. To completely clear a character, toggle "Fix" to not visible and erase all points. Hit either *Enter* or *Esc* to finish a character. When you're through, reply *Ctrl-Enter* (or *Ctrl-Esc*) to the "Character:" prompt. The last prompt, "Output file?", is for the save file. Hit *Enter* to abandon your changes. Output is in MASM format, listed by ascending ASCII code.

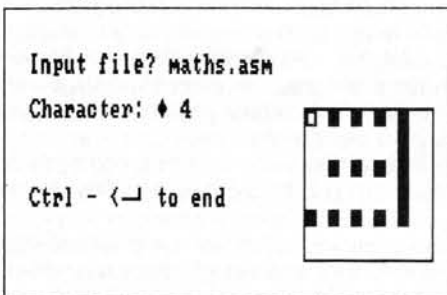


Figure 7-1
Sample Screen Display for
Designing a Special Math Character

Using the *Ctrl-Enter* sequence to stop work allows you to use *Enter* as a character key, entering download data for 13 (0D). This may seem odd — how can you use CR as a printable character, when it

means "Carriage Return"? Well, you can't use it directly, but... I explained in the first article (June 1989) that 12 characters (11 in the article plus *I* that got left out) can be re-assigned to foreign languages. The pin data for these characters has to be stocked somewhere, so Epson slipped it into the zone 0 - 31 (and 128 - 159 for italics). When you select "Denmark" with *Esc "R" 4*, your printer will use the character 13 (Å) instead of character 93 (J), and so on for six other substitutions. If you've downloaded a new character 13 and selected Character Set 4 and RAM, you get your own redefined character, but for ASCII code 93. The mappings used are:

| | | | | | |
|-----------------------|------------------------|-------|------|-------|-------|
| US: 35=# | 36=\$ | 64=@ | 91=[| 92=\ | 93=] |
| 94=- | 96=' | 123={ | 124= | 125=} | 126=- |
| à (0) → 64 F, 123 I | Š (16) → 64 D, 93 F | | | | |
| é (1) → 125 F, I | ß (17) → 126 D | | | | |
| ê (2) → 96 I, 124 F | Æ (18) → 91 DK, N | | | | |
| ø (3) → 124 I | æ (19) → 123 DK, N | | | | |
| ï (4) → 126 I | ø (20) → 92 DK, N | | | | |
| ° (5) → 91 F, I | ø (21) → 124 DK, N | | | | |
| £ (6) → 35 GB | ˆ (22) → 123 SP, 126 F | | | | |
| ı (7) → 91 SP | Å (23) → 91 D, S | | | | |
| ı̇ (8) → 93 SP | Ö (24) → 92 D, S | | | | |
| Ŋ (9) → 92 SP | Ü (25) → 93 D, 94 S | | | | |
| ñ (10) → 124 SP | ä (26) → 123 D, S | | | | |
| o (11) → 36 S | ö (27) → 124 D, S | | | | |
| ř (12) → 35 SP | ü (28) → 125 D, 126 S | | | | |
| Å (13) → 93 DK, N, S | É (29) → 64 N, S | | | | |
| å (14) → 125 DK, N, S | é (30) → 93 I, 96 N, S | | | | |
| ç (15) → 92 F | ÿ (31) → 92 J, 123 F | | | | |

D: Germany, DK: Denmark, F: France, GB: Britain,
I: Italy, J: Japan, S: Sweden, SP: Spain
N: Norway, some emulations only

Figure 7-2
National Character Set Mappings

Most of these characters can be printed either directly with their own ASCII code, by using *Esc "I" 1*, or indirectly with their transposed code and the *Esc "R" n* national sequence. Those marked with a vertical bar (|), however, can only be printed by international mapping (ß, DC1, should be available but often isn't).

If you want to see whether your MS-DOS is trapping end-of-file markers, try three of these international characters:

```
LPRINT CHR$(27)"I"CHR$(1);CHR$(28);
CHR$(25);CHR$(28)
```

If you get "üÜü, ^Z is being printed ok. If your printout reads "üü", then ^Z (25) characters are being suppressed, and you should consider other ways of accessing your printer.

Now, what can you do with the download file? First, you may be working on a document which uses a few characters which aren't on a standard printer — like three points, or a backwards E, for math text. You just choose some character you don't use, and redefine it to what you want, including the italics forms (ASCII code + 128) if required. The standard character shows on your screen, but the modified character prints in its place. This is how I did the printer pin diagrams in this series of articles, with high, low and double blobs on some of the IBM math characters. Since I'm using an IBM character set for this article, I had to do the same

to get π , ϕ , Φ and $\cdot\cdot$ in the character mapping table, because these are only available in the Epson sets. As an example, the following little program redefines three characters, 3, 4, and 5 ASCII (at lines 50 thru 70), and then prints two math formulae. *I\$* and *U\$* set italics on and off, *G\$* and *N\$* control emphasized bold print, and *S\$* and *F\$* handle subscripts. The *DATA* came straight from the BASIC font design program (see Figure 7-1, which gives the screen display corresponding to line 60):

```
10 REM Demo: maths expressions
20 REM John Day, Oct 1989
30 DEFINT A-Z
40 DATA 27,58,0,0,0,27,73,1,27,54
50 DATA 27,38,0,3,3,139,240,8,36,2,32,2,36,8,240,
0,0
60 DATA 27,38,0,4,4,139,130,0,146,0,146,0,146,0,
254,0,0
70 DATA 27,38,0,5,5,139,56,68,146,0,146,0,146,0,
146,0,0
80 DATA 27,37,1,0
90 WIDTH "LPT1:":255
100 FOR I = 1 TO 65
110 READ Z
120 LPRINT CHR$(Z);
130 NEXT I
140 E$ = CHR$(27)
150 I$ = E$ + "4"
160 U$ = E$ + "5"
170 G$ = E$ + "E" + E$ + "G"
180 N$ = E$ + "F" + E$ + "H"
190 S$ = E$ + "S" + CHR$(1)
200 F$ = E$ + "T"
210 LPRINT "♦ "G$A"N$": ♥ x♦G$A"N$
" and ♥ y♦G$A"N$": x+x = 0, xy(x+y) = 0"
220 LPRINT "♥ x"SS"1"FS", x"SS"2"FS" ♦ "G$E"N$
": "I$"f"U$(x"SS"1"FS"*x"SS"2"FS") = "I$"f"
U$(x"SS"1"FS)"G$T"N$;I$"f"U$(x"SS"2"FS")"
```

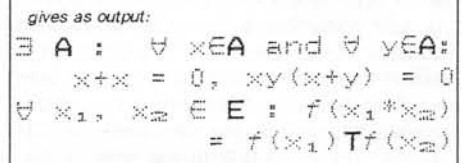


Figure 7-3
Using Downloaded Characters

This time, I've used *LPRINT* instructions as I know there are no troublesome bytes in the character redefinitions. For BASIC, I have to insert my own attribute control bytes for bold, etc.; with a word processor, you just use the standard selections (like I did for typing up the explanatory text). WordStar 4, WordStar 2000 and Word all accept most of the special characters not on the keyboard, such as the card suits I chose to redefine; you should be able to use almost any character with most modern word processors.

If you need a lot of redefined characters, you should build a completely new font, and include user code in your word processor to switch to RAM characters, *ESC "% " 1 0*, and back to ROM characters, *ESC "% " 0 0*. You can usually use "red" and "black" ribbon commands for this. Then, ribbon changes will switch you from one font to the other, allowing you about 200 upright characters instead of the hundred or so you get as standard. I have found this technique very useful with the Cen-

tronics Horizon series, now alas! obsolete, which could handle IBM GP emulation and downloading at the same time. With my setup, "Black" gave semi-graphics characters in ROM for outlining boxes, while "Red" gave me the extra characters that I had loaded into RAM. Usually, you would define italics versions of your character set from 128 on; but nothing stops you from using the italics space for upright characters unconnected with the standard (0-127) space, which gives you 300 different usable characters . . . who could need more?

You will have noticed in the previous articles that the FX emulation has a lot of useful things which a strict GP printer doesn't give, such as 12 cpi printing for Elite characters, and Esc "j" n for roll-up/roll-down superscripts and subscripts (tricky! This one was in the FX-80 specs when running without the optional forms tractor, but was never available on the original FX-100. See Jack Uretsky's article on Fancy Word, in the Oct '89 REMark). My GP/FX printer is always switched to Epson emulation, with all the IBM characters downloaded in place of the italics. Well, nearly all — you can't download the semi-graphics characters, which are 12-pin and not 9-pin. But I have defined similar eight-pin characters in the same code places. Four-pin descenders, which replace some less-useful characters, slot into place with a roll-down/roll-up. Your word processor will handle this bastard combination if you select an FX printer driver, then disable the character compose table with a user patch.

Finally — object of the next program — you can handle character generation yourself by including your characters in graphics data set up by your processor. The only difficult part of this job is getting the pin patterns, and the first program does that for you. I've chosen bar charting as an example. I wrote a bar chart program some years back for course planning, using the standard GP character set for drawing the boxes. I never really like it; I was tied down to the standard pitches of 17, 12 or 10 units/inch, I could only use vertical pitches of 2 bars per inch because each box took up at least two lines plus a line space, and it was difficult to include text inside a bar. But at that time, I didn't have character drawing routines at hand. Once this article is finished, I shall be looking at the old program so as to completely re-do the graphics side. The only program logic not covered in this article will be how to cover date formats — how to set up a page header with a section of the calendar, and how to input beginning/end dates in a text file. The rest is right here. With graphics data, you define a space — 960 × 16 dots, for example, if you use two print lines on 8" wide paper — and just do what you like inside it.

In our case, we are going to draw bar

charts at 4-1/2 bars per inch (two lines of 1/9" each), with the bar length written inside each bar. The characters are going to be half-and-half across the two lines, just to show how easy it is; we shall print the top 5 pins of the character on the first line, and the remaining 3 on the second. The text is proportionally spaced; this, mixed with fixed-pitch semi-graphics characters, as you would to bar-chart in text mode, would drive any respectable word processor crazy.

The program reads a text file *BAR.DTA*, with a series of values to plot. The page width is fixed at 6-1/2", plus 1" left margin. The scale is 1" for 10 units, so we can handle up to 65 as a maximum value. The program prints the bar to a precision of 1/120", corresponding to increments of 0,08 in the value plotted. Even with proportional spacing, checks on overflow and overprinting, and I got the program down to 110 lines (by stripping out printer status checks to keep it short — if the printer's not ready, the program hangs).

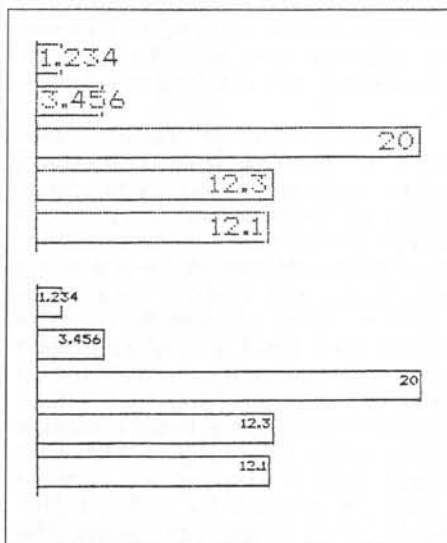


Figure 7-4
Sample Bar Charts: (top) Normal
Density, (bottom) High Density

The *DEFINT* leaves the letter V for decimal values, because our bar length is not necessarily an integer. Everything else is. The *DATA* was obtained from the other program, and contain attribute and data bytes for the ten digits 0 thru 9, together with a period in position 10. *L1* and *L2* are the first- and second-line print buffers, for 781 bytes. Line 260 picks up the printer port address for direct output. It's unsafe to use *LPRINT* instructions, because any byte containing 13 will be trapped by BASIC, and this will blow the byte count for the Esc Y command. Try *LPRINT#* on a "relative" print file if you like, but check what's happening to 25 (^Z). I prefer to play safe.

If the length works out to more than 6-1/2" (value 65), *OFLW* is set so as to leave the end of the bar open. 255 is "all

bits set", which fires all 8 pins at lines 450, 460 for a vertical rule along the left of the paper. Lines 470 to 500 fire pins 2 (on the first line) and 5 (on the second line) for the horizontal lines. Note that only even bytes are set, corresponding to the minimum firing pitch of 1/60". If the bar is to be closed at the right-hand end (plotted value 65 or less), line 510 sets pins 2 thru 8 for the first line and 1 thru 5 (128 + 64 + 32 + 16 + 8) for the second. If W, the last byte to plot, is odd, this would give two adjacent bytes set for pins 2 and 5, which is forbidden; so the previous byte is cleared to avoid this. On some of the bars in the sample printout, you can just see a small gap at the end of the bar, where the last dot is on 3/120" centers instead of 1/60". You could use Esc "L" . . . and 1/120" centers instead of Esc "Y" . . ., but at the price of half-speed printing; and vertical resolution would still be only 1/72".

The printed value is right-justified if there's room inside the bar, left-justified otherwise, so we have to check the overall width — allowing an extra 5/120" to set the characters off a bit from the left rule. The last column in the character is given by bytes 4 thru 7 in the attribute byte TA(C), which we extract with TA(C) AND 15. The first column is in bits 1 thru 3 — AND 112 — which we must shift right 4 bits by dividing by 16. The width of the character is the difference between these two values, plus 1. If the overall width of the text goes beyond the end-line of the bar, line 570 blanks pins 4 thru 8 for the upper half, 1 thru 3 for the lower, so as to leave a gap for the digits.

This is one of the useful aspects of graphics printing; you can set up features in the buffers using simple algorithms, then come back and erase them before printing if you want to change your mind. For a production program doing resource planning, you would want vertical rules all across the page for weeks and days, except where there are boxes indicating resource usage. The easiest way to lay this out would be by drawing vertical rules all across the page in a simple loop, then blanking them while drawing the boxes. This would be obtained for each column with an *AND NOT* instruction with a mask to blank the unwanted part of the vertical rule, followed by an *OR* instruction to add the box element to any unblanked parts of the ruler line — you would want to leave in any of the ruler line outside the box.

The start column, *C1* at line 580, is set somewhere from byte 4 on, which leaves a 3/120" space (bytes 1, 2 and 3) after the left rule. The character data table was set up with 12 columns — *DIM T(10,11)* — to make an explicit zero byte available for column 12; this is implicit in the download data. With this extra zero available, lines 590 thru 660 just loop through the characters, moving each one

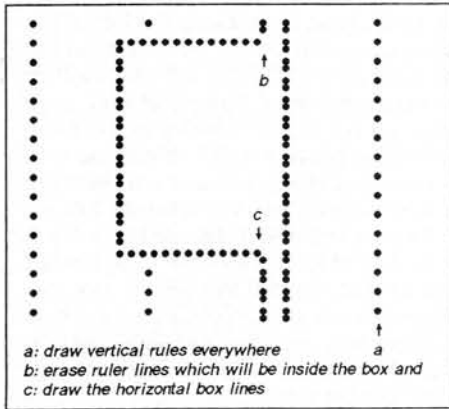


Figure 7-5
Stages in Drawing Boxes
Over a Vertical Grid

into the two halves of the bar. The OR instructions combine the existing contents of the buffer with the new bits, meaning that the characters will be added to the bar lines that are already there. Dividing by 8 moves the character down 3 bits, the bottom three bits being lost. When moving the character up 5 bits for the lower line, however, the top five bits must be stripped off (AND 7), otherwise, they will shift into the high-order byte and prevent the CHR\$ function from working.

Lines 1040 and 1060 are where you risk going wrong most. These give an integer value, over two bytes, telling the printer how many bytes of graphics to expect: Esc Y n1 n2 . . . n bytes of graphics . . . If you send more bytes than you declare, the excess will print as a few garbage ASCII characters. If, on the other hand, you send fewer — the printer will read the next control sequence as being the last few graphics characters, miss the switch from ASCII to graphics, and spew out most of the next graphics as garbage ASCII.

Line 890 could read Z = L1(I) (and Z = L2(I) at 760). This, however, might let data through which would fire pins too often. Your printer may be checking for this itself, in which case, you can use the shorter, faster code. If you use the BASIC interpreter, checking for adjacent pins will slow the program by about 15%; compiled BASIC will run at the same speed with or without checking, because it puts bytes on the port much faster than the printer can accept them and spends some time anyway waiting for "printer ready". Look in your manual at the description of "Esc Y". If it says, "Horizontally adjacent dots cannot be printed" (Epson FX manual), it's safer to eliminate them. If it says, "Adjacent dot columns are not printed" (Centronics Horizon and PrintStation manuals), you can test to see if the printer eliminates adjacent dots before printing. Send:

```
LPRINT CHR$(27)"Y"CHR$(9);CHR$(0);
CHR$(64);CHR$(0);CHR$(64);
```

```
CHR$(0);CHR$(64);CHR$(112);
CHR$(64);CHR$(0);CHR$(64)
```

keeping your finger on the power switch, and at the slightest abnormal sound, switch off. Look at what was printed using a magnifier.

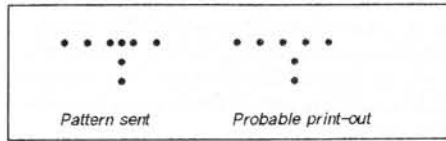


Figure 7-6
Eliminating Adjacent Pins

If the printout corresponds to the right-hand sketch, with regular spacing between points, then your printer is eliminating adjacent points and you need not code for this yourself. Otherwise, you must eliminate them in your program. The code given is over-cautious and will eliminate too many points, as it compares each column with what should have been printed previously, before checks. If you are using such checks to clean up intentional over-printing, modify to compare with what was really printed in the previous column, after checking and elimination.

Starting with these two basic programs, you can easily build up your own custom applications. Full alphanumeric printing on charts just requires putting a full alphabet in DATA — preferably indexed on the ASCII code — and taking pin 9 into account. If bit 0 of the attribute is not set, you should shift the character down 4 bits instead of 3, which I didn't bother with because I knew that none of the eleven characters I used for the demonstration require pin 9. Better resolution is given by Esc "Z" . . . and a 1/240" pitch instead of 1/120" with Esc "Y" . . . and a double pass for each line, with a 1/216" fine line feed, to give NLQ letters. This does multiply processing time by four, but gives good printout. I prepared sample output for this option, using the same pin layouts for the characters which gave me half-size letters. I defined four double-length print buffers, L1A, L1B, L2A and L2B; the B buffers are printed on a second pass 1/216" below the A buffers, using Esc "J" 1, and between 1 and 2 the paper is moved 23/216" instead of 24/216". The characters fit completely into the upper half of the bar, using the following sleight-of-hand instructions to move even character pattern bits (0 to 6) into L1A from pins 4 to 7 (bits 3 thru 6), and the odd bits (1 to 7) into the same positions in L1B:

```
L1A(C1) = L1A(C1) OR ((T(C,J) AND 128)
x 8) OR ((T(C,J) AND 32) x 4)
OR ((T(C,J) AND 8) x 2) OR
(T(C,J) AND 2)
L1B(C1) = L1B(C1) OR ((T(C,J) AND 64) x
4) OR ((T(C,J) AND 16) x 2)
OR (T(C,J) AND 4) OR ((T(C,J)
AND 1) * 2)
```

Half- and full-size characters are far from the only possibilities. I invite you to add code on to the first program to get the size of characters you need. The drawing box on the screen can be expanded to any reasonable size, and the square blob can be split into four or six for 1/240", two- or three-pass fonts. But you'll have to change the formats used to stock your characters on disk. The download format is fine for draft 9-pin characters, where a full font fits into 3K bytes. If you used the same layout for 18-pin NLQ characters, you would end up with a good 18K for fixed pitch — three passes at double the horizontal resolution — and worse for proportional fonts. Larger-size characters would take up an astronomical space. The trick is not to stock leading and trailing spaces for each character. Standard NLQ download formats have three attribute bytes for each character, the number of zero bytes (spaces) suppressed before and after, and the number of bytes of effective character data. They are usually arranged with three consecutive bytes for each print column, "top half", "bottom half", and "descender". For generating my own characters, I find it's simpler to stock all bytes for the first head pass, then for the second, then for the third, so that each set of bytes can be loaded straight into the print buffer. If the character takes up more than eight pins, I repeat the structure for succeeding print lines. Descender data is only stored if it actually exists, as I suppress trailing lines of zeros. This is not convenient, however, for drawing on the screen. The easiest I have found is to use one byte for the same pin/column combination on two or three passes. This way, moving vertically on the screen takes you down through the byte for the dots which are handled by a particular pin, albeit on different passes, then changes bytes as you change pins. "4" can indicate "fire on pass 1", "2" "fire on pass 2", and "1" "fire on pass 3", with all the combinations possible, such as "7" "fire on all three passes". It is surprisingly easy to convert from one format to another, once you've gotten into the habit. A conversion program sweeps through the destination format byte-by-byte, picking up the corresponding bits wherever they are in the source format. With a bit of practice, you can even pick up character outlines from laser fonts, which are in a fourth format, reading horizontally with all the points for the top raster line, then everything for the second raster line, and so on.

All the figures in the previous articles in this series were printed in this way. The characters were drawn on the screen in maximum resolution, three passes at 1/240" pitch, and stocked on disk. The figures were prepared as ordinary text using WordStar 2000 and output to disk using a custom print driver. They were then

OS/2, Unix, the ZDS Group Sale, ZDS and QED, the eaZy PC

William M. Adney
P.O. Box 531655

Grand Prairie, TX 75053-1655

Copyright © 1990 by William M. Adney. All rights reserved.

Based on the responses I received when I asked for any comments you had about OS/2, it is quite clear that virtually all HUG members seem to be as skeptical as I am about this new operating system. Even though some of you told me that you used OS/2, most of those responses indicated that DOS was still your primary operating system, and only one user said that he used OS/2 as the only operating system for his computer. As I said last year, I do not intend to write about something that seems to be of little interest to most HUG members, but there is little doubt that most computer users are quite reluctant to forge ahead with OS/2 for a lot of reasons, even though there is some outstanding software that is now available, such as HyperACCESS/5, I mentioned last month.

DOS is Dead?

Don't believe that for an instant. By most estimates, there are now some 30 or 40 million microcomputers in users' hands, and fully half of them cannot run OS/2. Why? Simply because they don't have a CPU (Central Processing Unit) that is capable of running OS/2. One of the OS/2-specific hardware requirements is that you must have at least an 80286 CPU, so that is a specific limitation resulting from the operating system. One of the problems associated with OS/2 is that hardware requirement which is a direct result of this operating system being able to take advantage of some of the features of 80286 and later CPUs, such as multi-tasking. Older computers that are based on the 8088 CPU just do not have the "power" required to run OS/2.

In any case, DOS is absolutely NOT dead, no matter how much Microsoft and IBM would like to "kill" it. There are many reasons for that, and the obvious one is that OS/2 has not done well in the mar-

ketplace — at least not nearly as well as Microsoft had hoped and expected. Users are not standing in line to buy OS/2, and it is appropriate to examine some of the most likely reasons for that.

OS/2 Obstacles

For those of you unfamiliar with my background and experience, I spend a considerable amount of time doing computer consulting in addition to writing. As a result, I spend a lot of time working with large companies (mostly in the Fortune 100), and your reaction to OS/2 is typical of what I see. There are some good reasons to change over to OS/2, but I find that very few people in these companies have sufficient justification to do so.

Of those I find who have changed to OS/2, most tell me they have done so because they believe OS/2 is the "operating system of the future." Part of that rationale is that they want to "be familiar with OS/2 when it replaces DOS." Personally, I do not believe that OS/2 is necessarily the operating system of the future, at least not yet. Why? Well, there is convincing historical evidence that OS/2 is not likely to do very well until something happens. And that "something" is usually called a *killer application* that "all" users believe they MUST have.

A Brief History

Historically, the first killer application usually recognized as having any significant impact on the microcomputer market is VisiCalc — an electronic spreadsheet that was developed for the old Apple II computer running an operating system generally referred to as AppleDOS. This occurred in the late 1970s.

The second killer application was WordStar, which ran under the CP/M operating system. WordStar was originally released in 1979. The real reason it is usually

recognized as a killer application is simply because it was the first real word processor available for a microcomputer. The only real alternative was the commonly available stand-alone word processors, such as the Lanier, that cost in the \$15,000 to \$20,000 range. One could generally buy a CP/M computer, printer, and software (including WordStar) for under \$4,000 at the time. And contrary to what was stated in the article on "The Evolution of DOS" in the December 1989 issue (page 43), Gary Kildall did NOT design the CP/M operating system for an Apple computer. The original CP/M operating system was designed on an Intel 8080 CPU, not the 6502 (used on the Apple I and II).

Nearly all computer users have at least heard of the third and most famous killer application: Lotus 1-2-3. Lotus 1-2-3 was the application that really caused the sales of the IBM PC to skyrocket because many business managers saw how the "what-if" features of the electronic spreadsheet could be applied to solving business problems. As a result of entering the microcomputer market, IBM was able to get businesses to recognize that a small computer was an acceptable tool that could be used to help solve business problems, and Lotus 1-2-3 was the specific application generally recognized for that purpose. Indeed, one could develop a rather convincing argument that the IBM PC was not a commonly accepted business tool until after the introduction of Lotus 1-2-3 because that is when sales of the PC really began to take off.

As you can see, it is the killer application that really drives the purchase of new hardware and additional software, such as new operating systems like the DOS used on today's microcomputers. Regardless of its capabilities, no operating system for a microcomputer has been so great that it has been the ONLY reason for people

buying new hardware, not to mention application software. People buy microcomputers to USE for some specific purpose, such as word processing, spreadsheets, and games. Most people could care less about what the operating system is, so long as the computer can perform whatever task is required. And that is one of the problems with OS/2.

Back to OS/2

I believe that the lack of a killer application for OS/2 is probably the major reason that it is not doing as well as some industry pundits and vendors had anticipated or predicted. Even though OS/2 was released nearly two years ago, there is still not a whole lot of application software that has been released specifically for OS/2. Most OS/2-specific software that has been released is even more expensive than its DOS equivalent, and that is another general problem that has helped cause the lukewarm reception.

Regardless of any other problems, OS/2 is expensive. As previously mentioned, it requires at least an 80286 CPU. But even if you have that, OS/2 is still expensive. First, you must have a hard drive just to run OS/2, even though most 80286 and later systems usually have a hard drive. To run the current version of OS/2 with Presentation Manager (like Windows), you must have at least four megabytes of extended memory (NOT expanded memory), and more is recommended. Even though the cost of memory is finally coming back to more reasonable levels, memory is still expensive. Buying OS/2 as an operating system is inherently expensive because the version 1.1, which includes OS/2 and Presentation Manager, is in the \$500-700 price range. The "Extended" version of OS/2, which includes some application software (e.g., a data base with SQL and communications), is rumored to have a list price of nearly \$1,000. By comparison, the latest ZDS MS-DOS version 3.3 Plus has a list price of \$149. If you really need communications software, I think you are far better off using HyperACCESS/5 under DOS than anything available as part of OS/2. And there are far better data base software packages you can buy that run under either OS/2 and/or DOS. In terms of cost, you can generally expect that OS/2-specific software will be more expensive than its DOS equivalent, primarily because it takes more time to develop it. There is also another element of the cost of OS/2 that you also need to consider.

Let's say that you already have an 80286 or later system with a hard drive, an EGA video display, and five megabytes of extended memory. Let's also assume that you have the usual variety of application software that includes a word processor, spreadsheet, at least one desktop utility that is memory-resident, at least one disk

utility package (e.g., Mace Utilities or Norton Utilities), communications software (e.g., an older version of HyperACCESS), and a few games. Perhaps you even have a data base package like FoxBase or PC-File:db. Now you want to know what runs under OS/2 and what does not.

OS/2 has two modes, usually referred to as the REAL (or DOS "compatibility box") mode and the PROTECTED mode. Based on the assumption that all of the old software was purchased for the "old" DOS, you can only run some software in the OS/2 real mode. In other words, the old DOS software cannot take advantage of any of the OS/2 features because it does *not run* in the protected mode. Specifically, you cannot use the OS/2 multitasking feature because the old DOS software was not designed for it. If you really want to use OS/2's multitasking, you must buy another word processor and spreadsheet that was specifically designed to run under OS/2 or you must at least check with the manufacturer to see if you can upgrade to an OS/2-specific version of that particular application. Even for well-known DOS applications, there are still not a lot of OS/2-specific versions available yet. Obviously, that will add to the cost of implementing OS/2 when more major applications are available. More importantly, it turns out that there are four "minor" categories of general DOS programs that will not currently run under OS/2 in any mode.

The first category of programs that will not run under OS/2 includes any memory-resident or TSR (Terminate-and-Stay-Resident) program that was developed for DOS. This category includes such things as print spoolers, keyboard enhancers, notepads, disk managers, menu software, popup calculators, help programs, mouse drivers, screen savers, and so on. Although I have heard some unconfirmed rumors that Microsoft is trying to make the OS/2 real mode more compatible with DOS programs, the technical problems are formidable because OS/2 uses a completely different memory management scheme (to implement multitasking) than DOS does. In short, you can plan on spending a bunch of money to buy new memory-resident programs that will run under OS/2.

The second category of programs that will not run under OS/2 is usually referred to as disk or file utilities, such as the DOS-based Mace Utilities or Norton Utilities. This category includes virtually any program that requires what is usually referred to as low-level disk access. Programs that usually require low-level disk access include sector editors, hard disk "optimizer" or defragmentation programs, undelete and file recovery programs, and some disk caching programs. These programs will not work under OS/2

because of a different file management scheme that Microsoft calls the "High Performance File System" or HPFS. In contrast, DOS uses what is currently referred to as the "FAT (File Allocation Table) File System." When I first heard about that, I could not help but wonder if the DOS file system was a "low performance file system" because I have mentioned how fragile it is in various articles. The important point is that most of the programs in this category perform low-level disk access, so these programs will simply not work with OS/2 because its file management scheme is different. I should note that OS/2 does allow you to read and write floppy disks that were formatted with DOS, so you can still use old application software and "old" disks in the OS/2 real mode. This restriction only applies to programs that use low-level disk access techniques, and it applies only to disks formatted under the protected mode, such as a hard drive.

The third category of programs that will not run under OS/2 is all communications software that was developed for DOS. DOS-based communications programs will not run under OS/2 because these programs require direct port access because of their nature. That is, any program that bypasses DOS and goes directly to the computer hardware will not run under OS/2 in any mode because of the changes in the way OS/2 handles hardware ports, such as required for communications software to talk to a modem. Although I have defined this third category as being communications software, it really includes any software that uses direct port addressing. Some print spoolers and mouse drivers may also fall into this category too, but they were previously mentioned in the memory-resident program category. I have specifically discussed communications programs as a third category because this software is generally considered a major application. Fortunately, Hilgraeve has developed HyperACCESS/5 that includes both DOS and OS/2 versions that eliminate this kind of problem.

The fourth category of programs is slightly different than the first three. Some forms of copy-protected programs (e.g., games) will not run under OS/2, depending on how the copy protection is implemented. In general, any form of copy protection that uses any memory-resident programming, unusual disk format tricks or direct port access will not run under OS/2. In particular, I have been told that this restriction specifically applies to ALL software that uses a hardware "key" (called a dongle) that typically plugs into a serial or parallel port. A few major applications, such as some CAD and accounting software, use a dongle for copy protection, and because these applications were designed for DOS, they will not run under

OS/2 because the application cannot access the serial or parallel port.

Other General Problems

From what I see and hear, cost is a major factor in the decision of moving to OS/2. The lack of a wide variety of popular OS/2-specific application software is also a significant problem because virtually all of us frequently use one or more applications in each of the four "problem categories" mentioned above, and equivalent software is not yet available for OS/2. That will undoubtedly change, although many software manufacturers are apparently reluctant to commit financial and programming resources to develop new software for an "unproven" operating system . . . which leads to another current problem.

Despite everything else, OS/2 is not yet a stable operating system or platform for software development. Many of the tools required for software development, especially those required to develop applications for Presentation Manager, are not even close to the sophistication of those available for DOS. And because OS/2 is still somewhat unstable because it is not fully "developed," many software manufacturers are also having problems developing good OS/2 applications without bugs. Obviously, that leads to a vicious circle, which is where OS/2 and its applications are today. How soon that will change is anyone's guess. One question remains: "Are there any real alternatives to OS/2?"

OS/2 Alternatives

If you really think you need most, any or some of the advantages of OS/2 on a microcomputer, there are two viable alternatives available today. The first, and most obvious alternative, is the Unix (sometimes still referred to as Xenix for a microcomputer) operating system. The second alternative is to stick with DOS and use the incredible DesqView software that provides most of the advantages of OS/2 without the cost.

Unix is becoming more and more popular for lots of reasons, including the fact it has been around for a long time and is far more mature (i.e., stable) and fully developed than OS/2. Moreover, Unix has virtually all of the key features that OS/2 is supposed to have, not to mention some things that OS/2 will probably not have. Are you looking for a multitasking operating system? Unix and Xenix have had that for years, and they are quite reliable. Do you need a multiuser operating system? Unix and Xenix have had that for years too. Are you looking for an operating system that is essentially compatible with all kinds of hardware platforms, from mainframes to microcomputers? Unix runs on an incredible variety of computer hardware and has for a long time. Even

IBM has announced that Unix will be fully supported in their mainframe computers, not to mention any of the other extremely popular computer systems such as the DEC VAX series of minicomputers and microcomputers. Virtually all of the clients I see have a considerable investment in VAX systems (usually running Unix), and the trend seems to be to replace large mainframes with smaller VAX minis for any one of a number of business reasons, including cost. If Unix is so great, why don't you hear more about it for microcomputers?

As you might expect for any operating system that has both multitasking and multiuser features, Unix takes a lot of hardware, just like OS/2. Moreover, Unix is sometimes said to be difficult to learn, but that is rapidly changing now with the development of additional "shells" for new users. More and more application software runs under Unix, although it is still not even close to the wide variety that is available for DOS.

No matter how much some manufacturers would like to kill DOS, I don't see it happening in the near future. By the end of the century perhaps, but not in the next couple of years because there are too many computers which cannot run OS/2. I think it will also be at least another year before OS/2 becomes stable enough before many users will want to trust it as a production operating system.

The ZDS Group Sale

By now, I'm sure that all of you have heard about the sale of the "ZDS Group" to Groupe Bull. For those of you not familiar with the term "ZDS Group", it includes three different organizations: Zenith Data Systems (ZDS), Heath Company, and Veritechnology Electronics Corporation (which operates the Heath/Zenith Computer Centers). I have purposely stayed away from any comments about that because there was not much information available that seemed to directly affect ZDS computer users, and from what I can see, that still seems to be true. To date, there have been some minor "administrative" kinds of changes that still do not affect ZDS computer users, but there is one kind of change that you may have noticed. That change directly affects *REMark* in terms of how we use descriptions of various items, and you will see those changes in my columns and other articles in this magazine.

Perhaps the most important point is that the word "Zenith" (with the jagged Z) is a trademark of Zenith Electronics Corporation, which means that "Zenith", by itself, no longer applies to any of the items, including computers, produced by the ZDS Group. Because that affects everyone who writes an article for *REMark*, let's take a look at the practical implementation of how this works.

When referring to a very specific product manufactured by the ZDS Group, such as MS-DOS, one can call it "ZDS MS-DOS", "Zenith Data Systems MS-DOS", "Heath/Zenith MS-DOS" or "H/Z MS-DOS". Computer models, such as the Z-386/16 or Z-386/33, still retain the "Z" because that is part of a model number, but it is also correct to refer to a ZDS or Heath computer or other hardware with an additional "name" as previously mentioned, such as ZDS Z-386 or H/Z-386/25. If you have been a member of HUG for a while, you may remember that I mentioned a long time ago that I thought a model number like H/Z-386/25 was somewhat confusing, so I normally have used the ZDS assembled model number in the form of Z-386/25 to indicate both kit computers (from Heath) and assembled computers. And I usually refer to my own Heath 386 system as a Z-386/16, even though I assembled it as a Heathkit.

If you have already submitted articles using the other terminology, they will be edited to include this change, so you don't have to worry about that. You can make the production, typesetting, and editing process a little faster if you include the correct terminology in future articles.

QED

Perhaps you are wondering what the Latin QED (quod erat demonstrandum) has to do with Heath/Zenith computers. According to my Webster's New Collegiate Dictionary, QED means that "which was to be demonstrated." For all three companies in the ZDS Group, it has another meaning: Quality in Everything We Do. This process was introduced in April to help improve quality throughout the ZDS Group and its products. Art Lambert, Vice President of Sales and Marketing, is chairman of the company-wide QED Steering Committee. He says: "Right now, we are a hassled company. Every employee and customer probably has at least one problem that they wish 'someone' would fix. If we identify and develop ways to prevent those problems from occurring, we will improve our overall quality."

Based on many of the letters I receive from HUG members, there is no doubt that his statement is true, at least from the customer point of view. In fact, he has also noted that: "We want and need to become a hassle-free company, one that . . . enables customers to have confidence in us and the quality of our products and services." I believe that he is quite aware of the poor image that ZDS products have in terms of service and support, despite the best efforts of some dealers, notably the Heath/Zenith Computer Centers. For example, some hardware reviews of ZDS products in other magazines consistently mention the lack of factory-direct technical support. That causes a problem for

many users, either because they do not have a local Heath/Zenith dealer or because the question (usually technical) is far beyond a dealer's knowledge.

Perhaps the most interesting point about the QED process is that it is one of the few times I have seen a company come to grips with and admit this kind of problem exists. This is a "known" problem that most companies have to a greater or lesser extent, and I find it quite refreshing that ZDS is taking a proactive stance in trying to solve it.

Another interesting point is that the QED process involves virtually everyone, both customers and employees. If you have ever worked in a large company, you have no doubt run into "trivial" problems, such as finding office supplies, getting people to answer memos and return phone calls, and generally getting other people to do their jobs so that you can do yours, especially when they do not work directly for you. I have never worked for ZDS as an employee, but I would be surprised if at least one of these problems needs some attention. And that is also the focus of the QED process. It is designed to help solve both the internal and external types of problems.

As a long-time Heath/Zenith computer user, I am quite encouraged to see that high-level management attention is being focused on the general problem. I am quite optimistic that this program will help be successful in improving the general level of ZDS support for all of us. And I would like to present some constructive suggestions that I have noted over the past seven years that I have written this column.

Some Suggestions for ZDS

In light of the QED process, as well as the purchase of the ZDS Group by Groupe Bull, I think it is appropriate to mention some suggestions for improvement of ZDS computers. You may recognize the first one as having appeared in this column before.

A hardware reset switch. I think it is extremely important that ALL computers, including laptops, have a hardware reset switch. Software continues to get more sophisticated and complex, and it is a rare user who has not had to power-off a computer because of a HARD system freeze which will not respond to a CTRL-ALT-DEL reset. That happens because some kind of software problem or conflict causes a keyboard lock-up, and the keyboard will not respond to any command, including CTRL-ALT-DEL. The only alternative is to power-off the computer, which effectively resets the hardware of course. If you have a hard system freeze, you should power-off the computer, wait at least 10 seconds, and power it on again. Regardless of what anyone says, I do not like to run a power off/on reset because

of the possible impacts on my hard drive, among other things. Besides, a hardware reset switch is a trivial and inexpensive addition to a computer, especially for a manufacturer. To give you an idea of how inexpensive it is, I spent less than three dollars to add a hardware reset switch to my Z-386/16. Besides, many of the VERY cheap compatibles have a hardware reset switch, and there is no reason to expect less from a premium-quality system.

On the other hand, a SOFT system freeze will respond to a CTRL-ALT-DEL (or CTRL-ALT-INS for most ZDS computers), and that is typically the only command which will work. Other CTRL-key, ALT-key, and Function-Key commands are "frozen" and do not work. I should note that you cannot tell whether you have a soft or hard system freeze until you try to use CTRL-ALT-DEL. If it works, then it is a soft freeze. If it doesn't, it is a hard freeze which requires a power-off reset. In either case, a soft reset (with CTRL-ALT-DEL) or a hard reset (by powering off) will destroy anything in memory that has not been saved to disk, so you will lose whatever work has been done since the last "save" command.

The system ROM. So far as I know, ZDS was the first manufacturer to incorporate the SETUP command in ROM, and that has now become the standard rather than the exception. In case you did not know, the original IBM ATs (both the 6 MHz and 8 MHz versions) had a special disk with a SETUP command that was like any other DOS command. I've had some difficult times working with these IBM computers because NOBODY knew where the Setup disk was if the computer was more than three days old. In any case, I am beginning to wonder if it is a good idea for ZDS to continue extensive ROM development when there are lots of good third-party ROMs available, such as Award and Phoenix.

Make no mistake about it, I think that ZDS ROMs are great because of the neat features, such as the ROM debugger, TEST command, SETUP command, and the ability to boot from any drive or hard disk partition. But it seems like it could be more cost-effective to buy standard ROM code, such as the Award or Phoenix that I mentioned, and add some of the usual ZDS features to it. Again, I recognize that this is much easier to say than to do, but it might help reduce some of the problems. And who knows, it might also help reduce costs to the point that ZDS prices would become more reasonable and competitive, which is another major point.

Prices. This comment will no doubt generate a lot of interest, but I believe that ZDS needs to seriously evaluate their pricing structure to bring it more in line with today's market. Many of you have written to me stating that you would like to buy a more current ZDS computer,

such as an 80386 system, but it is difficult to justify \$4,000 for a ZDS computer when you can get a similar 80386 system for about half that. When one is considering that kind of price differential, it is difficult to cost-justify buying a ZDS computer, no matter how good it is and regardless of what features it has. For example, the April 10, 1990 issue of *PC Magazine* (page 254) contains an advertisement for a well-known brand of 80386, 33 MHz computer with prices ranging from \$3395 to \$5995, depending on the configuration. One comment I have also heard is that businesses are willing to pay more for a computer, but that is not true so much any more. Most businesses, as well as individuals, are now buying computers based on price as much as anything else. Several people in the ZDS Group have told me that ZDS prices are higher because there is more support available, and you have to pay for that with higher prices. Nuts! There are lots of people who disagree that ZDS support has been that great, which is another issue.

Service and support is another big issue. When the ZDS Consultation Group was discontinued, that was done because it was supposed that an authorized dealer could provide whatever support was needed. Unfortunately, there were lots of "unauthorized" dealers (e.g., discount computer stores) that were selling ZDS computers, especially laptops, at prices far below what an authorized dealer could even come close to matching. Unfortunately, these "unauthorized" dealers could not provide much support (if any), so the chore fell upon the authorized dealers who quite reasonably felt some resentment at having to support computers they did not sell. Many of these "unauthorized" dealers also have no on-site service facilities whatever, so any warranty service also fell on the shoulders of the authorized dealers. This kind of situation has inevitably caused a large number of problems, but ZDS has already started to deal with at least one of them.

ZDS has started the Medallion program, which basically requires that all dealers sell ZDS computers "locally" — in other words, NO mail order. That is a good start and will eliminate the problem of "unauthorized" dealers who cannot support ZDS computers. This applies ONLY to assembled ZDS computers and does not affect the Heathkit systems that are still available by mail order. The apparent basis for the Medallion program is that the authorized dealer is available for support, but that is somewhat questionable in my mind. The problem is that many of the authorized dealers I know of only have just a very few people who really know enough about computers to answer detailed questions about them. Training is a problem, and that takes time and some technical skills. For example, many sales

representatives don't have the foggiest idea of what the differences are between MFM, RLL, ESDI, and SCSI hard drives. They cannot even answer a basic question about them, let alone discuss the advantages and disadvantages of each type. How many sales representatives can explain the differences between extended and expanded memory? Do you know any sales representative who can explain the REAL difference between the 386SX and an 80386 CPU, and what that means to a user? As I said, training is one solution to the problem, and it will take time.

Another way to help fix the service and support problem is to start up a direct support line, like the old ZDS Consultation Group. There are lots of objections to this, but even Compaq has been forced to do it, despite the support provided by their dealers. Regardless of how good a training program is for authorized dealers, there is still a considerable need for detailed information that is generally not available from anyone else but the manufacturer.

HELP — For the eaZy PC

I continued to get letters from eaZy PC owners asking about upgrades for their computers. The basic eaZy PC includes 512 K of RAM, a parallel port, and a "mouse" (COM2) port; but it does not include a standard serial COM1 port. I have suggested that the Modem/Memory Expansion/Serial Port module is a necessary addition to this computer, but I have not been able to find anyone who knows where to buy it. Although many users may not need the modem, most are finding that they need 640 K of RAM, which can only be added by using either the 128 K Memory Module or the combination module. A standard serial port that can be used to drive a serial printer is only available in the combination module, accord-

Continued from Page 6

to buffer and disk, and transmitting files from disk to the TNC. It's still usable as a telephone node connector (TNC) or telephone. It's a CP/M program, requires 64K of RAM, and is running in an unmodified 2 MHz machine.

I'll send a copy to anyone who sends me a formatted 5-1/4" disk and a stamped, self-addressed disk mailer — or \$10 in lieu of! The disk can be 10 hole hard-sector or double-density (180/360k) soft-sector. An ASM file is included to permit changing the time constants for anyone with a 4 MHz mod. installed in the H-89 or to make any other changes that might appeal.

Warmest regards and best of luck to all of you.

Ray Isenson, N6UE
4168 Glenview Drive
Santa Maria, CA 93455

ing to the information I have. Unfortunately, I have not been able to locate a source for either module, probably because the eaZy PC was discontinued a couple of years ago, and supplies seem to be exhausted.

If anyone knows where either or both of these modules are currently available, please write to me at the address listed at the beginning of this article so that I can make this information available in this column to HUG members who own eaZy PCs. Both HUG and I have received a number of inquiries about a source for these modules, and we have not been able to locate any source. If you know where one or both of these modules can be purchased, be sure to include ordering information, such as price, ordering address and phone number, business hours, and what credit cards are accepted. I would also ask that you let me know right away so that I can get this information published in REMark as soon as possible. Any vendors who have these eaZy PC modules are especially invited to respond to this request.

Powering Down

When many of you ordered the FlipFast books as discussed in the March 1990 issue, you took the time to mention that you found my *Powering Up* book and column extremely helpful. A few of you even mentioned that my articles were one of the key reasons you have renewed your subscription to REMark, and I can only say that I deeply appreciate those compliments. In the nearly seven years I have written this column, I have always tried to provide useful information about the Heath/Zenith computers and software we use. From your letters, it appears that I have been somewhat successful in that objective which was the original purpose of this column. Please accept my

X-10 Won't Work on AT

Dear Jim:

Wondering if (Buggin' HUG) might be able to help me.

Purchased an X-10 Powerhouse CP-290 for a clone XT. Worked fine for years. Removed and placed in my new H-2526-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 AT. Wondering if anyone has written new software for the AT.

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

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

personal thanks for the kind words and suggestions included in your letter.

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

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

Products Discussed

HUG Software

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

HELP!

Dear HUG:

Does anyone have some spare CMOS for the ZP-150?

Jim Casner
1832 N.E. 104th Street, Loop #5
Vancouver, WA 98686 *



Want New And Interesting Software?
Check Out HUG Software

FREE MICROSOFT WINDOWS * DETAILS BELOW

Seagate HARD DRIVES

| MODEL | CAPACITY/FORMAT/SPEED/SIZE | DRIVE ONLY | XT KIT |
|-------------|----------------------------------|------------|----------|
| * ST-125 | 21 MEG / MFM / 40 MS / 3.5" | \$229.00 | \$279.00 |
| * ST-138 | 32 MEG / MFM / 40 MS / 3.5" | \$277.00 | \$327.00 |
| * ST-138-1 | 32 MEG / MFM / 28 MS / 3.5" | \$307.00 | \$357.00 |
| * ST-151 | 42 MEG / MFM / 24 MS / 3.5" | \$353.00 | \$403.00 |
| * ST-138R | 32 MEG / RLL / 40 MS / 3.5" | \$258.00 | \$313.00 |
| * ST-157R | 49 MEG / RLL / 40 MS / 3.5" | \$286.00 | \$341.00 |
| * ST-225 | 21 MEG / MFM / 65 MS / 5.25" | \$199.00 | \$254.00 |
| * ST-251-1 | 42 MEG / MFM / 28 MS / 5.25" | \$312.00 | \$362.00 |
| * ST-4096 | 80 MEG / MFM / 28 MS / 5.25" FH | \$582.00 | \$633.00 |
| * ST-238R | 32 MEG / RLL / 65 MS / 5.25" | \$218.00 | \$273.00 |
| * ST-277R-1 | 65 MEG / RLL / 28 MS / 5.25" | \$348.00 | \$403.00 |
| * ST4144R | 122 MEG / RLL / 28 MS / 5.25" FH | \$623.00 | \$678.00 |

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

*** ZENITH PC COMPUTER UPGRADES ***

SmartWatch from FBE RESEARCH

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

Z-150/160 MEMORY UPGRADE

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

Z-150 SERIES HARD DISK DRIVE KIT

⇒ These kits include high speed Seagate drives with autopak 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, | \$283.00 |
| * ST-138/Z150 Kit | 32 Meg, 40 MS, | \$331.00 |
| * ST-151/Z150 Kit | 42 Meg, 24 MS, | \$409.00 |

Z-148 HARD DISK DRIVE KIT

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

| | | |
|-------------------|----------------|----------|
| * ST-125/Z148 Kit | 21 Meg, 40 MS, | \$354.00 |
| * ST-138/Z148 Kit | 32 Meg, 40 MS, | \$402.00 |
| * ST-151/Z148 Kit | 42 Meg, 24 MS, | \$478.00 |

Z-148 EXPANSION CARD

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

Z-150 VIDEO ELIMINATOR

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

2400 BAUD MODEMS

⇒ Fully Hayes compatible 2400/1200/300 Baud with Software. Internal \$99.00 External Model \$128.00 Cable for External Modem \$8.50



15718 SYLVAN LAKE * HOUSTON TX 77062

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

FLOPPY DISK DRIVES

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

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

PAYLOAD CUSTOM ASSEMBLED COMPUTERS

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

VIDEO MONITORS

| | | |
|------------|---------------------------------|-----------|
| ⇒ ZCM-1492 | ZENITH Color Flat Screen VGA | \$ 679.00 |
| ⇒ MA2585 | SAMSUNG Amber TTL 720x350 | \$ 89.00 |
| ⇒ CW4656 | SAMSUNG Color RGB 640x200 | \$ 223.00 |
| ⇒ CM4592 | SAMSUNG Color EGA 640x350 | \$ 339.00 |
| ⇒ CJ4681 | SAMSUNG VGA 720x400 | \$ 360.00 |
| ⇒ CVB4581 | SAMSUNG Multi-sync VGA 1024x768 | \$ 469.00 |
| ⇒ CM1440 | SEIKO VGA 1024x768 .25 dot | \$ 609.00 |

VIDEO CARDS

| | | |
|-----------------|------------------------------|-----------|
| ⇒ EGA480 | PARADISE AUTOSWITCH 640x480 | \$ 99.00 |
| ⇒ VGAPLUS | PARADISE AUTOSWITCH 800x600 | \$ 168.00 |
| ⇒ VGAPLUS16 | PARADISE AUTO 16 BIT 800x600 | \$ 207.00 |
| ⇒ VGA 1024-256k | PARADISE AUTO 1024x768 | \$ 249.00 |
| ⇒ VGA 1024-512k | PARADISE AUTO 1024x768 | \$ 309.00 |

MEMORY CHIPS, ETC.

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

| | | |
|--------------------------------|-----------------------|---------|
| ⇒ 41256 256x1 80 ns.....\$2.95 | SIP 1Mx9 80 ns | \$80.00 |
| ⇒ 41256 256x1 100 ns | SIM 1Mx9 80 ns | \$79.00 |
| ⇒ 41256 256x1 120 ns | SIM 256x9 80 ns | \$24.00 |

Z-100 SERIES COMPUTER UPGRADES

High Density 1.2 Meg Drives

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

ZMF100A by FBE Research

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

UCI Memory Upgrade Pal Chip Set

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

UCI EasyWin HARD DRIVE SYSTEM

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

Z-100 SERIES SOFTWARE

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

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

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

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

PAYLOAD * PAYLOAD * PAYLOAD * PAYLOAD

Accessing the System Bus on Z-181 Laptop Computers

Dennis L. Myers
717 Clover Lane
Temple, TX 76502

Part 2

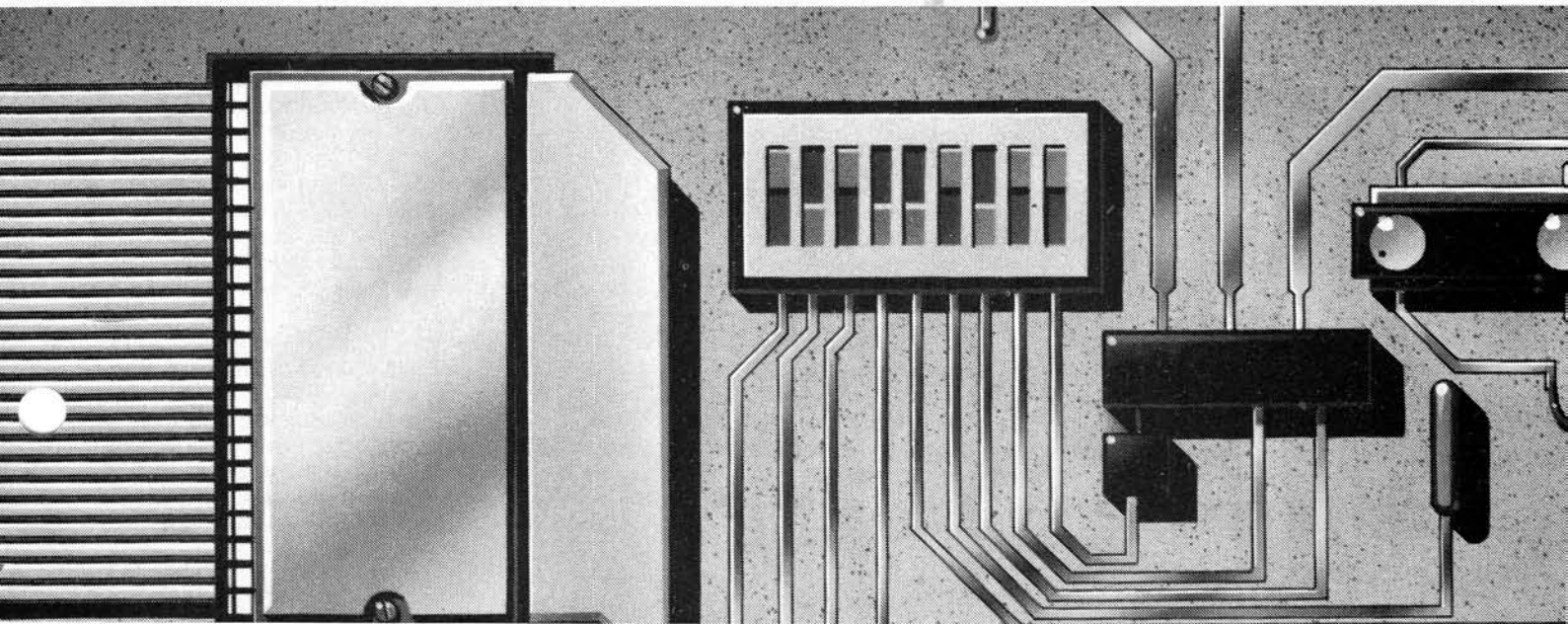
Warning: The system described below is "custom built". Building it requires knowledge of circuit boards, soldering, pin configurations, power supplies, and electronic troubleshooting. Do not undertake this project if you do not have the necessary experience! Although I believe this system will allow adding any IBM-PC compatible hard disk system to the Z-181-93, it has been tried ONLY with the power supply, hard disk controller, and hard disk described in the article. I take no responsibility for assuring the successful use of other drives or controllers. NO documentation is available (either from me or Zenith Data Systems) other than that supplied in the article. Although using a ZDS connecting cable and interface board should make the project relatively "safe" in terms of risk to the computer motherboard itself, it is always possible to damage electronic components by mis-wiring, etc. I take absolutely no responsibility for any damage to any user components caused by attempting this project! The cost of this project is significant . . . be certain you wish to attempt it!

Part 1 of this article described the efforts required for me to accumulate all the information needed to implement non-Zenith Data Systems external floppy and hard disk drives on the Z-181-93 laptop computer. Readers considering projects that modify hardware of ZDS machines (other than kit machines) may wish to read of my trials and tribulations to see the types of problems encountered. Part 2 of the article, however, shows that significant hardware modifications can be made to business-oriented ZDS systems.

Adding a non-Zenith Data Systems external hard disk to a Z-181-93 requires the following items:

1. An external disk drive case with power supply (I used American Design Components surplus part #14541, \$59.50, phone 1-800-776-3700).
2. Heath part #970-1809 "Interface PC Assembly".
3. An IBM-PC (not AT) compatible "half-card" hard disk controller.
4. A half-height hard disk compatible with the controller.
5. A ZDS "laptop to interface board" cable (part #HCA-78).
6. Constructing a short 62-pin "gender changer" cable.
7. Assorted connectors and small hardware.

Solutions for two problems must be considered when choosing the disk drive case/power supply to be used. The Interface PC Assembly must be mounted in the case (along with the hard disk), and must derive power from the disk drive power supply present in the case. Similarly, the hard disk controller must be mounted in the case (not in its usual location in an expansion slot in an IBM-PC compatible computer), and must also be powered (it will derive power from the Interface PC Assembly). Therefore, the case must be of a size and configuration which will allow the mounting of two circuit boards not normally present (in addition to the hard disk drive itself), and it must provide enough power (and proper voltages) to handle these boards. I chose a surplus external case/power supply with two full-height drive slots. By using a half-height hard disk coupled to a "half-card" hard disk controller, ample space was present to accommodate not only the hard disk and "extra" circuit boards, but two external half-height floppy drives as well. (See pictures). The case/power supply was originally designed for a Burroughs system. It only cost \$59.50, and arrived in very good (used) condition. Its grey-white color nearly matched the Z-181 case. The power supply provides an ample 60 watts



(the original IBM-PC only produced 65 watts!)

The Interface PC Assembly appears to provide an emulation of the IBM-PC I/O channel (expansion slots) present in an IBM-PC computer (see part one of this article). The expansion cards on an IBM-PC have 62 pins, consisting of two rows of 31 pins (A1-A31 and B1-B31). Four voltages are present on the IBM-PC expansion slot: +5v, -5v, +12v, -12v. The Interface PC Assembly requires only +5v and +12v to operate itself, but "passes through" -12v to its proper pin location on the IBM-PC compatible male card edge connector present on the Interface board. Note that the Interface PC Assembly appears to be MORE than just a circuit dedicated to connecting a hard disk controller to the Z-181. As best as I can tell, it is a buffer board that conceptually could allow connection to the Z-181 of ANY IBM-PC compatible expansion card! In addition to +5v and +12v, some of these cards require -12v to operate; virtually none require -5v, even though it is provided by the IBM-PC expansion slots. IBM-PC compatible hard disk controller cards should require only +5v and +12 volts (since the original IBM hard disk controller required only these voltages), and all floppy and hard disk drives themselves require only these two voltages. Thus, any external disk drive power supply should have the voltages necessary to install an external hard disk system on the Z-181. One reason I chose the Burroughs case/power supply, however, was that it additionally provides -12v and -5v. I wanted to "pass through" -12v to the card edge connector on the Interface PC Assembly, since in the future I may wish to try other expansion cards (such as serial or parallel cards, modem cards, video cards, etc.) on my system. My only concern then will be physically placing the cards in the case . . . I will be limited to "half-cards" in order to get them in.

The Zenith Data Systems HCA-78 cable is a real necessity. Both the 50-pin ultraminiature female connector which hooks the cable to the Z-181, and the 62-pin 3-row AMP male connector that hooks the cable to the interface board are expensive, hard-to-find special order items from electronic supply houses. The cable itself MUST be shielded, since the output lines of the Z-181 are limited to a rather low 5 volts. Connecting wires to an ultraminiature connector is torture. The ZDS cable costs an outrageous \$99.00, but building one will cost over \$50.00 in parts alone due to the cost of the connectors. It's best to order the cable from Zenith Data Systems. I did determine pin definitions for the cable itself, using a continuity tester and lots of tedious "pin mapping". (ZDS listed two different pin descriptions, and nobody could tell me which was correct!) I will supply the listing

to any person foolhardy enough to try to build the cable).

The Interface PC Assembly is a must. It costs an astounding \$8.55! The board is the size of a small half-height card. Its bottom edge is identical to the male card edge on an IBM-PC type expansion card. DO NOT PLUG THIS CARD INTO THE EXPANSION SLOTS OF AN IBM-PC compatible COMPUTER! I converted the male card edge connector to a female 62-pin "slot connector" (similar to an IBM-PC expansion slot) by making a "gender changer" consisting of a 62-wire ribbon cable with female 62-pin connectors on each end.

The Interface PC Assembly does not come with any documentation, and none is available from Zenith Data Systems. It has a four-pin DIP switch present, which comes pre-set as: 1-ON, 2-OFF, 3-ON, 4-OFF. A hard disk works fine with these settings, so do not change them. (I suspect the switch tells the board how much memory the host computer has, for applications such as adding external memory . . . but who knows?) The board also has a 6-pin power supply connector. The pins are numbered on the board 1-6. Pins 1 and 2 require +5v. Pins 3 and 4 are grounds. Pin 5 is +12v, and pin 6 is -12v. (Again, pin 6 need not be connected to use the board with a hard disk system).

I chose a Seagate ST-238R hard disk drive for my system. Its 65ms access speed is better than that of the original IBM-PC hard disk. Even with the Z-181 running at 8 meg., paying the extra price for a 28-30 ms access hard disk is probably not worth while, since the slow 8088 CPU will keep the drive from reaching its maximum I/O speed. The Seagate is a half-height drive (needed to leave room for the extra boards to be placed in the drive case). It is an RLL (run-length-limited) certified drive, so it can allow up to 32 meg. formatting. I matched the drive to an OMTI 5527A RLL controller. The controller is on a half-card (a full card would not fit in the drive case), and is IBM PC/XT compatible.

Preparing the disk drive case for the additional boards is not too hard. The Burroughs case is nice in that the two additional circuit boards required (interface and hard disk controller) can be easily mounted on either side of the internal chassis. No matter how you mount the drives, or which case you select, you must make sure that the boards will be well ventilated, and will fit with the case cover on. You must be certain that the 62-pin 3 row AMP connector on the disk drive end of the ZDS laptop-to-board cable will fit with the drive case cover in place, and that the cable has a way to exit the case. You may need to modify the back of the case chassis to allow the cable out. Alternatively, both boards could be mounted onto the hard disk drive itself. Doing so

would require fabricating brackets, etc. I found it much easier to mount the boards on the case chassis, using screws, threaded spacers, and nuts.

Power for the Interface PC Assembly must be obtained from the case power supply. The Burroughs case/power supply had an "extra" 4-wire standard disk drive power cable (+5v, GND, GND, +12v) that I easily modified by cutting off the usual 4-pin connector, replacing it with a 6-pin connector compatible with the pins on the Interface PC Assembly (I jumpered the two 5v and GND pins at the 6-pin connector itself). If no extra cables (other than those you will use for the hard disk drive and any floppy disk drives you plan to add) are available in the case you choose, you will need to splice wires into one of the cables to build a cable for the Interface PC Assembly. Alternatively, you can bring a cable directly from the power supply — just be certain the voltages are correct! I also brought a separate -12v wire from the power supply itself to pin 6 of the 6-pin power connector for the Interface PC Assembly. Once again, it is not necessary to supply -12v to the Interface PC Assembly if it will only be used with a hard disk controller. Remember that wire color is NOT a reliable guide to voltage! The Burroughs power supply uses RED wire for +12v, while IBM uses it for +5v! Be CERTAIN power to the Interface PC Assembly is wired properly before powering up. Check the voltages on the pins of the power connectors with a voltmeter for whichever power supply/case you use.

After modifying the case to hold the components and creating a power cable for the Interface PC Assembly, mount the assembly and the hard disk controller in the disk drive case and connect the power cable to the Assembly. Connect the male card edge connectors present on both the Interface PC Assembly and the hard disk controller by building a "gender changer" using 12 inches of 62-pin ribbon cable and two crimp-on 62-pin IBM-PC type female card edge slot connectors. These components may be a little hard to find, but a large electronics supply house should have them or be able to order them for you. You may wish to have the supply house crimp the connectors on for you . . . 62 wires are hard to "squish" without a press vise. When installing the cable, be certain that pin A1 of the board connects to pin A1 of the disk controller. Both cards have numbered connectors. The "A" side is the component side of the card.

Now set the jumpers on the hard disk drive, mount the drive in the case, and connect the cables from the drive to the disk controller, following the instructions that accompany the drive. Follow the controller installation instructions carefully regarding configuration. For both sets of

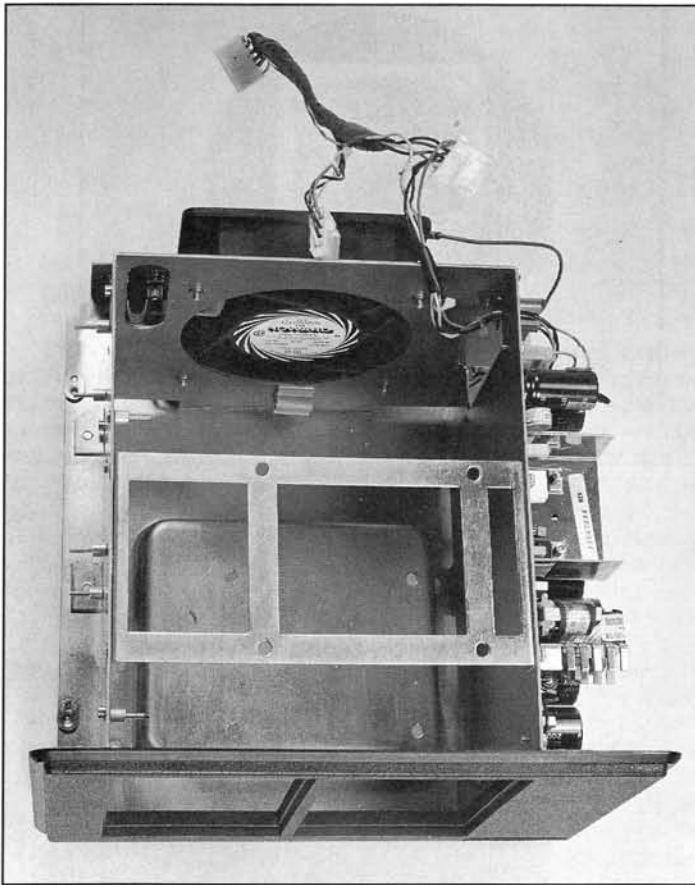


Photo 2

The surplus Burroughs computer case has an ample power supply and lots of room. The taped 6-pin connector powers the "Interface PC Board."

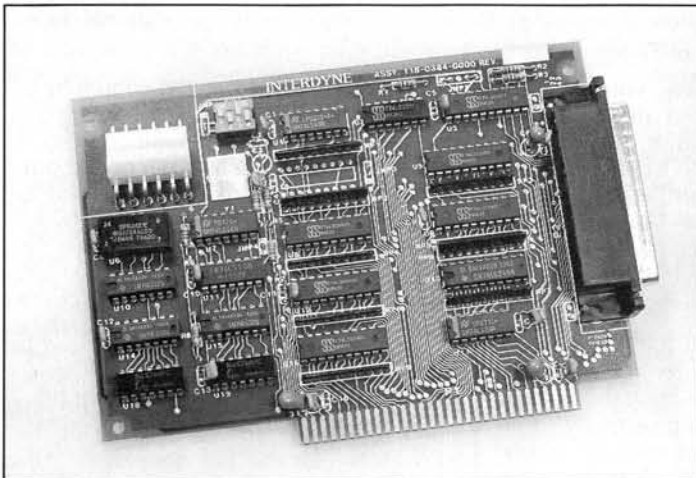


Photo 3

The "Interface PC Board." Note the 6-pin power connector at the top, and the 62-pin double-row IBM PC style male connector at the bottom. The cable connector at the right accepts the ZDS HCA-78 cable.

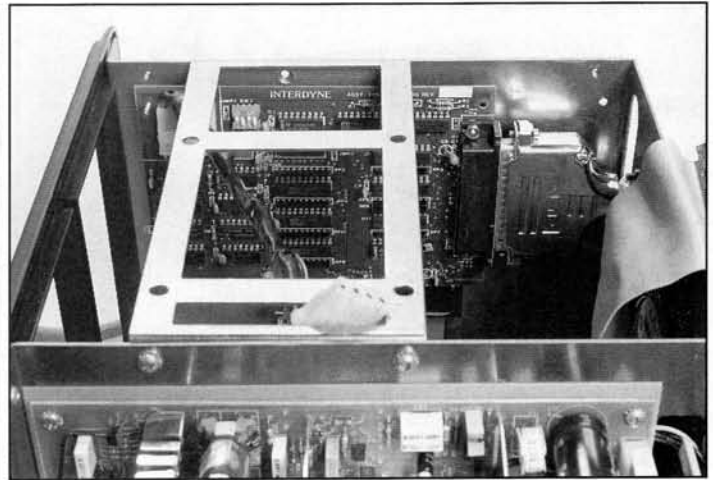


Photo 4

The drive case must have room for both the Interface PC Board and the large connector on the ZDS HCA-78 cable. Note the 62-wire ribbon cable "gender changer" coupled to the interface board with a female connector. The other end of the cable also terminates in a female connector.

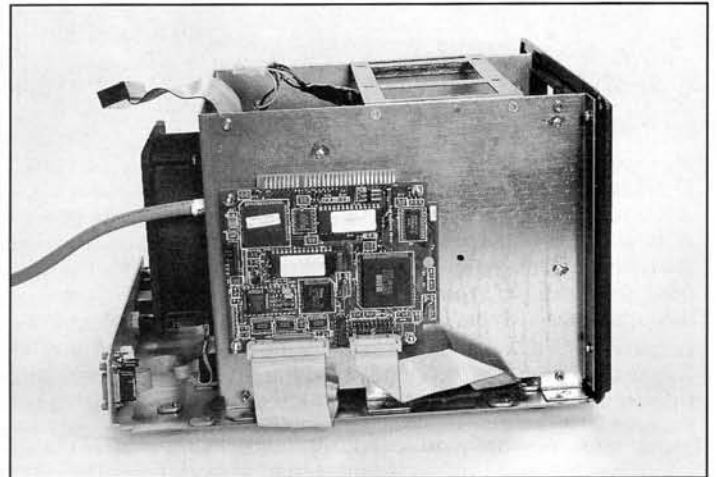



Photo 5

The "half-card" hard disk controller also mounts in the drive case. It derives its power from lines present in the "gender changer" cable that connects it to the Interface PC Board. That 62-pin ribbon cable can be seen at the top of the disk controller card. The cables on the bottom are from the controller to the drive.

MOVING?



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

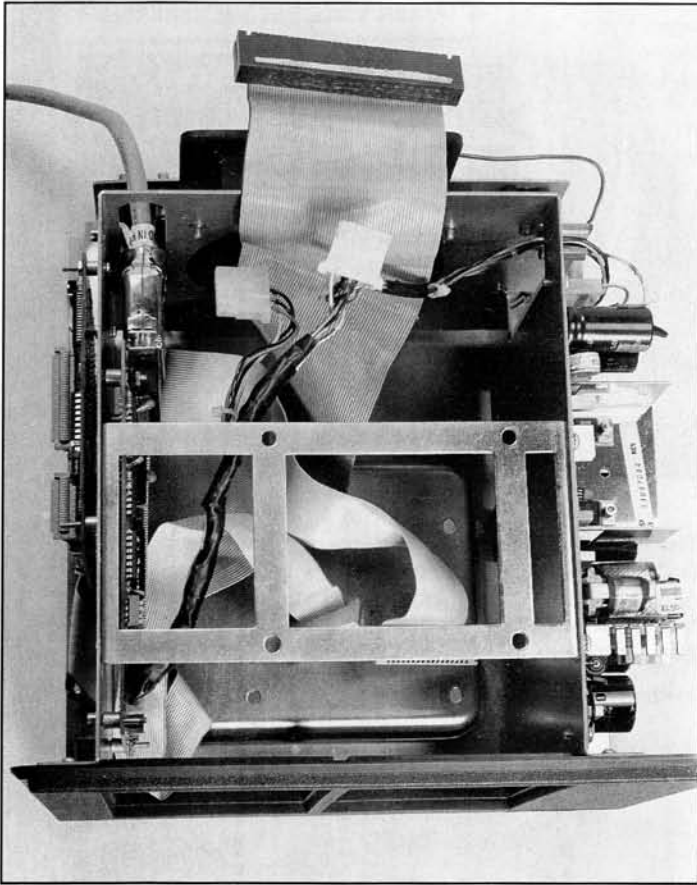


Photo 6

There must be room after mounting the Interface PC Board and the hard disk controller for hard and floppy drives. Two half-height floppies and one half-height hard drive will fit here. After installation of the drives, the large "gender changer" cable will lay over the drives and connect from the Interface PC Board inside to the hard disk controller mounted outside.

instructions, assume you are mounting hard drive "C" as a single hard drive in an IBM-PC. If the installation instructions ask, the Z-181 assumes that the controller BIOS ROM resides at address C8000 hex, and that the hard disk port is 320 hex.

The Z-181 does NOT need any modifications to accept the hard drive. For testing, set the 6 DIP switches on the bottom of the Z-181 for: no external floppy drives, external hard drive labeled "C", 4.7 meg clock (1-ON, 2-ON, 3-ON, 4-ON, 5-OFF, 6-ON). I use MS-DOS 3.21 supplied with my laptop. Other versions (greater than 3.2) should work, but may require device driver specifications in the config.sys file. (I don't know.)

Testing the Hard Drive System

Once the system is assembled, recheck all connections and configuration jumpers on the Interface PC Board, controller card, and hard disk again! Connect the disk drive to the laptop via the ZDS cable, and turn the drive system on first. Then turn on the laptop and boot the machine using a system disk in drive A. If the

computer boots normally, enter DEBUG and type the command dC800:0 (NO SPACES) at the "-" prompt, followed by the ENTER key. If the Z-181 is addressing the controller BIOS ROM, you will see a full screen of various numbers and letters representing the hex code of the BIOS ROM. This means the Interface PC Assembly is working properly.

If you see a screen full of "F's", the laptop does not recognize the hard disk controller BIOS ROM. Although the hard disk controller itself could be faulty, more likely there is a problem with the operation of the Interface PC Assembly. Recheck the cable connection between the controller and the interface card, the cable between the Z-181 and the interface card, and the power cable connector to the interface card. Also check the jumpers (if present) on the controller card that set the starting address of BIOS ROM (C8000 hex) and the hard disk port (320 hex). Do not worry about the drive connections to the controller, or the jumpers on the drive itself (yet) . . . the problem is located somewhere between the Z-181 and the



**Want New And Interesting Software?
Check Out HUG Software**

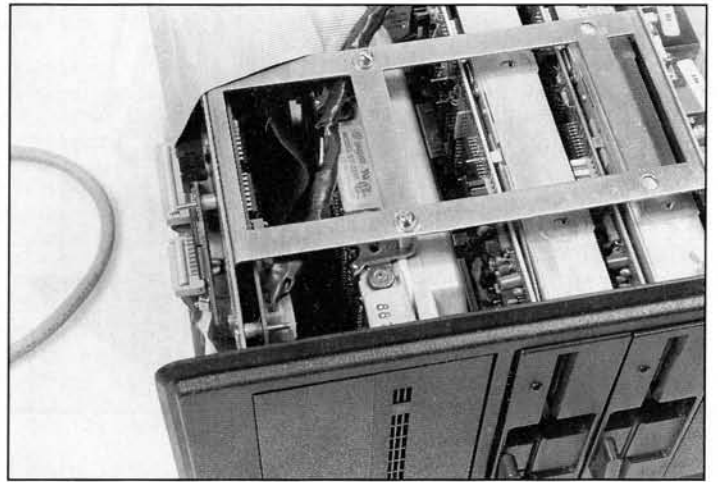


Photo 7

All drives installed. A full-height hard drive would not leave room for the two cards (hard disk controller and Interface PC Board), since they are not normally present in an external drive case.

disk controller itself. Test the continuity of the gender changing cable you constructed. Each pin on one connector must be connected to the same pin on the other connector. If everything else checks out, check the hard disk controller by removing it and installing it along with the drive in an IBM-PC compatible machine. If all connections and the controller check out, the problem must be in the interface board or the ZDS cable. Check the power at the board using a voltmeter. If it's ok, consider ordering another board (the first one I got was defective!) or checking the ZDS cable (get the pin definitions from me).

You may find that the boot process takes a long time (about 30 seconds). Keep watching the screen. If you see "1701" appear, followed by a normal boot from the floppy disk, it indicates the Z-181 MONITOR ROM was aware of an external hard disk system (that's good, it means the interface system is working), but that the controller or hard drive failed the Z-181 MONITOR ROM test sequence at boot. Go ahead and check for control-

Continued on Page 48

Jan Axelson
2209 Winnebago Street
Madison, WI 53704

GDU

(Zenith Data Systems' General Disk Utilities)

Part 2

This is Part 2 of an introduction to Zenith Data Systems' General Disk Utilities, a set of utilities included in ZDS' MS-DOS 3.3 plus. Last time, we looked at GDU's basic functions for undeleting files, as well as other functions for sorting directories and displaying disk information. This time, we'll see how to examine and edit disk data and how to undelete directories and files that can't be undeleted in other ways.

Examining files

Undeleting a file works best if it's done immediately after deleting. When a file is deleted, the disk clusters it used are marked "free for use" in the disk's FAT (file allocation table). If files have been saved to disk after deleting, there's a chance that a file was stored in the deleted file's clusters, overwriting the deleted file in the process.

If this occurs, when you try to undelete with GDU, you'll see the following message:

"Clusters used by erased files have been allocated to other files. You may still be able to recover some lost data by using the REBUILD FILE feature."

Or you may find that for one reason or another none of the other undelete options in GDU are successful. This is most likely to happen if you're undeleting a long file (one that takes up several clusters), especially if you've recently deleted many files from the disk.

Using REBUILD FILE is a little more involved than GDU's other undelete procedures. It requires you to view possible clusters and decide whether or not to include each in the new, restored file. To understand the process better, let's first use GDU's LIST/EDIT option to view a file. Select LIST/EDIT IN HEX AND ASCII from the main menu and select one of your practice files (a file you don't need). You'll then see a display showing the file, byte for byte, as it is stored on the disk. Figure 1 shows an example of such a display.

```
File Name: B:\GDU.TXT
Cluster: 27H (Cluster 1 of 7) Sector: 86 1st 1/2 sector 1
2E 50 4C 36 38 2F 2E 46 44 36 36 2F 2E 4F 46 31 0 .PL66/.FD66/.OF1
32 2F 2E 52 4D 36 30 2F 2E 54 53 35 2C 31 30 2C 1 2/.RM60/.TS5,10,
31 35 2F 2E 4C 53 32 2F 2E 46 4C 2F 0D 0A 2E 46 2 15/.LS2/.FL/.F
43 2F 47 65 74 74 69 6E 67 20 53 74 61 72 74 65 3 C/Getting Starte
64 20 77 69 74 68 0D 0A 20 47 44 55 20 28 47 65 4 d with GDU (Ge
6E 65 72 61 6C 20 44 69 73 6B 20 55 74 69 6C 69 5 neral Disk Utili
74 69 65 73 29 0D 0A 2E 46 4C 2F 0D 0A 09 49 66 6 ties) .FL/ If
20 79 6F 75 20 88 61 76 65 20 5A 65 6E 69 74 68 7 you have Zenith
27 73 20 4D 53 2D 44 4F 53 20 20 20 76 65 72 73 8 's MS-DOS vers
69 6F 6E 20 33 2E 33 20 50 6C 75 73 2C 20 79 6F 9 ion 3.3 Plus, yo
75 20 61 6C 73 6F 20 68 61 76 65 20 61 20 73 65 A u also have a se
74 20 6F 66 20 75 74 69 6C 69 74 69 65 73 20 63 B t of utilities c
61 6C 6C 65 64 20 47 44 55 2C 20 77 68 69 63 68 C alled GDU, which
20 73 74 61 6E 64 73 20 66 6F 72 20 47 65 6E 65 D stands for Gene
72 61 6C 20 44 69 73 6B 20 55 74 69 6C 69 74 69 E ral Disk Utiliti
65 73 2E 20 20 4D 75 63 68 20 6F 66 20 47 44 55 F es. Much of GDU

End of file occurs in sector 2 of file cluster 7
move 1/2 sector
HOME/END first/last half sector PgUp/Dn next cluster ESC for main menu
```

Figure 1

The box on the left shows the contents of a file in hexadecimal, while the box on the right shows it in ASCII. (This printout of a screen display was created with SHIFT-PrtSc. The printer ignored some non-printing codes in the display, so a few of the characters and part of one side of the box in the ASCII display are shifted left from how they appear on screen.)

play. If the file is short enough to fit on one screen, you'll see a blinking byte that indicates the end of the file.

Along the top of the screen is some information about where your file is stored on the disk, including the number of the displayed cluster, how many clusters make up the file, and which half-sector in the cluster is now displayed.

The file is shown two ways simultaneously. The display on the left uses hex notation. (Hex is short for hexadecimal, a number system based on 16. In hex, the letters A-F represent the numbers 10-15, and each 2-digit number represents one byte of information (8 bits, signifying a decimal number from 0 to 255).)

The display on the right shows the ASCII characters represented by the hex numbers. Each character in the ASCII display corresponds to a 2-character number in the hexadecimal display. For example, the first character in Figure 1 is a period, and the corresponding hexadecimal byte

is 2E, the hex ASCII code for the period character. (Most BASIC manuals have a list of ASCII codes.)

If you're displaying a text file, you'll see the text, or words, contained in the file. You'll probably also see characters that perform formatting and other functions. These characters may be invisible when you view the file with a word processor or other text editor.

Press F2, and extended ASCII characters (those with values greater than 127 or hex 7F) will be reduced by subtracting 128 from their values. In addition, most non-text characters with decimal values less than 32 become invisible. The result is usually a more readable, less cluttered display of the file's text. This should be especially helpful for files like WordStar's, which use the eighth ASCII bit (with value 128) for their own special purposes.

Stepping Through a File

If the file is longer than the half sector

displayed, use PGDN to step to the final cluster, and use the DOWN-ARROW key to step to the file's last half-sector — the one with the blinking end-of-file byte.

Unless the file fills the last cluster exactly, the last cluster will contain information after the end of the file. You may see a series of F6s or 00s. Or you may recognize information that looks like it belongs in a different file.

How can the cluster contain information from another file, if two files can't share a cluster? Remember that when a file is deleted, it remains on the disk, but its space is freed for use by another file. The same thing happens when you edit a file and then save the new version. The edited file is usually stored in different clusters from those that the original file used.

Reading and writing to disk is done in complete sectors. Whatever is in memory when the file is saved may be used to fill out the sector the file ends in. But any unused sectors in the file's final cluster are left untouched.

So if a file is saved to a previously used cluster, and if the file does not use every sector in the cluster, anything previously stored in the unused sectors will remain. The old data isn't part of the new file, but the file reserves the entire cluster, including any unused sectors.

You can use DISPLAY/EDIT to edit a file, though you would normally do this only in special circumstances. Word processors and other text editors are far better suited for everyday file editing. You might call on DISPLAY/EDIT to remove or add codes that are otherwise invisible, however.

To edit a file, press F8 to change to EDIT mode and use the arrow keys to move around the file and select a byte to change. (Do this only with a practice file — one you can afford to lose!) You can edit by typing hex numbers in the hex display, or by typing text characters in the ASCII display. F2 toggles the cursor between the hex and ASCII sides of the display.

To cancel any changes you've made, press ESC. To save your changes, press F8. You must either save your changes or quit without saving before you move on to view a different half sector. To exit DISPLAY/EDIT, press ESC.

Rebuilding a File

Now we're ready to try rebuilding a file. Select REBUILD FILE OR DIRECTORY from the main menu, then select SEARCH ERASED CLUSTERS. As before, select an erased file from the list and type the first character of its file name.

Select REBUILDING A FILE and you'll see a display much like what you saw with LIST/EDIT FILE. The display will show the first half sector of an erased cluster, with a list of options along the bottom of the

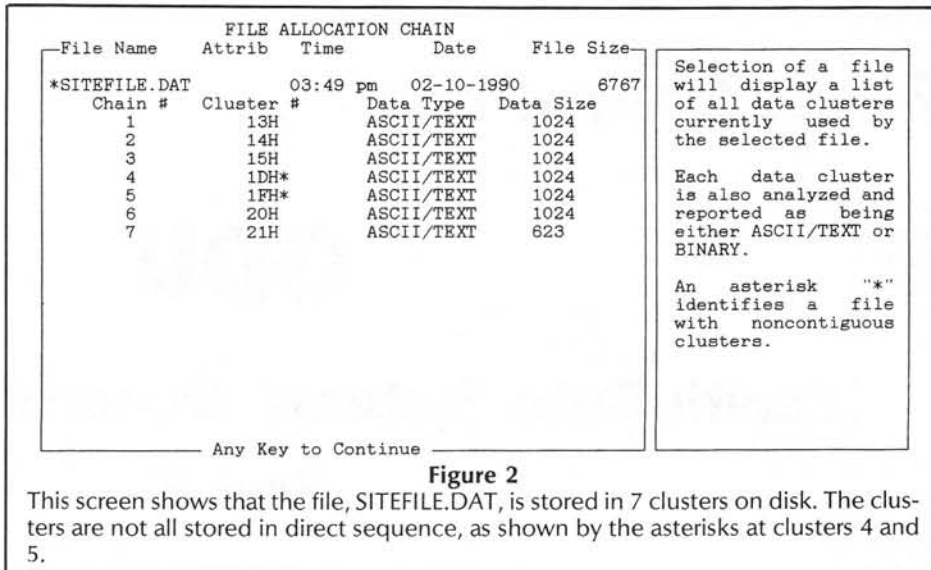


Figure 2

This screen shows that the file, SITEFILE.DAT, is stored in 7 clusters on disk. The clusters are not all stored in direct sequence, as shown by the asterisks at clusters 4 and 5.

screen.

Examine the contents of the displayed cluster. Since the file's first cluster number is saved in the file's directory listing, chances are good that the cluster displayed will be the correct first cluster for the file.

You can examine the entire cluster by stepping through its half-sectors with the arrow keys. If you believe the displayed cluster belongs in your restored file, press F6 to include it.

The display tells you how many clusters you need to restore. If the entire file is contained in the displayed cluster, press F4 (DONE) to complete the file restoration.

If the cluster displayed is not part of your file, or if you need to find the next cluster, you have several options. You can press F5 to examine the next erased cluster on the disk and proceed as before to include or exclude it. This is probably worth a try, especially if the first cluster was correct, since many files are stored in contiguous clusters.

Or you can give GDU a string of data (some text) to search for. Try to come up with a word or phrase that exists in the next cluster of your file, but is not likely to be found in other files. The search is NOT case-sensitive — if you ask it to find "Remark," it will also find "REMark." After you enter your text, GDU will search the disk for data matching what you enter, and will display the cluster when it finds a match.

From there you can include or exclude the cluster, then search for another string by pressing F9, or search for the next erased cluster by pressing F5.

If you press F10, you can select whether your search will be through all erased clusters, or through erased ASCII, binary, or directory clusters only, or you can select a cluster number in which to begin your search.

When you're done searching, press F1 to review the clusters you've included. If you're satisfied with the results, press ESC, then F4 to save your rebuilt file and exit. Otherwise, press ESC, then F3 to bail out and quit without saving the rebuilt file.

If you view a restored file with DOS's TYPE command or a text editor, you may find that the end of the file contains some "garbage" that doesn't belong. If so, use a text editor to delete it.

Recovering a Copied-Over File

What can you do if you copy over a file by mistake? DOS gives no warning if you copy a file over an existing file. For example, the command COPY FILE1 FILE2 copies the contents of FILE1 into FILE2, and in doing so removes any previously existing file with the name FILE2. Can you get the original FILE2 back? Since no files were deleted, undelete won't help here.

Or perhaps you've edited an existing file, saved the edited version, and then realized that this was a mistake and you must have the original back. In both cases, all is (probably) not lost. If there is room on the disk, a new version of a file is stored in different clusters than those used by the original.

So there is hope, especially if you can recognize the contents of your file when you see them. For experimenting, select two files on your practice disk and copy one (we'll call it FILE1) over the other (FILE2). Now let's try to get the original FILE2 back.

Run GDU and select REBUILD FILE OR DIRECTORY, then PROMPT FOR STARTING CLUSTER NUMBER. Press the INS key and enter the file name you want for your recovered file. Since FILE2 already exists, choose a different name, FILE3 perhaps. Select REBUILDING A FILE, and enter 2 for starting cluster number. If the displayed cluster belongs in your restored file, press F6 to include it as before. Other-

wise, press F9 and enter a string of text that was contained toward the beginning of the overwritten file. Or, press F5 to view the next erased cluster.

When a cluster is displayed, proceed as you did when rebuilding a deleted file. Examine the cluster, include or exclude it, and continue searching if necessary. When you're finished, examine the results and save the restored file (or not, as you wish).

Follow the same procedure to find a previous version of an existing file. If a file has been saved many times, you may find that the disk contains more than one erased copy of the file. If so, it's up to you to decide which clusters you want in the restored file.

Restoring Directories

If you delete all of the files in a directory, then delete the directory, and then realize that you need one or more of the deleted files back, GDU can help. Restore the directory before you try to restore its files, since the directory contains the starting cluster numbers for its deleted files.

To restore a deleted directory, select REBUILD FILE OR DIRECTORY, then SEARCH ERASED DIRECTORY CLUSTERS, and proceed as you did when rebuilding a file. Deleted directories are labeled E<dir> in GDU's list of deleted files.

The first character of each file name

in the directory will be changed (to "e" in ASCII, the Greek sigma in hex). But the rest of the directory information should be intact. After restoring the directory, you may use the techniques described previously to undelete its files.

Other Functions

Finally, GDU has a few other functions you might use in examining or editing a disk. The option DISPLAY FAT CLUSTER CHAIN in GDU's main menu will show you how many clusters, and which ones, are used by a file. Figure 2 shows an example. This information is what GDINDEX saves when you delete a file with GDUTSR installed. Non-contiguous clusters — those that don't occur in direct sequence on the disk — are marked by asterisks. The screen also shows data type (binary or ASCII), and data size (how many bytes in each cluster are in use).

If many of the files on a hard disk are non-contiguous, you may want to use the COMPACT command in DOS 3.3 Plus to put the files in contiguous clusters. This can give faster disk access, and contiguous files are also easier to undelete if the need arises. The DOS manual recommends backing up your disk before running COMPACT.

SEARCH/DISPLAY DATA CLUSTERS lets you examine data clusters, much as you did when rebuilding a file. You can

search active clusters (those currently holding data), erased clusters, or both. You can search an entire disk for the data you request, but it will be located only by cluster number, not by file name. Since this option includes no editing capabilities, it's safe to use for exploring a disk without risking damage.

DISPLAY/EDIT LOGICAL SECTORS lets you examine and edit a disk, including the sectors containing the boot record, FAT, and root directory.

DO NOT select this option unless you know what you're doing. Editing a disk with this option can quickly turn a good disk into a worthless one. On the positive side, if a disk's boot record, FAT, or root directory has been damaged, causing all or part of the disk to be unusable, it may be possible to bring it back to life with this option. But to do so you need to know what information is needed and where to put it, and this topic is beyond the scope of this article.

If you're curious, however, you should be safe exploring with DISPLAY/EDIT, as long as you don't press F8 to enable editing.

These are the basics of GDU. After this practice run, you should have an idea of when GDU can be of help to you, and of how to use GDU when the need arises.

*

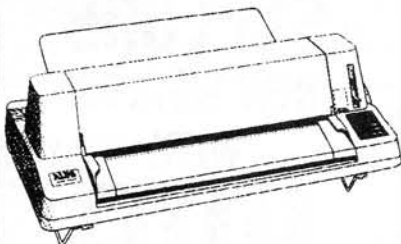
W S Electronics **ALPS**

AMERICA

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

1106 State Route 380, Xenia, Ohio 45385

YOU AND YOUR BIG IDEAS.



ALPS Allegro 500XT™

Call for pricing

- * 300 cps draft speed
- * 100 cps letter quality
- * Seven resident fonts
- * Flatbed paper path
- * 3 macro default settings
- * Wide carriage
- * 24 pin printing
- * Front panel controls

enable™



2.0 Retail \$695.

Sale \$99.*

*Quantities are limited

Attention: Federal Government Offices
We stock ALL ALPS Printer Models

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

****Government Discounts Offered****

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

Continued from Page 19

higher. If you have MS C use:

cl /AC /Ox /Gt28001 showega.c /link /E

If you have Quick C use:

gcl /AC /Ox /Gt28001 showega.c /link /E

The /Gt option sets the data size threshold. This is necessary because the linker will place all variables under 32K in the same data segment by default. And will cause an error since ShowEGA's varia-

bles exceed 64K and none are over 32K.

Next month, we will continue with graphics printing and OS/2 display programs.



Continued from Page 27

Again, this means that you must read your application software manuals. Some of today's software (e.g., the Quattro Pro spreadsheet) will automatically sense that you have a mouse driver installed and nothing else is necessary. Other software may "include" a built-in mouse driver so that you need not install a specific driver included with a mouse. And older software typically requires that you explicitly define the fact that you are using a mouse before you can actually use it. There are so many variations that it is impossible to describe them all, but the information should be in one of your manuals.

Powering Down

The next article will talk about the details of connecting a printer to your computer. This will include the construc-

tion of the various kinds of null-modem cables. Also included in this article is information about connecting a parallel printer to your computer, including a description of what the signals are for in this interface.

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

Products Discussed

HUG Software

| | |
|-------------------------------|---------|
| Powering Up (885-4604) | \$12.00 |
| HUGMCP (885-3033) | 40.00 |
| Heath/Zenith Computer Centers | |
| Heath/Zenith Users' Group | |
| P.O. Box 217 | |
| Benton Harbor, MI 49022-0217 | |

(616) 982-3463

Software

| | |
|---|----------|
| HyperACCESS/5 (DOS & OS/2) | \$199.00 |
| Upgrade (Registered HyperACCESS Users)(+ shipping) | 49.00 |
| Hilgraeve, Inc. | |
| Genesis Centre | |
| 111 Conant Avenue, Suite A | |
| Monroe, MI 48161 | |
| (800) 826-2760 (Orders only) | |

Hardware

| | |
|--|---------|
| DMM — Digital Multi-Meter (SM-2300-A) | \$19.95 |
| Heath Company | |
| P.O. Box 8589 | |
| Benton Harbor, MI 49022 | |
| (800) 253-0570 | |
| (Heath Catalog orders only) | |



Continued from Page 32

transferred to a 9-pin matrix printer (Centronics PS 220) using my homemade graphics program, running at 24-pin resolution. Incidentally, printing was physically at 9 lines/inch, the natural pitch for the print head, and logically at 6 lines/inch, the usual pitch. Even logical lines were printed as one physical line; odd lines were shifted so that the upper half was printed first, then the paper moved up to complete the bottom half. Any printer in reasonable mechanical condition will do this with a perfect line-up between the two halves. This means that — to come back to bar charting — you can not only use pitches of 4-1/2 and 3 bars/inch (2/9", 3/9"), but also intermediate values like 5/18" (no, I don't know what that is in lpi), if you shift every other bar four bits. Once you're used to this, you can use algorithms for any pitch, just shifting bars down until they're right where you want

them. As long as you stick to a 1/72" vertical increment, which means that data for the first head pass is always used for the first pass and so on, the shift routines are reasonably easy to write. I do not intend to use a 1/216" increment if I can avoid it, as this would mean that pass 1 data would be used for pass 2 or 3, and I think that this would be much more complicated.

To sum up: Small printers can do a lot more than they're usually used for. Special effects — bold, italic, and so on — are available outside word processing, if you include proper control sequences in your text. They can be customized for printing technical documents by downloading special characters. As long as these can be defined within an 8x11 matrix (or even across two or three consecutive 8x11 characters), designing is easy and fast using simple BASIC drawing programs. Standard word processors will accept the pres-

ence of download fonts, as long as they don't completely reinitialize the printer, and can switch from ROM to RAM and back if you can specify a ribbon control sequence. Full graphics use, where the program sets up an entire image, is within the scope of any good programmer, provided that the image can be broken down into repeating, independent blocks, each fitting on a few horizontal lines (like horizontal bar charts). By printing on outside paper and scaling down on a photocopier, outstanding resolution can be obtained (the drawings accompanying this text are all about 60% of the original size). And BASIC is powerful enough to handle all the possibilities available on a printer, although you will need compiled code to get a reasonable speed.

If readers are interested by these possibilities, it will be a pleasure to keep you informed on anything I develop along these lines.

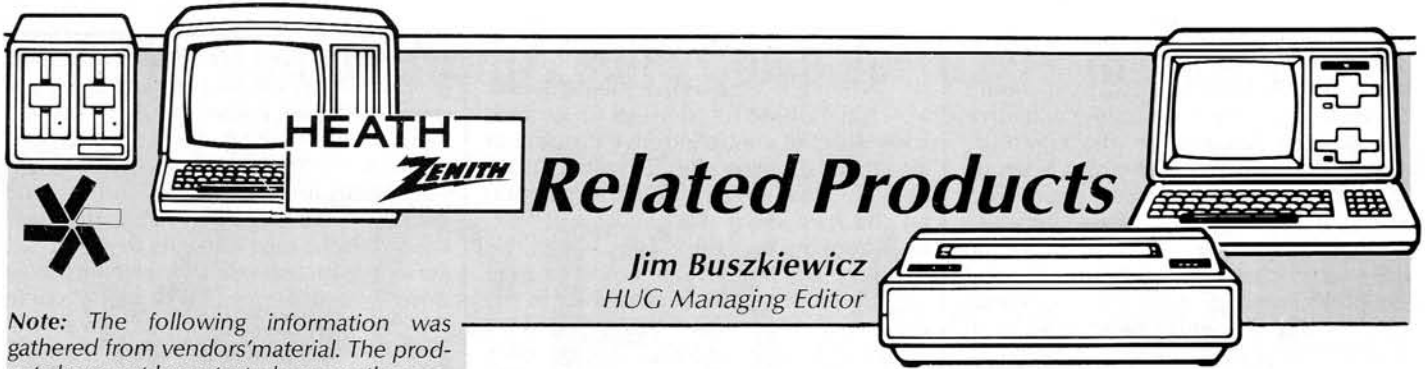


Z-100 or PC? With our I/O library you can write code in Microsoft's 'C' for both computers at once. Draw boxes with line-drawing characters; scroll any screen region; set screen colors; emulate PC characters on Z-100; refer to function keys symbolically; many other features. \$20.00 postpaid. No royalties. Write or call for free catalog with details.

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



**EXPLORE
NEW WORLDS
WITH
HUG
GAME
SOFTWARE**



Jim Buszkiewicz
HUG Managing Editor

Note: The following information was gathered from vendors' material. The products have not been tested nor are they endorsed by HUG. We are not responsible for errors in descriptions or prices.

**HDOS 3.02 Ready to Ship
The System and On-Disk Manual are Now Available**

An upgrade to HDOS 3.0, as distributed by William G. Parrott, Jr. for some years, is now available to the H-8 and H/Z-89/90 users. The system was prepared by Richard Musgrave (MIGHTY-SOFT, Kansas City, MO) and the voluminous documentation was written and edited by Daniel N. Jerome (Burnsville, MN) who keyed and updated the HDOS 2.0 manual for *The Staunch 8/89'er*.

The system will boot on virtually all '89/90's. The H-8 must be capable of remapping memory as when booting CP/M. Hence, the latter requires a Z-80 CPU ("ORG-0") card and front panel ROM, such as XCON8. (PAM-8 will NOT work!) Like ver. 3.0, 3.02 remaps memory and loads the ENTIRE system at the bottom of memory. There are no overlays. Moreover, most software written for HDOS 2.0 will run without modification under 3.02. Exceptions include those programs which directly access the MTR or H-17 ROMs (those addresses are now used for other purposes), Steve Robbins' EDIT19 (but a patched version is available from *Staunch*), and Softshop's UD.DVD. In most cases, commercial or public domain substitutes are available for these programs. Contact *Staunch* for further information.

Device drivers for all common media types (H-17, H-37, and H-47) are included. Drivers for other types may be obtained from the HDOS 3.0 seven-disk set. Almost any generic terminal may also be used (even the H-9!) because the terminal driver is separate from the system core. However, a number of the utilities on the distribution disks require the H/Z-19/89. If you are not using the latter, include this information when ordering.

If you are presently running HDOS 2.0, you should be able to move your printer device driver to 3.02 without any problem. The one known exception is Softshop's UD.DVD; it will lock up any 3.0 system! If you need a printer driver, contact *Staunch*; it has a library of them for many off makes and models, including a replacement for UD.DVD by Rick Street-er. Suitable drivers are also available from commercial sources, such as Lindley Systems.

Enhancements beyond those provided by HDOS 3.0 include: MEGAPIP, a DOS shell; many new BATCH commands; HALT, which will execute SHUTDOWN .ABS or .BAT before exiting the system; and 8 user areas, similar to CP/M's USER or MSDOS' subdirectories. As in 3.0, HDOS 2.0 has been integrated into PIP,

also from earlier versions, FLAGS.ABS has been added.

The documentation for this immediate release is on disk; when printed on standard 9-1/2" x 11" fanfold, it is 3 inches thick! The package includes a 3 inch, D-ring binder and section dividers for the 14 chapters. A printed manual is in preparation and if you order the system with the latter, the system will be promptly shipped with adequate documentation to get you started. The printed manual will be shipped as soon as it becomes available at no additional cost.

The price of the package, no matter which manual version you order, is \$60, including U.P.S. shipping. When ordering, clearly indicate the manual (on disk or commercially printed) you want and the media (standard or double-sided hard-sector; single- or double-sided soft-sector; or eight inch) you need. All 5-1/4" disks are formatted at 40-track (48-tpi). Send your order to:

Kirk L. Thompson
Editor, *The Staunch 8/89'er*
P.O. Box 548
#6 West Branch Mob. Home Vil.
West Branch, IA 52358
Home Phone: (319) 643-7136 *

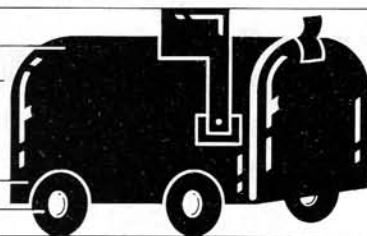
CLASSIFIED ADS

FOR SALE: Z-100 WITH SEPARATE COLOR MONITOR. Diablo 360 letter quality printer. Loads of software. Manuals and modem. Solid oak desk and stand, \$1500.00 or Offer. Mary (616) 429-1710.

MPI PRINTER OWNERS: CYBER FORCE. Phone (713) 682-0668. Has Parts. Thank you Al Lindo, Arkansas for answering my classified ad for help in January issue. Bert Desmarais.

WANTED OLD WESTERN DIGITAL FILE CARD. 10 and 20 Meg., 1.3 Height Hard Drive New or Used. Call Jim Evenings, (616) 429-3583.

MOVING?



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

Continued from Page 42

ler BIOS ROM with DEBUG . . . it should be there. Then check the cables and drive specification jumpers on the controller and hard drive itself. Something is wrong with the controller (other than BIOS ROM), the drive, the cables between them, or the jumpers on them. Follow the drive and controller installation sheet troubleshooting instructions. If all else fails, install the controller and drive in an IBM-PC compatible machine. If it can't be made to work there, the problem is not with the Z-181 or the interface.

If the machine boots quickly and passes the DEBUG test, proceed with the hard disk controller instructions for formatting the drive (it may or may not need low-level formatting). If you have problems, consult the installation manuals for the drive and the controller — it is very unlikely the problem is with the Z-181

interface.

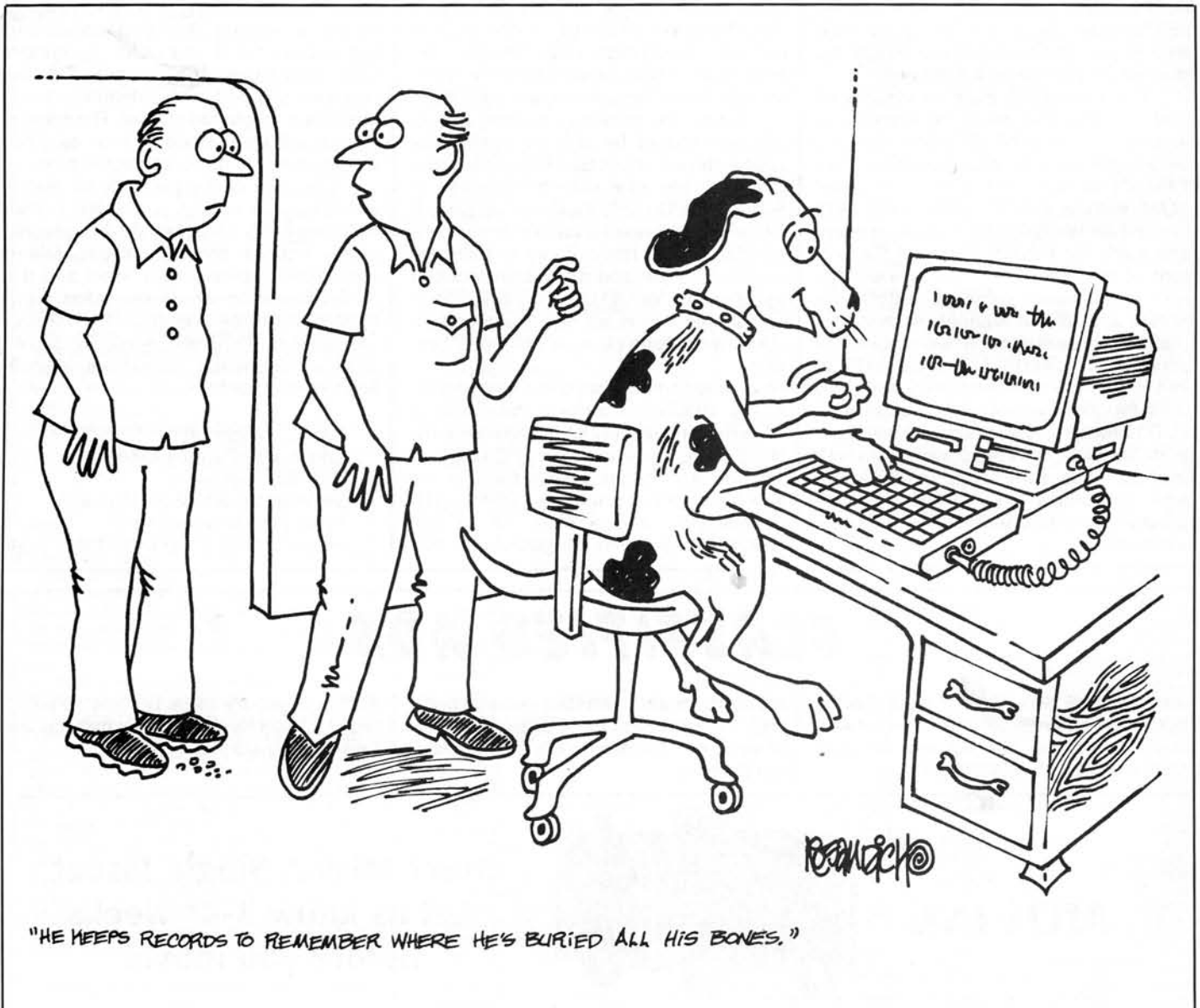
When it is time to use the high-level MS-DOS FORMAT command (in accordance with the controller/drive installation instructions), using the /S switch (FORMAT C:/S) will create a bootable partition on the hard drive. The Z-181 will then boot from the hard drive if you leave drive A empty. The MONITOR command BWO will not work exactly as described in the Z-181 owner's manual (bypassing drive A), since the disk controller you will use is not a Zenith Data Systems device (ZDS modifies the EPROM on controllers sold under the Zenith Data Systems name, adding code that will respond to the MONITOR ROM's extensive set of instructions). BWO will cause the same action as a cold or warm boot — a search of drive A, followed by a boot from the hard disk if drive A is empty.

You are now ready to set the com-

puter for the presence of external drives (if you have them), and to reset the clock speed if desired. Follow the Z-181 owners manual instructions to set the DIP switches on the bottom of the Z-181. If you configure for external drives, the Z-181 will automatically label the hard drive with the next available device letter. For example, one external drive becomes drive C, and the hard disk will become drive D. Two external drives causes the hard drive to become drive E. Do NOT change any drive specification jumpers on the hard disk controller or drive when adding external floppy disk drives to the Z-181. No matter what letter the operating system assigns to the hard disk, it must still "see" it as hard disk "C" (hard disk 0) at the hardware level.

Now, start filling up that hard disk!

✱



A decorative border of stylized yellow and orange flames surrounds the central text. The flames are jagged and pointed, with a red outline. The background is a light gray.

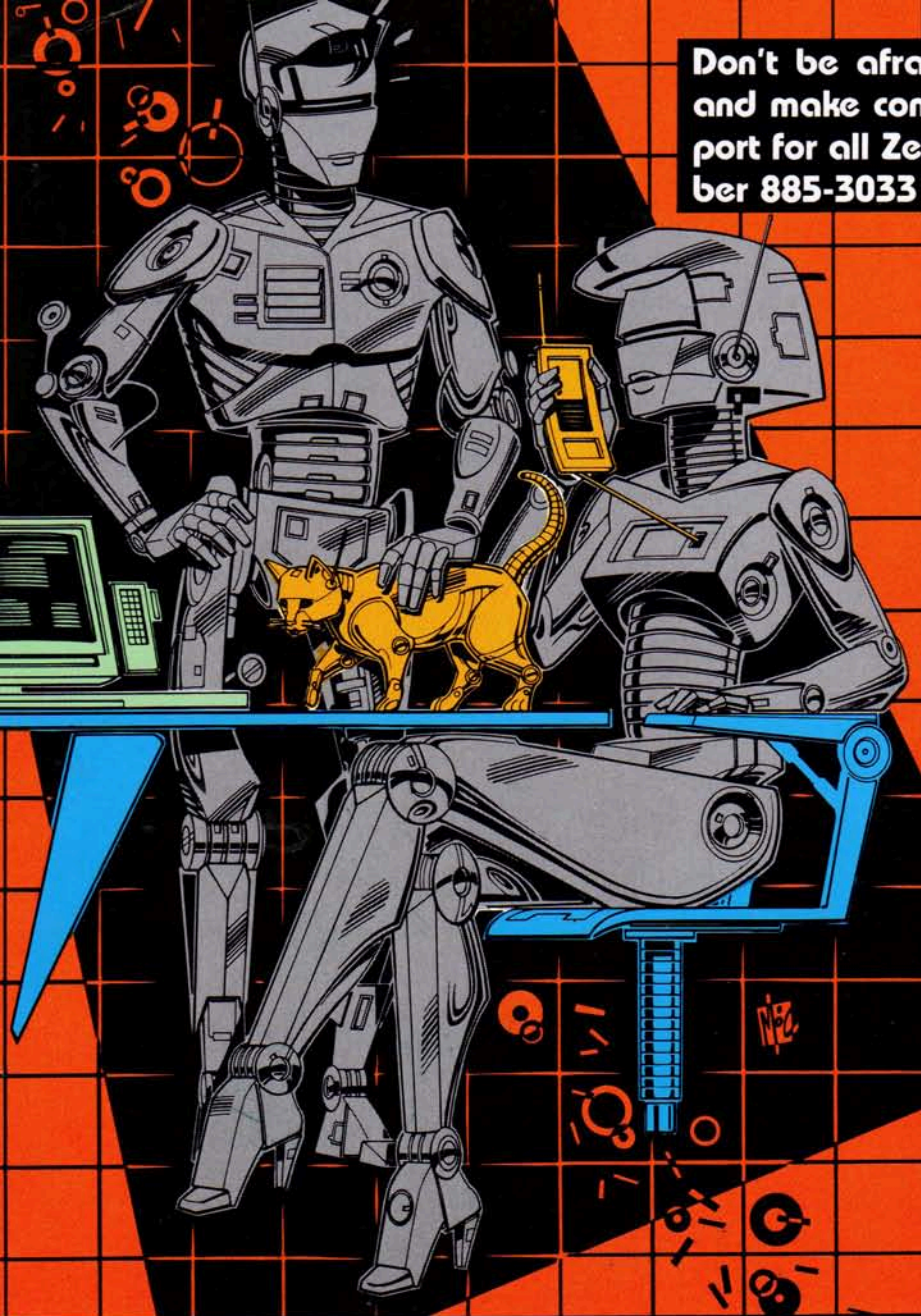
HADES II

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 (F5) 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 Window By Way Of MODEM Protocol.
- F5 -- Allow Sending Data From Disk, Using Either NON-MOBF, Which Optionally Can Be Ignored, Or MODEM Protocol.
- F6 -- Enters The Setup Mode So This Software Can Be Configured.
- F7 -- Cleans 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 (0-0), (F1) To List, Anything Else To Abort --> _

F1-List F2-Msg F3-BufR F4-Sav F5-Snd F6-Cfg F7-Clr F8-Print F9-Exit CM

HUGMCP Configuration Menu #1

- F -- This Function Allow The Baud Rate To Be Changed. Depending Upon Which Mode You're Using, Normally It Would Be Set In Either 1200, 1800, Or 2400 Baud. Some Connection To A Host, Will Allow Higher Baud Rates.
- E -- This Function Allow You To Change The Word Parity. Normally, you Should Change "No Parity". This Is Acceptable By Most Remote Systems, and It Is Also Necessary For MODEM Protocol To Work Properly.
- C -- This Function Allow The Changing Of The Word Length. Normally The Length Should Be Set To 8 Data Bits. This Length Is Acceptable By Most Remote Systems, and It Is Necessary For MODEM Protocol To Work Properly.
- S -- This Selection Allow You To Enter Messages Which Can Be Automatically Set With The F2 Key. Up To 14, 10-Character Messages Can Be Saved. Selection (0) Is Special. It Should Contain Your Computer's ID Number And Password. Selection (9) Is Also Special. This Selection Can Be Set Actually Be Set When This Program Is First Executed To Selecting The Proper Option During Setup.

Type (F6)C (F4)F: For New Help, Anything Else To Continue:

F1-List F2-Msg F3-BufR F4-Sav F5-Snd F6-Cfg F7-Clr F8-Print F9-Exit CM

HUGMCP Configuration Menu:

- 0 --> Modify Baud Rate
- 1 --> Modify Parity Type
- 2 --> Modify Word Length
- 3 --> Modify Or Add Auto-Messages
- 4 --> Miscellaneous Functions
- 5 --> Change Screen Color Assignments
- 6 --> Display Current Configuration
- 8 --> Make Changes Permanent

Select 0-6, (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-List F2-Msg F3-BufR F4-Sav F5-Snd F6-Cfg F7-Clr F8-Print F9-Exit CM



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

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

[BLKJEXP: 111490 ID: 007354
R Mehaffey
P.O. Box 7334
Fort Landerdale, FL

33338-7334

POSTMASTER: If undeliverable,
please do not return.

\$2.50
P/N 885-2125